GENERACIÓN AUTOMÁTICA DE MARCHAS PARA UN ROBOT HUMANOIDE (BIOLOID)

Autor
Cristian David Villate Martínez

FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍA MECÁNICA, MECATRÓNICA E INDUSTRIAL
PROGRAMA DE INGENIERÍA MECATRÓNICA
PAMPLONA, DICIEMBRE DE 2016

GENERACIÓN AUTOMÁTICA DE MARCHAS PARA UN ROBOT HUMANOIDE (BIOLOID)

Cristian David Villate Martínez

Trabajo de grado presentado como requisito para optar por el título de:
Ingeniero en Mecatrónica

Director
PhD. CÉSAR AUGUSTO PEÑA CORTÉS
Docente Universidad de Pamplona

FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍA MECÁNICA, MECATRÓNICA E INDUSTRIAL
PROGRAMA DE INGENIERÍA MECATRÓNICA
PAMPLONA, DICIEMBRE DE 2016

Dedicatorias por Cristian Villate:

Quiero dedicarle este logro que me ha permitido llegar hasta aquí a pesar de las adversidades.

De una manera muy especial a mis padres Humberto Villate y Nancy Martínez, quienes incansablemente han hecho todo por sacarme adelante, por

Estar conmigo en esos momentos tan difíciles que se presentaron y no desfallecer los amo inmensamente.

A mi hermana Yenny Villate, por apoyarme en mis decisiones y confiar en mis capacidades por enorgullecerse de mis logros gracias "china" yo la quiero mucho.

Se lo dedico a mis familiares por todo ese amor y cariño que me han prestado toda la vida, por confiar en mis capacidades y siempre apoyarme

Por último, se lo dedico a mis amigos quienes me han acompañado y brindado su apoyo en todo momento y a los que hoy les digo: "Gracias por todo".

AGRADECIMIENTOS

.

Agradecerle al director del proyecto PhD. CÉSAR AUGUSTO PEÑA CORTÉS, por su conocimiento en mi formación académica, sus consejos y apoyo en cada parte del proceso, que como resultado llevo a la culminación del proyecto.

Al PhD. OSCAR EDUARDO GUALDRON GUERRRO por sus conocimientos en mi formación académica su colaboración para que este proyecto culminara satisfactoriamente.

A mi amigo Ing.(c). JULIAN ANDRES BAREÑO, gracias a su amistad y toda la colaboración que me presto cuando la necesite para no solo cumplir este proyecto si no en mi formación académica y fuera de la universidad muchas gracias.

A mi amigo Ing. (c). LEONARDO LOPEZ GELVEZ, gracias por su amistad y su ayuda estoy muy agradecido por todo esto y lo tendré muy presente.

A mi amigo Ing. FABIO ANDRES MORALES por todos esos años de amistad y estar presente así estemos lejos inmensamente agradecido y espero que la vida nos traiga buenas oportunidades a los dos.

A mi primo JUAN GABRIEL MENDEZ MARTINEZ por estar conmigo en todo momento, por la confianza entregada en todos instantes y la ayuda brindada desde niños.

A los compañeros y amigos JOSE LUIS DIAZ, YEISON PEREA, JULIAN MUÑOZ, DANIEL VILLALBA Y RUBEN SANCHEZ por su colaboración en el desarrollo de este proyecto.

A todas las personas que en me colaboraron cuando lo necesite gracias por su ayuda.

Y finalmente a la Universidad de Pamplona por todas estas experiencias vividas en mi etapa universitaria gracias.

RESUMEN

En este proyecto se propone el desarrollo e implementación de algoritmos computacional para generar marchas autónomas para el robot humanoide Bioloid. Este robot posee dieciocho grados de libertad. El algoritmo contempla las restricciones cinemáticas presentes dentro del modelo físico del robot y crea las trayectorias de forma estocástica para generar la marcha, se realizó el desarrollo desde su análisis matemático pasando por el desarrollo de un modelado CAD de cada pieza que compone al robot para su implementación en el simulador de trayectorias donde se visualiza de una forma más sencilla todos movimiento que se le ingresen del resultado del generador de trayectorias autónomas. El cual opera bajo el concepto de colisiones en espacios de n dimensiones.

La validación de los algoritmos se hace por medio de pruebas en simulación donde se obtuvo tanto la marcha final del robot como simulaciones que representasen colisiones en espacios bidimensionales y tridimensionales para finalmente realizar experimentales con el robot físico donde se obtiene la mejor marcha en cuestión de estabilidad y así generar autonomía al robot para poder realizar una marcha.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. PRELIMINARES.

1. PRELIMINARES.

1.1. INTRODUCIÓN.

Este trabajo desea presentar el desarrollo e implementación de un algoritmo computacional para generar marchas automáticas para un robot humanoide (Bioloid).

La definición de un robot humanoide son los robots que presentan características similares en la anatomía con los seres humanos, en esta investigación el modelo robótico a tratar es el Bioloid Premium Type A, que presenta movilidad bípeda.
Al ser un robot bípedo su locomoción se realiza de forma similar a la de un humano, eso incluye todas las restricciones presentes para generar una trayectoria.

Al existir restricciones para generar una trayectoria debe surgir un algoritmo que contemple y analice las posibilidades para realizar una trayectoria, se plantea la pregunta ¿Qué es un algoritmo? Un algoritmo es un conjunto ordenado de secuencias que permiten dar solución a un problema.

Por su velocidad de procesamiento superior se debe realizar el algoritmo en un máquina electrónica de computo (computador) que brinde las garantías para desarrollar la trayectoria para el robot humanoide bioloid.

Cuando se trabaja con robots se desea que no posea intervención cuando se esté desarrollando una tarea esto quiere decir que debe ser automática su respuesta, si se acopla todos los ítems anteriormente expuestos se tendría una trayectoria automática, pero al ser un robot humanoide se debe desarrollar una marcha que básicamente es una serie de trayectorias acopladas entre sí.

En este trabajo se describe de forma detallada como se desarrolló el proyecto desde su fase de concepción incluyendo las dificultades que se presentaron hasta su implementación final.

1.2. PLANTEAMIENTO DEL PROBLEMA.

La robótica humanoide debido a su semejanza a la morfología humana presenta características similares en su locomoción, desarrollando las mismas dificultades de desplazamiento donde mantener el equilibrio son conflictos bastante recurrentes dentro de su operatividad. Para poder solucionar estos problemas se debe generar en primera instancia una solución al problema básico de desplazamiento.
Desde un enfoque tradicional en robótica para generar trayectorias se debe conocer el recorrido que el robot realice para determinada tarea, pero en ocasiones las soluciones son tan múltiples, los gastos computacionales de todos los puntos para que realice la trayectoria son enormes o en definitiva solo se tiene a la mano el punto inicial y final del recorrido, sin contar que al operar el robot su autonomía y adaptabilidad a la hora de realizar un desplazamiento son muy mínimas.
Por estos motivos se propone realizar un algoritmo computacional que de forma estocástica genere una trayectoria en el espacio de la tarea del robot humanoide para que este realice una marcha.

1.3. JUSTIFICACIÓN.

El desplazamiento es el pilar fundamental de la operatividad del robot, ya que si no se presenta difícilmente realice una tarea que se le asigne. Todo desplazamiento está compuesto por una serie de marchas que junto a su morfología irregular vuelve muy recurrente que al realizar un movimiento ocurra una caída o ruptura estructural que pueda estropear el robot. En base a esto surge la necesidad de realizar un algoritmo que genere marchas de forma automática.

Adicionalmente para desarrollar una marcha se contemplan matemáticamente todas las coordenadas por las cuales el robot debe pasar, pero al realizar esto el robot pierde autonomía, por esta razón el algoritmo a su vez debe ser estocástico para que crear así la mayor autonomía posible en el robot.

1.4. OBJETIVOS.

1.4.1. OBJETIVO GENERAL.

- Desarrollar e implementar algoritmos computacionales para generar marchas automáticas para un robot humanoide bípedo (Bioloid Premium A)

1.4.2. OBJETIVOS ESPECÍFICOS.

- Desarrollar un simulador cinemático para la implementación de los algoritmos computacionales.

- Realizar los algoritmos de generación automática de trayectorias, contemplando las restricciones cinemáticas.

- Verificar de forma experimental el comportamiento de los algoritmos computacionales implementados en el robot humanoide bípedo (Bioloid Premium A).

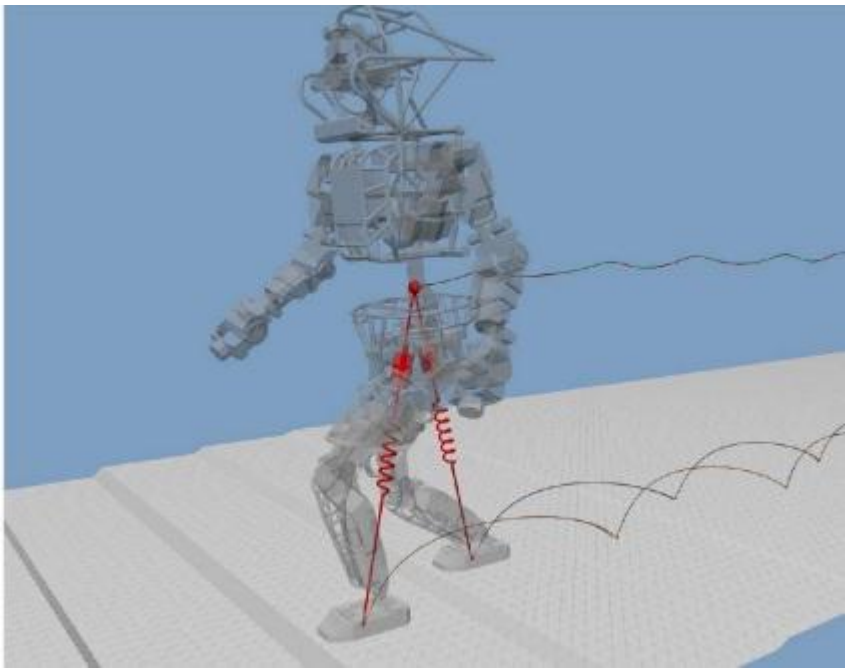
CAPÍTULO 2.
ESTADO DEL ARTE.

2. ESTADO DEL ARTE.

En la robótica humanoide donde se presenta movilidad bípeda la estabilidad juega un papel imprescindible que debe tomarse en consideración por su estructura anatómica. Por esta razón las piernas de un robot humanoide se debe hacer hincapié ya que deben ser capaces de realizar una variedad de tareas, incluyendo la interacción con el medio ambiente mientras se mantiene el equilibrio [7]. Existen delimitaciones básicas Para hacer frente a estos escenarios como por ejemplo, el robot debe ser capaz de moverse en entornos no estructurados [8].

Se ha abordado la problemática del desplazamiento bípedo en humanoides analizando los puntos ciegos que puede tener un robot al desplazarse, llegando a estrategias de soluciones como la utilización del metodo“dual-SLIP 3D” [4]. El cual consiste en un sistema que detecta cuando el terreno es plano e irregular para así caminar de una manera más eficiente economizando recursos tanto energéticos como computacionales. Siendo eficiente para un solo paso en terrenos irregulares el entrenamiento se realizó off line y el terreno no era previamente conocido por esta razón para algunos tramos hubo un desconocimiento de ciertos puntos pero logro adaptarse eficientemente.

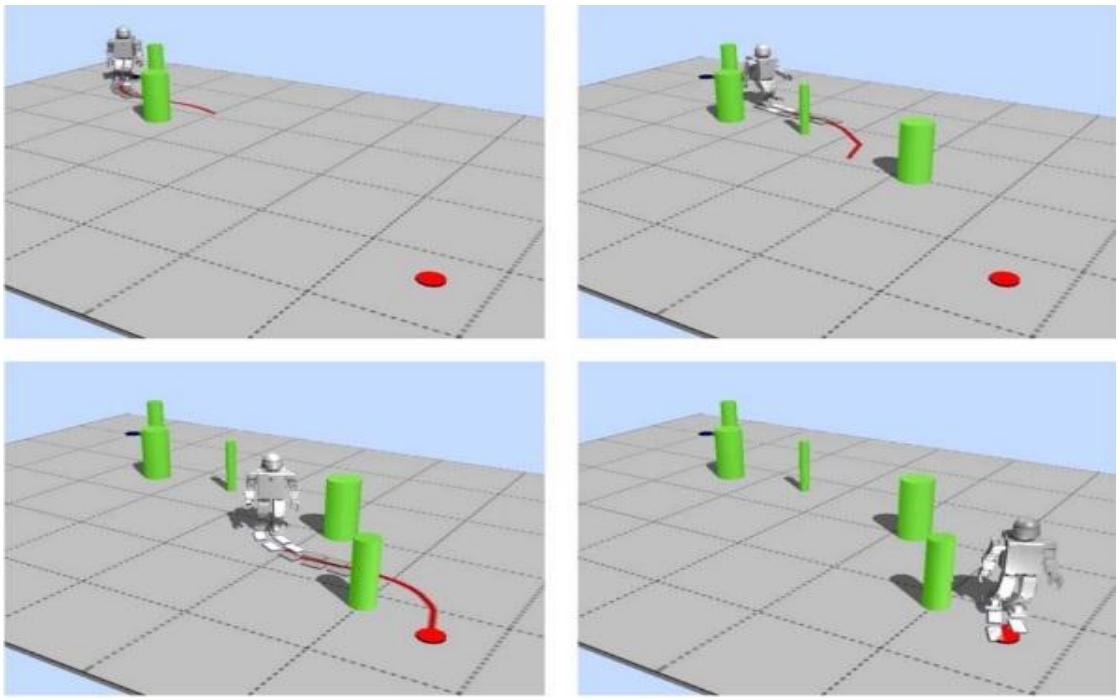
FIGURA 1: MOVIMIENTO DEL ROBOT HUMANOIDE APLICANDOLE EL METODO “DUAL-SLIP 3D”.



FUENTE: K. TEACHASRISAKSAKUL, Z. ZHANG, G. YANG, AND B. LO, “IMITATION OF DYNAMIC WALKING WITH BSN FOR HUMANOID ROBOT,” VOL. 19, NO. 3, PP. 794–802, 2015.

Mantener el equilibrio en un sistema robótico humanoide es una tarea ardua, ahora realizar una marcha donde involucre evadir obstáculos será más complicado aún para esta tarea una estrategia de soluciones es reducir la navegación a multiobjetivos midiendo las distancias que existen entre el objetivo, el obstáculo y el robot activando así algoritmos de marcha para las diferentes situaciones que se le presenten [5].

FIGURA 2: ESTRATEGIA DE SOLUCIÓN REDUCCIÓN A MULTIOBJETIVOS.

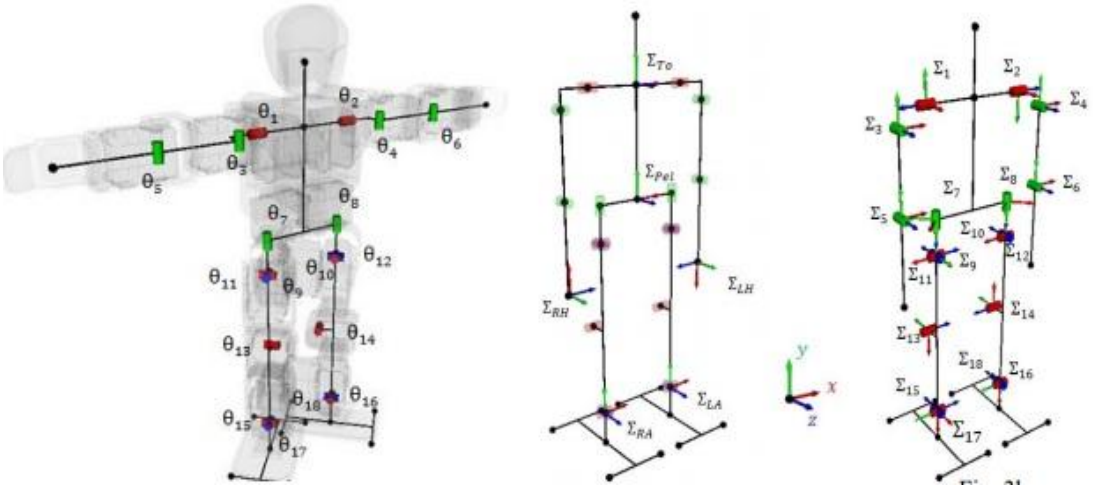


FUENTE: K. LEE, H. MYUNG, AND J. KIM, "ONLINE MULTIOBJECTIVE EVOLUTIONARY APPROACH FOR NAVIGATION OF HUMANOID ROBOTS," VOL. 62, NO. 9, PP. 5586–5597, 2015.

Otras estrategias de solución puede ser la imitación de movimiento por que en el aprendizaje humano se utilizan los ejemplos como método de enseñanza, por esta razón también aplica cuando se desea enseñar a caminar a una persona, en este orden de ideas se podría extrapolar esta enseñanza a un robot, gracias a su capacidad de procesamiento mucho más rápida que un ser biológico desarrollando un aprendizaje más veloz, las imitaciones se realizaron con mecanismos de captaciones de movimientos estos se guardaron en la memoria del robot para que este los procesara y finalmente los replicara [2].

El modelo físico de la investigación es el "Bioloid Premium Humanoid type A" el cual es un robot tanto de entretenimiento como una plataforma robótica de investigación que es usada recurrentemente en diferentes investigaciones. Como por ejemplo un análisis cinemático directa e inversos, contemplando todos los puntos singulares presentes en el robot [3].

FIGURA 3: ANÁLISIS CINEMATICO ROBOT BIOLOID.

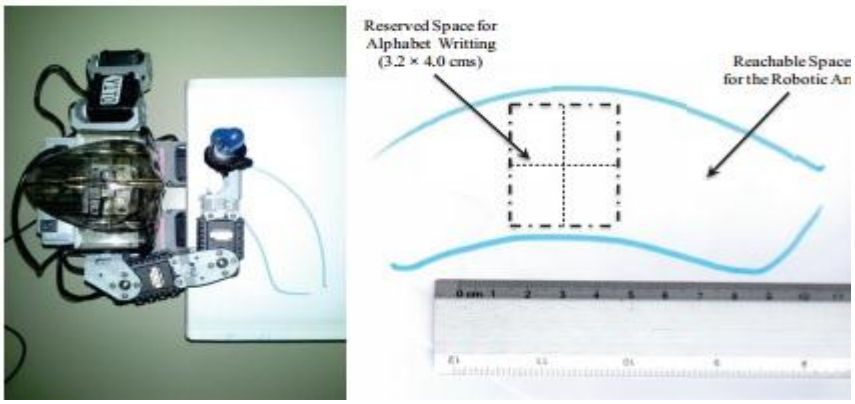


FUENTE: J. V. NUNEZ, A. BRISENO, D. A. RODRIGUEZ, J. M. IBARRA, AND V. M. RODRIGUEZ, "EXPLICIT ANALYTIC SOLUTION FOR INVERSE KINEMATICS OF BIOLOID HUMANOID ROBOT *," PP. 33–38, 2012.

Debido a su versatilidad el robot bioloid se puede utilizar para diversas aplicaciones ejemplo de ellos es el análisis de desplazamiento y más puntualmente en el análisis de velocidades cuando se incorporó sensores propioceptivos, donde es corroborado de forma interna tanto el análisis matemático que se realizó al modelo físico del robot como su respuesta real, se realizó pruebas experimentales tanto simuladas como implementadas en el robot y el desfase de respuesta del modelo matemático al fisco es casi insignificante cuando el robot está completamente cargado o su carga se encuentra en limites operables, cuando esta empieza a disminuir su desfase se hace más significativo. Concluyendo así que la alimentación juega un papel relevante en la operatividad del robot puesto que si no se obtiene la energía necesaria para realizar la operación los controladores dinámicos harán todo lo posible por acercarse al modelo cinemático matemático[1].

El robot bioloid posee investigaciones tan variadas que algunas son tareas que por su anatomía humana sería factible realizar como el desarrollo de una rutina para que el humanoide dependiendo de una codificación de puntos realizara una letra del abecedario español. Se podría utilizar para personas con alguna discapacidad locomotriz siendo este un intérprete con el resto de personas que lo rodean escribiendo lo que el desee por el mando aparte de atenderlo si este lo desea, a su vez podría comunicar alguna eventualidad de forma más práctica [6].

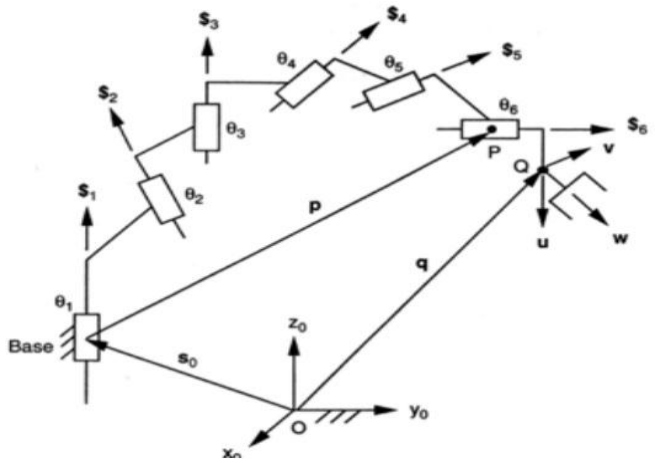
FIGURA 4: ROBOT BIOLOID REALIZANDO PRUEBAS PARA REALIZAR LETRAS DEL ABECEDARIO ESPAÑOL.



FUENTE: P. DIVISION AND H. DE LEON, "DEVELOPMENT OF MOTION MODELS FOR WRITTING OF THE SPANISH ALPHABET ON THE HUMANOID BIOLOID ROBOTIC PLATFORM," PP. 217–224, 2014.

Como se pudo analizar la investigaciones en el campo de la robótica son bastante extensas y diversas. En este caso particular el componente teórico en que se basa la investigación consiste en analizar el robot bioloid cinematicamente por el método de los srews [9].

FIGURA 5: MÉTODO DE DESPLAZAMIENTO SUCESIVO SCREW.



FUENTE: TSAI - ROBOT ANALYSIS.

Se facilita el análisis en morfologías irregulares, a su vez el análisis de desplazamiento será pseudoestatico para no se afectado tan significativamente por las fuerzas dinámicas.

Los algoritmos serán de forma estocástica incluyendo las limitaciones cinemáticas y mecánicas que el robot presente generando una trayectoria para un desplazamiento requerido

CAPÍTULO 3.

PLANTEAMIENTOS DE LOS ALGORITMOS.

3. PLANTEAMIENTO DE LOS ALGORITMOS.

3.1. FUNDAMENTO MATEMÁTICO.

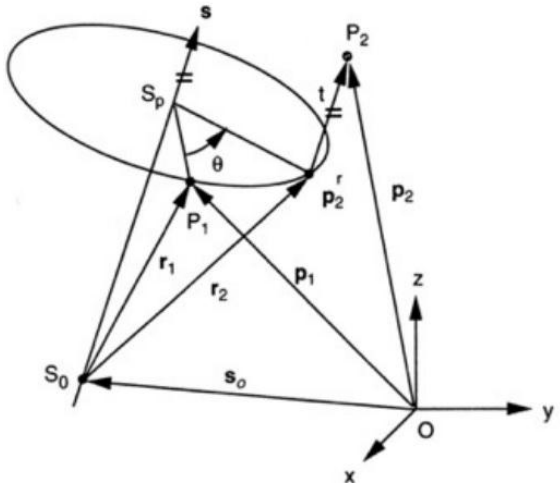
Los robots son máquinas que involucran aspectos como el análisis cinemático directo o inverso, igual que la dinámica y la planificación trayectorias. En este proyecto se plantea un análisis de movimiento de forma pseudoestatica (las velocidades desarrolladas por las articulaciones son bajas), con la finalidad de desestimar las fuerzas dinámicas involucradas en el movimiento (fuerzas de coriolis, centrifuga, centrípeta, etc.), que si se contemplase todo estos aspectos por su gasto computacional no se podría implementar de una manera eficiente.

El análisis cinemático directo del robot humanoide, si se le analizase de forma tradicional utilizando el método de Denavit-Hartenberg [5], tendría un grado elevado de complejidad dada su estructura. Con el fin realizar el análisis se propone el uso de métodos modernos. Por su simplicidad y eficacia a la hora de la realización de los cálculos en geometrías complejas se utilizó el método de desplazamiento Screws

3.1.1. MÉTODO DE DESPLAZAMIENTO SCREWS.

Según el teorema de Charles, el desplazamiento general de un cuerpo rígido en el espacio, puede ser representado por medio una rotación más una translación La combinación de una rotación seguida de una translación sobre un eje es conocido como un desplazamiento screw [9].

FIGURA 6: REPRESENTACION VECTOTIAL DE UN DESPLAZAMIENTO SCREW.



FUENTE: TSAI - ROBOT ANALYSIS.

Dado la fórmula de rodríguez:

$$\boldsymbol{P}_2 = \boldsymbol{S}_0 + t\boldsymbol{s} + (\boldsymbol{P}_1 - \boldsymbol{S}_0)cos\,\emptyset + \boldsymbol{S} \times (\boldsymbol{P}_1 - \boldsymbol{S}_0)sin\,\emptyset + [(\boldsymbol{P}_1 - \boldsymbol{S}_0)\boldsymbol{T}\,\boldsymbol{s}]\boldsymbol{s}(1 - cos\,\emptyset) \quad (1)$$

Donde $\boldsymbol{P}_n$ representa el punto de desplazamiento de un punto, $\boldsymbol{S}_0)$ es el vector perpendicular del sistema de origen al punto de desplazamiento y $\boldsymbol{S}$ representa el vector screw, según la fórmula de Rodríguez. Esta formula también se puede escribir en términos de una matriz de transformación homogénea definida por $\boldsymbol{A}$.

Donde sus componentes son:

$$a_{11} = \left(S_x{}^2 - 1\right)(1 - \cos\emptyset) + 1 \quad (2)$$

$$a_{12} = S_x S_y(1 - \cos\emptyset) - S_z \sin\emptyset \quad (3)$$

$$a_{13} = S_x S_z(1 - \cos\emptyset) + S_y \sin\emptyset \quad (4)$$

$$a_{21} = S_y S_x(1 - \cos\emptyset) + S_z \sin\emptyset \quad (5)$$

$$a_{22} = \left(S_y{}^2 - 1\right)(1 - \cos\emptyset) + 1 \quad (6)$$

$$a_{23} = S_y S_z(1 - \cos\emptyset) - S_x \sin\emptyset \quad (7)$$

$$a_{31} = S_z S_x(1 - \cos\emptyset) - S_y \sin\emptyset \quad (8)$$

$$a_{32} = S_z S_y(1 - \cos\emptyset) + S_x \sin\emptyset \quad (9)$$

$$a_{33} = \left(S_z{}^2 - 1\right)(1 - \cos\emptyset) + 1 \quad (10)$$

$$a_{14} = tS_x - S_{0x}(a_{11} - 1) - S_{0y}a_{12} - S_{0z}a_{13} \quad (11)$$

$$a_{24} = tS_y - S_{0x}a_{21} - S_{0y}(a_{22} - 1) - S_{0z}a_{23} \quad (12)$$

$$a_{34} = tS_z - S_{0x}a_{31} - S_{0y}a_{32} - S_{0z}(a_{33} - 1) \quad (13)$$

$$a_{41} = 0 \quad (14)$$

$$a_{42} = 0 \quad (15)$$

$$a_{43} = 0 \quad (16)$$

$$a_{44} = 1 \quad (17)$$

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{34} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \quad (18)$$

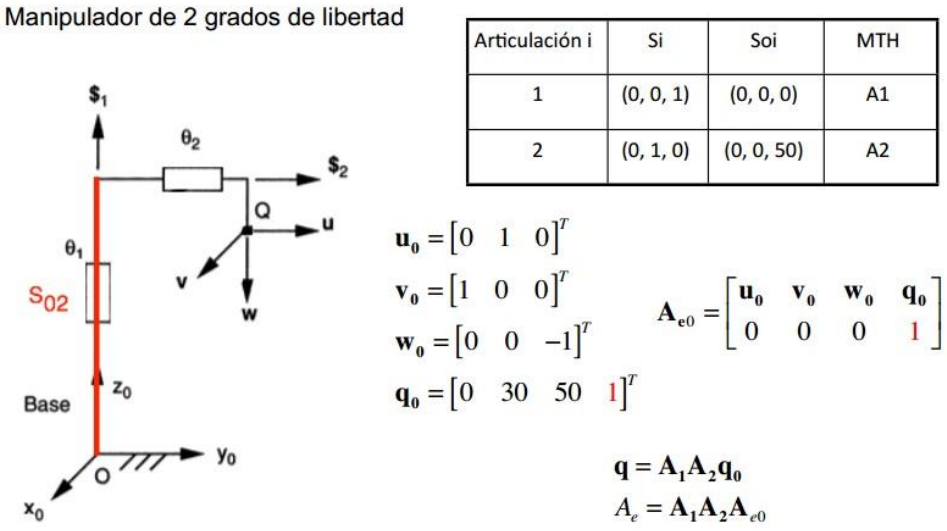
Para calcular el desplazamiento y del efector final del robot desde la posición de referencia ($\boldsymbol{q}_0$) hasta la posición objetivo ($\boldsymbol{q}$) se considera como la resultante de desplazamiento de n screws sucesivos, lo cual puede ser calculado por medio de la denominada ecuación de cierre:

$$\boldsymbol{A_h = A_1 A_2 A_3 A_4 A_5 A_{n-1} A_n} \quad (19)$$

$$\boldsymbol{q = A_1 A_2 A_3 A_4 A_5 A_{n-1} A_n q_0} \quad (20)$$

A manera de ejemplo para comprender el concepto se genero una imagen (ver figura 7) con un robot manipulador de dos grados de libertad donde se aprecia la disposición de los Screws, las matrices de transformación homogénea y las ecuación de cierre correspondiente.

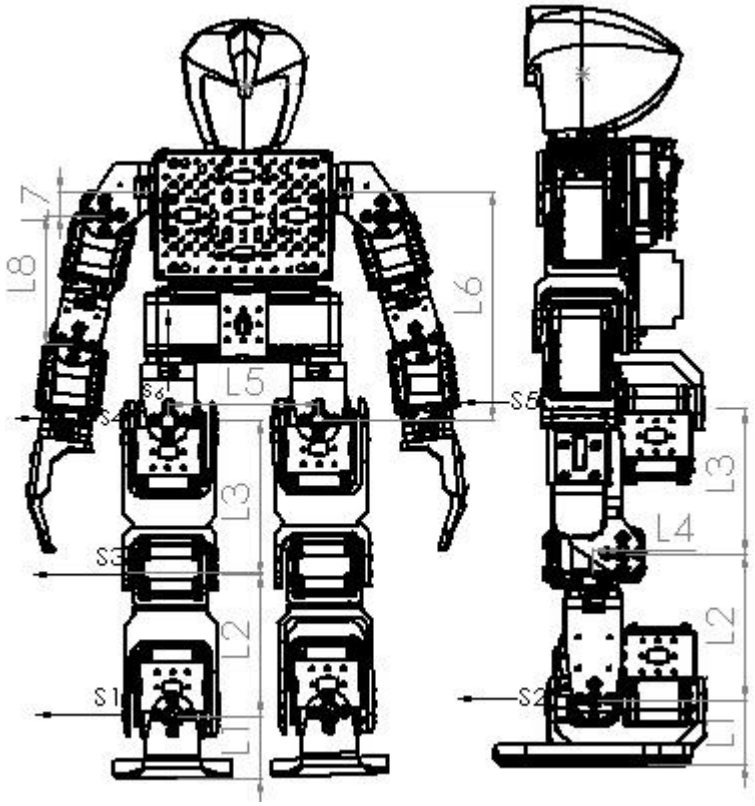
FIGURA 7: EJEMPLO DE MOVIMIENTO SCREW MANIPULADOR DE DOS GRADOS.



Articulación i	Si	Soi	MTH
1	(0, 0, 1)	(0, 0, 0)	A1
2	(0, 1, 0)	(0, 0, 50)	A2

FUENTE: TSAI - ROBOT ANALYSIS.

FIGURA 8: REPRESENTACION VECTOTIAL DEL DESPLAZAMIENTO SCREW ROBOT BIOLOID.



FUENTE: AUTOR.

Los cálculos cinemáticos por el método de los Screws para el robot bioloid se empezaron en los pies puestos que estos son los que afectan más significativamente la marcha, luego se decidió realizar los cálculos para los brazos para así analizar todas las articulaciones que el robot posee (figura8).

Los resultados se agruparon en tablas donde se representa la articulación a analizar, el vector screw, el vector perpendicular desde el origen y la matriz de transformación homogénea que representa

TABLA 1: ANÁLISIS POR EL MÉTODO DE LOS SCREWS DEL PIE DE APOYO DEL ROBOT BIOLOID.

Análisis del pie de apoyo del robot bioloid			
Articulación (i)	S_i	S_{0i}	***MTH***
1	(0, -1, 0)	(0, 0, L1)	A1
2	(1, 0, 0)	(0, 0, L1)	A2
3	(0, -1, 0)	(-L4, 0, L1+ L2)	A3
4	(0, -1, 0)	(0, 0, L1+ L2+L3)	A4
5	(1, 0, 0)	(0, 0, L1+ L2+L3)	A5
6	(0, 0, 1)	(0, 0, 0)	A6

FUENTE: AUTOR

TABLA 2: ANÁLISIS POR EL MÉTODO DE LOS SCREWS DEL PIE DE MÓVIL DEL ROBOT BIOLOID.

Análisis del pie de apoyo del robot bioloid			
Articulación (i)	S_i	S_{0i}	***MTH***
7	(0, -1, 0)	(0, 0, L1)	A7
8	(1, 0, 0)	(0, L5, L1)	A8
9	(0, -1, 0)	(-L4, 0, L1+ L2)	A9
10	(0, -1, 0)	(0, 0, L1+ L2+L3)	A10
11	(1, 0, 0)	(0, L5, L1+ L2+L3)	A11
12	(0, 0, 1)	(0, L5, 0)	A12

FUENTE: AUTOR

TABLA 3: ANÁLISIS POR EL MÉTODO DE LOS SCREWS DEL BRAZO DERECHO DEL ROBOT BIOLOID.

Análisis del pie del brazo derecho del robot bioloid			
Articulación (i)	S_i	S_{0i}	***MTH***
13	(0, -1, 0)	(0,0,L1+L2+L3+L6)	A13
14	(1, 0, 0)	(0,-L7,L1+L2+L3+L6-L8)	A14
15	(0, -1, 0)	(0,-L7-L9,L1+L2+L3+L6-L8)	A15

FUENTE: AUTOR

TABLA 4: ANÁLISIS POR EL MÉTODO DE LOS SCREWS DEL BRAZO IZQUIERDO DEL ROBOT BIOLOID.

Análisis del pie del brazo derecho del robot bioloid			
Articulación (i)	S_i	S_{0i}	***MTH***
16	(0, -1, 0)	(0,L5,L1+L2+L3+L6)	A16
17	(1, 0, 0)	(0,L7+L5,L1+L2+L3+L6-L8)	A17
18	(0, -1, 0)	(0,L7+L9+L5,L1+L2+L3+L6-L8)	A18

FUENTE: AUTOR

3.1.2. **CENTROS DE MASAS DEL ROBOT BIOLOID.**

Si se desea realizar un análisis de posición es de suma importancia tener claridad donde se concentra la mayor cantidad de masa dentro de un sistema a esto se le conoce como centro de masas.

Como se realizó un análisis pseudoestatico del robot bioloid se debe conocer donde se encuentran los centros de masas ya que son afectados directamente por las fuerzas dinámicas. Para calcular los centros de masas estos dependen de la geometría del material y el tipo de material o materiales que estén presentes.

Gracias a que el software de diseño incluye opciones de cálculo de diferentes propiedades (ver página 33) no se incluirá las ecuaciones del cálculo de centros de masa de cada eslabón, pero el centro de masa final se calculó dada la fórmula:

$$\overline{X_T} = \frac{\sum_{i=1}^{n} \overline{X_i} * m_i}{\sum_{i=1}^{n} m_i} \quad (20)$$

$$\overline{Y_T} = \frac{\sum_{i=1}^{n} \overline{Y_i} * m_i}{\sum_{i=1}^{n} m_i} \quad (21)$$

$$\overline{Z_T} = \frac{\sum_{i=1}^{n} \overline{Z_i} * m_i}{\sum_{i=1}^{n} m_i} \quad (22)$$

Donde $\overline{X_T}$, $\overline{Y_T}$, $\overline{Z_T}$ son las coordenadas del centro de masas final $\overline{X_i}$, $\overline{Y_i}$, $\overline{Z_i}$ son las coordenadas de cada centro de masa de cada eslabón y m_i es la masa correspondiente de cada eslabón.

3.1.3. **ALGORITMO COMPUTACIONAL METODOS DE LOS SCREWS.**

A razón de los múltiples cambios angulares que se desarrollan mientras se está operando el robot y por ser parte fundamental del desarrollo del algoritmo computacional generador de trayectorias se decidió realizar un algoritmo computacional en el lenguaje de programación Matlab ® que representara todo los cálculos por el método de los Screws del robot.

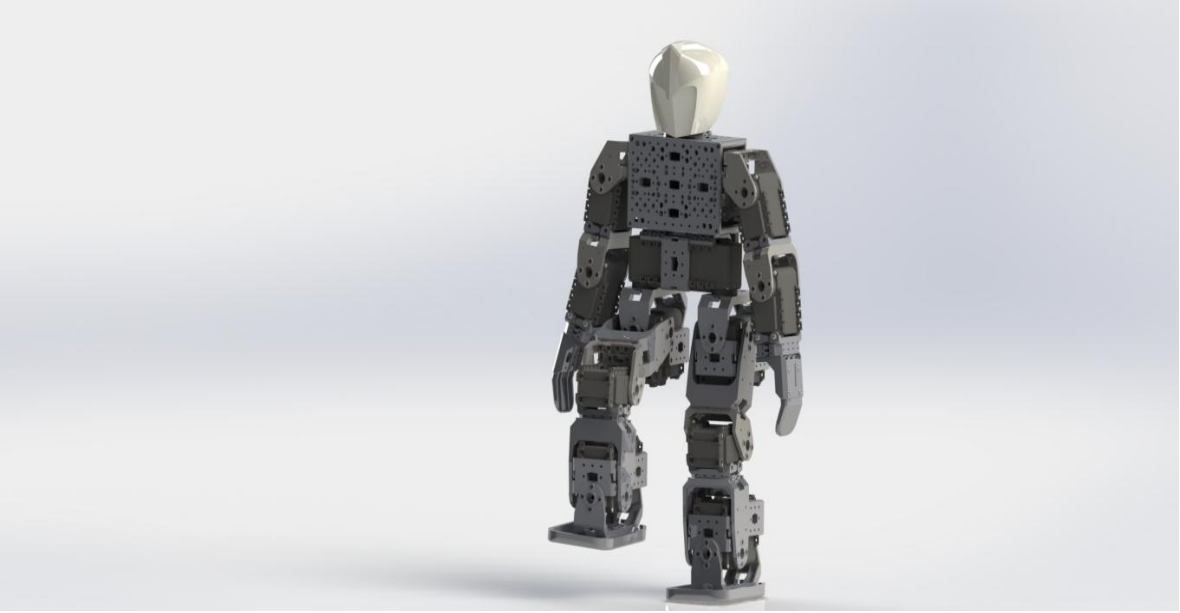
El código completo se anexo al apéndice del libro (ver página 70).

3.2. DISEÑO CAD.

Se realizó un diseño asistido por computador (CAD) para poder digitalizar el modelo físico del robot bioloid Premium type A, a su vez sirvió para poder analizar los tipos de desplazamientos que realizaría al reflejarlos al robot en físico. De igual forma se pudo determinar con precisión los centros de masa de los eslabones y las medidas.

El diseño CAD sirvió a su vez para poder generar las piezas con sus acabados finales e importarlas desde el simulador de marchas del robot. De forma sinérgica se unieron tanto los datos obtenidos por el modelo CAD y los proporcionados por el fabricante.

FIGURA 9: RENDERIZADO DEL ROBOT BIOLOID.



FUENTE: AUTOR.

3.2.1. **OBTENCIÓN DEL MODELO CAD.**

La realización de un modelo digital del robot bioloid es un apartado de suma importancia debido a que por medio de este modelado se obtienen los centros de masas de todos los eslabones que permiten obtener el centro de masas total del robot, además de servir de intermediario para el ensamblaje final dentro del simulador.

Por el tiempo que llevaría modelar todas las piezas presentes en el robot bioloid type A, ya que es un diseño bastante robusto y el objetivo de la investigación no es el modelado de este robot se decidió buscar las piezas en la web en páginas de libre tráfico de diseños digitales donde se obtuvieron las piezas en la página GrabCad.

En esta página se obtuvieron las piezas individuales que componen al robot bioloid type A para realizar el ensamblaje final del robot. Las piezas se pasaron a formato .prt que es el formato de modelado del software de diseño solidworks ®.

3.2.2. **PIEZAS BÁSICAS DEL ROBOT BIOLOID TYPE A.**

Por ser un robot destinado a personas que no poseen conocimientos en ingeniería su ensamblaje debe ser lo más fácil posible y es así que bastantes piezas de este robot se utilizan para formar diferentes partes del cuerpo de una forma similar a piezas de Lego ® (ver anexos).

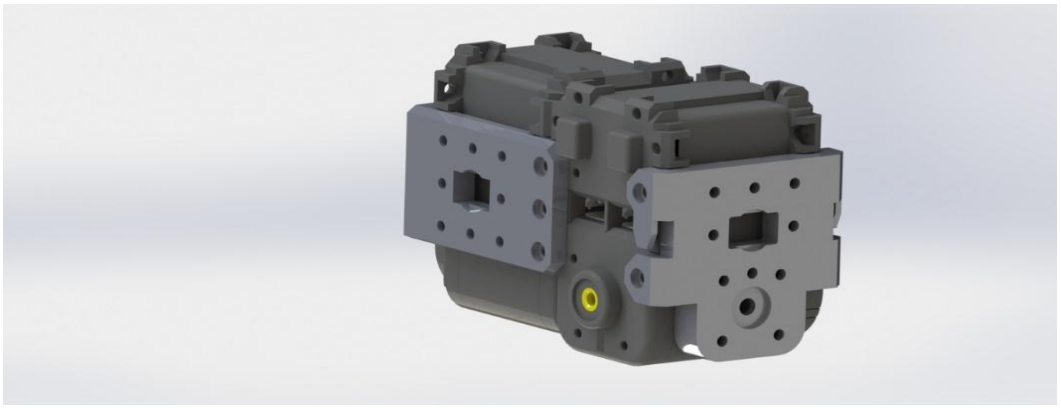
3.2.3. **ESLABONES DEL ROBOT BIOLOID TYPE A.**

Los cálculos matemáticos del robot necesitan un medio físico donde se reflejen los movimientos que realizan los servomotores, estos medio físicos deben enlazarse entre sí para generar una sola cadena cinemática, estos son los atributos que debe poseer los eslabones del robot. Cada eslabón está compuesto por las piezas básicas que se encuentran presentes dentro de la estructura del mismo

3.2.3.1. ESLABONES NÚMERO UNO, CUATRO, OCHO Y ONCE DEL ROBOT BIOLOID TYPE A.

Está relacionado con el primer, cuarto, octavo y onceavo movimiento articular del robot al nivel matemático es el eslabón presente dentro de las matrices de transformación homogénea A1, A4, A8 y A11 (ver tabla 1 y tabla 2).

FIGURA 10: RENDERIZADO DEL PRIMER ESLABON DEL ROBOT.

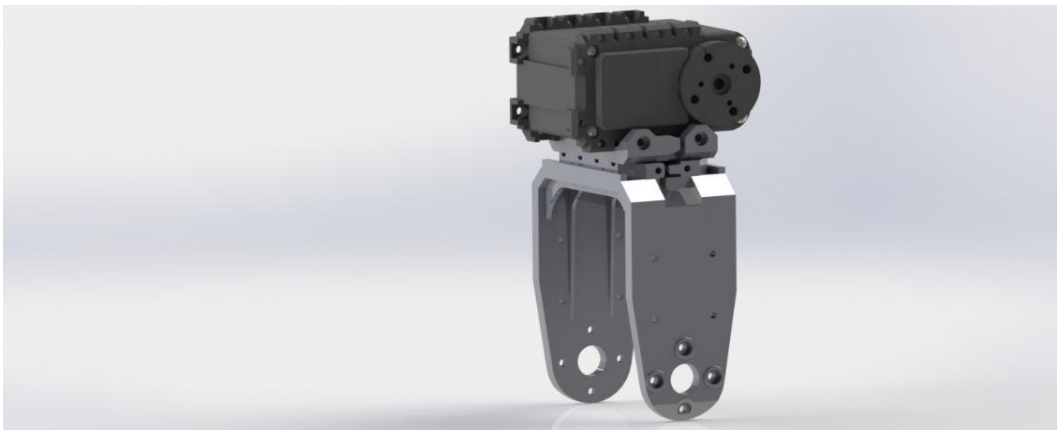


Fuente: Autor.

3.2.3.2. ESLABONES NÚMERO DOS Y NUEVE DEL ROBOT BIOLOID TYPE A.

Está relacionado con el segundo y noveno movimiento articular del robot al nivel matemático es el eslabón presente dentro de las matrices de transformación homogénea A2 y A9 (ver tabla 1 y tabla 2).

FIGURA 11: RENDERIZADO DEL SEGUNDO ESLABON DEL ROBOT.

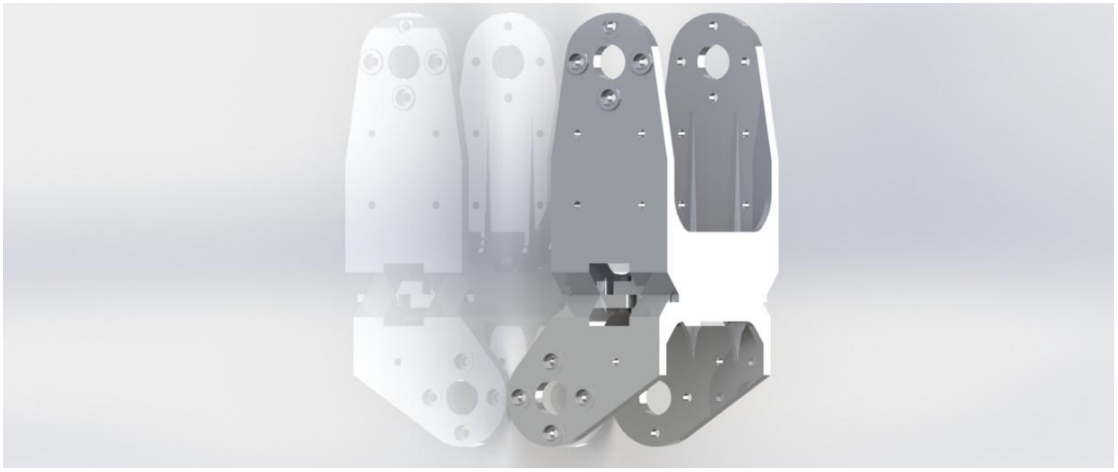


Fuente: Autor.

3.2.3.3. ESLABONES NÚMERO TRES Y DIEZ DEL ROBOT BIOLOID TYPE A.

Está relacionado con el tercero y decimo movimiento articular del robot al nivel matemático es el eslabón presente dentro de las matrices de transformación homogénea A2 y A10 (ver tabla 1 y tabla 2).

FIGURA 12: RENDERIZADO DEL TERCERO Y OCTAVO ESLABON DEL ROBOT.



Fuente: Autor.

3.2.3.4. ESLABONES NÚMERO CINCO DEL ROBOT BIOLOID TYPE A.

Está relacionado con el quinto movimiento articular del robot al nivel matemático es el eslabón presente dentro de las matrices de transformación homogénea A5 (ver tabla 1).

En primera instancia el eslabón incluía los brazos del robot como una sola estructura debido a que se le otorgo prioridad al análisis de las piernas.
(Ver figura 13).

FIGURA 13: RENDERIZADO DEL QUINTO ESLABON EN SUS PRIMERAS ETAPAS DEL ROBOT.

Fuente: Autor.

Luego de conseguir una respuesta del movimiento se decidió omitir del eslabón los brazos ya que estos se convirtieron en nuevos eslabones, que operan de forma independiente del resto de la estructura del cuerpo (ver figuras14 y 15).

FIGURA 14: RENDERIZADO DEL QUINTO ESLABON FINAL DEL ROBOT.

Fuente: Autor.

3.2.3.5. ESLABONES NÚMERO SEIS Y DOCE DEL ROBOT BIOLOID TYPE A.

Está relacionado con el sexto y doceavo movimiento articular del robot al nivel matemático es el eslabón presente dentro de las matrices de transformación homogénea A6 y A12 (ver tabla 1 y tabla 2).

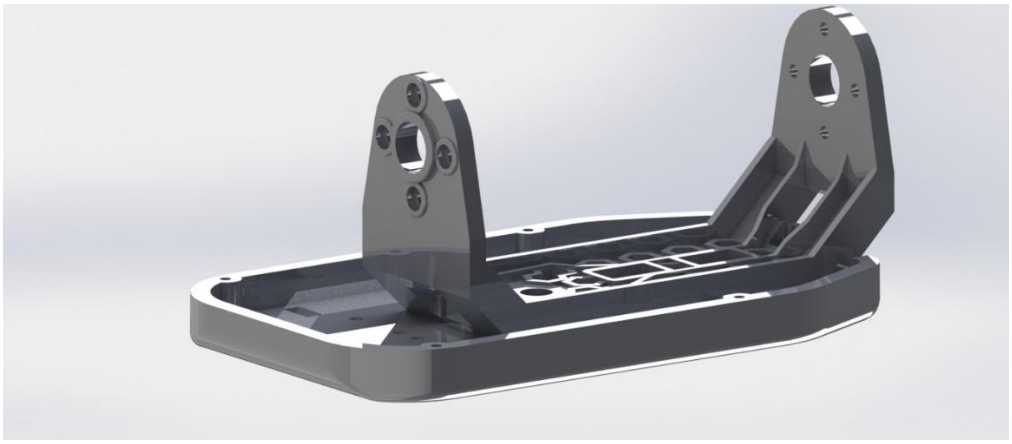
Este eslabón se compone por una pieza única sirviendo solo para movimientos rotacionales (ver figura anexos).

3.2.3.6. ESLABONES NÚMERO SIETE DEL ROBOT BIOLOID TYPE A.

Está relacionado con el séptimo movimiento articular del robot al nivel matemático es el eslabón presente dentro de las matrices de transformación homogénea A7 (ver tabla 2).

Este eslabón tiene la característica de que representa tanto la base del robot como el efector final.

FIGURA 15: RENDERIZADO DEL SEPTIMO ESLABON DEL ROBOT.



Fuente: Autor.

3.2.3.7. ESLABONES NÚMERO TRECE Y DIECISÉIS DEL ROBOT BIOLOID TYPE A.

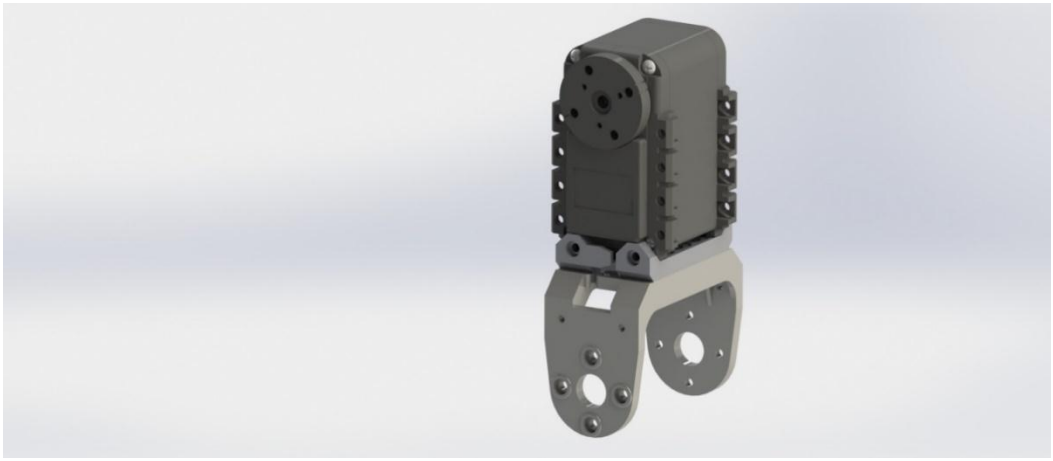
Está relacionado con el treceavo y dieciseisavo movimiento articular del robot al nivel matemático es el eslabón presente dentro de las matrices de transformación homogénea A13 y A16 (ver tabla 3 y tabla 4).

Este eslabón se compone por una pieza única donde sirviendo solo para movimientos rotacionales (ver figura anexos).

3.2.3.8. ESLABONES NÚMERO CATORCE Y DIECISIETE DEL ROBOT BIOLOID TYPE A.

Está relacionado con el catorceavo y diecisieteavo movimiento articular del robot al nivel matemático es el eslabón presente dentro de las matrices de transformación homogénea A14 y A17 (ver tabla 3 y tabla 4).

FIGURA 16: RENDERIZADO DEL CATORCEAVO ESLABON DEL ROBOT.



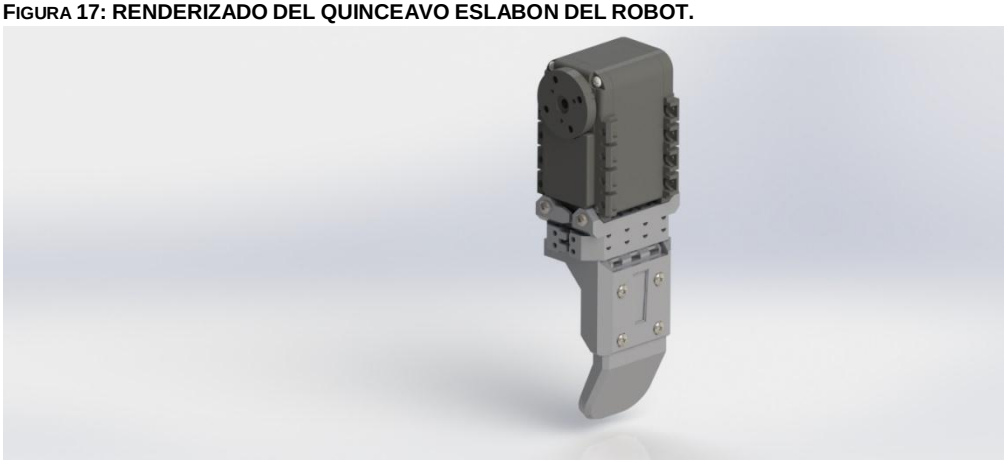
Fuente: Autor.

3.2.3.9. ESLABONES NÚMERO QUINCE Y DIECIOCHO DEL ROBOT BIOLOID TYPE A.

Está relacionado con el quinceavo y dieciochoavo movimiento articular del robot al nivel matemático es el eslabón presente dentro de las matrices de transformación homogénea A15 y A18 (ver tabla 3 y tabla 4).

Este eslabón representa el efector final de los brazos del robot.

FIGURA 17: RENDERIZADO DEL QUINCEAVO ESLABON DEL ROBOT.



Fuente: Autor.

3.2.4. **PROPIEDADES FISICAS BÁSICAS DEL ROBOT BIOLOID TYPE A.**

Por medio del modelado digital de los eslabones se pueden obtener de una manera más sencilla algunas propiedades físicas del eslabón, como por ejemplo la masa, la inercia, el volumen etc..., debido a su facilidad el software solidworks ® arrojo con exactitud varios parámetros físicos, dependiendo las características del eslabón.

Puesto que cada eslabón está asociado a una matriz de trasformación homogénea se dispuso a generar sistemas de coordenadas auxiliares en el modelo en la misma localización que su matriz asociada.

De los parámetros físicos obtenidos se necesitan solo la masa del eslabón y su centro de masa referenciado al sistema de coordenadas de la matriz asociada, ya que estos valores sirven más adelante para saber dónde se encuentra el centro de masas total del robot y así tener claridad si la proyección de este se encuentra dentro del área del pie de apoyo del robot.

Se decidió organizar una serie de tablas recopilatorias de cada eslabón referenciando su masa en gramos, sus coordenadas en milímetros del centro de masas y la matriz de transformación homogénea asociada.

TABLA 5: TABLA RECOPILATORIA DE PROPIEDADES ESLABONES DEL ROBOT BIOLOID TYPE A.

Eslabones uno, cuatro, ocho y once del robot bioloid type A				
Masa (g)	***Centro de masa coordenada X (mm)***	***Centro de masa coordenada Y (mm)***	***Centro de masa coordenada Z (mm)***	***Matrices de transformación homogénea***
181.80	*-18.40*	*7.94*	*4.21*	$A_{01}, A_{04}, A_{08\ y}\ A_{011}$
Eslabón dos del robot bioloid type A				
Masa (g)	***Centro de masa coordenada X (mm)***	***Centro de masa coordenada Y (mm)***	***Centro de masa coordenada Z (mm)***	***Matriz de transformación homogénea***
103.64	*-9.03*	*6.87*	*63.90*	A_{02}
Eslabón tres del robot bioloid type A				
Masa (g)	***Centro de masa coordenada X (mm)***	***Centro de masa coordenada Y (mm)***	***Centro de masa coordenada Z (mm)***	***Matriz de transformación homogénea***
21.04	*13.48*	*0.03*	*31.64*	A_{03}
Eslabón cinco del robot bioloid type A				
Masa (g)	***Centro de masa coordenada X (mm)***	***Centro de masa coordenada Y (mm)***	***Centro de masa coordenada Z (mm)***	***Matriz de transformación homogénea***
1065.99	*1.64*	*40.59*	*79.52*	A_{05}
Eslabón cinco del robot bioloid type A				

Masa (g)	*Centro de masa coordenada X (mm)*	*Centro de masa coordenada Y (mm)*	*Centro de masa coordenada Z (mm)*	*Matriz de transformación homogénea*
654.42	*12.14*	*37.72*	*100.66*	A_{05}
Eslabón seis y doce del robot bioloid type A				
Masa (g)	*Centro de masa coordenada X (mm)*	*Centro de masa coordenada Y (mm)*	*Centro de masa coordenada Z (mm)*	*Matriz de transformación homogénea*
10.82	*-17.31*	*0*	*18.94*	$A_{06\ y}A_{012}$
{Eslabón siete del robot bioloid type A				
Masa (g)	*Centro de masa coordenada X (mm)*	*Centro de masa coordenada Y (mm)*	*Centro de masa coordenada Z (mm)*	*Matriz de transformación homogénea*
30.89	*-4.7*	*0*	*-25.83*	A_{07}
Eslabón nueve del robot bioloid type A				
Masa (g)	*Centro de masa coordenada X (mm)*	*Centro de masa coordenada Y (mm)*	*Centro de masa coordenada Z (mm)*	*Matriz de transformación homogénea*
103.64	*5.47*	*9.67*	*-11.10*	A_{09}
Eslabón diez del robot bioloid type A				
Masa (g)	*Centro de masa coordenada X (mm)*	*Centro de masa coordenada Y (mm)*	*Centro de masa coordenada Z (mm)*	*Matriz de transformación homogénea*
21.04	*-1.02*	*0.03*	*-43.36*	A_{010}
Eslabón trece y dieciséis del robot bioloid type A				
Masa (g)	*Centro de masa coordenada X (mm)*	*Centro de masa coordenada Y (mm)*	*Centro de masa coordenada Z (mm)*	*Matriz de transformación homogénea*
7.93	*0.08*	*-10.73*	*-3.73*	A_{013} y A_{016}
Eslabón catorce y diecisiete del robot bioloid type A				
Masa (g)	*Centro de masa coordenada X (mm)*	*Centro de masa coordenada Y (mm)*	*Centro de masa coordenada Z (mm)*	*Matriz de transformación homogénea*
95.97	*9.92*	*5.13*	*-8.50*	A_{014} y A_{017}
Eslabón quince y dieciocho del robot bioloid type A				
Masa (g)	*Centro de masa coordenada X (mm)*	*Centro de masa coordenada Y (mm)*	*Centro de masa coordenada Z (mm)*	*Matriz de transformación homogénea*
101.88	*9.34*	*5.44*	*-12.13*	A_{015} y A_{018}

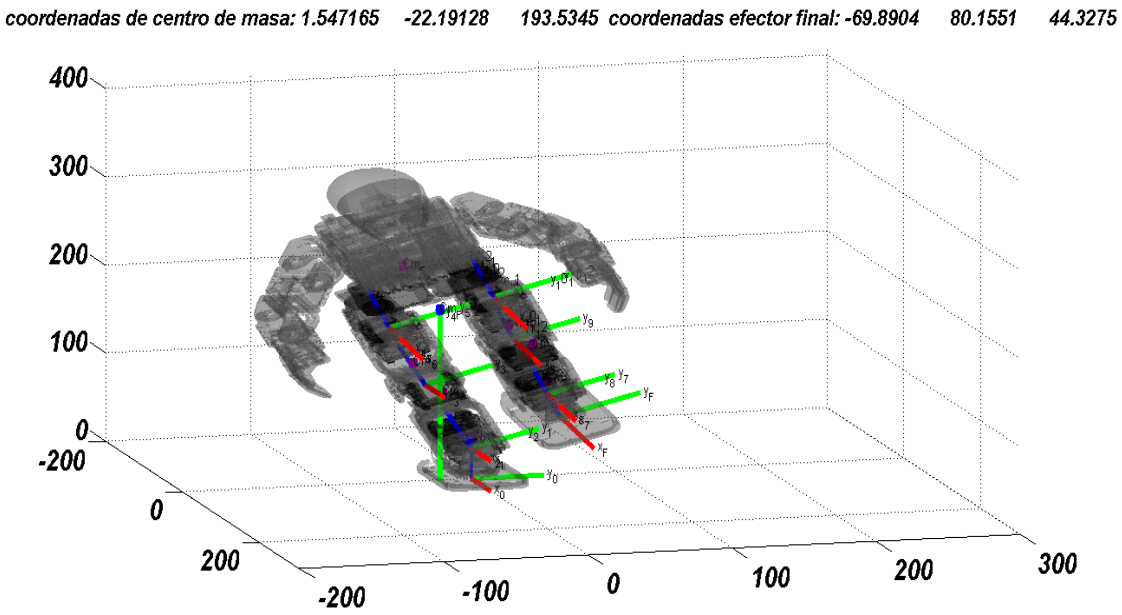
3.3. SIMULADOR DE TRAYECTORIAS.

Para poder visualizar con mayor precisión los movimientos de una trayectoria se deben digitalizar las piezas y esto conlleva a crear un simulador con las mismas características del robot real.

El objetivo principal de la creación del simulador es poder visualizar, recrear y probar los ángulos obtenidos por los algoritmos computacionales sin golpear o estropear el robot, además permite corroborar el funcionamiento de una forma más rápida que en la vida real. Su desarrollo se hizo con base al análisis cinemático realizado anteriormente [3].

Las piezas del diseño CAD del robot se transformaron en datos matriciales utilizando el formato estéreo-litografía (basado en la triangulación de superficies), con el fin de manipular su posición y orientación al acoplar los conceptos matemáticos de las matrices de transformación homogéneas.

FIGURA 18: SIMULADOR DE TRAYECTORIAS.



Fuente: Autor.

3.3.1. **TIPOS DE SIMULADOR DE TRAYECTORIAS.**

El robot bioloid es un robot bastante robusto compuesto por muchas piezas acopladas entre sí, por esta razón si se realizase el simulador en primera instancia con la totalidad de las piezas fuse bastante complejo desarrollarlo completamente y al digitalizar las piezas completas se aumenta el tiempo de cómputo necesario para realizar el simulador.

Es por esta razón que se decidió realizar dos simuladores:

Uno donde el robot bioloid se representa por un modelo digital en alambres y el otro es el modelo digital con la totalidad de las piezas tridimensional, el modelo de alambres por su desarrollo computacional más veloz fue el que se decidió realizar primero sirviendo como base para finalmente realizarlo con la totalidad de las piezas.

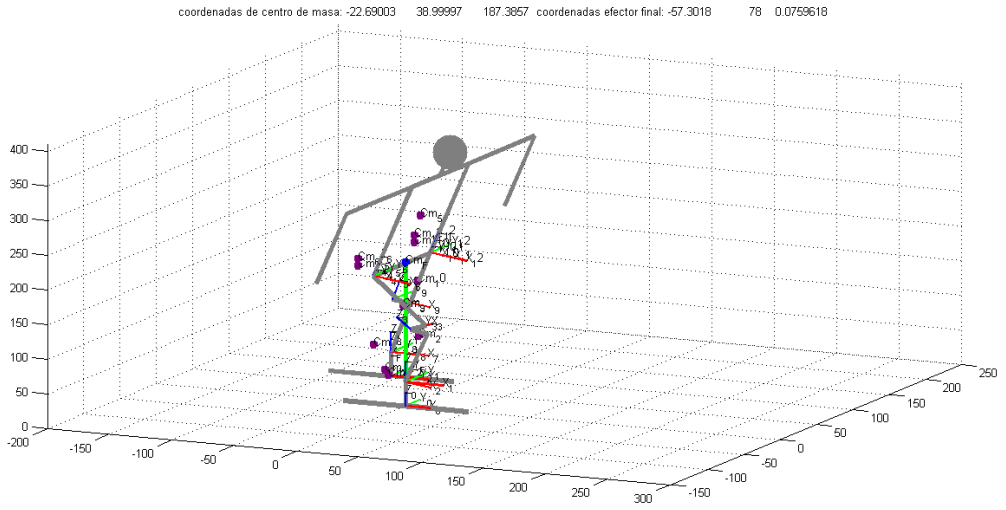
3.3.2. **SIMULADOR DE TRAYECTORIAS MODELO EN ALAMBRES.**

Por su velocidad computacional se optó por realizarse primero este simulador posee todas las características necesarias para visualizar los movimientos cuando se introducen los valores angulares.

Este simulador muestra en forma de alambres todo el robot completo y sus ejes coordenados correspondientes a cada matriz de transformación homogénea. El desarrollo del simulador se realizó en el lenguaje de programación Matlab ®.

El simulador contiene todos los centros de masa de cada eslabón, el centro de masas total del robot y a su vez la proyección del centro de masas total del robot en el pie de apoyo, posteriormente se muestra los tipos de proyecciones que existen presentes en el robot (ver páginas 46 al 47).

FIGURA 19: SIMULADOR DE TRAYECTORIAS EN ALAMBRES.



Fuente: Autor.

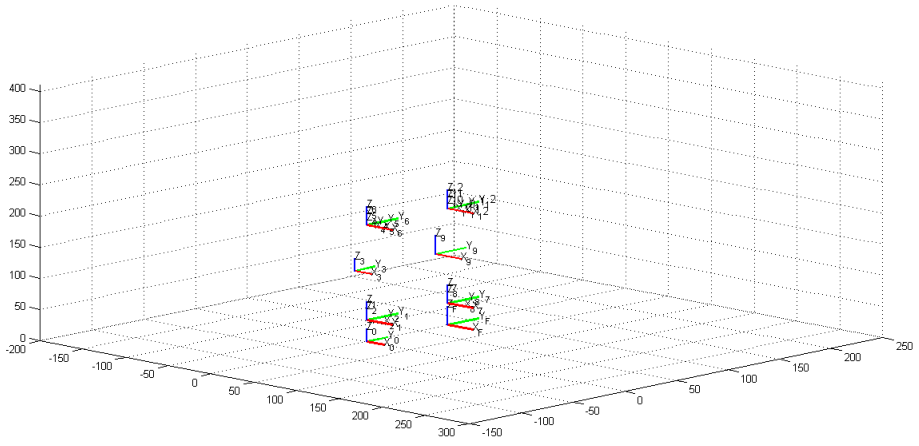
3.3.3. **DESARROLLO DEL SIMULADOR DE TRAYECTORIAS MODELO EN ALAMBRES.**

Los ejes coordenados de cada matriz de transformación homogénea con respecto al origen es la parte cinematicamente más relevante ya que estos son los que determinan la localización final de cada eslabón del robot.

Tomando en consideración lo anterior propuesto se optó por graficar primero los ejes coordenados (ver figura 20).

Por ser más relevante a la hora de realizar una trayectoria las piernas se decidió analizarlo primero, siendo así que la primera parte desarrollada del simulador fuese los ejes coordenados de las piernas del robot.

FIGURA 20: SIMULADOR DE TRAYECTORIAS EN ALAMBRES PRIMERA FASE.

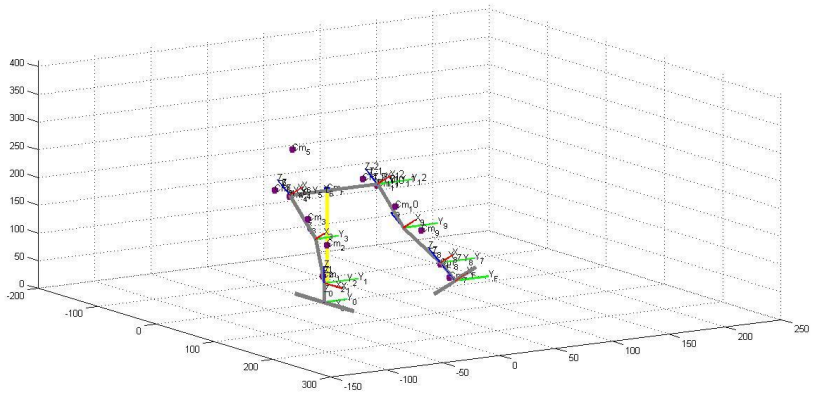


Fuente: Autor.

Se realizó pruebas para validar su orientación al ser satisfactoria su respuesta se decidió graficar los centros de masa de cada eslabón y el centro de masas final del robot.

Como se poseía las coordenadas de los centros de masa de cada eslabón referidos a las matrices de transformación homogénea asociada, se ubicaron partiendo del origen de cada matriz. Se calculó el centro de masas final (ver página 25), luego se anidaron todos los ejes coordenados en forma de los eslabones del robot, se realizó el diseño el alambres de la base del robot y el efector final y finalmente se graficó la proyección del centro de masas final con respecto al pie de apoyo del robot. (Ver figura 21).

FIGURA 21: SIMULADOR DE TRAYECTORIAS EN ALAMBRES SEGUNDA FASE.

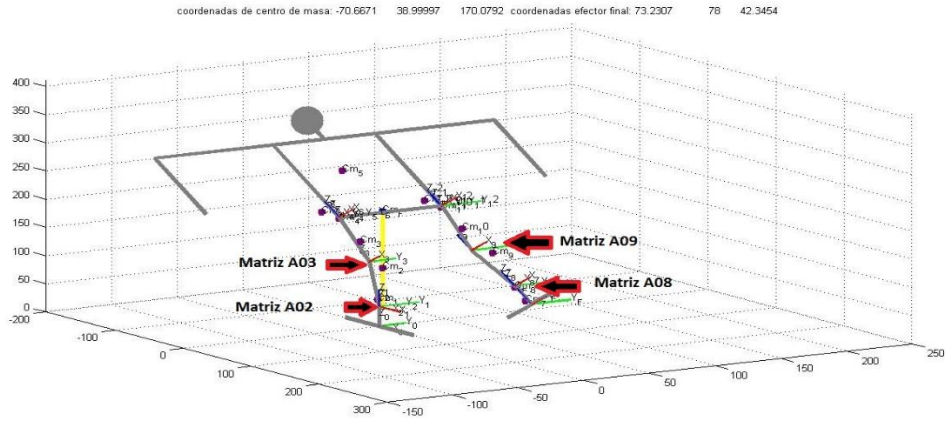


Fuente: Autor.

Por ser una primera aproximación al diseño de alambres del robot se contempló solo las piernas del robot, pero se necesita visualizar el resto del cuerpo ya que visualizando completamente el robot se puede observar cómo sería un movimiento implementado al modelo físico.

Debido a esto se realizó el modelo con el resto del cuerpo como una única estructura excluyendo los movimientos de los brazos del robot bioloid (ver página 30). Se graficó en el simulador en la parte superior las coordenadas del centro de masas final del robot a su vez que las coordenadas del pie móvil del robot (efector final).

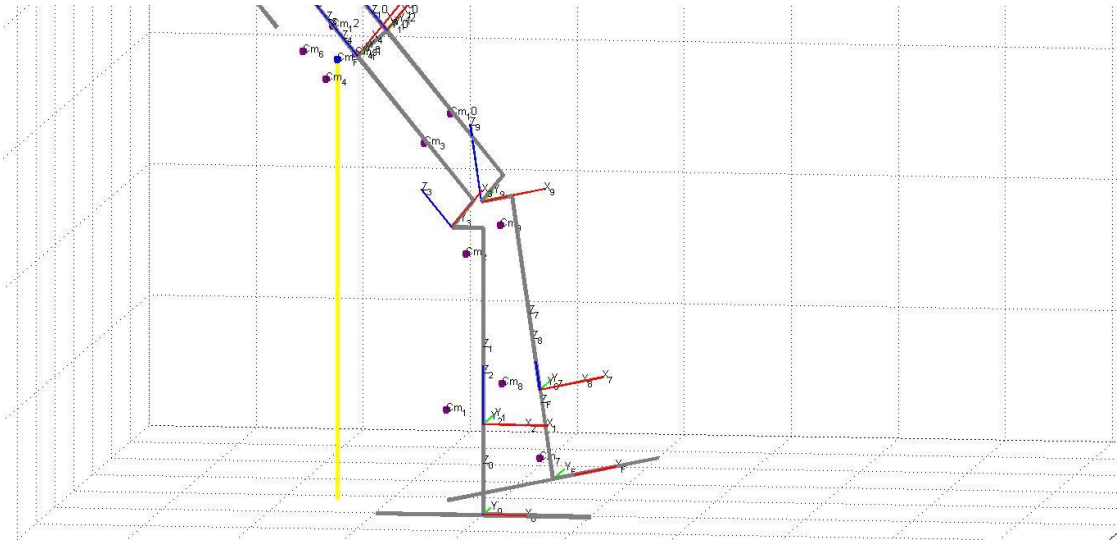
FIGURA 22: SIMULADOR DE TRAYECTORIAS EN ALAMBRES TERCERA FASE.



Fuente: Autor.

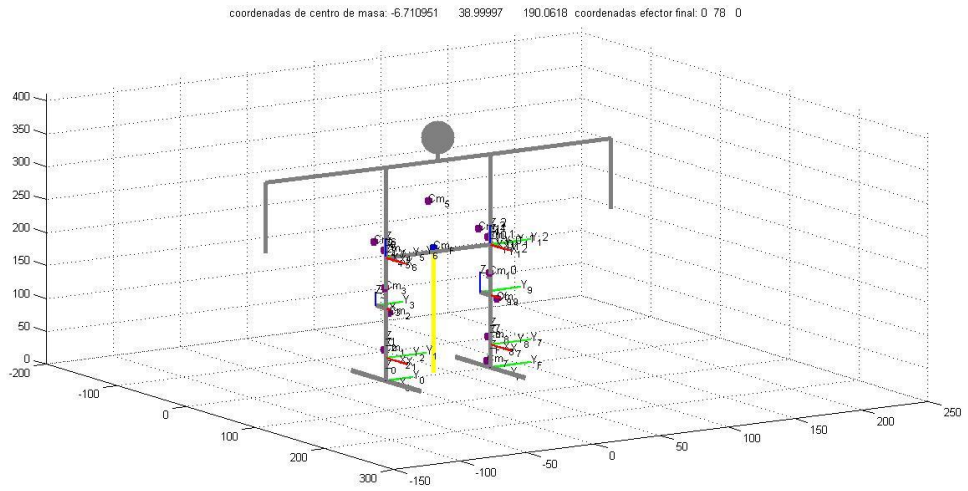
El eslabón que unen los ejes coordenados de las matriz de transformación homogénea A_{02} y A_{03} no corresponden a cómo deberían ser en realidad de igual forma el eslabón que une los ejes coordenados de la matriz de transformación homogénea A_{08} y A_{09} presenta esta anomalía también (ver figura 22), es por esta razón que se realizó la corrección de estos eslabones, generando un modelo completo del robot bioloid donde se analicen los movimientos de las piernas al generar una marcha como afecta al resto del cuerpo (ver figura 23).

FIGURA 23: SIMULADOR DE TRAYECTORIAS EN ALAMBRES CORRECCIÓN DE ESLABONES.



Fuente: Autor.

FIGURA 24: SIMULADOR DE TRAYECTORIAS EN ALAMBRES CUARTA FASE.

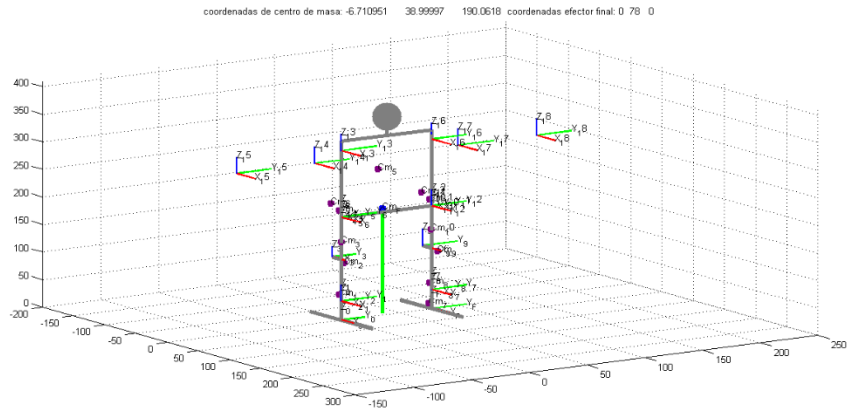


Fuente: Autor.

Luego de haber realizado las pruebas con los algoritmos generadores de trayectorias (ver páginas 48 a la 66) y realizar las pruebas experimentales en el modelo físico (ver páginas 61 a la 68) se decidió implementar el análisis a los brazos del robot, al igual que las piernas del robot se analizaron por el método de los Screws [3].

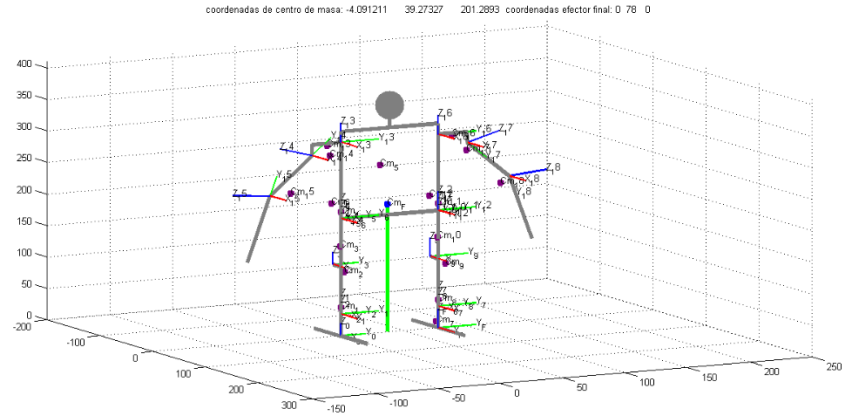
De forma análoga a como se realizó con las piernas del robot se debe graficar primero los ejes coordenados de todos los brazos (ver figura 25), validar su orientación y finalmente graficar todos los eslabones que componen los brazos del robot (ver figura 26), realizando así el modelo en alambres final del robot bioloid type A, concluyendo así todo el modelo en alambres del robot bioloid.

FIGURA 25: SIMULADOR DE TRAYECTORIAS EN ALAMBRES QUINTA FASE.



Fuente: Autor.

FIGURA 26: SIMULADOR DE TRAYECTORIAS EN ALAMBRES FASE FINAL.

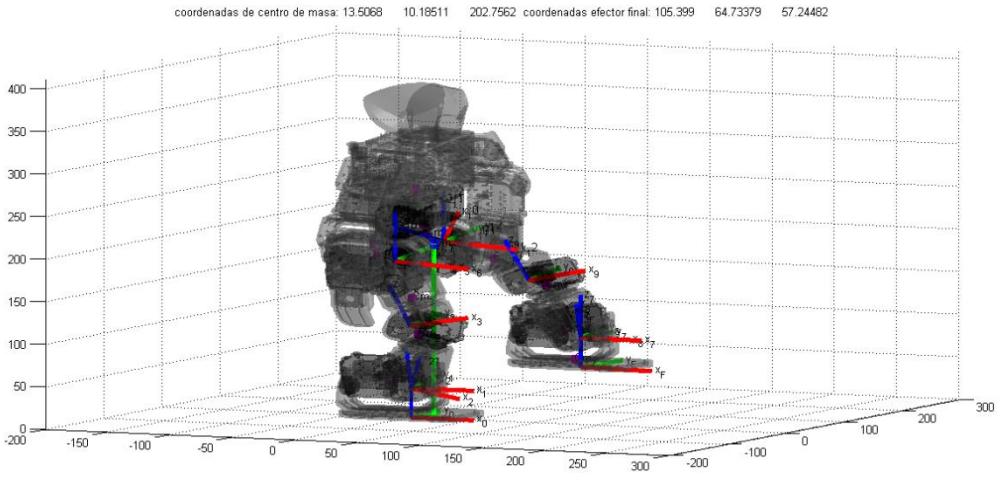


Fuente: Autor.

3.3.4. **SIMULADOR DE TRAYECTORIAS TRIDIMENSIONAL.**

Al realizar el simulador de trayectorias del robot bioloid en alambres da claridad para poder realizar el simulador contemplando todas las piezas, este simulador utiliza las piezas diseñadas anteriormente (ver página 26 a la 34), este simulador ilustra de manera tridimensional todo el robot bioloid type A. Incluye todos los centros de masa de cada eslabón, el centro de masas total del robot y a su vez la proyección del centro de masas total del robot en el pie de apoyo, posteriormente se muestra los tipos de proyecciones que existen presentes en el robot (ver paginas 46 al 47).

FIGURA 27: SIMULADOR DE TRAYECTORIAS TRIDIMENSIONAL.



Fuente: Autor.

En este modelo finalmente se pueden apreciar como seria si se aplicase ciertos valores angulares al modelo real.

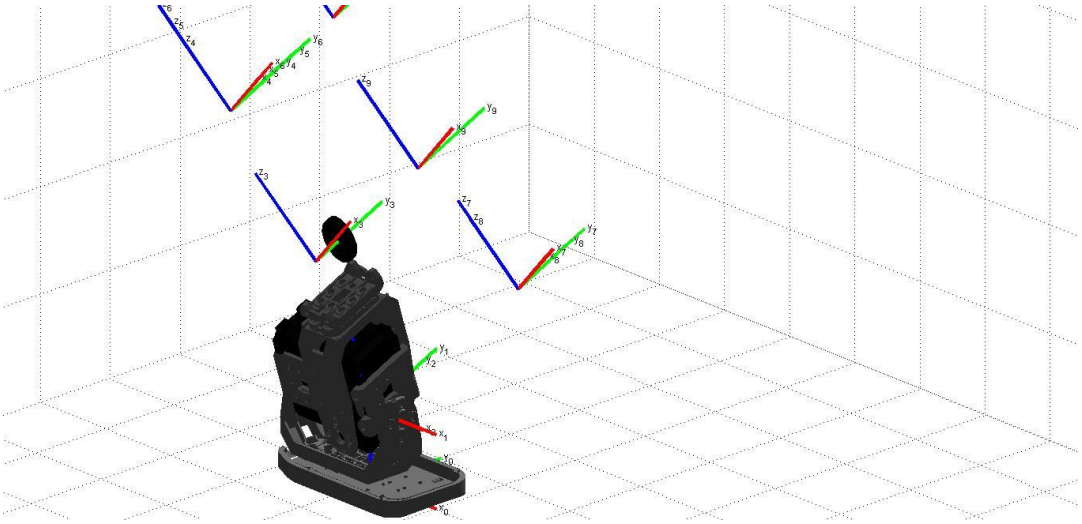
3.3.5. **DESARROLLO DEL SIMULADOR DE TRAYECTORIAS TRIDIMENSIONAL.**

Cada pieza diseñada se convierte al formato .STL (estéreo litografía) ya en este formato se ingresa a un algoritmo desarrollado en el lenguaje de programación Matlab ® que lo convierte en datos matriciales que representan todas las propiedades (textura, vértices, caras, etc...) que posee cada pieza (ver apéndice 70).

Posteriormente se realiza una serie de gráficas tridimensionales que representan secciones de las piezas que cuando se unen completamente se recrea en totalidad la pieza.
En seguida se realiza un ensamble de las piezas para generar el modelo tridimensional del robot bioloid type A.

Para desarrollar el ensamble se debe graficar primero al igual que en el modelo en alambres los ejes coordenados (ver páginas 37 y ver grafica 15), luego de realizar el ensamble se debe acoplar cada pieza que este asociada al eje coordenado correspondiente (ver figura 28).

FIGURA 28: SIMULADOR DE TRAYECTORIAS TRIDIMENSIONAL PRIMERA FASE.

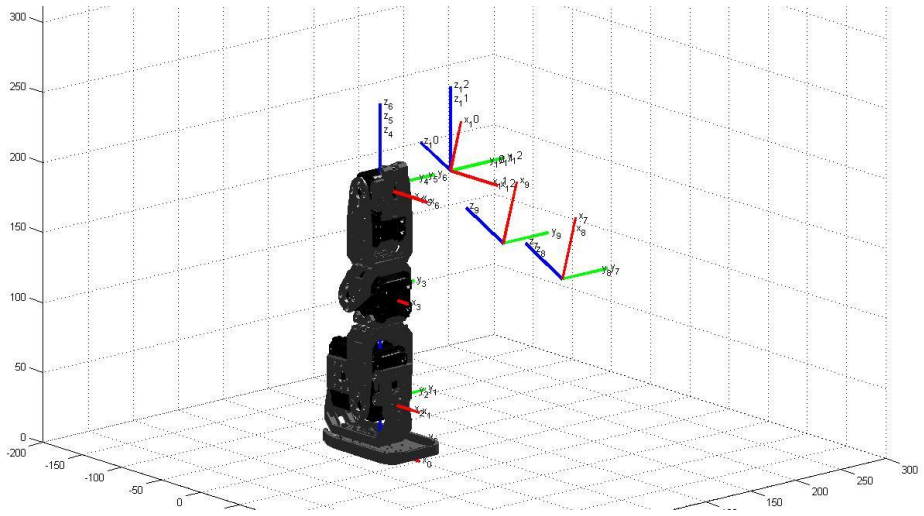


Fuente: Autor.

Para poder corroborar si los movimientos del simulador coincidiera con el modelo real se decidió graficar una pierna con las piezas ensambladas y la otra solo con los ejes coordenados (ver figura 28).

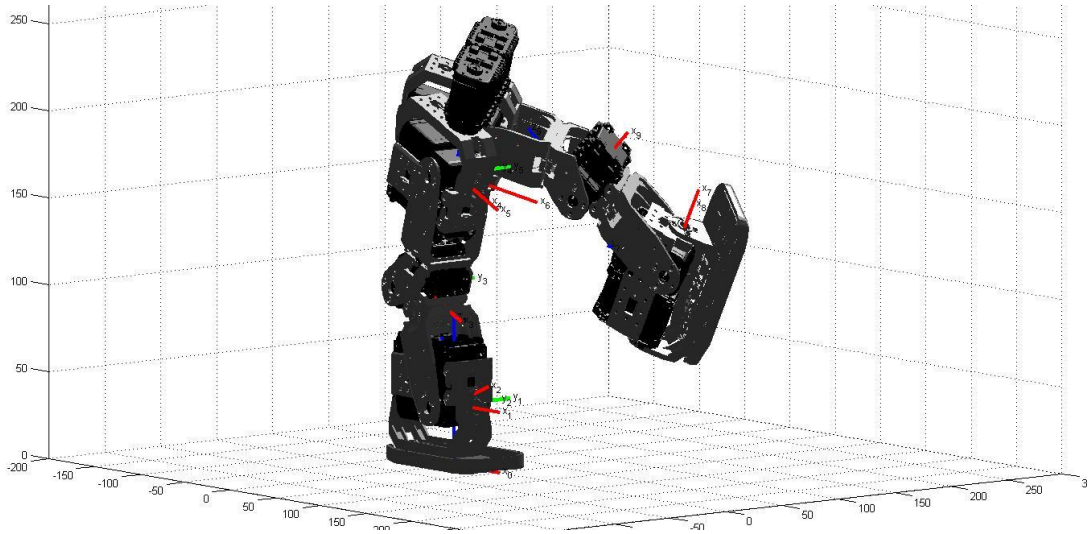
Las piernas del robot bioloid es la parte más importante si se desea realizar una marcha por ende se debe graficar antes de seguir con el resto del cuerpo (ver figura 29), cuando se graficó completamente las piernas del robot se realizaron pruebas para analizar la localización tanto de los eslabones como de la planta móvil del pie (efector final).

FIGURA 29: SIMULADOR DE TRAYECTORIAS TRIDIMENSIONAL SEGUNDA FASE.



Fuente: Autor.

FIGURA 30: SIMULADOR DE TRAYECTORIAS TRIDIMENSIONAL TERCERA FASE.

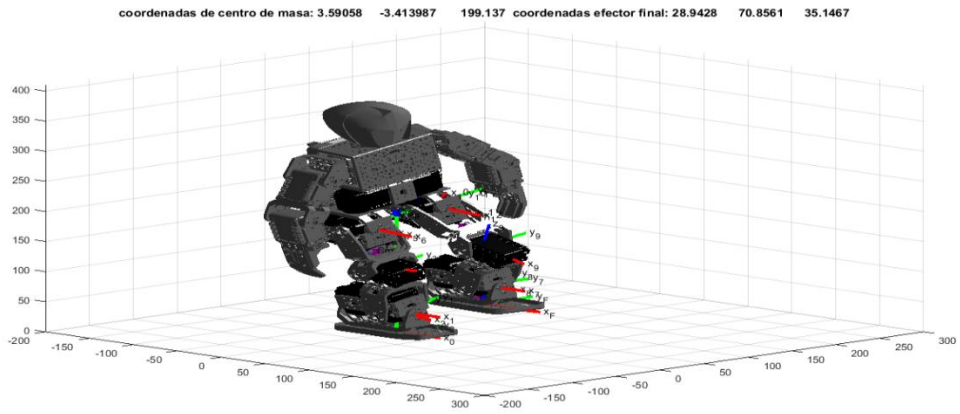


Fuente: Autor.

Posteriormente se inició con el desarrollo en el simulador de la estructura del robot completa. Los movimientos articulares de los brazos no se realizaron y se organizaron en una posición donde permita una marcha sin problemas de colisiones por su propia estructura, a su vez se localizó en una posición balanceada y así no desequilibrar el robot (ver figura 30).

De forma paralela se implementó la visualización de las coordenadas de los centros de masas y el efector final (ver figura 31). Como la visualización de la proyección de su centro de masa en el área del pie (ver páginas 46 al 47)

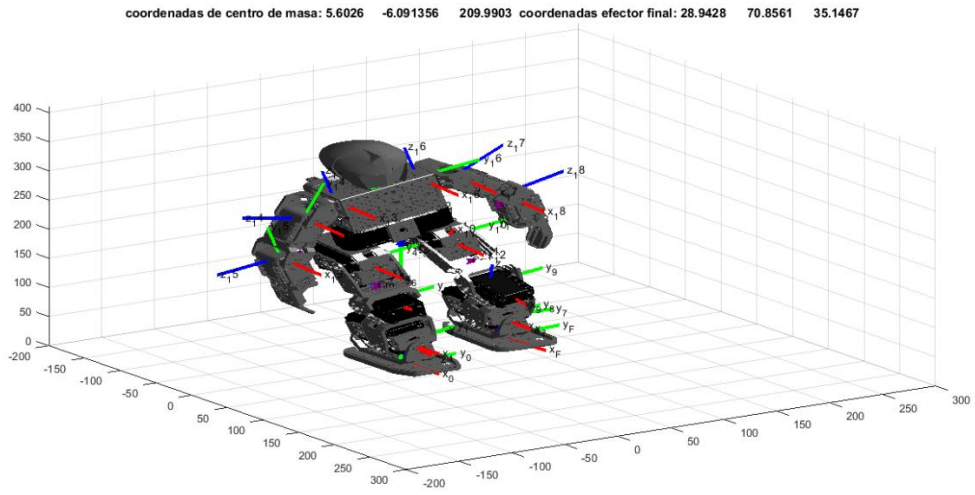
FIGURA 31: SIMULADOR DE TRAYECTORIAS TRIDIMENSIONAL CUARTA FASE.



Fuente: Autor.

Finalmente se recreó el modelo digital tridimensional final del robot bioloid type A con todos sus movimientos articulares implementados y a su vez cada eslabón acoplalado a su matriz de transformación homogénea correspondiente (ver página 25 y 26), para su implementación final con los algoritmos generadores de trayectorias (ver páginas 48 a la 56)

FIGURA 32: SIMULADOR DE TRAYECTORIAS TRIDIMENSIONAL FINAL.



Fuente: Autor.

3.3.6. **MOVIMIENTOS PRESENTES EN EL SIMULADOR DE TRAYECTORIAS.**

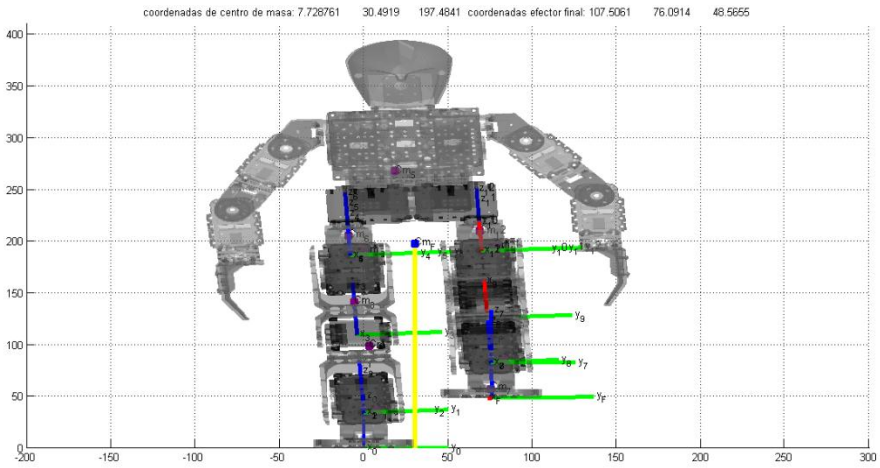
Para poder determinar qué tipos de movimientos cuando se esté realizando una trayectoria son los permitidos o no se debe tener un indicador, en este caso el indicador se presenta si las componentes verticales y horizontales (X y Y) se encuentran dentro del área del pie de apoyo del robot.

Se pudo determinar tres tipos de movimientos: movimiento fuera del área del pie de apoyo, movimiento con colisión mecánica y movimiento valido, a continuación se explicara en qué consisten estos movimientos.

3.3.7. **MOVIMIENTO FUERA DEL AREA DEL PIE DE APOYO.**

Cuando se realiza un movimiento que puede ocasionar una caída al robot este se representa como un movimiento que esté fuera del área del pie de apoyo del robot.
Estos movimientos son representados dentro del simulador como una proyección desde el centro de masas hasta el piso de color amarillo (ver figura 33) y no son tomados en cuenta cuando se quiere operar el robot.

FIGURA 33: MOVIMIENTO FUERA DEL AREA DEL PIE DE APOYO.

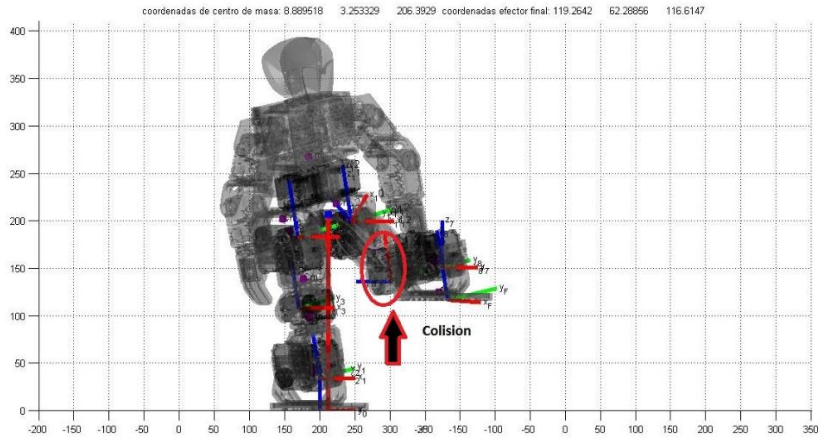


Fuente: Autor.

3.3.8. **MOVIMIENTO CON COLISION MECANICA.**

Estos movimientos se presentan cuando se le ingresa al simulador movimientos angulares que puedan generar rupturas en la estructura del robot.
Estos movimientos deben ser evitados a toda costa y son los movimientos más peligrosos que se puedan presentar.
Se representan dentro del simulador con una proyección desde el centro de masas hasta el piso de color rojo (ver figura 34).

FIGURA 34: MOVIMIENTO CON COLISION MECANICA.



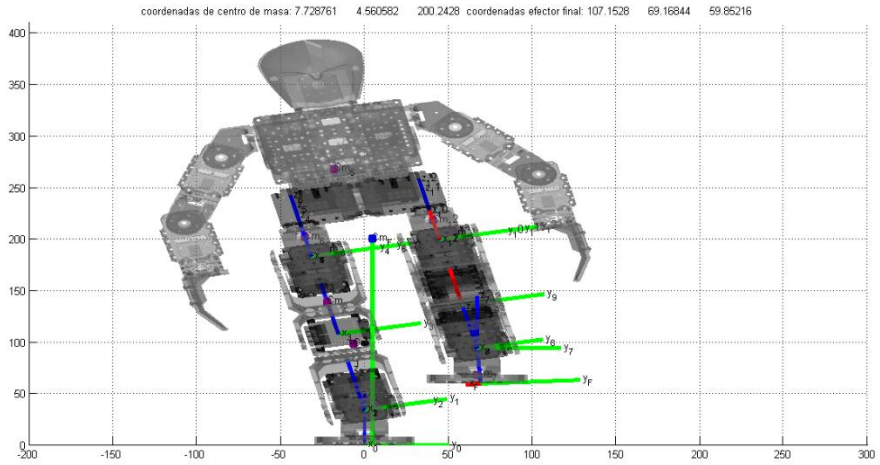
Fuente: Autor.

3.3.9. **MOVIMIENTO VALIDO.**

Son los movimientos cuando la proyección del centro de masas se encuentra en el área del pie de apoyo del robot (ver figura 35), estos movimientos son los que se necesitan para generar un desplazamiento y finalmente una marcha para el robot.

Se representan dentro del simulador como una proyección en verde desde el centro de masas hasta el piso.

FIGURA 35: MOVIMIENTO VALIDO.



Fuente: Autor.

3.4. GENERADOR DE TRAYECTORIAS AUTONOMAS.

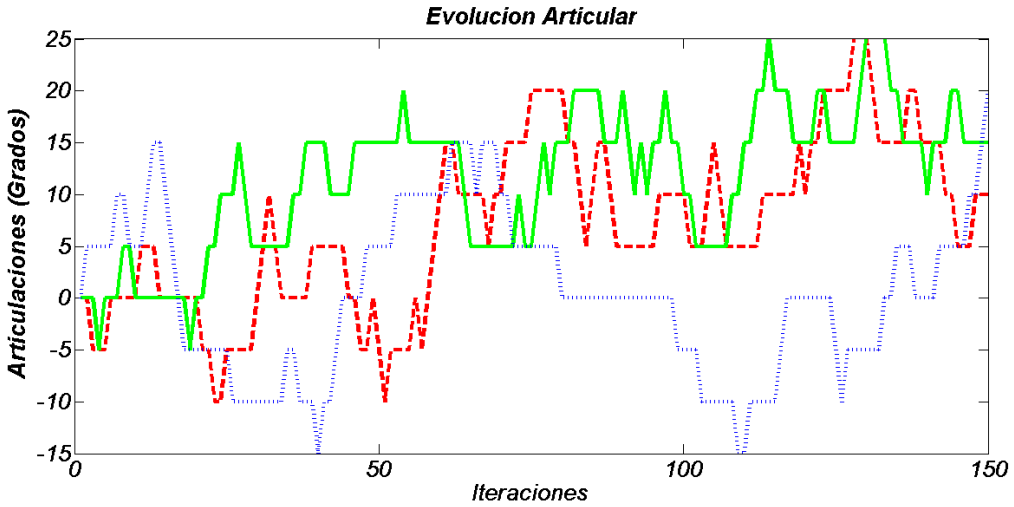
Para que el robot se desplace debe existir algún método autónomo donde por medio de algoritmos los datos iteren entre sí para generar la evolución angular que deben tener los servomotores para poder realizar ese desplazamiento sin que se produzca ninguna colisión mecánica o se produzca una caída en el robot.

La realización de estos métodos puede ser por diferentes alternativas tanto heurísticas, estocásticas como técnicas avanzadas de inteligencia artificial. Puntualmente en esta investigación se desarrollaron unos algoritmos de evolución articular de forma estocástica los algoritmos se realizaron en el lenguaje de programación Matlab ® para realizar su implementación con el simulador y el modelo real.

3.4.1. PRIMERAS FASES DEL ALGORITMO GENERADOR DE TRAYECTORIAS AUTONOMAS

Se desarrollaron unos algoritmos de evolución articular de forma aleatoria que por medio de iteraciones (cada articulación tiene 3 alternativas: giro en un sentido a un ángulo fijo, giro en sentido contrario o permanecer inmóvil durante la iteración) se generó una trayectoria desde un punto inicial a uno final (ver figura 36).

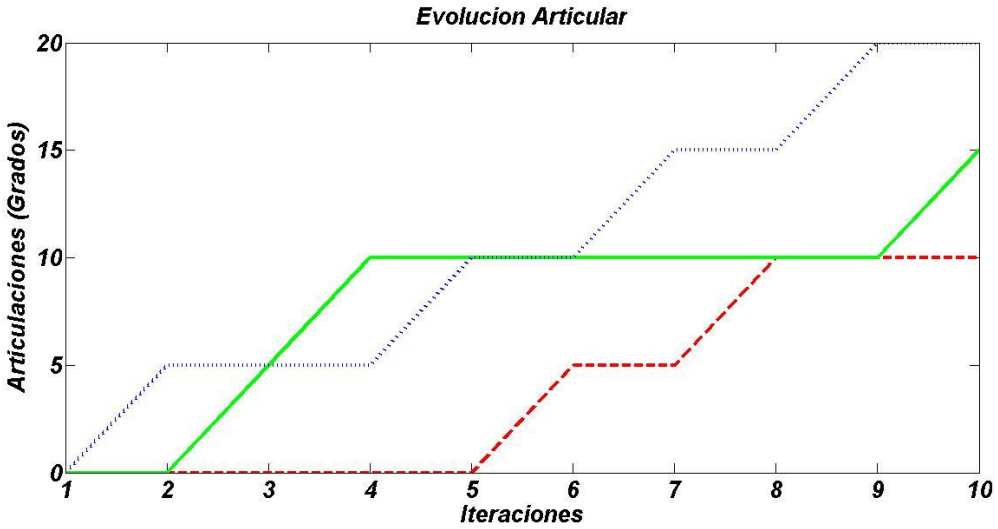
FIGURA 36: TRAYECTORIAS ARTICULARES PRIMERA FASE.



Fuente: Autor.

Por su naturaleza aleatoria el algoritmo en ocasiones varía su número de iteraciones obteniendo una solución diferente cada vez que se realizaba cada prueba.

FIGURA 37: TRAYECTORIAS ARTICULARES CORTA PRIMERA FASE.



Fuente: Autor.

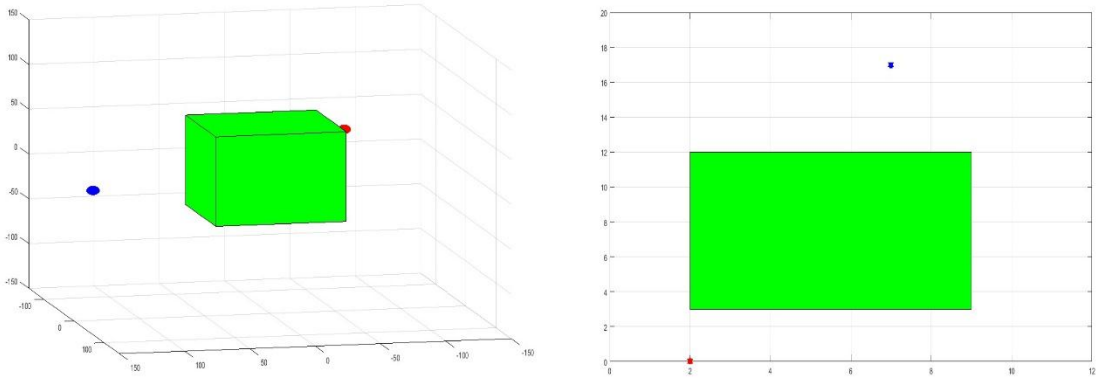
Luego de haber corroborado el fenómeno se decidió aumentar la cantidad de articulaciones (ver páginas 77 a la 83), por su elevado tiempo de ejecución de las trayectorias obtenidas, por no tener un orden definido y no contemplar las restricciones cinemáticas del robot. Se decidió realizar una optimización más eficiente al algoritmo.

3.4.2. **COLISICIONES EN N DIMENSIONES.**

Una trayectoria es el recorrido que se realiza desde un punto inicial a uno final, esta trayectoria puede tomar múltiples soluciones y en ocasiones se presentan dificultades que producen cambios en el rumbo que llevaba la trayectoria, estas dificultades son conocidas como obstáculos.

Si se avanza por el rumbo del obstáculo se genera una colisión y estas dificultan el avance da la trayectoria, en espacios bidimensionales son representadas como áreas donde no se puede avanzar y en espacios tridimensionales como volúmenes donde no se puede avanzar (ver figura 55). Estos obstáculos deben ser evitados y no tenerlos en cuenta en la trayectoria final.

FIGURA 38: REPRESENTACIÓN DE OBSTACULOS EN 3D Y 2D.



Fuente: Autor.

Debido a la forma que los seres humanos perciben la realidad solo se puede observar fenómenos menores o iguales a tres dimensiones, pero matemáticamente existen un número infinito de dimensiones, partiendo de este punto pueden existir colisiones en dimensiones mayores a las que percibimos.

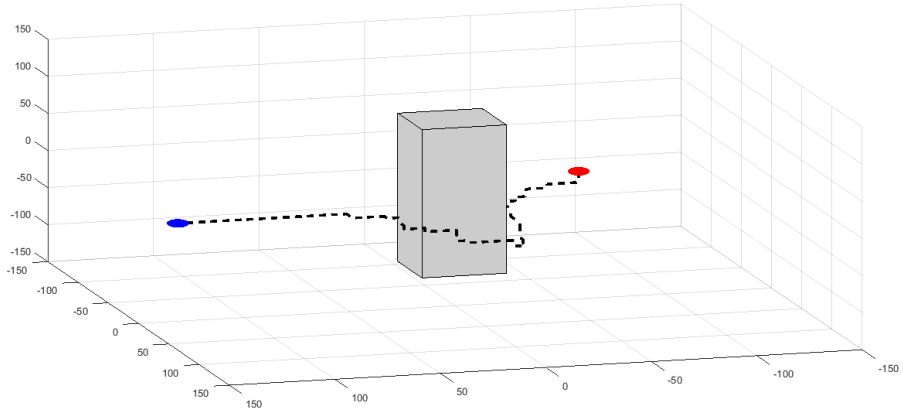
Se realizó una extrapolación donde cada articulación que presenta el robot bioloid type A es una dimensión, los obstáculos son representados como movimientos fuera del área del pie y movimientos donde existen colisión mecánica (ver paginas 46 al 47).

Operando así en un espacio de doce dimensiones cuando se está trabajando solo las piernas del robot.

3.4.2.1. COLISIONES DONDE EXISTE AVANCE.

Es el tipo de colisión donde se impide el avance por una o más dimensiones pero existen otras dimensiones por donde el avance es permitido, este tipo de colisión es bastante recurrente cuando se está realizando una trayectoria con obstáculos, al realizar la implementación en el algoritmo debe realizar una búsqueda de que dimensiones permiten el avance y así evitar el obstáculo (ver figura 39).

FIGURA 39: COLISIONES DONDE EXISTE AVANCE.

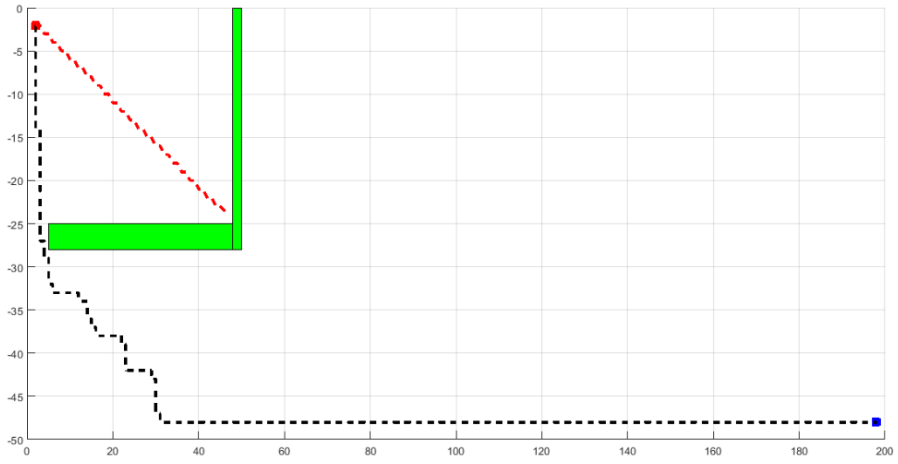


Fuente: Autor.

3.4.2.2. COLISIONES DONDE NO EXISTE AVANCE.

Es el tipo de colisión donde no se puede avanzar en ninguna dimensión por lo tanto el rumbo que se asumió no fue el indicado, se retorna hasta el punto de colisión más cercano a sus valores finales y se debe cambiar el rumbo de algunas dimensiones para poder esquivar el obstáculo (ver figura 40).

FIGURA 40: COLISIONES DONDE NO EXISTE AVANCE.



Fuente: Autor.

3.4.3. **SENTIDO DE AVANCE.**

Para poder generar un avance se debe contemplar que los movimientos que realice no sean redundantes ya que esto conlleva a más tiempo de ejecución y obtención de la trayectoria final, por lo tanto el avance debe ser lo más corto posible.

Matemáticamente la forma más corta para llegar a dos puntos es una línea recta, pero si se presenta un obstáculo por el rumbo de avance no se puede seguir a menos que se implemente un método donde evite el obstáculo (ver página 62).

Cuando se retornó a un punto donde el obstáculo ha sido evitado el rumbo del avance debe retomar su comportamiento rectilíneo. Para poder determinar la ruta más recta posible se debe obtener los signos de avances de todos los valores de las dimensiones (o articulaciones) en ese instante con respecto a los valores finales de todas las dimensiones (o articulaciones) finales.

3.4.4. **PASOS DE AVANCE.**

En una trayectoria los avances individuales que se le da a los valores de las dimanaciones (o articulaciones) se conocen con el nombre de pasos, estos pasos pueden tener valores constantes o variables. Al introducir obstáculos se puede producir eventos como la imposibilidad de avance con un paso constante obligando a aumentar el valor de los pasos para evitar el obstáculo.

3.4.5. **EXCLUCION DE DATOS.**

A medida de que se aumenta las dimensiones (o articulaciones) las permutaciones que se deben realizar aumentan de forma exponencial, a nivel computacional eso significa que la convergencia de un punto inicial a un punto final se incrementa enormemente haciendo inoperable el algoritmo.

Se decidió que a medida que cada dimensión (o articulación) llegara a su valor final esta se excluyera de la búsqueda para disminuir así cada vez más las permutaciones necesarias que debe realizar el algoritmo convergiendo mucho más rápido y convirtiéndose en un algoritmo operable.

3.4.6. **ELIMINACION DE COORDENADAS INTERMEDIAS.**

Luego de obtener las trayectorias articulares finales contemplando las restricciones cinemáticas (ver páginas 21 a la 26) estas no son redundantes en el avance de cada articulación, pero si se realiza un cambio de coordenadas articulares a coordenadas cartesianas del efector final (pie móvil) se obtiene que ciertos movimientos son redundantes, lo que conlleva aplicar un método de eliminación entre cada coordenada redundante para sí optimizar la trayectoria final.

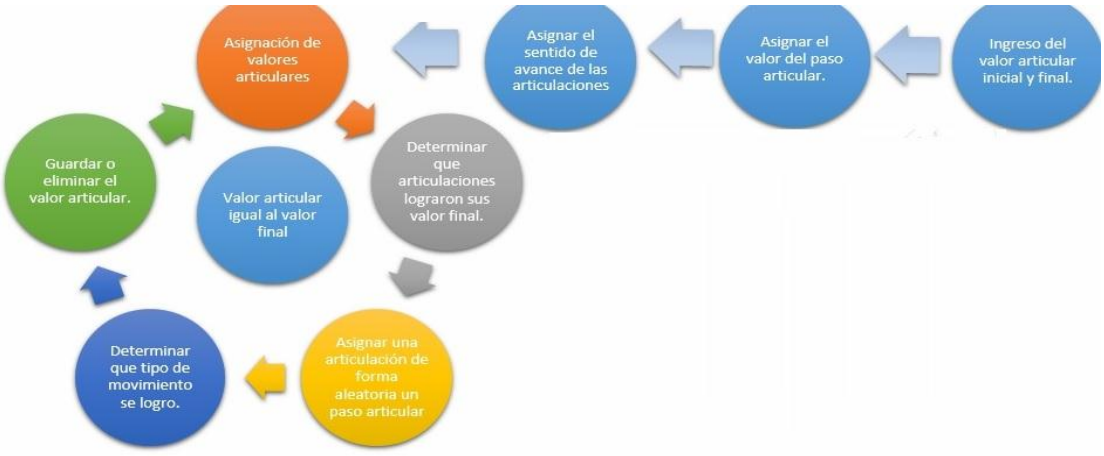
Posteriormente se pasa de nuevo a las coordenadas articulares pero ya optimizado y finalmente estas se aplicaran o al modelo digital del robot o al robot físico.

3.4.7. **SECUENCIAS DEl ALGORIMO GENERADOR DE TRAYETORIA AUTOMATÍCAS.**

Implementando los puntos anteriores mencionados de una forma ordenada en un solo algoritmo se genera trayectorias automáticas para dos coordenadas articulares donde el robot bioloid pueda realizarlo sin presentar caídas o ruptura estructural, por facilidad se organizó una serie de secuencias que expresa el comportamiento del algoritmo a las diferentes eventualidades que puedan suceder.

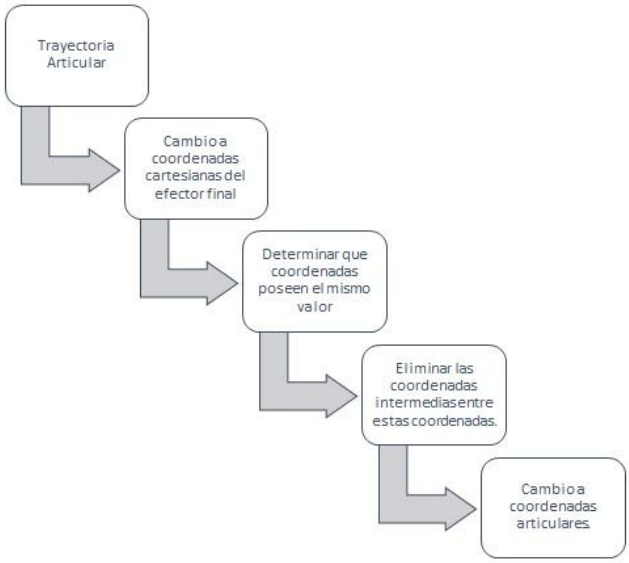
Se genero un dos flujograma con los procesos del algoritmo.

FIGURA 41: FLUJOGRAMA DEL ALGORITMO PRIMERA PARTE.



Fuente: Autor.

FIGURA 42: FLUJOGRAMA DEL ALGORITMO SEGUNDA PARTE.



Fuente: Autor.

La secuencia completa es:

1. Determinar el sentido
2. Fijar pasos
3. Escoger aleatoriamente la articulación a variar
4. Verificar si alguna articulación llego al objetivo sino llego continuar con el algoritmo y si llego se excluye esa articulación de la búsqueda
5. Verificar restricciones
   a) Centros de masas
   b) Limites

   Si cumple se anexa las coordenadas generalizadas del robot y sigue al paso 3

6. Si se sale de los limites articulares o la proyección del centro de masas no se encuentra en el área del pie de apoyo del robot.
   Desestimar las nuevas coordenadas articulares y retomar el valor anterior.
   Empieze desde la primera articulación le realiza el incremento según el sentido y verificar si el nuevo vector de coordenadas articulares es válido o no valido, si es válido y esa articulación no llego a su objetivo con ese paso, retome el vector original y continúe al paso 3.

En caso contrario determinar si es un movimiento no valido o si al darle el paso llego a su objetivo, se incrementa el contador dependiendo de qué situación se presente.
Y seguir con la siguiente articulación y repetir este pasó hasta la última articulación presente.

7. Si dentro de la búsqueda se llega hasta la última articulación, se compara si el contador de movimientos no válidos y el contador de llegadas son iguales a la cantidad de articulaciones presentes en el comienzo de la búsqueda de trayectorias.
   Si es así se guarda el valor y la posición de la articulación más alejada de su objetivo y se permite que las articulaciones que llegaron a su objetivo superen ese valor, reinicie todos los contadores y luego pasar al paso 3
   Y cada vez que inicie un ciclo compare si el valor de la articulación actual en la posición guardada es diferente al valor guardado de la articulación más alejada,
   Si es así recalcular el sentido de avance de las articulaciones y se continúa normalmente la búsqueda.

8. Si dentro de la búsqueda se llega hasta la última articulación se compara si el contador de movimientos no validos si es igual a la cantidad de articulaciones presentes en el comienzo de la búsqueda de trayectorias.
   Se compara si el paso articular es menor o igual a un cuarto de la diferencia inicial de la articulación más cercana de su objetivo.
   Si es así reinicie todos los contadores aumente una unidad al paso articular y se retoma al paso 6
   Si no es menor y el contador de movimientos no validos es igual a la cantidad de articulaciones presentes al comienzo de la búsqueda.
   Se remplaza las trayectorias con los valores guardados de las trayectorias antes de existir un movimiento no valido y los valores articulares actuales con los valores articulares antes de existir el primer movimiento no valido.
   Se obtiene el valor y la posición de la articulación en ese instante más alejada de su objetivo, Se invierte el sentido de avance de todas las articulaciones excepto la que se esté más alejada de su objetivo se reinicia todos los contadores y se pasa al paso 3.
   Cada vez que inicie un ciclo compare si el valor articular actual en la posición guardad es diferente al valor guardado de la articulación más alejada de su objetivo.
   Si es así se recalcula el sentido actual de las articulaciones y se continua normalmente la búsqueda.

9. Si existen movimientos no validos se guardan las trayectorias obtenidas hasta el instante anterior de existir el movimiento no valido al igual que su posición articular en el instante anterior de existir el movimiento no valido.
   Si es primera vez que existe un movimiento no valido se guarda su trayectoria articular y sus valores articulares.
   Si no es la primera vez se compara los valores articulares guardados y los valores articulares presentes antes de existir un movimiento no valido con los valores articulares objetivos.
   Se compara quien presenta mayor número de articulaciones que llegaron a sus objetivos.
   Si el valor articular presente antes de existir un movimiento no valido posee más articulaciones que llegaron a su valor objetivo se remplazan las trayectorias y los valores articulares presentes antes de existir un movimiento no valido.

10. Si al cabo de un número limitado de ciclos no se obtiene solución se eliminan los valores de las articulaciones, las trayectorias y el valor articular vuelve a su valor inicial y se pasa al paso 3

11. Luego de obtener las trayectorias articulares finales se transforman a coordenadas espaciales del efector final (ósea el pie móvil) y se eliminan las coordenadas intermedias entre dos coordenadas iguales.

12. Se transforman nuevamente en coordenadas articulares y se implementan los valores dentro del robot.

Estos algoritmos fueron implementados en primera instancia en el simulador para poder analizar sus resultados detectar errores y luego implementarlos en el modelo físico.

3.5. TRASFERENCIA DE MARCHAS AL ROBOT BIOLOID.

Luego de obtener una trayectoria articular final (ver paginas 48 ala 56) se deben pasar o al simulador de trayectorias (ver paginas 35 ala 47) o al modelo físico (ver página 66). Se debe tener en cuenta algunos dispositivos y procesos intermedios que juegan un papel relevante en la comunicación entre la máquina de cómputo que posee los algoritmos y el robot real.

Todos los dispositivos deben estar interconectados entre sí para poder tener una comunicación idónea y así trasmitir toda la marcha completa y a su vez que los servomotores envíen al computador la posición actual para controlar su movimiento mientras se está operando el robot.

3.5.1. **SERVOMOTOR DYNAMIXEL AX-12.**

Los servomotores son motores de corriente directa donde se puede hacer control del ángulo de giro, por esta propiedad son bastantes usados en la robótica, el servomotor Dinamixel AX-12 se utilizó principalmente por que el robot bioloid type A está diseñado para que funcione con estos servomotores, pero debido a sus propiedades se pueden interconectar todos los servomotores que se estén operando y conectarse todos a su vez con el controlador del robot, a su vez que cada uno posee un id un identificador único que es trasmitido al controlador y con este número puede identificarlo y realizar las acciones de control pertinentes.
Sin contar que posee control de toque, velocidad, temperatura, voltaje etc. [10].

FIGURA 43: SERVOMOTOR DYNAMIXEL AX-12.

FUENTE: USER'S MANUAL, DYNAMIXEL AX-12, ROBOTIS, 2006-06-14.

3.5.2. **CONTROLADOR CM-530.**

El controlador del robot bioloid type A es el CM-530 el cual se encarga en enviar y recibir datos de cada uno de los servomotores del robot, se encarga de sumistrarle a cada servomotor la energía necesaria para su operatividad y a su vez se puede controlar desde el software RoboPlus ® que es desarrollado por el fabricante poder programarle ciertas rutinas para que la ejecute [11].

En esta investigación nos comunicaremos al controlador CM-530 por protocolo TTL.

FIGURA 44: CONTROLADOR CM-530.

FUENTE: USER'S MANUAL2006-06-14, CM-530, ROBOTIS.

3.5.3. **USB2DYNAMIXEL.**

Para poder enviar los datos desde el computador se debe hacer un paso donde los datos salgan por algún puerto del computador y se transformen a datos TTL que se enviaran al controlador CM-530 para que lo envíen a los servomotores.

Para poder realizar esta conversión existe un dispositivo llamado usb2dynamixel este dispositivo permite la comunicación no solo en TTL si no en R232, R485 y comunicación serial, por este último puerto se puede realizar comunicación inalámbrica esta relación se realiza con el protocolo zigbee pero en esta investigación solo se abordara la comunicación TTL [12].

FIGURA 45: USB2DYNAMIXEL.

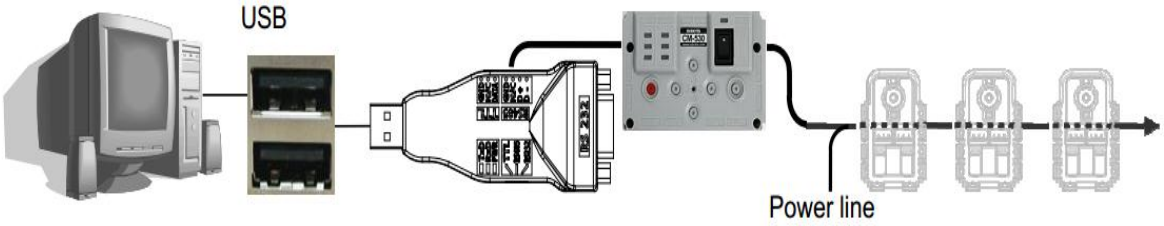
FUENTE: USER'S MANUAL2006-06-14, USB2DYNAMIXEL, ROBOTIS.

3.5.4. **ESQUEMA DE CONEXIONES PARA LA TRASMISIÓN.**

Todos los dispositivos deben estar conectados y listos para que el algoritmo que hace comunicación entre el lenguaje de programación y el usb2dynamixel y así pueda realizar toda marcha que se le envíen.

La comunicación debe ser bidireccional para poder tener control de los ángulos y a su vez saber si se presenta algún tipo de anomalía como sobrecarga, falta de corriente etc.

FIGURA 46: ESQUEMA DE CONEXIONES PARA LA TRASMISIÓN.



FUENTE: USER'S MANUAL2006-06-14, USB2DYNAMIXEL, ROBOTIS.

3.5.5. **ALGORITMOS DE TRANSMISIÓN DE MARCHAS.**

Luego de estar todos los dispositivos interconectados se debe habilitar un puerto virtual (COM) donde se transmiten información estos puertos son habilitados por un algoritmo computacional este algoritmo se programó en el lenguaje Matlab ® al ser Matlab ® un lenguaje de alto nivel el tiempo de ejecución es mucho más alto que un lenguaje de bajo nivel, se decidió utilizar un compilador en C en el lenguaje Matlab ® con el fin de controlar los servomotores de una forma más rápida ya que no solo se le envían los ángulos a los servomotores si no que de forma análoga se le está enviando velocidad, carga etc. además de que recibe datos de los servos por estas razones si no se utilizara un lenguaje de bajo nivel no se podría realizar eficientemente a su vez este algoritmo carga unas librerías escritas en C donde habilitan todas las propiedades de los servomotores.

Luego de realizar comunicación se organiza el robot en su posiciones iniciales y se guardan sus valores iniciales en un vector, se crea otro vector donde contiene todos los ids de los servomotores y finalmente se crea otro vector que contiene las velocidades de los servomotores estas velocidades deben ser bajas (ver paginas 21 ala 26), todos los vectores en la misma posición deben referenciar el mismo servomotor.

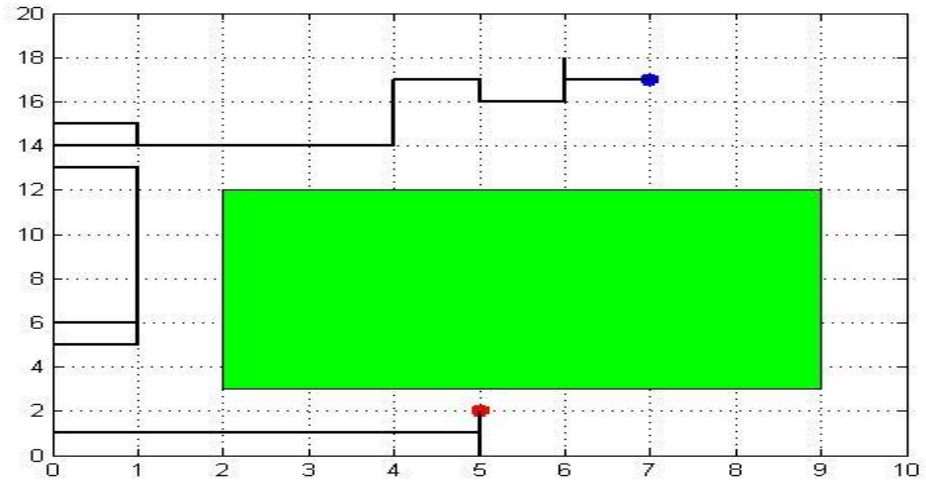
Se crea otro algoritmo donde se unen las diferentes trayectorias creadas por el generador de trayectorias (ver páginas 48 a la 56) para formar una sola marcha estas se analiza el sentido de giro de cada servomotor dependiendo de esto se crea un vector indicador de el sentido de giro y luego se utiliza para restar o sumar cada movimiento de la marcha a la posición inicial y esta se unen con el algoritmo que se encarga en cargar la librería en C para poder mover y controlar su velocidad de los servomotores y así finalmente realiza la marcha el robot real.

PRUEBAS Y RESULTADOS

Se realizaron tanto para simuladas como para el modelo real, Las pruebas simuladas fueron:

En primera instancia para probar el concepto se realizaron pruebas en un espacio bidimensional en donde dos puntos uno de inicio y otro de llegada realizan una trayectoria sobrepasando un área no permitida (ver figura 47).

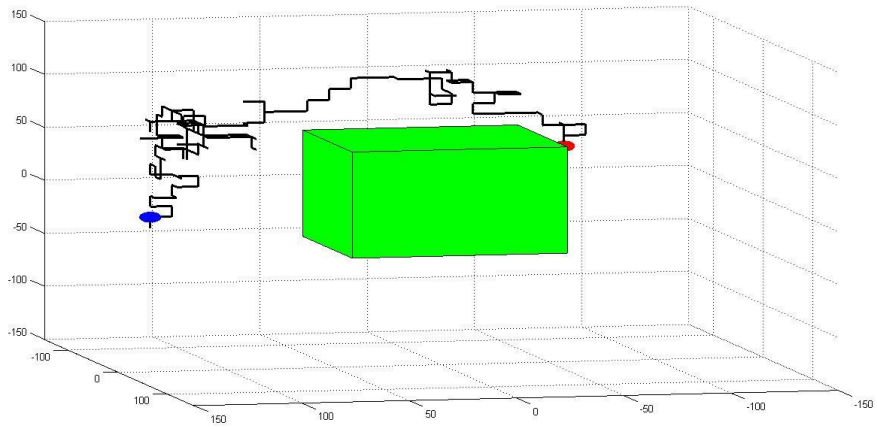
FIGURA 47: SIMULACIÓN DE OBSTACULOS EN 2D FASE DE PRUEBA.



Fuente: Autor.

Haciendo una extrapolación (donde cada eje representa una articulación), se realizaron experimentos tridimensionales con las mismas condiciones. En sus primeras pruebas el algoritmo de forma poco ordenada generaba una trayectoria pero carecía de algún sentido aparente (ver figura 48).

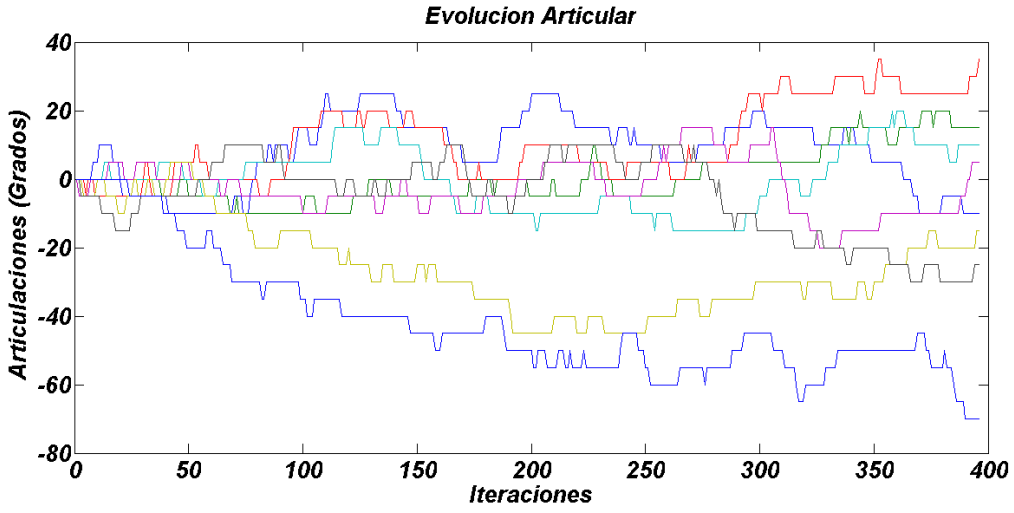
FIGURA 48: SIMULACIÓN DE OBSTACULOS EN 3D FASE DE PRUEBA.



Fuente: Autor.

Al resultar fructíferas las respuestas obtenidas en los experimentos se procedieron a realizar pruebas con la extrapolación para todas las articulaciones. En estas pruebas la generación de una trayectoria partiendo de unas articulaciones iniciales a unas finales, se realizaron de forma correcta, pero su respuesta fue muy lenta alrededor de dieciséis horas lo cual hace inoperable el algoritmo en la práctica.

FIGURA 49: TRAYECTORIAS DE OCHO ARTICULACIONES FASE DE PRUEBA.

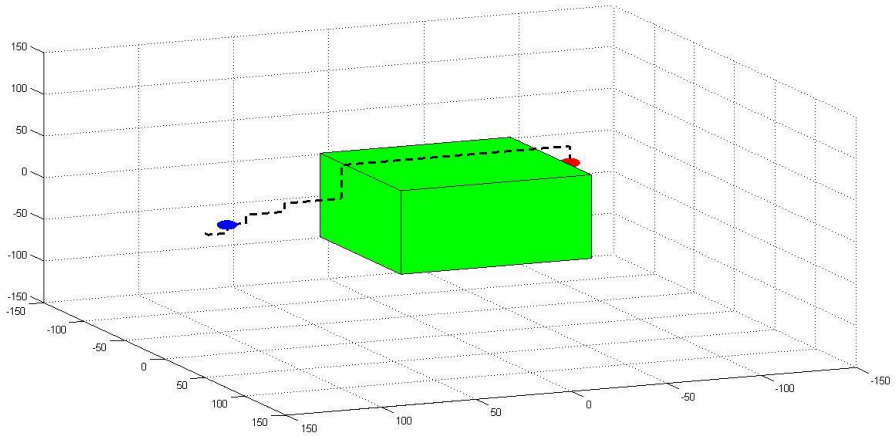


Fuente: Autor.

A razón de estos resultados se decidió optimizar el algoritmo (ver páginas 48 a la 56) y de manera simultánea se observaron los tipos de movimiento existentes y las clases de movimientos no validos presentes dentro del robot.

Realizando los ajustes ya mencionados dentro de la generación de trayectorias, se decidió nuevamente realizar experimentos en un espacio tridimensional con obstáculos para observar el fenómeno nuevamente.

FIGURA 50: SIMULACIÓN DE OBSTACULOS EN 3D FASE FINAL.



Fuente: Autor.

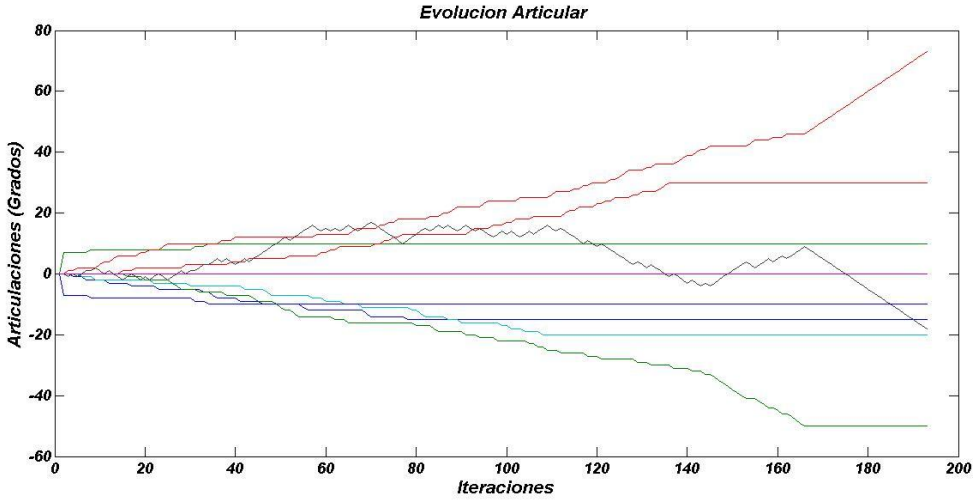
Los resultados fueron los esperados optimizando el algoritmo y la trayectoria de una manera bastante notoria, reduciendo tanto en cantidad de iteraciones, cantidad de movimientos y tiempo de ejecución. El tiempo de ejecución anterior fue aproximadamente de una hora al realizar la optimización su respuesta fue de dos segundo, la cantidad de movimientos para realizar la trayectoria antes de optimizarlo fue de 300 y con la optimización se redujo a 94 (datos aproximados), por su excelente respuesta se dispuso a dar paso a la implementación con las articulaciones de los pies del robot añadiéndole a su vez los cálculos cinemáticos y las restricciones mecánicas que posee el robot para poder optimizar el código se referenciaron tres articulaciones del pie móvil del robot con respecto a las nueve articulaciones restantes de los pies del robot, pasando solo para los pies de doce a nueve articulaciones con estas modificaciones se optimizo el tiempo de ejecución del algoritmo.

Al realizar las pruebas se mostró igualmente una mejora significativa en la cantidad de iteraciones, cantidad de movimientos y tiempo de ejecución, permitiendo que su operatividad este dentro de los parámetros permitido, generando la posibilidad de ser un algoritmo on line en el robot.

La respuesta de anterior fue aproximadamente 16 horas, con la optimización la realiza en 3 segundos.

Sus movimientos se redujeron de uno 470 a 190 aproximadamente.

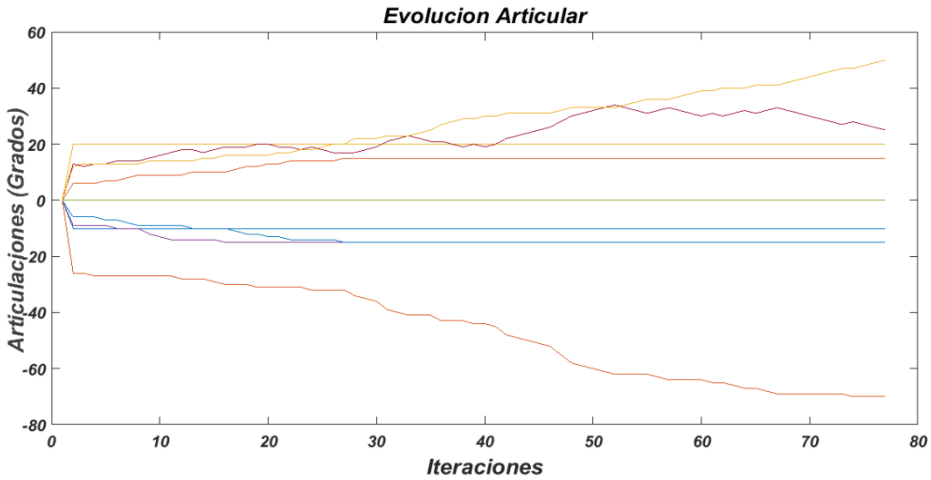
FIGURA 51: TRAYECTORIA ARTICULAR PIES DEL ROBOT FASE DE PRUEBA.



Fuente: Autor.

Los resultados obtenidos se pasaron a coordenadas cartesianas del pie móvil del robot, y se puedo visualizar movimientos redundantes que realizaba el efector final del robot (planta del pie móvil), se realizaron modificaciones para eliminar las coordenadas redundantes (ver página 70).

FIGURA 52: TRAYECTORIA ARTICULAR PIES DEL ROBOT FASE FINAL.

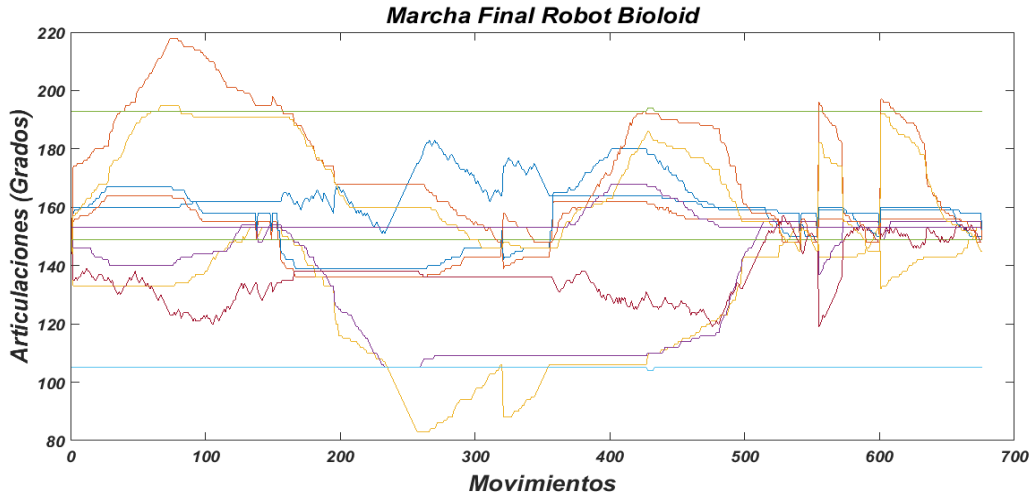


Fuente: Autor.

Posteriormente se acoplaron los resultados al simulador de trayectorias donde se encuentra el modelo gráfico del robot (ver páginas 50 a la 63).

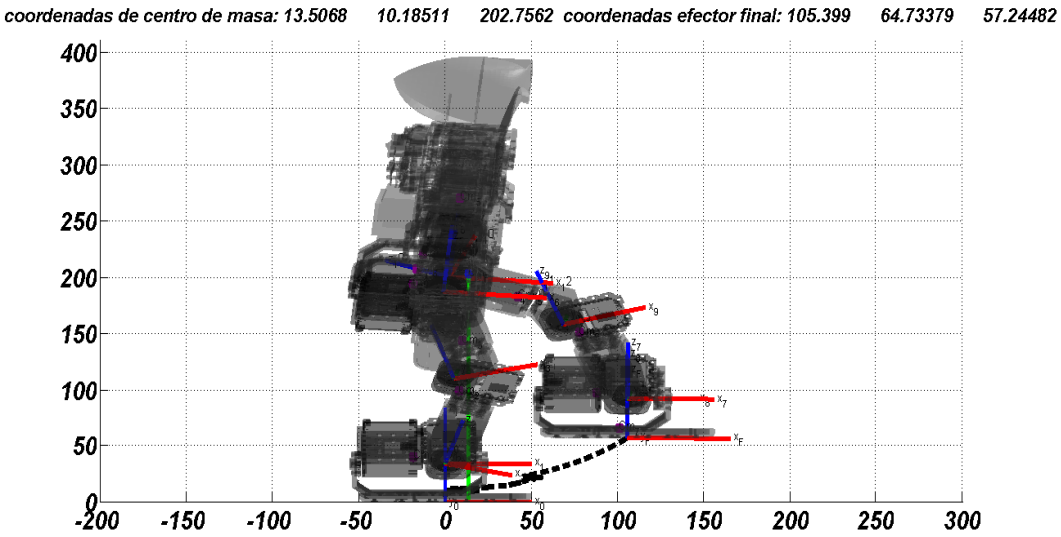
Como una marcha está compuesta por diferentes pasos se decidió dividirlo en secciones partiendo de un reposo, luego sube el pie izquierdo hasta un punto máximo, luego baja este pie nuevamente hasta el piso pero más alejado, luego el pie izquierdo se convierte en el pie fijo y el derecho lo desplaza hasta un punto máximo y lo pasa nuevamente hasta el piso pero más alejado. Nuevamente el pie izquierdo vuelve al punto más alto y finalmente apoya los dos pies de forma paralela en el piso generando así una marcha.

FIGURA 53: TRAYECTORIA DE LA MARCHA FINAL.



Fuente: Autor.

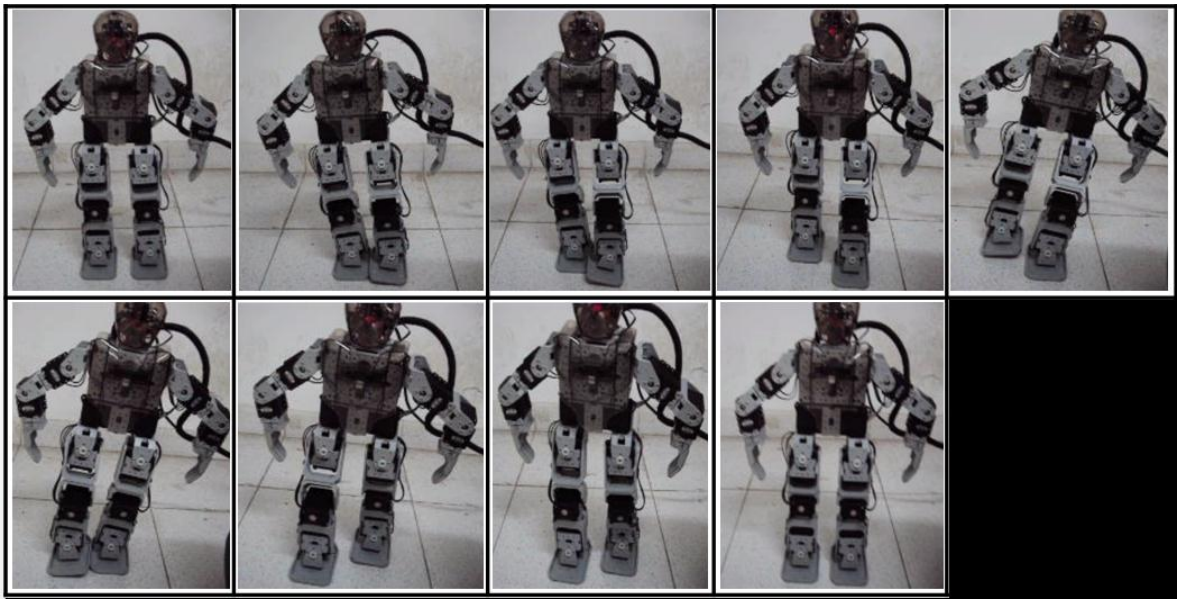
FIGURA 54: SIMULACION DE MARCHA FINAL.



Fuente: Autor.

Esta investigación hace corroboración en la parte experimental desde la fase de simulación a la implementación en el modelo real donde se realiza finalmente las pruebas de la generación automática de trayectorias.

FIGURA 55: SECUENCIA DE MARCHA FINAL ROBOT BIOLOID.



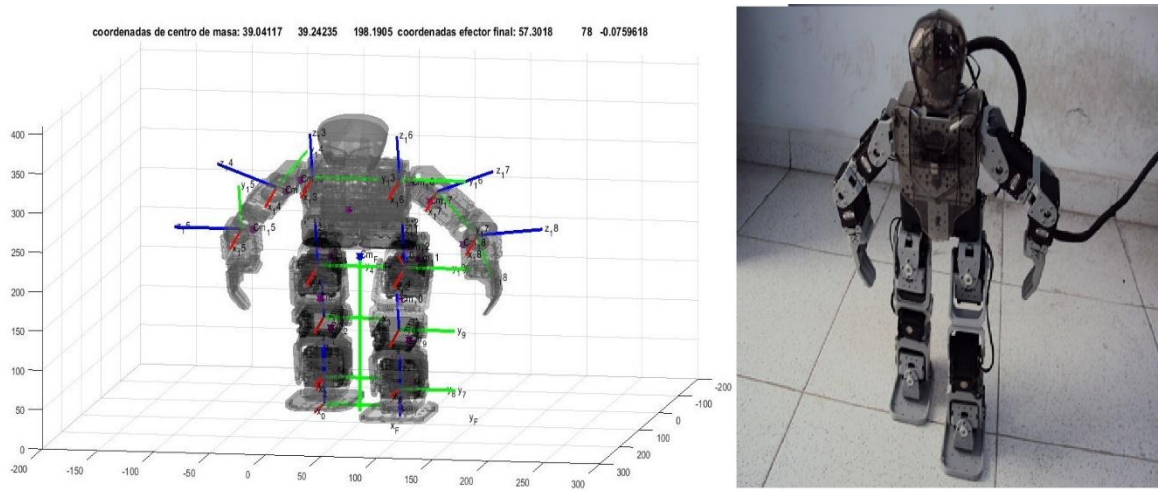
Fuente: Autor.

Cuando se realizaron las pruebas por primera vez en el robot físico hubo unos errores de aproximación puesto que no se había considerado los backslash presentes en los eslabones del robot (ver páginas 29 a la 48), en algunos eslabones era despreciables pero en los eslabones que están referenciados por las matrices de trasformación homogénea (ver página 21 a la 26) A1,A5,A7 y A11 fueron las partes más afectadas, otro inconveniente que se presento fue por el método de comunicación (TTL) entre el computador y el robot en ocasiones tiraba el cable aplicándole una fuerza extra que afectaba el equilibrio.

Estas dificultades se sobrepasaron reduciendo el área del pie de apoyo del robot, acomodando el centro de masas final del robot de forma que coincidiera de forma más real con los movimientos que el robot físico realiza.

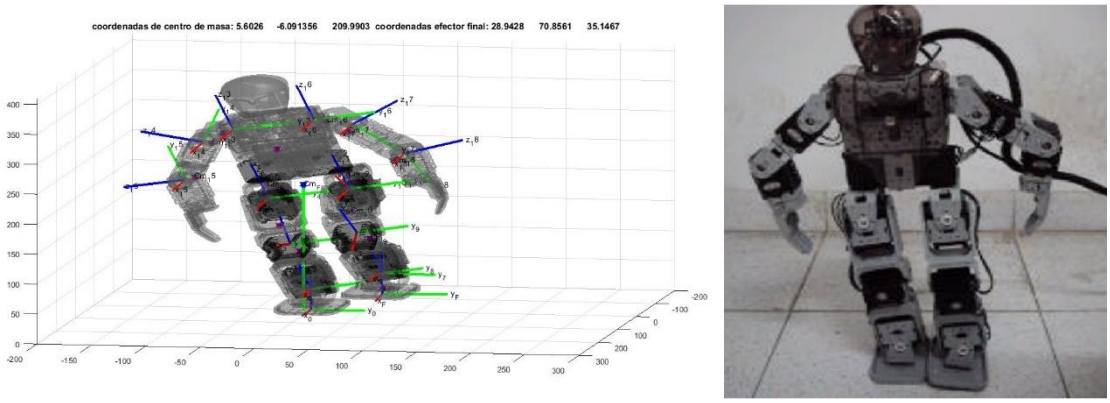
Luego de realizar las modificaciones se realizaron nuevamente las pruebas 17 de noviembre del 2016, su comportamiento fue el esperado realizando las pruebas de manera satisfactoria, para probar la discrepancia entre el simulador y el modelo real (ver figura 65)

FIGURA 56: COMPARACIÓN SIMULADOR VS ROBOT BIOLOID.



Fuente: Autor.

FIGURA 57: COMPARACIÓN SIMULADOR VS ROBOT BIOLOID DOS.



Fuente: Autor.

Se realizaron cien pruebas en el robot real y se obtuvo un 76% de efectividad a la hora de realizar una marcha, las pruebas validas se le aplicaron técnicas estadísticas y se organizó una tabla recopilatorio (ver tabla 5). Las muestras se obtuvieron al medir la distancia de la proyección del centro de masas del robot con el centro del pie de apoyo del robot.

TABLA 6: DATOS ESTADÍSTICOS DE LA MARCHA DEL ROBOT BIOLOID.

Datos estadísticos de la marcha del robot bioloid.			
Pruebas	***Medias***	***Desviaciones Estándar***	***Varianzas***
1.	*24,4332415346296*	*13,0365416282327*	*169,951417624644*
2.	*23,3384233719071*	*12,7549343986312*	*162,688351513386*
3.	*24,0995289740222*	*12,4631780660157*	*155,330807505214*
4.	*22,4816526087240*	*11,3662043068475*	*129,190600345000*
5.	*22,7917438396214*	*9,77658318627140*	*95,5815787980847*
6.	*23,3117633061970*	*11,1587888948316*	*124,518569599416*
7.	*22,3692777332629*	*10,4051047224540*	*108,266204285235*
8.	*21,3467558858540*	*9,14861323402258*	*83,6971241057331*
9.	*22,5093971679571*	*10,5587049518914*	*111,486250261096*
10.	*21,2075470159156*	*8,87971133132556*	*78,8492733276715*
11.	*23,9070131241363*	*12,0328303133206*	*144,789005349166*
12.	*22,3508463152019*	*10,5457156813614*	*111,212119232112*
13.	*22,6009568818043*	*10,1270549850865*	*102,557242670965*
14.	*22,8742024392594*	*10,1266500015036*	*102,549040252953*
15.	*23,2795090020783*	*10,0626316290803*	*101,256555302567*
16.	*23,5962686013919*	*11,3962350540671*	*129,874173407548*
17.	*24,5247288688172*	*11,4888275230575*	*131,993157854564*
18.	*22,1778334811985*	*10,1623442802816*	*103,273241270972*
19.	*21,7482422956397*	*10,1016411737928*	*102,043154404066*
20.	*24,0898708047822*	*11,1891439967036*	*125,196943378969*
21.	*23,6979155204036*	*10,6120946015030*	*112,616551831249*
22.	*24,0579919285346*	*11,8257943682211*	*139,849412439451*
23.	*23,6319912545514*	*10,3549468457319*	*107,224924177934*
24.	*22,6249325719794*	*10,7813879151223*	*116,238325376345*
25.	*21,6324545713272*	*9,93536483083013*	*98,7114743216962*
26.	*22,7392124453698*	*10,2461845977820*	*104,984298811826*
27.	*24,7148804317655*	*11,6815740031162*	*136,459171190280*
28.	*26,0607982121007*	*12,6618428692348*	*160,322264845193*
29.	*22,3744421918144*	*9,73098781705433*	*94,6921238956597*
30.	*22,7874609844585*	*9,13583507399401*	*83,4634824992191*
31.	*24,6175865781549*	*13,0058812031800*	*169,152945871231*
32.	*23,3173391146488*	*10,2956058249337*	*105,999499302408*
33.	*22,7480868997069*	*9,53801232715239*	*90,9736791529111*
34.	*23,9826286387843*	*11,4797573509999*	*131,784828837835*
35.	*22,3967494707548*	*11,3118046421386*	*127,956924261909*
36.	*23,8602417545675*	*10,0695361432203*	*101,395558139619*
37.	*23,2963129178027*	*10,8946289773702*	*118,692940554555*
38.	*22,8641183648449*	*9,87583743514650*	*97,5321650454410*
39.	*26,0607982121007*	*12,6618428692348*	*160,322264845193*
40.	*23,5820010090998*	*10,6516785179245*	*113,458255249214*

41.	*22,7576107113592*	*12,5783457179459*	*158,214781000168*
42.	*22,1615133952308*	*11,5897038184575*	*134,321234599568*
43.	*20,9846012935386*	*9,65443716225083*	*93,2081569198499*
44.	*22,8199453981615*	*11,7530288995948*	*138,133688314711*
45.	*23,2686423090084*	*10,9466370171408*	*119,828861985038*
46.	*23,3380463523893*	*13,3560185893919*	*178,383232560181*
47.	*23,3046879742416*	*11,2485066030572*	*126,528900799021*
48.	*21,8245780227271*	*11,6077007426984*	*134,738716532040*
49.	*22,4396703661350*	*9,97417480245442*	*99,4841629899167*
50.	*24,7646495312996*	*14,1248709009129*	*199,511977967457*
51.	*21,5688716587043*	*11,2594845011274*	*126,775991231129*
52.	*23,2250136301180*	*11,2683569762595*	*126,975868944417*
53.	*21,8885811861527*	*12,1408610473474*	*147,400506970997*
54.	*23,0999799023399*	*12,4294112293318*	*154,490263507840*
55.	*21,8715649763015*	*11,2597325247778*	*126,781576529540*
56.	*22,6687927329686*	*11,7169487098617*	*137,286887069530*
57.	*21,6752247784371*	*10,9286519943962*	*119,435434414620*
58.	*24,2988156358957*	*12,7145034199846*	*161,658597216800*
59.	*24,2988156358957*	*12,7145034199846*	*161,658597216800*
60.	*23,1483984269115*	*12,2341241121644*	*149,673792791842*
61.	*25,4933459729123*	*16,4832879809546*	*271,698782663083*
62.	*22,2183699362124*	*10,5812665291663*	*111,963201361254*
63.	*24,2792941425605*	*10,9487979257730*	*119,876176019411*
64.	*24,8962075761398*	*13,5540661391637*	*183,712708904823*
65.	*23,8737383831538*	*13,1354305489727*	*172,539535706885*
66.	*23,9889998715281*	*10,7248758864962*	*115,022962780747*
67.	*24,1695170937542*	*12,0069107222319*	*144,165905091646*
68.	*24,2154945720740*	*10,5721938488755*	*111,771282778200*
69.	*23,6798478463170*	*12,4610273792586*	*155,277203346632*
70.	*25,6248317763692*	*15,0032873099365*	*225,098630104502*
71.	*23,4355493579923*	*10,3931871312486*	*108,018338745151*
72.	*24,2554979190152*	*10,8964744888552*	*118,733156286271*
73.	*25,6776416924861*	*11,8248207361176*	*139,826385441317*
74.	*23,3286063186516*	*11,2571475818532*	*126,723371679623*
75.	*24,7736042210456*	*11,6187393705697*	*134,995104561227*
76.	*23,4295652196499*	*10,2683235200681*	*105,438467912784*

FUENTE: AUTOR

La marcha más estable se determinó al analizar las desviaciones estándar y las varianzas la prueba que tuviera menor valor seria la que esté más cerca del pie de apoyo dando como resultado la prueba diez con una desviación estándar de 8.879 y una varianza de 78.849.

Analizando el resultado obtenido se puedo observar una respuesta similar en la simulación como en el modelo real corroborando así el algoritmo generador de trayectorias y el análisis cinemático (método de los Screws) y así corroborar todos los pasos utilizados en esta investigación

CONCLUSIONES

- ✓ Con base a los resultados experimentales obtenidos por el simulador del Robot Bioloid, se apreció que el análisis matemático concuerda satisfactoriamente al trasladar los resultados angulares a los servomotores del modelo físico del Robot Bioloid, corroborando así el fundamento teórico (método de los Screws).

- ✓ Al analizar la marcha final automática del robot es el mismo concepto utilizado los espacios bidimensionales y tridimensionales en este orden de ideas se podría asumir que el algoritmo generador de trayectorias funciona para n dimensiones donde cada dimensión se representa con un grado articular.

- ✓ El algoritmo se puede utilizar también para generar una trayectoria a cualquier punto valido que el robot posea.

- ✓ Al ser un algoritmo donde solo introduce el punto inicial y final posee ventaja en espacios donde solo se tiene a la mano esos valores operando de forma autónoma.

BIBLIOGRAFÍA

[1] P. Wawrzyński, J. Możaryn, and J. Klimaszewski, “Robust estimation of walking robots velocity and tilt using proprioceptive sensors data fusion,” Rob. Auton. Syst, 00-665,2014.

[2] K. Teachasrisaksakul, Z. Zhang, G. Yang, and B. Lo, “Imitation of Dynamic Walking With BSN for Humanoid Robot,” vol. 19, no. 3, pp. 794–802, 2015.

[3] J. V. Nunez, A. Briseno, D. A. Rodriguez, J. M. Ibarra, and V. M. Rodriguez, “Explicit analytic solution for inverse kinematics of Bioloid humanoid robot *,” pp. 33–38, 2012.

[4] Y. Liu, P. M. Wensing, J. P. Schmiedeler, and D. E. Orin, “Terrain-Blind Humanoid Walking Based on a 3-D Actuated Dual-SLIP Model,” vol. 1, no. 2, pp. 1073–1080, 2016.

[5] K. Lee, H. Myung, and J. Kim, “Online Multiobjective Evolutionary Approach for Navigation of Humanoid Robots,” vol. 62, no. 9, pp. 5586–5597, 2015.

[6] T. Kishi, S. Shimomura, H. Futaki, H. Yanagino, M. Yahara, S. Cosentino, T. Nozawa, K. Hashimoto, and A. Takanishi, “Development of a Humorous Humanoid Robot Capable of Quick-and-Wide Arm Motion,” vol. 1, no. 2, pp. 1081–1088, 2016.

[7] Y. Hosoda, S. Egawa, and J. Tamamoto, “Basic Design of Human-Symbiotic Robot EMIEW,” no. c, pp. 5079–5084, 2006.

[8] B. Henze, A. Dietrich, and C. Ott, “An Approach to Combine Balancing with Hierarchical Whole-Body Control for Legged Humanoid Robots,” vol. 1, no. 2, pp. 700–707, 2016.

[9] TSAI - Robot Analysis.

[10] User’s Manual, DYNAMIXEL AX-12, ROBOTIS, 2006-06-14.

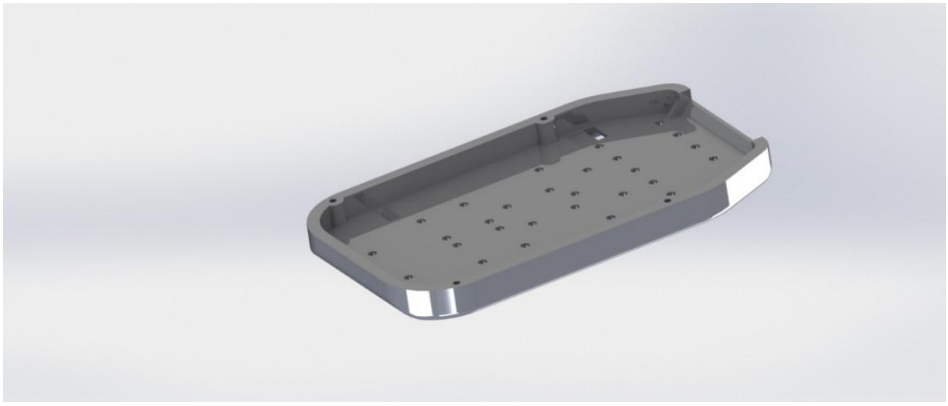
[11] User’s Manual, USB2DYNAMIXEL, ROBOTIS, 2006-06-14.

[12] User’s Manual, CM530, ROBOTIS, 2006-06-14.

APENDICES

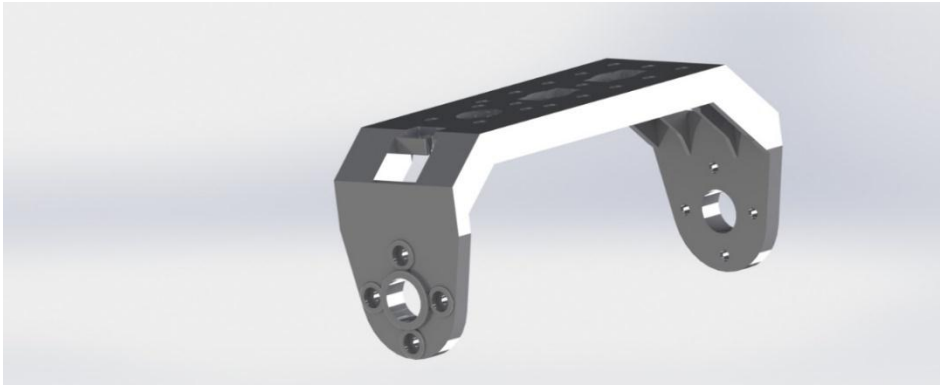
3.5.6. PIEZAS RENDERIZADAS

RENDERIZADO DE LA PLANTA DEL PIE DEL ROBOT.



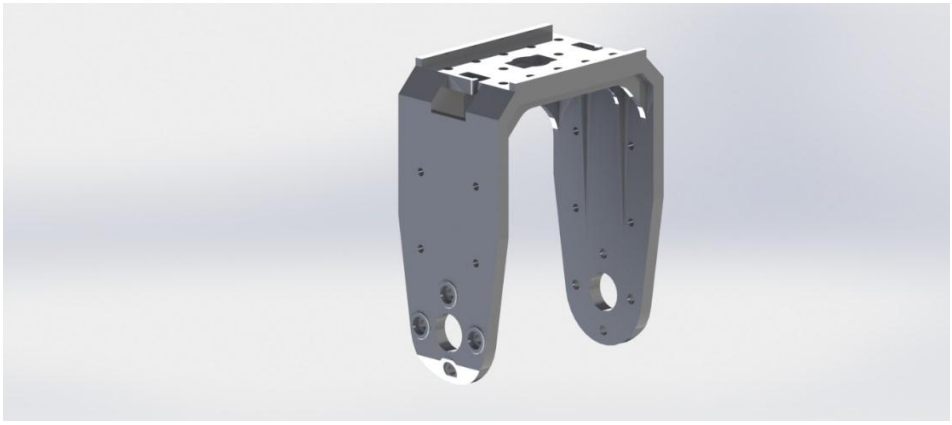
Fuente: Autor.

RENDERIZADO DE PIEZA DE APOYO DEL ROBOT.



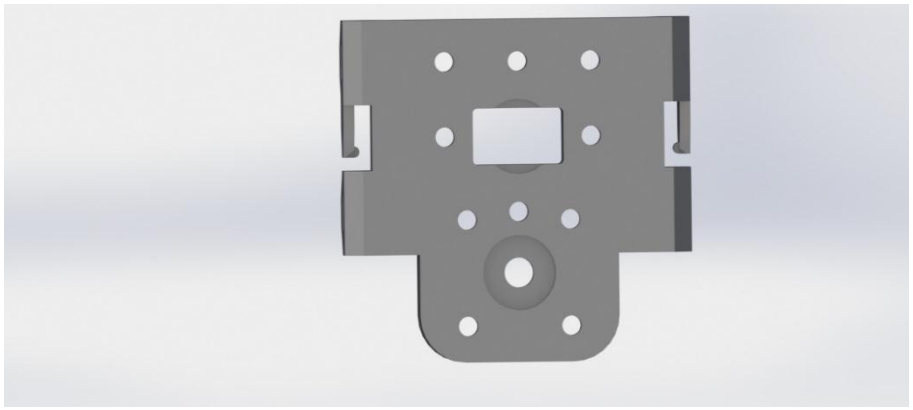
Fuente: Autor.

RENDERIZADO DE PIEZA ASIGNADA EN LOS ESLABONES MAS LARGOS DEL ROBOT.



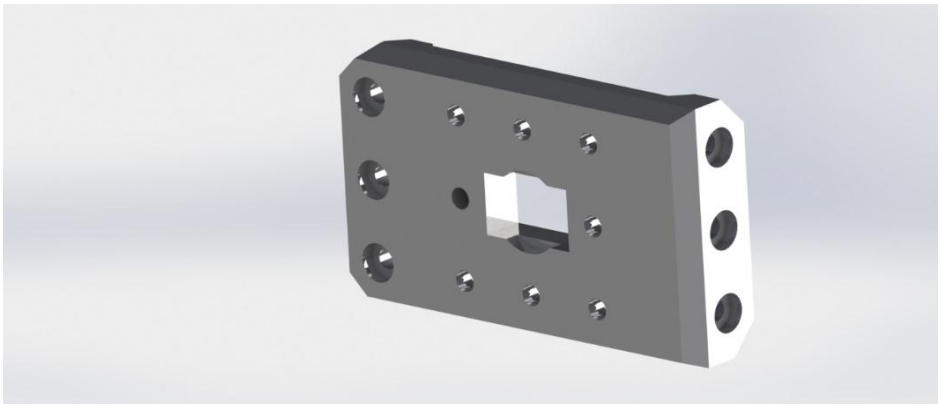
Fuente: Autor.

RENDERIZADO DE PIEZA DEL ROBOT.



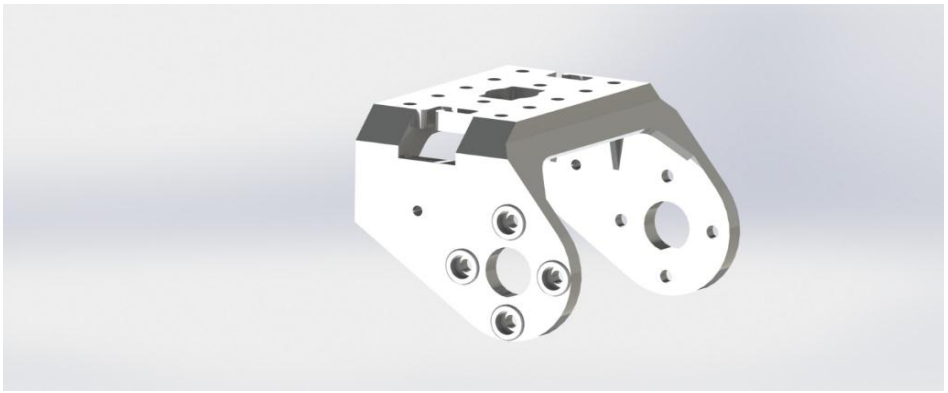
Fuente: Autor.

RENDERIZADO DE PIEZA PLANAR DE APOYO DEL ROBOT.



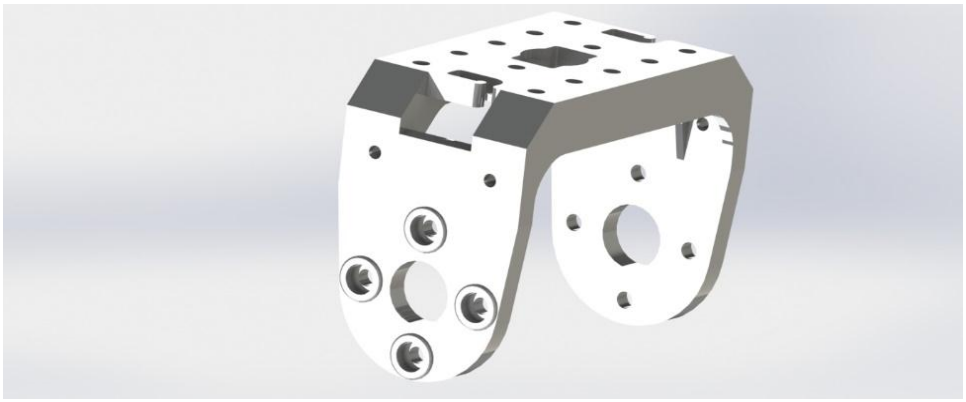
Fuente: Autor.

RENDERIZADO DE PIEZA EN FORMA DE HOMBRO O RODILLA DEL ROBOT.



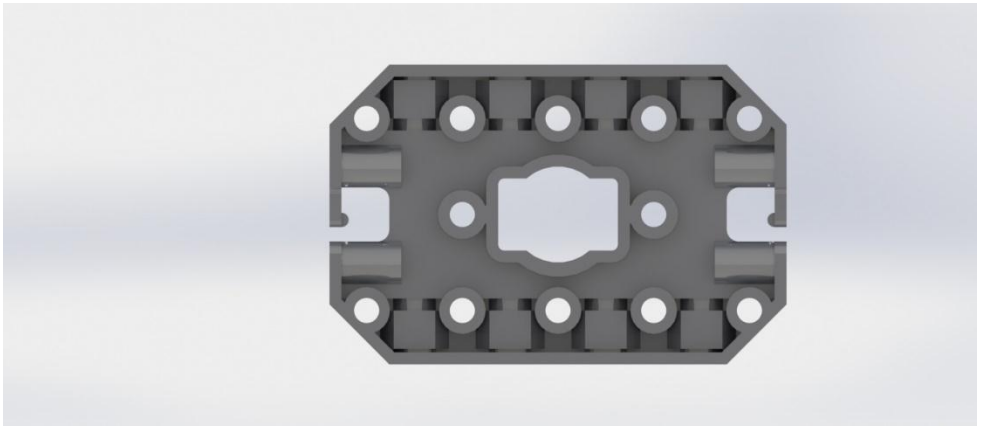
Fuente: Autor.

RENDERIZADO DE PIEZA EN FORMA DE BRAZO O ANTEBRAZO DEL ROBOT.



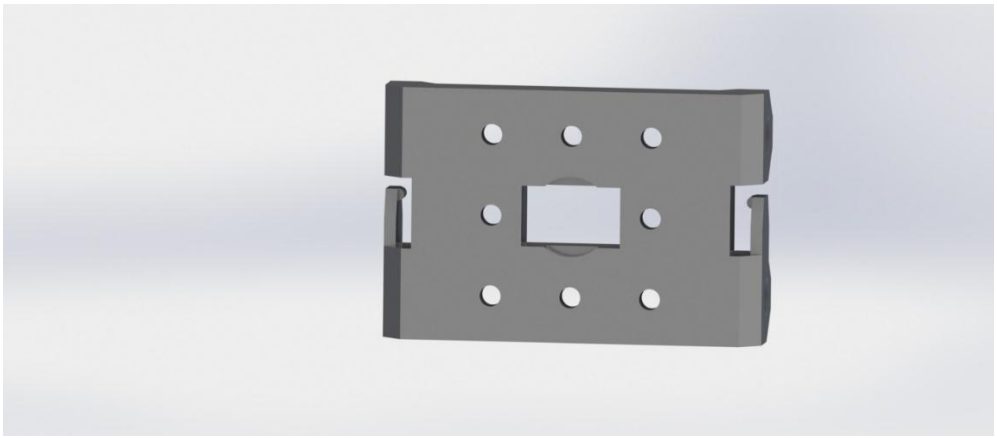
Fuente: Autor.

RENDERIZADO DE PIEZA DE ACOPLE ENTRE LA PIERNA DEL ROBOT.



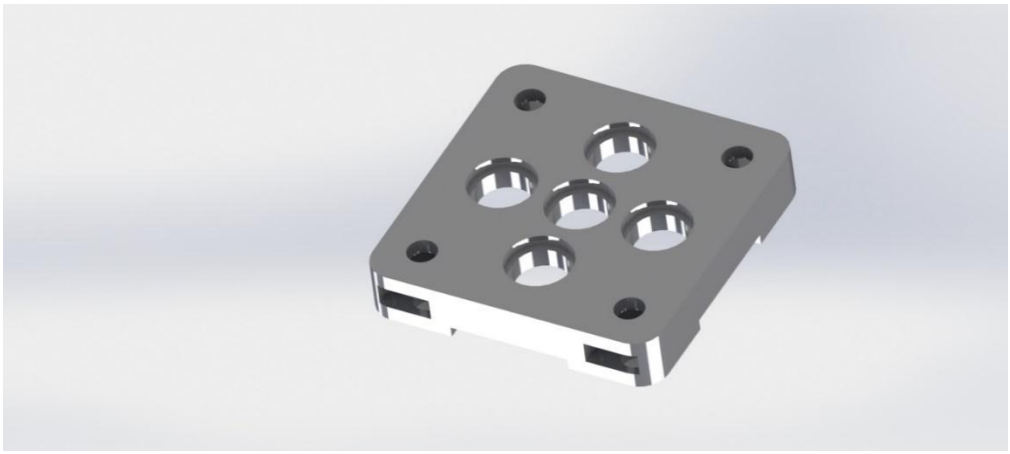
Fuente: Autor.

RENDERIZADO DE PIEZA DE ACOPLE ENTRE LA RODILLA DEL ROBOT.



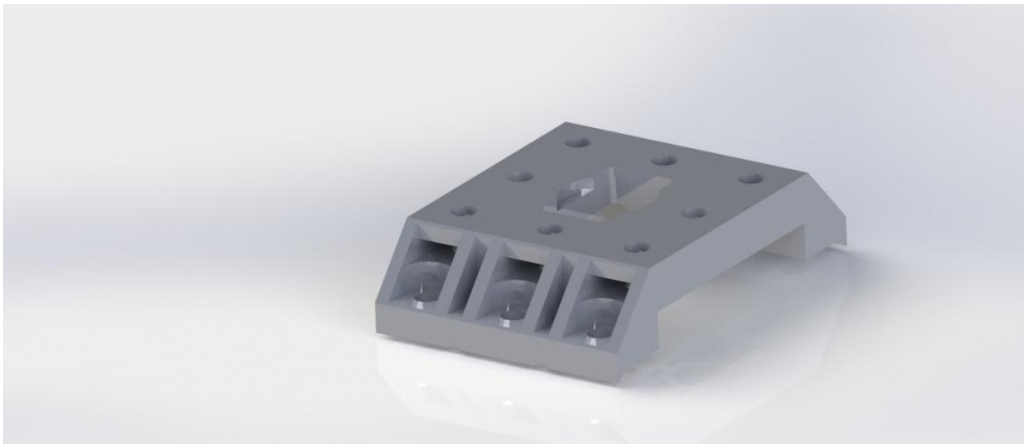
Fuente: Autor.

RENDERIZADO DE PIEZA DE ACOPLE ENTRE EL ABDOMEN DEL ROBOT.



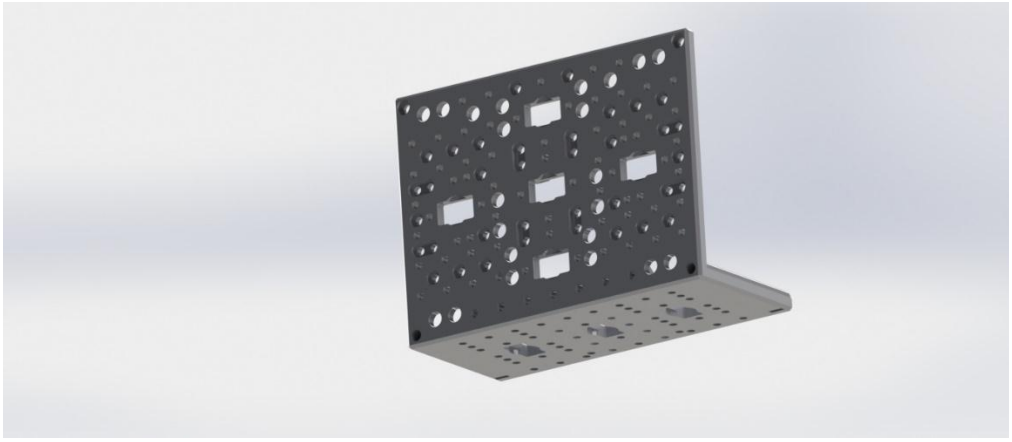
Fuente: Autor.

RENDERIZADO DE PIEZA DE ACOPLE ENTRE EL ABDOMEN Y LAS PIERNES DEL ROBOT.



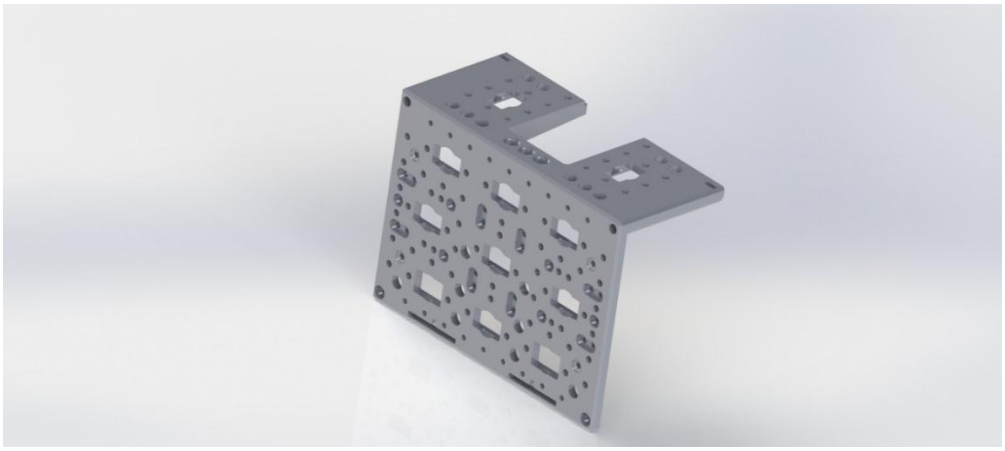
Fuente: Autor.

RENDERIZADO DEL TRONCO DELANTERO DEL ROBOT.



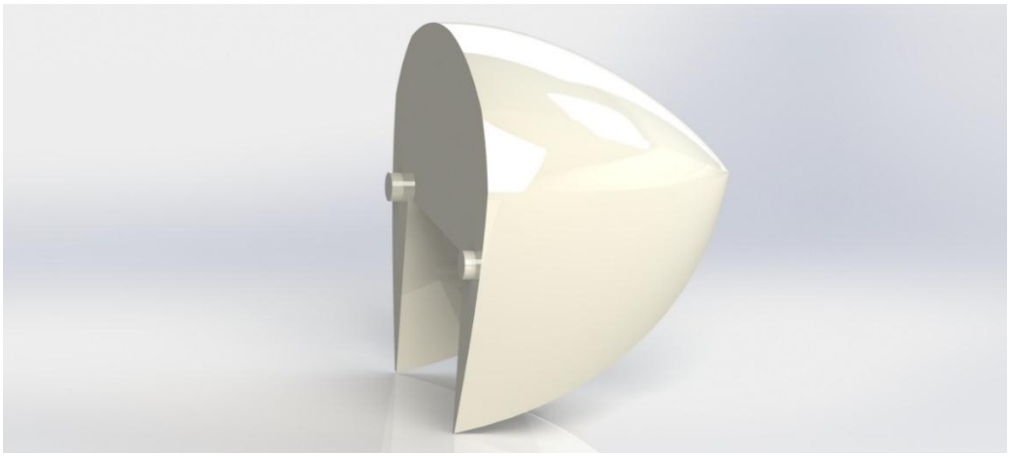
Fuente: Autor.

RENDERIZADO DEL TRONCO TRASERO DEL ROBOT.



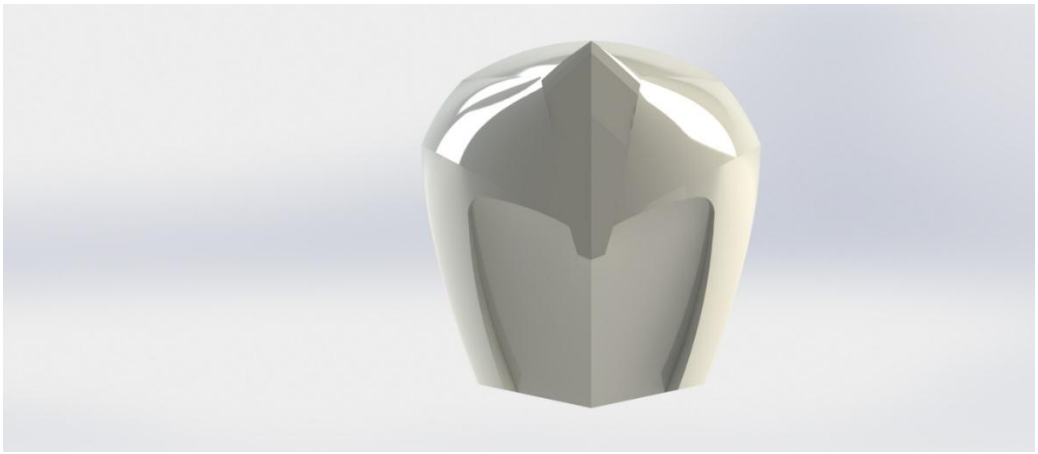
Fuente: Autor.

RENDERIZADO DE LA PARTE TRASERA DE LA CABEZA DEL ROBOT.



Fuente: Autor.

RENDERIZADO DE LA PARTE DELANTERA DE LA CABEZA DEL ROBOT.

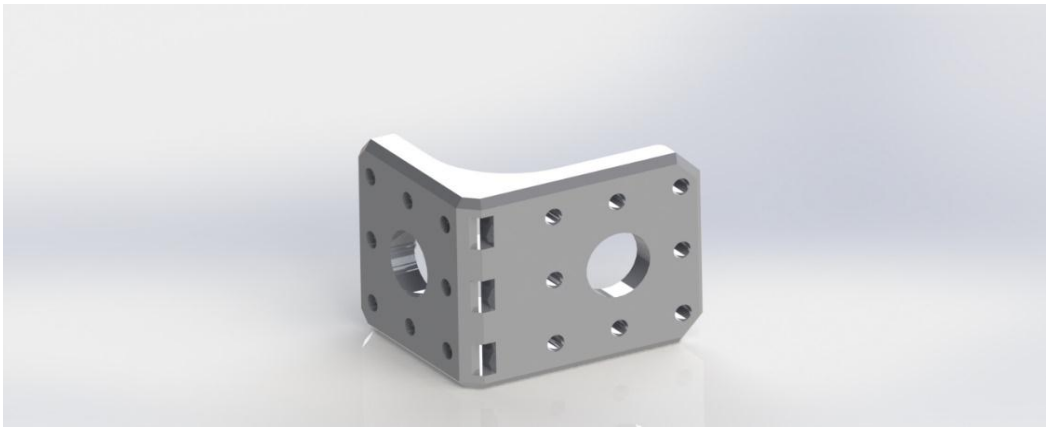


Fuente: Autor.

RENDERIZADO DE LA MANO DEL ROBOT.

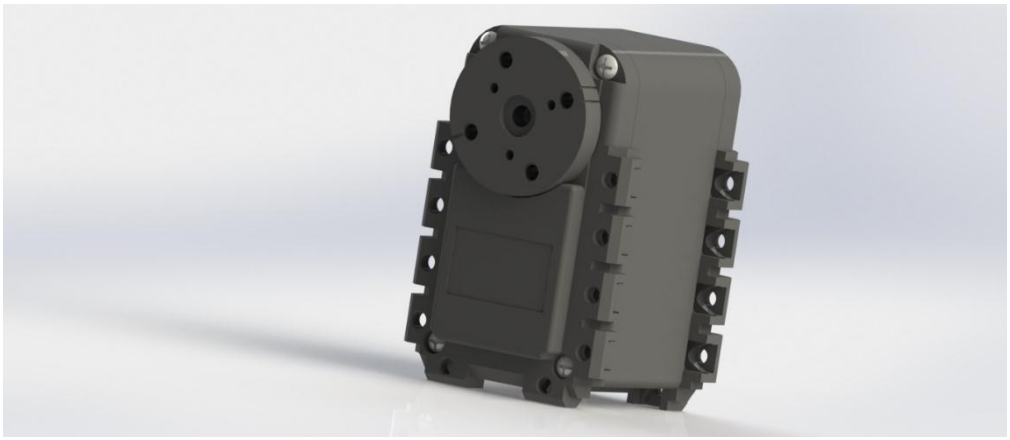
Fuente: Autor.

RENDERIZADO DEL CUELLO DEL ROBOT.



Fuente: Autor.

RENDERIZADO DEL SERVOMOTOR APLICADO EN EL ROBOT.



Fuente: Autor.

3.5.7. **CODIGOS**

```
% Bioloid_Calculos.m
%
% esta funcion se encarga de generar el tipo de opcion de movimiento y los
% angulos finales de las articulaciones.
%
% las opciones de movimiento son los siguientes:
% 0 es un movimiento no valido
% 1 es un movimiento valido
% 2 es un movimiento transitorio
%
% angulos es el vector articular
%
% ejemplo
%
% simulacion nueve articulaciones
%
% angulos=[0,0,0,0,0,0,0,0,0];
% [angulosf,movimiento]=Bioloid_Calculos(angulos)
%
% simulacion doce articulaciones
%
% angulos=[0,0,0,0,0,0,0,0,0,0,0,0];
% [angulosf,movimiento]=Bioloid_Calculos(angulos)
%
% Autor Cristian David Villate Martinez
%
function [angulosf,movimiento]=Bioloid_Calculos(angulos)
% bioloid calculos
% angulos a trabajar
dimen_a=length(angulos);
% obtencion de los tres grados articulares mediante las articulaciones
% introducidad (solo cuando son nueve grados articulares)
```

```
if(dimen_a==9)
    primer_a=angulos(1:6);
    segund_a=angulos(7:9);
    mov1_a=[angulos(1),angulos(3),angulos(4),angulos(7),angulos(8)];
    mov2_a=[angulos(2),angulos(5),angulos(9)];
    siete_a=-sum(mov1_a);
    ocho_a=-sum(mov2_a);
    doce_a=-angulos(6);
    angulos=[primer_a,siete_a,ocho_a,segund_a,doce_a];

end
% maximos y minimos grados articulares robot
 max_art=[23, 28, 135, 0, 28, 33, 99, 28, 0, 145, 28, 180];
 min_art=[-99, -28, 0, -145, -28, -180, -23, -28, -135, 0 -28 -33];
 if((sum(angulos>max_art)==0)&&(sum(angulos<min_art)==0))
% Conversión de grados a radianes
angulos=degtorad(angulos);
%longitudes
L1=34.5;
L2=75.07;
L4=14.50;
L3=77;
L5=78;
% posicion y localización del sistema de origen
M00=eye(4);
% masas
masa1=0.1818;
masa2=0.10364;
masa3=0.02104;
masa4=masa1;
masa5=1.06599;
% masa5=2.400;
masa6=0.01082;
masa7=0.03089;
% distancia minima en z
zmin=9.50580878560151;
% coordenadas de los centro de masas
masa1cor=[-18.40; 7.94; 4.21;1];
masa2cor=[-9.03;6.87;63.90;1];
masa3cor=[13.48;0.03;31.64;1];
masa5cor=[1.64;30.844;79.52;1];%masa5cor=[1.64;40.59;79.52;1];
masa6cor=[-17;0;18.94;1];
masa7cor=[-4.7;0;-25.83;1];
masa9cor=[5.47;9.67;-11.10;1];
masa10cor=[-1.02;0.03;-43.36;1];
centrosmasa.masast=[masa1 masa2 masa3 masa4 masa5 masa6 masa7 masa1 masa2 masa3
masa4 masa6];
% area del pie validad
areapiep=[39,19.76];
% screws
s1d=[0 -1 0];
s01d=[0 0 L1];
s2d=[1 0 0];
s02d=[0 0 L1];
```

```
s3d=[0 -1 0];
s03d=[-L4 0 (L1+L2)];
s4d=[0 -1 0];
s04d=[0 0 (L1+L2+L3)];
s5d=[1 0 0];
s05d=[0 0 (L1+L2+L3)];
s6d=[0 0 1];
s06d=[0 0 0];
s1i=[0 -1 0];
s01i=[0 0 L1];
s2i=[1 0 0];
s02i=[0 L5 L1];
s3i=[0 -1 0];
s03i=[-L4 0 (L1+L2)];
s4i=[0 -1 0];
s04i=[0 0 (L1+L2+L3)];
s5i=[1 0 0];
s05i=[0 L5 (L1+L2+L3)];
s6i=[0 0 1];
s06i=[0 L5 0];
%homogeneas
 A1=mat_tras_screw(s1d,s01d,(angulos(1)),0);
A2=mat_tras_screw(s2d,s02d,angulos(2),0);
A3=mat_tras_screw(s3d,s03d,angulos(3),0);
A4=mat_tras_screw(s4d,s04d,angulos(4),0);
A5=mat_tras_screw(s5d,s05d,angulos(5),0);
A6=mat_tras_screw(s6d,s06d,angulos(6),0);
A7=mat_tras_screw(s1i,s01i,angulos(7),0);
A8=mat_tras_screw(s2i,s02i,angulos(8),0);
A9=mat_tras_screw(s3i,s03i,angulos(9),0);
A10=mat_tras_screw(s4i,s04i,angulos(10),0);
A11=mat_tras_screw(s5i,s05i,angulos(11),0);
A12=mat_tras_screw(s6i,s06i,angulos(12),0);
% Posicion de referencia
q1d=[0; 0; L1;1];
q2dp=[-L4; 0; L2+L1;1];
q3d=[0;0;L3+L2+L1;1];
q1i=[0;L5;L1;1];
q2ip=[-L4; L5; L2+L1;1];
q3i=[0;L5;L3+L2+L1;1];
q0i=[0;L5;0;1];
% MTH Posición y orientación de referencia (arbitrarios)
A2r = [ eye(3), q1d(1:3,1) ; [0 0 0 1]];
A4r = [ eye(3), q2dp(1:3,1) ; [0 0 0 1]];
A5r = [ eye(3), q3d(1:3,1) ; [0 0 0 1]];
A6r = [ eye(3), q1i(1:3,1) ; [0 0 0 1]];
A8r = [ eye(3), q2ip(1:3,1) ; [0 0 0 1]];
A9r = [ eye(3), q3i(1:3,1) ; [0 0 0 1]];
A0r = [ eye(3), q0i(1:3,1) ; [0 0 0 1]];
% MTH Correspondiente a cada eslabon referida a la base
A01 = A2*A2r;
A02=A1*A2*A2r;
A03=A1*A2*A3*A4r;
A04=A1*A2*A3*A4*A5r;
```

```
A05=A1*A2*A3*A4*A5*A5r;
A06=A1*A2*A3*A4*A5*A6*A5r;
A07=A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A8*A6r;
A08=A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A6r;
A09=A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8r;
A010=A1*A2*A3*A4*A5*A6*A12*A11*A10*A9r;
A011=A1*A2*A3*A4*A5*A6*A12*A11*A9r;
A012=A1*A2*A3*A4*A5*A6*A12*A9r;
A0p7=A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8*A7*A0r;
Aefec=A0p7;
%centros de masa referidos a sus sistemas
C01=A01*masa1cor;
C02=A02*masa2cor;
C03=A03*masa3cor;
C04=A04*masa1cor;
C05=A05*masa5cor;
C06=A06*masa6cor;
C07=A07*masa7cor;
C08=A08*masa1cor;
C09=A09*masa9cor;
C010=A010*masa10cor;
C011=A011*masa1cor;
C012=A012*masa6cor;
xm(1)=C01(1);
ym(1)=C01(2);
zm(1)=C01(3);
xm(2)=C02(1);
ym(2)=C02(2);
zm(2)=C02(3);
xm(3)=C03(1);
ym(3)=C03(2);
zm(3)=C03(3);
xm(4)=C04(1);
ym(4)=C04(2);
zm(4)=C04(3);
xm(5)=C05(1);
ym(5)=C05(2);
zm(5)=C05(3);
xm(6)=C06(1);
ym(6)=C06(2);
zm(6)=C06(3);
xm(7)=C07(1);
ym(7)=C07(2);
zm(7)=C07(3);
xm(8)=C08(1);
ym(8)=C08(2);
zm(8)=C08(3);
xm(9)=C09(1);
ym(9)=C09(2);
zm(9)=C09(3);
xm(10)=C010(1);
ym(10)=C010(2);
zm(10)=C010(3);
xm(11)=C011(1);
```

```
ym(11)=C011(2);
zm(11)=C011(3);
xm(12)=C012(1);
ym(12)=C012(2);
zm(12)=C012(3);
% suma de los centros de masas de los eslabones
sumamasdx=xm.*centrosmasa.masast;
sumamasdy=ym.*centrosmasa.masast;
sumamasdz=zm.*centrosmasa.masast;
xf=sum(sumamasdx)/sum(centrosmasa.masast);
yf=sum(sumamasdy)/sum(centrosmasa.masast);
zf=sum(sumamasdz)/sum(centrosmasa.masast);
% color de los centros de masas por eslabon y centro de masas final
colormasa=[0.5019 0 0.5019];
% datos de centros de masas
% coordenadas del efector final
coorefec=Aefec(:,4);
coorefec(4)=[];
coorefec=coorefec';
Cmt=[xf yf zf];
Cmtp=Cmt;
Cmtp(3)=0;
centrosmasa.xm=xm;
centrosmasa.ym=ym;
centrosmasa.zm=zm;
centrosmasa.origen=[-5.1650   45.1022  193.8835];
centrosmasa.sistemas=length(angulos);
centrosmasa.color=colormasa;
% validacion de la proyecion del centro de masas con respecto al pie de
% apoyo
% desfase=2.8;
% desfase2=0;
 desfase=4;
 desfase2=2;
 aux1=25;
 aux2=7;
areapiep_max=[(M00(1,4)+areapiep(1)),(M00(2,4)+areapiep(2)+desfase)];
areapiep_min=[(M00(1,4)-areapiep(1)+desfase2+aux1),(M00(2,4)-
areapiep(2)+desfase2+aux2)];
if(((Cmtp(1)>=areapiep_min(1))&&(Cmtp(1)<=areapiep_max(1)))&&((Cmtp(2)>=areapiep
_min(2))&&(Cmtp(2)<=areapiep_max(2)))&&(coorefec(3)>0))||(((coorefec(3)<=0.1)&&(
coorefec(3)>=-0.1))&&(sum(angulos(1:6))-angulos(6)~=0))
    movimiento=1;
else
    if(coorefec(3)<zmin)&&(coorefec(3)~=0)
      movimiento=2;
    else
       if(sum(angulos(1:6))==0)
           movimiento=1;
       else
  movimiento=0;
       end
    end
end
```

```
 else
  angulos=degtorad(angulos);
     movimiento=0;
 end
angulosf=round(radtodeg(angulos));
end
```

```
% Bioloid_Calculos_centroid.m
%
% esta funcion se encarga de generar el tipo de opcion de movimiento y los
% y el centro de masas del robot.
%
% las opciones de movimiento son los siguientes:
% 0 es un movimiento no valido
% 1 es un movimiento valido
% 2 es un movimiento transitorio
%
% angulos es el vector articular
%
% ejemplo
%
% simulacion nueve articulaciones
%
% angulos=[0,0,0,0,0,0,0,0,0];
% [Cmtp,movimiento]=Bioloid_Calculos(angulos)
%
% simulacion doce articulaciones
%
% angulos=[0,0,0,0,0,0,0,0,0,0,0,0];
% [Cmtp,movimiento]=Bioloid_Calculos(angulos)
%
% Autor Cristian David Villate Martinez
%
function [Cmtp,movimiento]=Bioloid_Calculos_centroid(angulos)
% bioloid calculos
% angulos a trabajar
dimen_a=length(angulos);
% obtencion de los tres grados articulares mediante las articulaciones
% introducidad (solo cuando son nueve grados articulares)
if(dimen_a==9)
    primer_a=angulos(1:6);
    segund_a=angulos(7:9);
    mov1_a=[angulos(1),angulos(3),angulos(4),angulos(7),angulos(8)];
    mov2_a=[angulos(2),angulos(5),angulos(9)];
```

```
    siete_a=-sum(mov1_a);
    ocho_a=-sum(mov2_a);
    doce_a=-angulos(6);
    angulos=[primer_a,siete_a,ocho_a,segund_a,doce_a];

end
% maximos y minimos grados articulares robot
 max_art=[23, 28, 135, 33, 28, 33, 99, 28, 0, 145, 28, 180];
 min_art=[-99, -28, 0, -145, -28, -180, -23, -28, -135, -33 -28 -33];
 if((sum(angulos>max_art)==0)&&(sum(angulos<min_art)==0))
% Conversión de grados a radianes
angulos=degtorad(angulos);
%longitudes
L1=34.5;
L2=75.07;
L4=14.50;
L3=77;
 L5=78;
% masas
masa1=0.1818;
masa2=0.10364;
masa3=0.02104;
masa4=masa1;
masa5=1.06599;
% masa5=2.400;
masa6=0.01082;
masa7=0.03089;
% distancia minima en z
zmin=9.50580878560151;
% coordenadas de los centro de masas
masa1cor=[-18.40; 7.94; 4.21;1];
masa2cor=[-9.03;6.87;63.90;1];
masa3cor=[13.48;0.03;31.64;1];
masa5cor=[1.64;30.844;79.52;1];%masa5cor=[1.64;40.59;79.52;1];
masa6cor=[-17;0;18.94;1];
masa7cor=[-4.7;0;-25.83;1];
masa9cor=[5.47;9.67;-11.10;1];
masa10cor=[-1.02;0.03;-43.36;1];
centrosmasa.masast=[masa1 masa2 masa3 masa4 masa5 masa6 masa7 masa1 masa2 masa3
masa4 masa6];
% area del pie validad
areapiep=[39,19.76];
% screws
s1d=[0 -1 0];
s01d=[0 0 L1];
s2d=[1 0 0];
s02d=[0 0 L1];
s3d=[0 -1 0];
s03d=[-L4 0 (L1+L2)];
s4d=[0 -1 0];
s04d=[0 0 (L1+L2+L3)];
s5d=[1 0 0];
s05d=[0 0 (L1+L2+L3)];
s6d=[0 0 1];
```

```
s06d=[0 0 0];
s1i=[0 -1 0];
s01i=[0 0 L1];
s2i=[1 0 0];
s02i=[0 L5 L1];
s3i=[0 -1 0];
s03i=[-L4 0 (L1+L2)];
s4i=[0 -1 0];
s04i=[0 0 (L1+L2+L3)];
s5i=[1 0 0];
s05i=[0 L5 (L1+L2+L3)];
s6i=[0 0 1];
s06i=[0 L5 0];
%homogeneas
 A1=mat_tras_screw(s1d,s01d,(angulos(1)),0);
A2=mat_tras_screw(s2d,s02d,angulos(2),0);
A3=mat_tras_screw(s3d,s03d,angulos(3),0);
A4=mat_tras_screw(s4d,s04d,angulos(4),0);
A5=mat_tras_screw(s5d,s05d,angulos(5),0);
A6=mat_tras_screw(s6d,s06d,angulos(6),0);
A7=mat_tras_screw(s1i,s01i,angulos(7),0);
A8=mat_tras_screw(s2i,s02i,angulos(8),0);
A9=mat_tras_screw(s3i,s03i,angulos(9),0);
A10=mat_tras_screw(s4i,s04i,angulos(10),0);
A11=mat_tras_screw(s5i,s05i,angulos(11),0);
A12=mat_tras_screw(s6i,s06i,angulos(12),0);
% Posicion de referencia
q1d=[0; 0; L1;1];
q2dp=[-L4; 0; L2+L1;1];
q3d=[0;0;L3+L2+L1;1];
q1i=[0;L5;L1;1];
q2ip=[-L4; L5; L2+L1;1];
q3i=[0;L5;L3+L2+L1;1];
q0i=[0;L5;0;1];
% MTH Posición y orientación de referencia (arbitrarios)
A2r = [ eye(3), q1d(1:3,1) ; [0 0 0 1]];
A4r = [ eye(3), q2dp(1:3,1) ; [0 0 0 1]];
A5r = [ eye(3), q3d(1:3,1) ; [0 0 0 1]];
A6r = [ eye(3), q1i(1:3,1) ; [0 0 0 1]];
A8r = [ eye(3), q2ip(1:3,1) ; [0 0 0 1]];
A9r = [ eye(3), q3i(1:3,1) ; [0 0 0 1]];
A0r = [ eye(3), q0i(1:3,1) ; [0 0 0 1]];
% MTH Correspondiente a cada eslabon referida a la base
A01 = A2*A2r;
A02=A1*A2*A2r;
A03=A1*A2*A3*A4r;
A04=A1*A2*A3*A4*A5r;
A05=A1*A2*A3*A4*A5*A5r;
A06=A1*A2*A3*A4*A5*A6*A5r;
A07=A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A8*A6r;
A08=A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A6r;
A09=A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8r;
A010=A1*A2*A3*A4*A5*A6*A12*A11*A10*A9r;
A011=A1*A2*A3*A4*A5*A6*A12*A11*A9r;
```

```
A012=A1*A2*A3*A4*A5*A6*A12*A9r;
A0p7=A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8*A7*A0r;
Aefec=A0p7;
%centros de masa referidos a sus sistemas
C01=A01*masa1cor;
C02=A02*masa2cor;
C03=A03*masa3cor;
C04=A04*masa1cor;
C05=A05*masa5cor;
C06=A06*masa6cor;
C07=A07*masa7cor;
C08=A08*masa1cor;
C09=A09*masa9cor;
C010=A010*masa10cor;
C011=A011*masa1cor;
C012=A012*masa6cor;
xm(1)=C01(1);
ym(1)=C01(2);
zm(1)=C01(3);
xm(2)=C02(1);
ym(2)=C02(2);
zm(2)=C02(3);
xm(3)=C03(1);
ym(3)=C03(2);
zm(3)=C03(3);
xm(4)=C04(1);
ym(4)=C04(2);
zm(4)=C04(3);
xm(5)=C05(1);
ym(5)=C05(2);
zm(5)=C05(3);
xm(6)=C06(1);
ym(6)=C06(2);
zm(6)=C06(3);
xm(7)=C07(1);
ym(7)=C07(2);
zm(7)=C07(3);
xm(8)=C08(1);
ym(8)=C08(2);
zm(8)=C08(3);
xm(9)=C09(1);
ym(9)=C09(2);
zm(9)=C09(3);
xm(10)=C010(1);
ym(10)=C010(2);
zm(10)=C010(3);
xm(11)=C011(1);
ym(11)=C011(2);
zm(11)=C011(3);
xm(12)=C012(1);
ym(12)=C012(2);
zm(12)=C012(3);
% suma de los centros de masas de los eslabones
sumamasdx=xm.*centrosmasa.masast;
```

```
sumamasdy=ym.*centrosmasa.masast;
sumamasdz=zm.*centrosmasa.masast;
xf=sum(sumamasdx)/sum(centrosmasa.masast);
yf=sum(sumamasdy)/sum(centrosmasa.masast);
zf=sum(sumamasdz)/sum(centrosmasa.masast);
% color de los centros de masas por eslabon y centro de masas final
colormasa=[0.5019 0 0.5019];
% datos de centros de masas
% coordenadas del efector final
coorefec=Aefec(:,4);
coorefec(4)=[];
coorefec=coorefec';
Cmt=[xf yf zf];
Cmtp=Cmt;
Cmtp(3)=0;
centrosmasa.xm=xm;
centrosmasa.ym=ym;
centrosmasa.zm=zm;
centrosmasa.origen=[-5.1650   45.1022  193.8835];
centrosmasa.sistemas=length(angulos);
centrosmasa.color=colormasa;
% validacion de la proyecion del centro de masas con respecto al pie de
% apoyo
if(((Cmtp(1)>=-areapiep(1))&&(Cmtp(1)<=areapiep(1)))&&((Cmtp(2)>=-
areapiep(2))&&(Cmtp(2)<=areapiep(2)))&&(coorefec(3)>0))||((coorefec(3)==0)&&(sum
(angulos(1:6))-angulos(6)~=0))
    movimiento=1;
else
    if(coorefec(3)<zmin)&&(coorefec(3)~=0)
      movimiento=2;
    else
       if(sum(angulos(1:6))==0)
           movimiento=1;
       else
  movimiento=0;
       end
    end
end
 else
  angulos=degtorad(angulos);
     movimiento=0;
 end
angulosf=round(radtodeg(angulos));
end




% bioloid graficas
% esta funcion se encarga de graficar el robot bioloid cuando se realiza
% el paso con el pie izquierdo
% los datos de ingreso son:
% angulos el vector que contiene los angulos que posee el robot
% M00 es la matriz de trasformacion homogenea que representa la posicion de
```

```
% origen de la base del robot
% bioloid es la estructura que contiene todas las piezas digitalizadas del
% robot bioloid
% Bioloid_graficas_izquierdo(angulos,M00,bioloid)
% Autor: Cristian David Villate Martinez

function  Bioloid_graficas_izquierdo(angulos,M00,bioloid)
% angulos a trabajar
%  angulos=[-12 0 0 -1 0 0 -47 46 0];
%angulos=[-14 0 47 -46 0 0 0 1 0];
%   angulos=[0 -8 0 0 0 0 0 0 0];
dimen_a=length(angulos);
if(dimen_a==9)
    primer_a=angulos(1:6);
    segund_a=angulos(7:9);
    mov1_a=[angulos(1),angulos(3),angulos(4),angulos(7),angulos(8)];
    mov2_a=[angulos(2),angulos(5),angulos(9)];
    siete_a=-sum(mov1_a);
    ocho_a=-sum(mov2_a);
    doce_a=-angulos(6);
    angulos=[primer_a,siete_a,ocho_a,segund_a,doce_a];

end
%longitudes
angulos=degtorad(angulos);
pie=50;
L1=34.5;
L2=75.07;
L4=14.50;
L3=77;
pie6=10.3;
cadera1=30.4;
L5=78;
distancia_tronco=137.44;
brazo_uno=90.50;
brazo_dos=106.60;
cuello=35.11;
masa1=0.1818;
masa2=0.10364;
masa3=0.02104;
masa4=masa1;
masa5=1.06599;
% masa5=2.400;
masa6=0.01082;
masa7=0.03089;
masa1cor=[-18.40; 7.94; 4.21;1];
masa2cor=[-9.03;6.87;63.90;1];
masa3cor=[13.48;0.03;31.64;1];
% masa5cor=[1.64;30.844;79.52;1];%masa5cor=[1.64;40.59;79.52;1];
masa5cor=[1.64;-47.1560;79.52;1];
masa6cor=[-17;0;18.94;1];
masa7cor=[-4.7;0;-25.83;1];
masa9cor=[5.47;9.67;-11.10;1];
masa10cor=[-1.02;0.03;-43.36;1];
```

```
centrosmasa.masast=[masa1 masa2 masa3 masa4 masa5 masa6 masa7 masa1 masa2 masa3
masa4 masa6];
areapiep=[39,19.76];
% screws
s1d=[0 -1 0];
s01d=[0 0 L1];
s2d=[1 0 0];
s02d=[0 0 L1];
s3d=[0 -1 0];
s03d=[-L4 0 (L1+L2)];
s4d=[0 -1 0];
s04d=[0 0 (L1+L2+L3)];
s5d=[1 0 0];
s05d=[0 0 (L1+L2+L3)];
s6d=[0 0 1];
s06d=[0 0 0];
s1i=[0 -1 0];
s01i=[0 0 L1];
s2i=[1 0 0];
s02i=[0 L5 L1];
s3i=[0 -1 0];
s03i=[-L4 0 (L1+L2)];
s4i=[0 -1 0];
s04i=[0 0 (L1+L2+L3)];
s5i=[1 0 0];
s05i=[0 L5 (L1+L2+L3)];
s6i=[0 0 1];
s06i=[0 L5 0];
%homogeneas
A1=mat_tras_screw(s1d,s01i,(angulos(1)),0);
A2=mat_tras_screw(s2d,s02i,angulos(2),0);
A3=mat_tras_screw(s3d,s03i,angulos(3),0);
A4=mat_tras_screw(s4d,s04i,angulos(4),0);
A5=mat_tras_screw(s5d,s05i,angulos(5),0);
A6=mat_tras_screw(s6d,s06i,angulos(6),0);
A7=mat_tras_screw(s1i,s01d,angulos(7),0);
A8=mat_tras_screw(s2i,s02d,angulos(8),0);
A9=mat_tras_screw(s3i,s03d,angulos(9),0);
A10=mat_tras_screw(s4i,s04d,angulos(10),0);
A11=mat_tras_screw(s5i,s05d,angulos(11),0);
A12=mat_tras_screw(s6i,s06d,angulos(12),0);
%medidas
% M00=eye(4);
%  M00=[1 0 -2.22044604925031e-16 57.3018246119400;0 1 0 78;2.22044604925031e-16
0 1 -0.0759617536599628;0 0 0 1];
% M00=[1 0 0 0;0 1 0 L5;0 0 1 0;0 0 0 1];
desfase=MTHtrasx(M00(1,4));
color=[0.5 0.5 0.5];
q0d=[0; 0; 0;1];
q1d=[0; 0; L1;1];
q2d=[0; 0; L2+L1;1];
q2dp=[-L4; 0; L2+L1;1];
q3d=[0;0;L3+L2+L1;1];
q4d=[0;0;(L3+L2+L1+cadera1);1];
```

```
q0i=[0;L5;0;1];
% q0ip=[pie;L5;0;1];
% q0ip1=[-pie;L5;0;1];
q0ip=[pie;0;0;1];
q0ip1=[-pie;0;0;1];
q1i=[0;L5;L1;1];
q2i=[0; L5; L2+L1;1];
q2ip=[-L4; L5; L2+L1;1];
q3i=[0;L5;L3+L2+L1;1];
q4i=[0;L5;(L3+L2+L1+cadera1);1];
qtron=[0;0;distancia_tronco+L3+L2+L1;1];
qtron_2=[0;L5;distancia_tronco+L3+L2+L1;1];
qbrazo_uno=[0;-brazo_uno;distancia_tronco+L3+L2+L1;1];
q2brazo_uno=[0;L5+brazo_uno;distancia_tronco+L3+L2+L1;1];
qbrazo_dos=[0;-brazo_uno;distancia_tronco+L3+L2+L1-brazo_dos;1];
q2brazo_dos=[0;L5+brazo_uno;distancia_tronco+L3+L2+L1-brazo_dos;1];
qcuello=[0;(L5/2);distancia_tronco+L3+L2+L1+cuello;1];
qmedio=[0;(L5/2);distancia_tronco+L3+L2+L1;1];
A1r = [ eye(3), q0i(1:3,1) ; [0 0 0 1]];
A2r = [ eye(3), q1i(1:3,1) ; [0 0 0 1]];
A3r = [ eye(3), q2i(1:3,1) ; [0 0 0 1]];
A4r = [ eye(3), q2ip(1:3,1) ; [0 0 0 1]];
A5r = [ eye(3), q3i(1:3,1) ; [0 0 0 1]];
A6r = [ eye(3), q1d(1:3,1) ; [0 0 0 1]];
A7r = [ eye(3), q2d(1:3,1) ; [0 0 0 1]];
A8r = [ eye(3), q2dp(1:3,1) ; [0 0 0 1]];
A9r = [ eye(3), q3d(1:3,1) ; [0 0 0 1]];
A0r = [ eye(3), q0d(1:3,1) ; [0 0 0 1]];
A0rp = [ eye(3), q0ip(1:3,1) ; [0 0 0 1]];
A0rp1 = [ eye(3), q0ip1(1:3,1) ; [0 0 0 1]];
A0rtron = [ eye(3),qtron(1:3,1) ; [0 0 0 1]];
A0rtron_2 = [ eye(3),qtron_2(1:3,1) ; [0 0 0 1]];
A0brazo_uno = [ eye(3),qbrazo_uno(1:3,1) ; [0 0 0 1]];
A0brazo_uno2 = [ eye(3),q2brazo_uno(1:3,1) ; [0 0 0 1]];
A0brazo_dos = [ eye(3),qbrazo_dos(1:3,1) ; [0 0 0 1]];
A0brazo_dos2 = [ eye(3),q2brazo_dos(1:3,1) ; [0 0 0 1]];
A0cuello = [ eye(3),qcuello(1:3,1) ; [0 0 0 1]];
A0medio = [ eye(3),qmedio(1:3,1) ; [0 0 0 1]];
A01 =desfase*A2*A2r;
A02=desfase*A1*A2*A2r;
A03=desfase*A1*A2*A3*A4r;
A04=desfase*A1*A2*A3*A4*A5r;
A05=desfase*A1*A2*A3*A4*A5*A5r;
A06=desfase*A1*A2*A3*A4*A5*A6*A5r;
A07=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A8*A6r;
A08=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A6r;
A09=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8r;
A010=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9r;
A011=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A9r;
A012=desfase*A1*A2*A3*A4*A5*A6*A12*A9r;
A0p7=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8*A7*A0r;
Aefec=A0p7;
A0pp=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8*A7*A0rp;
A0pp1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8*A7*A0rp1;
```

```
A05tronf=desfase*A1*A2*A3*A4*A5*A6*A0rtron;
A010tronf=desfase*A1*A2*A3*A4*A5*A6*A0rtron_2;
A05brazo_unof=desfase*A1*A2*A3*A4*A5*A6*A0brazo_uno;
A010brazo_uno2f=desfase*A1*A2*A3*A4*A5*A6*A0brazo_uno2;
A05brazo_dosf=desfase*A1*A2*A3*A4*A5*A6*A0brazo_dos;
A010brazo_dos2f=desfase*A1*A2*A3*A4*A5*A6*A0brazo_dos2;
A010cuello=desfase*A1*A2*A3*A4*A5*A6*A0cuello;
A010medio=desfase*A1*A2*A3*A4*A5*A6*A0medio;
A03mod=desfase*A1*A2*A3r;
A04mod=desfase*A1*A2*A3*A3r;
A09mod=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A7r ;
A07mod=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7r;
coorefec=Aefec(:,4);
coorefec(4)=[];
coorefec=coorefec';
%centros de masa referidos a sus sistemas
C01=A01*masa1cor;
C02=A02*masa2cor;
C03=A03*masa3cor;
C04=A04*masa1cor;
C05=A05*masa5cor;
C06=A06*masa6cor;
C07=A07*masa7cor;
C08=A08*masa1cor;
C09=A09*masa9cor;
C010=A010*masa10cor;
C011=A011*masa1cor;
C012=A012*masa6cor;
xm(1)=C01(1);
ym(1)=C01(2);
zm(1)=C01(3);
xm(2)=C02(1);
ym(2)=C02(2);
zm(2)=C02(3);
xm(3)=C03(1);
ym(3)=C03(2);
zm(3)=C03(3);
xm(4)=C04(1);
ym(4)=C04(2);
zm(4)=C04(3);
xm(5)=C05(1);
ym(5)=C05(2);
zm(5)=C05(3);
xm(6)=C06(1);
ym(6)=C06(2);
zm(6)=C06(3);
xm(7)=C07(1);
ym(7)=C07(2);
zm(7)=C07(3);
xm(8)=C08(1);
ym(8)=C08(2);
zm(8)=C08(3);
xm(9)=C09(1);
ym(9)=C09(2);
```

```
zm(9)=C09(3);
xm(10)=C010(1);
ym(10)=C010(2);
zm(10)=C010(3);
xm(11)=C011(1);
ym(11)=C011(2);
zm(11)=C011(3);
xm(12)=C012(1);
ym(12)=C012(2);
zm(12)=C012(3);
sumamasdx=xm.*centrosmasa.masast;
sumamasdy=ym.*centrosmasa.masast;
sumamasdz=zm.*centrosmasa.masast;
xf=sum(sumamasdx)/sum(centrosmasa.masast);
yf=sum(sumamasdy)/sum(centrosmasa.masast);
zf=sum(sumamasdz)/sum(centrosmasa.masast);
centromasat.xm=xf;
centromasat.ym=yf;
centromasat.zm=zf;
centromasat.sistemas=1;
centromasat.color=[0 0 1];
colormasa=[0.5019 0 0.5019];
Cmt=[xf yf zf];
Cmtp=Cmt;
Cmtp(3)=0;
centrosmasa.xm=xm;
centrosmasa.ym=ym;
centrosmasa.zm=zm;
centrosmasa.origen=[-5.1650   45.1022  193.8835];
centrosmasa.sistemas=length(angulos);
centrosmasa.color=colormasa;
areapiep_max=[(M00(1,4)+areapiep(1)),(M00(2,4)+areapiep(2))];
areapiep_min=[(M00(1,4)-areapiep(1)),(M00(2,4)-areapiep(2))];
if(((Cmtp(1)>=areapiep_min(1))&&(Cmtp(1)<=areapiep_max(1)))&&((Cmtp(2)>=areapiep
_min(2))&&(Cmtp(2)<=areapiep_max(2)))&&(coorefec(3)>0))||((coorefec(3)<=0.1)&&(c
oorefec(3)>=-0.1))
    colorlineavalido=[0 1 0];
else
   colorlineavalido=[1 1 0];
end
opcion=2;
if(opcion==1 )
punto_ini=[M00(1,4);M00(2,4);M00(3,4)];
p0=[0 0 0];
p01=[pie 0 0];
p01=p01'+punto_ini;
p001=[-pie 0 0];
p02=punto_ini;
p02(1)=p02(1)-pie;
% p02=[7.3018;78.0000;-0.0760];
p03=[0 0 L1];
ppie6=[0; L4; L3;1];
ppie7=[0; 0; 0;1];
dis0=A01(:,4);
```

```
dis0(4)=[];
dis1=A02(:,4);
dis1(4)=[];
dis2=A03(:,4);
dis2(4)=[];
dis3=A04(:,4);
dis3(4)=[];
dis4=A05(:,4);
dis4(4)=[];
dis5=A06(:,4);
dis5(4)=[];
dis6=A07(:,4);
dis6(4)=[];
dis7=A08(:,4);
dis7(4)=[];
dis8=A09(:,4);
dis8(4)=[];
dis9=A010(:,4);
dis9(4)=[];
dis10=A011(:,4);
dis10(4)=[];
dis11=A012(:,4);
dis11(4)=[];
p04=A0p7(:,4);
p04(4)=[];
p05=A0pp(:,4);
p05(4)=[];
p051=A0pp1(:,4);
p051(4)=[];
dismed_cuer=A05tronf(:,4);
dismed_cuer(4)=[];
dismed_cuer2=A010tronf(:,4);
dismed_cuer2(4)=[];
dismed_brazo1=A05brazo_unof(:,4);
dismed_brazo1(4)=[];
dismed_brazo1_2=A010brazo_uno2f(:,4);
dismed_brazo1_2(4)=[];
dismed_brazo2=A05brazo_dosf(:,4);
dismed_brazo2(4)=[];
dismed_brazo2_2=A010brazo_dos2f(:,4);
dismed_brazo2_2(4)=[];
dismed_cuello=A010cuello(:,4);
dismed_cuello(4)=[];
dismed_medio=A010medio(:,4);
dismed_medio(4)=[];
dismed_03=A03mod(:,4);
dismed_03(4)=[];
dismed_04=A04mod(:,4);
dismed_04(4)=[];
dismed_09=A09mod(:,4);
dismed_09(4)=[];
dismed_07=A07mod(:,4);
dismed_07(4)=[];
dibujar_linea( p01,punto_ini, color )
```

```
hold on
 dibujar_linea(p02,punto_ini, color )
dibujar_linea( punto_ini,dis0, color )
dibujar_linea(dis0,dis1, color )
dibujar_linea(dis1,dismed_03, color )
dibujar_linea(dismed_03,dis2, color )
dibujar_linea(dis2,dismed_04, color )
dibujar_linea(dismed_04,dis3, color )
dibujar_linea(dis3,dis4, color )
dibujar_linea(dis4,dis5, color )
dibujar_linea(dis6,dis7, color )
dibujar_linea(dis7,dismed_07, color )
dibujar_linea(dismed_07,dis8, color )
dibujar_linea(dis8,dismed_09, color )
dibujar_linea(dismed_09,dis9, color )
dibujar_linea(dis9,dis10, color )
dibujar_linea(dis10,dis11, color )
dibujar_linea(dis11,dis5, color )
dibujar_linea(dis6,p04, color )
dibujar_linea(p04,p05, color )
dibujar_linea(p04,p051, color )
dibujar_linea(Cmtp,Cmt,colorlineavalido )
dibujar_linea(dis5,dismed_cuer2, color )
dibujar_linea(dis11,dismed_cuer, color )
dibujar_linea(dismed_cuer,dismed_cuer2, color )
dibujar_linea(dismed_cuer, dismed_brazo1, color )
dibujar_linea(dismed_cuer2,dismed_brazo1_2, color )
dibujar_linea(dismed_brazo1, dismed_brazo2, color )
dibujar_linea(dismed_brazo1_2, dismed_brazo2_2, color )
dibujar_linea(dismed_brazo1_2, dismed_brazo2_2, color )
dibujar_linea(dismed_medio, dismed_cuello, color )
plot3(dismed_cuello(1),dismed_cuello(2),dismed_cuello(3),'o', 'Color',color
,'LineWidth',25);
ejes(M00,20,'0')
ejes(A01,30,'1')
ejes(A02,20,'2')
ejes(A03,20,'3')
ejes(A04,10,'4')
ejes(A05,20,'5')
ejes(A06,30,'6')
ejes(A07,30,'7')
ejes(A08,20,'8')
ejes(A09,30,'9')
ejes(A010,10,'10')
ejes(A011,20,'11')
ejes(A012,30,'12')
ejes(Aefec,30,'F')
graf_centros_masa(centrosmasa)
graf_centros_masa(centromasat)
titulo1=('coordenadas de centro de masa: ');
titulo3=('coordenadas efector final: ');
titulo2=num2str(Cmt);
titulo4=num2str(coorefec);
tituloes=('  ');
```

```
titulo=[titulo1 titulo2  tituloes titulo3 titulo4];
title(titulo)
grid on
axis ([-200,300,-150,250,0,410]);
view(56,23)
end
if(opcion==2)
pie_z=8.50;
pie2_z=-34.5;
pie2_x=-15;
rotula_z=14.50;
rotula_x=21.1;
dinamixel1x=2;
dinamixel2x=32;
dinamixel2z=14.5;
dinamixel1y=20.215;
piezita1x=38;
piezita1x2=8;
piezita1y=17;
peronez=53.7;
piezita_piernaz=2.3;
piezita_brazo_z=4.3;
dinamixelz3=16.10;
rodillaz=23;
mariposa=3.84;
musloz=29;
distanciarodillax=14;
color2=[0 0 0];
q1d1=[rotula_x; 0; (L1+rotula_z);1];
q1d2=[dinamixel1x; 0; L1;1];
q1d3=[(-piezita1x-piezita1x2);-piezita1y; L1+dinamixel2z;1];
q1d4=[-dinamixel2x; 0; (L1);1];
q1d5=[(-dinamixel2x-piezita1y); 0; (L1);1];
q1d6=[dinamixel1x;dinamixel1y; L1;1];
q1d7=[(-piezita1x-piezita1x2);+piezita1y; L1+dinamixel2z;1];
q2d1=[0; 0; L1+peronez;1];
q2d2=[0; 0; (L1+peronez+piezita_piernaz);1];
q2d3=[-L4; 0; (L1+peronez+piezita_piernaz+piezita_brazo_z+dinamixelz3);1];
q3d1=[-L4;dinamixel1y ; L2+L1;1];
q3d2=[0;0 ; L2+L1+rodillaz+peronez;1];
q4d1=[0;dinamixel1y;L3+L2+L1;1];
q4d2=[rotula_x; 0; (L1+L3+L2-rotula_z);1];
q4d3=[(-dinamixel2x-piezita1y-mariposa);0; L1+L3+L2;1];
q4d4=[-dinamixel2x; 0; (L1+L3+L2);1];
q4d5=[(-piezita1x-piezita1x2);-piezita1y; L1-dinamixel2z+L3+L2;1];
q4d6=[(-piezita1x-piezita1x2);piezita1y; L1-dinamixel2z+L3+L2;1];
q5d1=[(pie2_x);0; L1+L3+L2;1];
q6d1=[0;0;( L1+L3+L2+(dinamixel2z*2)+mariposa);1];
q6d2=[0;0;( L1+L3+L2+(dinamixel2z*2)+(piezita1x/2));1];
q12i1=[0;L5;( L1+L3+L2+(dinamixel2z*2)+(piezita1x/2));1];
q12i2=[0;L5;( L1+L3+L2+(dinamixel2z*2)+mariposa);1];
q12i3=[(pie2_x);L5; L1+L3+L2;1];
q11i1=[(-piezita1x-piezita1x2);-piezita1y+L5; L1-dinamixel2z+L3+L2;1];
q11i2=[(-piezita1x-piezita1x2);piezita1y+L5; L1-dinamixel2z+L3+L2;1];
```

```
q11i3=[-dinamixel2x; L5; (L1+L3+L2);1];
q11i4=[(-dinamixel2x-piezita1y-mariposa);L5; L1+L3+L2;1];
q11i5=[rotula_x; L5; (L1+L3+L2-rotula_z);1];
q11i6=[0;dinamixel1y+L5;L3+L2+L1;1];
q10i1=[0;L5 ; L2+L1+rodillaz+peronez;1];
q10i2=[-distanciarodillax;L5;L2+L1;1];
q9i1=[-L4;dinamixel1y+L5 ; L2+L1;1];
q8i1=[-L4;L5; (L1+peronez+piezita_piernaz+piezita_brazo_z+dinamixelz3);1];
q8i2=[0; L5; (L1+peronez+piezita_piernaz);1];
q8i3=[0; L5; L1+peronez;1];
q8i4=[0; L5; L1;1];
q7i0=[dinamixel1x;L5; L1;1];
q7i1=[dinamixel1x;dinamixel1y+L5; L1;1];
q7i2=[(-dinamixel2x-piezita1y);L5; (L1);1];
q7i3=[(-piezita1x-piezita1x2);(-piezita1y+L5); L1+dinamixel2z;1];
q7i4=[(-piezita1x-piezita1x2);(piezita1y+L5); L1+dinamixel2z;1];
q7i5=[-dinamixel2x;L5; L1;1];
q7i6=[rotula_x; L5; (L1+rotula_z);1];
qefe1=[-dinamixel2z;0;L1];
qefe2=[0;0;pie_z];
qtorso=[-0;48.87;L3+L2+L1+90.39;1];
q10i2mod=[-distanciarodillax;0;L2+L1;1];
q8i4mod=[0; 0; L1;1];
A2r1 = [ eye(3), q7i6(1:3,1) ; [0 0 0 1]];
A01p1 =desfase*A2*A2r1;
A2r2 = [ eye(3), q7i0(1:3,1) ; [0 0 0 1]];
A01p2 = desfase*A2*A2r2;
A2r3 = [ eye(3), q7i3(1:3,1) ; [0 0 0 1]];
A01p3 = desfase*A2*A2r3;
A2r4 = [ eye(3), q7i5(1:3,1) ; [0 0 0 1]];
A01p4 = desfase*A2*A2r4;
A2r5 = [ eye(3), q7i2(1:3,1) ; [0 0 0 1]];
A01p5 = desfase*A2*A2r5;
A2r6 = [ eye(3), q7i1(1:3,1) ; [0 0 0 1]];
A01p6 = desfase*A2*A2r6;
A2r7 = [ eye(3), q7i4(1:3,1) ; [0 0 0 1]];
A01p7 = desfase*A2*A2r7;
A3r1 = [ eye(3), q8i3(1:3,1) ; [0 0 0 1]];
A02p1=desfase*A1*A2*A3r1;
A3r2 = [ eye(3), q8i2(1:3,1) ; [0 0 0 1]];
A02p2=desfase*A1*A2*A3r2;
A3r3 = [ eye(3), q8i1(1:3,1) ; [0 0 0 1]];
A02p3=desfase*A1*A2*A3r3;
A4r1 = [ eye(3), q9i1(1:3,1) ; [0 0 0 1]];
A03p1=desfase*A1*A2*A3*A4r1;
A4r2 = [ eye(3), q10i1(1:3,1) ; [0 0 0 1]];
A03p2=desfase*A1*A2*A3*A4r2;
A5r1 = [ eye(3), q11i6(1:3,1) ; [0 0 0 1]];
A04p1=desfase*A1*A2*A3*A4*A5r1;
A5r2 = [ eye(3), q11i5(1:3,1) ; [0 0 0 1]];
A04p2=desfase*A1*A2*A3*A4*A5r2;
A5r3 = [ eye(3), q11i4(1:3,1) ; [0 0 0 1]];
A04p3=desfase*A1*A2*A3*A4*A5r3;
A5r4 = [ eye(3), q11i3(1:3,1) ; [0 0 0 1]];
```

```
A04p4=desfase*A1*A2*A3*A4*A5r4;
A5r5 = [ eye(3), q11i1(1:3,1) ; [0 0 0 1]];
A04p5=desfase*A1*A2*A3*A4*A5r5;
A5r6= [ eye(3), q11i2(1:3,1) ; [0 0 0 1]];
A04p6=desfase*A1*A2*A3*A4*A5r6;
A6r1= [ eye(3), q12i3(1:3,1) ; [0 0 0 1]];
A05p1=desfase*A1*A2*A3*A4*A5*A6r1;
A7r1= [ eye(3), q12i2(1:3,1) ; [0 0 0 1]];
A06p1=desfase*A1*A2*A3*A4*A5*A6*A7r1;
A7r2= [ eye(3), q6d2(1:3,1) ; [0 0 0 1]];
A06p2=desfase*A1*A2*A3*A4*A5*A6*A7r2;
A13r1= [ eye(3), q12i1(1:3,1) ; [0 0 0 1]];
A012p1=desfase*A1*A2*A3*A4*A5*A6*A13r1;
A13r2= [ eye(3), q6d1(1:3,1) ; [0 0 0 1]];
A012p2=desfase*A1*A2*A3*A4*A5*A6*A12*A13r2;
A13r3= [ eye(3), q5d1(1:3,1) ; [0 0 0 1]];
A012p3=desfase*A1*A2*A3*A4*A5*A6*A12*A13r3;
A11r1= [ eye(3), q4d5(1:3,1) ; [0 0 0 1]];
A011p1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A11r1;
A11r2= [ eye(3), q4d6(1:3,1) ; [0 0 0 1]];
A011p2=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A11r2;
A11r3= [ eye(3), q4d4(1:3,1) ; [0 0 0 1]];
A011p3=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A11r3;
A11r4= [ eye(3), q4d3(1:3,1) ; [0 0 0 1]];
A011p4=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A11r4;
A11r5= [ eye(3), q4d2(1:3,1) ; [0 0 0 1]];
A011p5=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A11r5;
A11r6= [ eye(3), q4d1(1:3,1) ; [0 0 0 1]];
A011p6=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A11r6;
A10r1= [ eye(3), q3d2(1:3,1) ; [0 0 0 1]];
A010p1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A10r1;
A10r2= [ eye(3), q10i2mod(1:3,1) ; [0 0 0 1]];
A010p2=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A10r2;
A9r1= [ eye(3), q3d1(1:3,1) ; [0 0 0 1]];
A09p1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A9r1;
A8r1= [ eye(3), q2d3(1:3,1) ; [0 0 0 1]];
A08p1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8r1;
A8r2= [ eye(3), q2d2(1:3,1) ; [0 0 0 1]];
A08p2=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8r2;
A8r3= [ eye(3), q2d1(1:3,1) ; [0 0 0 1]];
A08p3=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8r3;
A8r4= [ eye(3), q8i4mod(1:3,1) ; [0 0 0 1]];
A08p4=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8r4;
A72r1= [ eye(3), q1d6(1:3,1) ; [0 0 0 1]];
A07p1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A72r1;
A72r2= [ eye(3), q1d5(1:3,1) ; [0 0 0 1]];
A07p2=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A72r2;
A72r3= [ eye(3), q1d3(1:3,1) ; [0 0 0 1]];
A07p3=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A72r3;
A72r4= [ eye(3), q1d7(1:3,1) ; [0 0 0 1]];
A07p4=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A72r4;
A72r5= [ eye(3), q1d4(1:3,1) ; [0 0 0 1]];
A07p5=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A72r5;
A72r6= [ eye(3), q1d1(1:3,1) ; [0 0 0 1]];
```

```
A07p6=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A72r6;
Aefe1= [ eye(3), qefe1(1:3,1) ; [0 0 0 1]];
A0efe1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A8*Aefe1;
Aefe2= [ eye(3), qefe2(1:3,1) ; [0 0 0 1]];
A0efe2=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A8*Aefe2;
Atorso= [ eye(3), qtorso(1:3,1) ; [0 0 0 1]];
Atorsof=desfase*A1*A2*A3*A4*A5*A6*Atorso;
dibujar_sistema_referencia_MTH(M00, 50, 3, '0');
dibujar_sistema_referencia_MTH(A01, 50, 3, '1');
dibujar_sistema_referencia_MTH(A02, 40, 3, '2');
dibujar_sistema_referencia_MTH(A03, 50, 3, '3');
dibujar_sistema_referencia_MTH(A04, 40, 3, '4');
dibujar_sistema_referencia_MTH(A05, 50, 3, '5');
dibujar_sistema_referencia_MTH(A06, 60, 3, '6');
dibujar_sistema_referencia_MTH(A07, 50, 3, '7');
dibujar_sistema_referencia_MTH(A08, 40, 3, '8');
dibujar_sistema_referencia_MTH(A09, 50, 3, '9');
dibujar_sistema_referencia_MTH(A010, 40, 3, '10');
dibujar_sistema_referencia_MTH(A011, 50, 3, '11');
dibujar_sistema_referencia_MTH(A012,60, 3, '12');
dibujar_sistema_referencia_MTH(Aefec,60, 3, 'F');
hold on
graf_centros_masa(centrosmasa)
graf_centros_masa(centromasat)
dibujar_linea(Cmtp,Cmt,colorlineavalido )
A0cad = MTHRotx(degtorad(-90));
A0cad1= MTHRotz(degtorad(180));
A0cad6= MTHtrasz(pie_z);
Rot0=M00*A0cad;
Rot0=A0cad6*Rot0;
dibujar_objeto_matlab_from_stl(bioloid.planta_pie_derecho,Rot0);
A0cad2 = MTHRotx(degtorad(180));
A0cad3= MTHRotz(degtorad(90));
A0cad4= MTHtrasz(-pie2_z);
A0cad5= MTHtrasx(pie2_x);
Rot0=M00*A0cad2*A0cad3;
Rot0=A0cad4*Rot0;
Rot0=A0cad5*Rot0;
dibujar_objeto_matlab_from_stl(bioloid.planta_pie1_derecho,Rot0);
A1cad = MTHRotx(degtorad(90));
A1cad1 = MTHRotz(degtorad(90));
A1cad2 = MTHRoty(degtorad(180));
Rot1=A01p1*A1cad*A1cad1*A1cad2;
dibujar_objeto_matlab_from_stl(bioloid.rotula_derecho,Rot1);
A1cad3 = MTHRotx(degtorad(-90));
A1cad4 = MTHRotz(degtorad(180));
Rot1=A01p2*A1cad3;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot1);
A1cad5 = MTHRoty(degtorad(180));
A1cad9 = MTHRotx(degtorad(180));
Rot1=A01p3*A1cad5;
Rot11=A01p7*A1cad5*A1cad9;
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot1);
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot11);
```

```
A1cad6 = MTHRoty(degtorad(90));
A1cad7 = MTHRotx(degtorad(180));
A1cad8 = MTHRotz(degtorad(180));
Rot1=A01p4*A1cad3*A1cad6*A1cad7*A1cad8;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot1);
A1cad8 = MTHRoty(degtorad(90));
Rot1=A01p5*A1cad3*A1cad8;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot1);
Rot1=A01p6*A1cad3;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot1);
A2cad0 = MTHRotz(degtorad(90));
Rot2=A02*A2cad0;
dibujar_objeto_matlab_from_stl(bioloid.perone,Rot2);
A2cad1 = MTHRotx(degtorad(90));
Rot2=A02p1*A2cad0*A2cad1;
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna2,Rot2);
Rot2=A02p2*A2cad1;
dibujar_objeto_matlab_from_stl(bioloid.piezita_brazo,Rot2);
A2cad2 = MTHRotz(degtorad(90));
A2cad3 = MTHRoty(degtorad(90));
Rot2=A02p3*A2cad2*A2cad3;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot2);
A3cad0 = MTHRotx(degtorad(-90));
Rot3=A03p1*A3cad0;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot3);
A3cad1 = MTHRotz(degtorad(-90));
Rot3=A03*A3cad1;
dibujar_objeto_matlab_from_stl(bioloid.rodilla,Rot3);
A3cad2 = MTHRoty(degtorad(180));
A3cad3 = MTHRotz(degtorad(-90));
Rot3=A03p2*A3cad2*A3cad3;
dibujar_objeto_matlab_from_stl(bioloid.perone,Rot3);
Rot4=A04p1*A3cad0;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot4);
A4cad1 = MTHRotx(degtorad(90));
A4cad2 = MTHRoty(degtorad(180));
Rot4=A04*A4cad1* A4cad2;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot4);
A4cad3 = MTHRotz(degtorad(90));
A4cad4 = MTHRoty(degtorad(-90));
Rot4=A04p2*A4cad3*A4cad4;
dibujar_objeto_matlab_from_stl(bioloid.rotula_derecho,Rot4);
Rot4=A04p3*A4cad4;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot4);
Rot4=A04p4*A4cad1*A4cad4;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot4);
Rot4=A04p5*A1cad5;
Rot44=A04p6*A1cad5*A1cad9;
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot4);
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot44);
Rot5=A05p1*A4cad3;
dibujar_objeto_matlab_from_stl(bioloid.planta_pie1_derecho,Rot5);
Rot6=A06p1;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot6);
```

```
A6cad1 = MTHRotz(degtorad(180));
Rot6=A06p2*A6cad1*A3cad2;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot6);
A12cad1 = MTHRoty(degtorad(180));
Rot12=A012p1*A12cad1;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot12);
Rot12=A012p2;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot12);
Rot12=A012p3*A4cad3;
dibujar_objeto_matlab_from_stl(bioloid.planta_pie1_derecho,Rot12);
Rot11= A011p1*A1cad5;
Rot111=A011p2*A1cad5*A1cad9;
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot11);
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot111);
Rot11=A011p3*A4cad1*A4cad4;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot11);
Rot11= A011p4*A4cad4;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot11);
Rot11= A011p5*A4cad3*A4cad4;
dibujar_objeto_matlab_from_stl(bioloid.rotula_derecho,Rot11);
Rot11=A011*A4cad1* A4cad2;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot11);
Rot11=A011p6*A3cad0;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot11);
Rot10=A010p1*A3cad2*A3cad3;
dibujar_objeto_matlab_from_stl(bioloid.perone,Rot10);
Rot9=A010p2*A3cad1;
dibujar_objeto_matlab_from_stl(bioloid.rodilla,Rot9);
Rot9=A09p1*A3cad0;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot9);
Rot8=A08p1*A2cad2*A2cad3;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot8);
Rot8=A08p2*A2cad1;
dibujar_objeto_matlab_from_stl(bioloid.piezita_brazo,Rot8);
Rot2=A08p3*A2cad0*A2cad1;
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna2,Rot2);
rotiz1=MTHRotz(degtorad(180));
Rot8=A08p4*A2cad0;
dibujar_objeto_matlab_from_stl(bioloid.perone,Rot8);
Rot7=A07p1*A1cad3;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot7);
Rot7=A07p2*A1cad3*A1cad8;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot7);
Rot7=A08*A3cad2*A4cad1*A12cad1;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot7);
Rot7=A07p3*A1cad5;
Rot77=A07p4*A1cad5*A1cad9;
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot7);
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot77);
Rot7= A07p5*A1cad3*A2cad3;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot7);
Rot7=A07p6*A1cad*A1cad1*A1cad2;
dibujar_objeto_matlab_from_stl(bioloid.rotula_derecho,Rot7);
Rot7=A0efe1*A12cad1*A4cad3;
```

```
dibujar_objeto_matlab_from_stl(bioloid.planta_pie1_derecho,Rot7);
Rotf=A0efe2*rotiz1*A1cad3;
dibujar_objeto_matlab_from_stl(bioloid.planta_pie_derecho,Rotf);
Rottor=Atorsof*A4cad1*A2cad3 ;
dibujar_objeto_matlab_from_stl(bioloid.torso,Rottor);
titulo1=('coordenadas de centro de masa: ');
titulo3=('coordenadas efector final: ');
titulo2=num2str(Cmt);
titulo4=num2str(coorefec);
tituloes=('  ');
titulo=[titulo1 titulo2  tituloes titulo3 titulo4];
title(titulo)
grid on;
axis ([-200,350,-200,350,0,410]);
view(56,32);
camlight(40,20); lighting phong;
end
end

% bioloid graficas
% esta funcion se encarga de graficar el robot bioloid cuando se realiza
% el paso con el pie derecho
% los datos de ingreso son:
% angulos el vector que contiene los angulos que posee el robot
% M00 es la matriz de trasformacion homogenea que representa la posicion de
% origen de la base del robot
% bioloid es la estructura que contiene todas las piezas digitalizadas del
% robot bioloid
% Bioloid_graficas_izquierdo(angulos,M00,bioloid)
% Autor: Cristian David Villate Martinez.

function  Bioloid_graficas_derecho(angulos,M00,bioloid)
% bioloid simulado
% angulos a trabajar
dimen_a=length(angulos);
if(dimen_a==9)
    primer_a=angulos(1:6);
    segund_a=angulos(7:9);
    mov1_a=[angulos(1),angulos(3),angulos(4),angulos(7),angulos(8)];
    mov2_a=[angulos(2),angulos(5),angulos(9)];
    siete_a=-sum(mov1_a);
    ocho_a=-sum(mov2_a);
    doce_a=-angulos(6);
    angulos=[primer_a,siete_a,ocho_a,segund_a,doce_a];

end
%longitudes
angulos=degtorad(angulos);
pie=50;
L1=34.5;
L2=75.07;
L4=14.50;
L3=77;
pie6=10.3;
```

```
cadera1=30.4;
L5=78;
distancia_tronco=137.44;
brazo_uno=90.50;
brazo_dos=106.60;
cuello=35.11;
masa1=0.1818;
masa2=0.10364;
masa3=0.02104;
masa4=masa1;
masa5=1.06599;
% masa5=2.400;
masa6=0.01082;
masa7=0.03089;
masa1cor=[-18.40; 7.94; 4.21;1];
masa2cor=[-9.03;6.87;63.90;1];
masa3cor=[13.48;0.03;31.64;1];
masa5cor=[1.64;30.844;79.52;1];%masa5cor=[1.64;40.59;79.52;1];
masa6cor=[-17;0;18.94;1];
masa7cor=[-4.7;0;-25.83;1];
masa9cor=[5.47;9.67;-11.10;1];
masa10cor=[-1.02;0.03;-43.36;1];
centrosmasa.masast=[masa1 masa2 masa3 masa4 masa5 masa6 masa7 masa1 masa2 masa3
masa4 masa6];
areapiep=[39,19.76];
% screws
s1d=[0 -1 0];
s01d=[0 0 L1];
s2d=[1 0 0];
s02d=[0 0 L1];
s3d=[0 -1 0];
s03d=[-L4 0 (L1+L2)];
s4d=[0 -1 0];
s04d=[0 0 (L1+L2+L3)];
s5d=[1 0 0];
s05d=[0 0 (L1+L2+L3)];
s6d=[0 0 1];
s06d=[0 0 0];
s1i=[0 -1 0];
s01i=[0 0 L1];
s2i=[1 0 0];
s02i=[0 L5 L1];
s3i=[0 -1 0];
s03i=[-L4 0 (L1+L2)];
s4i=[0 -1 0];
s04i=[0 0 (L1+L2+L3)];
s5i=[1 0 0];
s05i=[0 L5 (L1+L2+L3)];
s6i=[0 0 1];
s06i=[0 L5 0];
%homogeneas
A1=mat_tras_screw(s1d,s01d,(angulos(1)),0);
A2=mat_tras_screw(s2d,s02d,angulos(2),0);
A3=mat_tras_screw(s3d,s03d,angulos(3),0);
```

```
A4=mat_tras_screw(s4d,s04d,angulos(4),0);
A5=mat_tras_screw(s5d,s05d,angulos(5),0);
A6=mat_tras_screw(s6d,s06d,angulos(6),0);
A7=mat_tras_screw(s1i,s01i,angulos(7),0);
A8=mat_tras_screw(s2i,s02i,angulos(8),0);
A9=mat_tras_screw(s3i,s03i,angulos(9),0);
A10=mat_tras_screw(s4i,s04i,angulos(10),0);
A11=mat_tras_screw(s5i,s05i,angulos(11),0);
A12=mat_tras_screw(s6i,s06i,angulos(12),0);
%medidas
desfase=MTHtrasx(M00(1,4))*MTHtrasy(M00(2,4));
color=[0.5 0.5 0.5];
q0d=[0; 0; 0;1];
q1d=[0; 0; L1;1];
q2d=[0; 0; L2+L1;1];
q2dp=[-L4; 0; L2+L1;1];
q3d=[0;0;L3+L2+L1;1];
q4d=[0;0;(L3+L2+L1+cadera1);1];
q0i=[0;L5;0;1];
q0ip=[pie;L5;0;1];
q0ip1=[-pie;L5;0;1];
q1i=[0;L5;L1;1];
q2i=[0; L5; L2+L1;1];
q2ip=[-L4; L5; L2+L1;1];
q3i=[0;L5;L3+L2+L1;1];
q4i=[0;L5;(L3+L2+L1+cadera1);1];
qtron=[0;0;distancia_tronco+L3+L2+L1;1];
qtron_2=[0;L5;distancia_tronco+L3+L2+L1;1];
qbrazo_uno=[0;-brazo_uno;distancia_tronco+L3+L2+L1;1];
q2brazo_uno=[0;L5+brazo_uno;distancia_tronco+L3+L2+L1;1];
qbrazo_dos=[0;-brazo_uno;distancia_tronco+L3+L2+L1-brazo_dos;1];
q2brazo_dos=[0;L5+brazo_uno;distancia_tronco+L3+L2+L1-brazo_dos;1];
qcuello=[0;(L5/2);distancia_tronco+L3+L2+L1+cuello;1];
qmedio=[0;(L5/2);distancia_tronco+L3+L2+L1;1];
A1r = [ eye(3), q0d(1:3,1) ; [0 0 0 1]];
A2r = [ eye(3), q1d(1:3,1) ; [0 0 0 1]];
A3r = [ eye(3), q2d(1:3,1) ; [0 0 0 1]];
A4r = [ eye(3), q2dp(1:3,1) ; [0 0 0 1]];
A5r = [ eye(3), q3d(1:3,1) ; [0 0 0 1]];
A6r = [ eye(3), q1i(1:3,1) ; [0 0 0 1]];
A7r = [ eye(3), q2i(1:3,1) ; [0 0 0 1]];
A8r = [ eye(3), q2ip(1:3,1) ; [0 0 0 1]];
A9r = [ eye(3), q3i(1:3,1) ; [0 0 0 1]];
A0r = [ eye(3), q0i(1:3,1) ; [0 0 0 1]];
A0rp = [ eye(3), q0ip(1:3,1) ; [0 0 0 1]];
A0rp1 = [ eye(3), q0ip1(1:3,1) ; [0 0 0 1]];
A0rtron = [ eye(3),qtron(1:3,1) ; [0 0 0 1]];
A0rtron_2 = [ eye(3),qtron_2(1:3,1) ; [0 0 0 1]];
A0brazo_uno = [ eye(3),qbrazo_uno(1:3,1) ; [0 0 0 1]];
A0brazo_uno2 = [ eye(3),q2brazo_uno(1:3,1) ; [0 0 0 1]];
A0brazo_dos = [ eye(3),qbrazo_dos(1:3,1) ; [0 0 0 1]];
A0brazo_dos2 = [ eye(3),q2brazo_dos(1:3,1) ; [0 0 0 1]];
A0cuello = [ eye(3),qcuello(1:3,1) ; [0 0 0 1]];
A0medio = [ eye(3),qmedio(1:3,1) ; [0 0 0 1]];
```

```
A01=desfase*A2*A2r;
A02=desfase*A1*A2*A2r;
A03=desfase*A1*A2*A3*A4r;
A04=desfase*A1*A2*A3*A4*A5r;
A05=desfase*A1*A2*A3*A4*A5*A5r;
A06=desfase*A1*A2*A3*A4*A5*A6*A5r;
A07=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A8*A6r;
A08=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A6r;
A09=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8r;
A010=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9r;
A011=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A9r;
A012=desfase*A1*A2*A3*A4*A5*A6*A12*A9r;
A0p7=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8*A7*A0r;
Aefec=A0p7;
A0pp=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8*A7*A0rp;
A0pp1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8*A7*A0rp1;
A05tronf=desfase*A1*A2*A3*A4*A5*A6*A0rtron;
A010tronf=desfase*A1*A2*A3*A4*A5*A6*A0rtron_2;
A05brazo_unof=desfase*A1*A2*A3*A4*A5*A6*A0brazo_uno;
A010brazo_uno2f=desfase*A1*A2*A3*A4*A5*A6*A0brazo_uno2;
A05brazo_dosf=desfase*A1*A2*A3*A4*A5*A6*A0brazo_dos;
A010brazo_dos2f=desfase*A1*A2*A3*A4*A5*A6*A0brazo_dos2;
A010cuello=desfase*A1*A2*A3*A4*A5*A6*A0cuello;
A010medio=desfase*A1*A2*A3*A4*A5*A6*A0medio;
A03mod=desfase*A1*A2*A3r;
A04mod=desfase*A1*A2*A3*A3r;
A09mod=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A7r ;
A07mod=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7r;
coorefec=Aefec(:,4);
coorefec(4)=[];
coorefec=coorefec';
%centros de masa referidos a sus sistemas
C01=A01*masa1cor;
C02=A02*masa2cor;
C03=A03*masa3cor;
C04=A04*masa1cor;
C05=A05*masa5cor;
C06=A06*masa6cor;
C07=A07*masa7cor;
C08=A08*masa1cor;
C09=A09*masa9cor;
C010=A010*masa10cor;
C011=A011*masa1cor;
C012=A012*masa6cor;
xm(1)=C01(1);
ym(1)=C01(2);
zm(1)=C01(3);
xm(2)=C02(1);
ym(2)=C02(2);
zm(2)=C02(3);
xm(3)=C03(1);
ym(3)=C03(2);
zm(3)=C03(3);
xm(4)=C04(1);
```

```
ym(4)=C04(2);
zm(4)=C04(3);
xm(5)=C05(1);
ym(5)=C05(2);
zm(5)=C05(3);
xm(6)=C06(1);
ym(6)=C06(2);
zm(6)=C06(3);
xm(7)=C07(1);
ym(7)=C07(2);
zm(7)=C07(3);
xm(8)=C08(1);
ym(8)=C08(2);
zm(8)=C08(3);
xm(9)=C09(1);
ym(9)=C09(2);
zm(9)=C09(3);
xm(10)=C010(1);
ym(10)=C010(2);
zm(10)=C010(3);
xm(11)=C011(1);
ym(11)=C011(2);
zm(11)=C011(3);
xm(12)=C012(1);
ym(12)=C012(2);
zm(12)=C012(3);
sumamasdx=xm.*centrosmasa.masast;
sumamasdy=ym.*centrosmasa.masast;
sumamasdz=zm.*centrosmasa.masast;
xf=sum(sumamasdx)/sum(centrosmasa.masast);
yf=sum(sumamasdy)/sum(centrosmasa.masast);
zf=sum(sumamasdz)/sum(centrosmasa.masast);
centromasat.xm=xf;
centromasat.ym=yf;
centromasat.zm=zf;
centromasat.sistemas=1;
centromasat.color=[0 0 1];
colormasa=[0.5019 0 0.5019];
Cmt=[xf yf zf];
Cmtp=Cmt;
Cmtp(3)=0;
centrosmasa.xm=xm;
centrosmasa.ym=ym;
centrosmasa.zm=zm;
centrosmasa.origen=[-5.1650   45.1022  193.8835];
centrosmasa.sistemas=length(angulos);
centrosmasa.color=colormasa;
areapiep_max=[(M00(1,4)+areapiep(1)),(M00(2,4)+areapiep(2))];
areapiep_min=[(M00(1,4)-areapiep(1)),(M00(2,4)-areapiep(2))];
if(((Cmtp(1)>=areapiep_min(1))&&(Cmtp(1)<=areapiep_max(1)))&&((Cmtp(2)>=areapiep
_min(2))&&(Cmtp(2)<=areapiep_max(2)))&&(coorefec(3)>0))||((coorefec(3)<=0.1)&&(c
oorefec(3)>=-0.1))
    colorlineavalido=[0 1 0];
else
```

```
   colorlineavalido=[1 1 0];
end
opcion=2;
if(opcion==1 )
p0=[0 0 0];
p01=[pie 0 0];
p001=[-pie 0 0];
p02=[0 0 0];
p03=[0 0 L1];
ppie6=[0; L4; L3;1];
ppie7=[0; 0; 0;1];
dis0=A01(:,4);
dis0(4)=[];
dis1=A02(:,4);
dis1(4)=[];
dis2=A03(:,4);
dis2(4)=[];
dis3=A04(:,4);
dis3(4)=[];
dis4=A05(:,4);
dis4(4)=[];
dis5=A06(:,4);
dis5(4)=[];
dis6=A07(:,4);
dis6(4)=[];
dis7=A08(:,4);
dis7(4)=[];
dis8=A09(:,4);
dis8(4)=[];
dis9=A010(:,4);
dis9(4)=[];
dis10=A011(:,4);
dis10(4)=[];
dis11=A012(:,4);
dis11(4)=[];
p04=A0p7(:,4);
p04(4)=[];
p05=A0pp(:,4);
p05(4)=[];
p051=A0pp1(:,4);
p051(4)=[];
dismed_cuer=A05tronf(:,4);
dismed_cuer(4)=[];
dismed_cuer2=A010tronf(:,4);
dismed_cuer2(4)=[];
dismed_brazo1=A05brazo_unof(:,4);
dismed_brazo1(4)=[];
dismed_brazo1_2=A010brazo_uno2f(:,4);
dismed_brazo1_2(4)=[];
dismed_brazo2=A05brazo_dosf(:,4);
dismed_brazo2(4)=[];
dismed_brazo2_2=A010brazo_dos2f(:,4);
dismed_brazo2_2(4)=[];
dismed_cuello=A010cuello(:,4);
```

```
dismed_cuello(4)=[];
dismed_medio=A010medio(:,4);
dismed_medio(4)=[];
dismed_03=A03mod(:,4);
dismed_03(4)=[];
dismed_04=A04mod(:,4);
dismed_04(4)=[];
dismed_09=A09mod(:,4);
dismed_09(4)=[];
dismed_07=A07mod(:,4);
dismed_07(4)=[];
dibujar_linea( p0, p01, color )
hold on
dibujar_linea( p01, p001, color )
dibujar_linea(p02,p03, color )
dibujar_linea(dis0,dis1, color )
dibujar_linea(dis1,dismed_03, color )
dibujar_linea(dismed_03,dis2, color )
dibujar_linea(dis2,dismed_04, color )
dibujar_linea(dismed_04,dis3, color )
dibujar_linea(dis3,dis4, color )
dibujar_linea(dis4,dis5, color )
dibujar_linea(dis6,dis7, color )
dibujar_linea(dis7,dismed_07, color )
dibujar_linea(dismed_07,dis8, color )
dibujar_linea(dis8,dismed_09, color )
dibujar_linea(dismed_09,dis9, color )
dibujar_linea(dis9,dis10, color )
dibujar_linea(dis10,dis11, color )
dibujar_linea(dis11,dis5, color )
dibujar_linea(dis6,p04, color )
dibujar_linea(p04,p05, color )
dibujar_linea(p04,p051, color )
dibujar_linea(Cmtp,Cmt,colorlineavalido )
dibujar_linea(dis5,dismed_cuer, color )
dibujar_linea(dis11,dismed_cuer2, color )
dibujar_linea(dismed_cuer,dismed_cuer2, color )
dibujar_linea(dismed_cuer, dismed_brazo1, color )
dibujar_linea(dismed_cuer2,dismed_brazo1_2, color )
dibujar_linea(dismed_brazo1, dismed_brazo2, color )
dibujar_linea(dismed_brazo1_2, dismed_brazo2_2, color )
dibujar_linea(dismed_brazo1_2, dismed_brazo2_2, color )
dibujar_linea(dismed_medio, dismed_cuello, color )
plot3(dismed_cuello(1),dismed_cuello(2),dismed_cuello(3),'o', 'Color',color
,'LineWidth',25);
ejes(M00,20,'0')
ejes(A01,30,'1')
ejes(A02,20,'2')
ejes(A03,20,'3')
ejes(A04,10,'4')
ejes(A05,20,'5')
ejes(A06,30,'6')
ejes(A07,30,'7')
ejes(A08,20,'8')
```

```
ejes(A09,30,'9')
ejes(A010,10,'10')
ejes(A011,20,'11')
ejes(A012,30,'12')
ejes(Aefec,30,'F')
graf_centros_masa(centrosmasa)
graf_centros_masa(centromasat)
titulo1=('coordenadas de centro de masa: ');
titulo3=('coordenadas efector final: ');
titulo2=num2str(Cmt);
titulo4=num2str(coorefec);
tituloes=('  ');
titulo=[titulo1 titulo2  tituloes titulo3 titulo4];
title(titulo)
grid on
axis ([-200,300,-150,250,0,410]);
view(56,23)
end
if(opcion==2)
pie_z=8.50;
pie2_z=-34.5;
pie2_x=-15;
rotula_z=14.50;
rotula_x=21.1;
dinamixel1x=2;
dinamixel2x=32;
dinamixel2z=14.5;
dinamixel1y=20.215;
piezita1x=38;
piezita1x2=8;
piezita1y=17;
peronez=53.7;
piezita_piernaz=2.3;
piezita_brazo_z=4.3;
dinamixelz3=16.10;
rodillaz=23;
mariposa=3.84;
musloz=29;
distanciarodillax=14;
color2=[0 0 0];
q1d1=[rotula_x; 0; (L1+rotula_z);1];
q1d2=[dinamixel1x; 0; L1;1];
q1d3=[(-piezita1x-piezita1x2);-piezita1y; L1+dinamixel2z;1];
q1d4=[-dinamixel2x; 0; (L1);1];
q1d5=[(-dinamixel2x-piezita1y); 0; (L1);1];
q1d6=[dinamixel1x;dinamixel1y; L1;1];
q1d7=[(-piezita1x-piezita1x2);+piezita1y; L1+dinamixel2z;1];
q2d1=[0; 0; L1+peronez;1];
q2d2=[0; 0; (L1+peronez+piezita_piernaz);1];
q2d3=[-L4; 0; (L1+peronez+piezita_piernaz+piezita_brazo_z+dinamixelz3);1];
q3d1=[-L4;dinamixel1y ; L2+L1;1];
q3d2=[0;0 ; L2+L1+rodillaz+peronez;1];
q4d1=[0;dinamixel1y;L3+L2+L1;1];
q4d2=[rotula_x; 0; (L1+L3+L2-rotula_z);1];
```

```
q4d3=[(-dinamixel2x-piezita1y-mariposa);0; L1+L3+L2;1];
q4d4=[-dinamixel2x; 0; (L1+L3+L2);1];
q4d5=[(-piezita1x-piezita1x2);-piezita1y; L1-dinamixel2z+L3+L2;1];
q4d6=[(-piezita1x-piezita1x2);piezita1y; L1-dinamixel2z+L3+L2;1];
q5d1=[(pie2_x);0; L1+L3+L2;1];
q6d1=[0;0;( L1+L3+L2+(dinamixel2z*2)+mariposa);1];
q6d2=[0;0;( L1+L3+L2+(dinamixel2z*2)+(piezita1x/2));1];
q12i1=[0;L5;( L1+L3+L2+(dinamixel2z*2)+(piezita1x/2));1];
q12i2=[0;L5;( L1+L3+L2+(dinamixel2z*2)+mariposa);1];
q12i3=[(pie2_x);L5; L1+L3+L2;1];
q11i1=[(-piezita1x-piezita1x2);-piezita1y+L5; L1-dinamixel2z+L3+L2;1];
q11i2=[(-piezita1x-piezita1x2);piezita1y+L5; L1-dinamixel2z+L3+L2;1];
q11i3=[-dinamixel2x; L5; (L1+L3+L2);1];
q11i4=[(-dinamixel2x-piezita1y-mariposa);L5; L1+L3+L2;1];
q11i5=[rotula_x; L5; (L1+L3+L2-rotula_z);1];
q11i6=[0;dinamixel1y+L5;L3+L2+L1;1];
q10i1=[0;L5 ; L2+L1+rodillaz+peronez;1];
q10i2=[-distanciarodillax;L5;L2+L1;1];
q9i1=[-L4;dinamixel1y+L5 ; L2+L1;1];
q8i1=[-L4;L5; (L1+peronez+piezita_piernaz+piezita_brazo_z+dinamixelz3);1];
q8i2=[0; L5; (L1+peronez+piezita_piernaz);1];
q8i3=[0; L5; L1+peronez;1];
q8i4=[0; L5; L1;1];
q7i1=[dinamixel1x;dinamixel1y+L5; L1;1];
q7i2=[(-dinamixel2x-piezita1y);L5; (L1);1];
q7i3=[(-piezita1x-piezita1x2);(-piezita1y+L5); L1+dinamixel2z;1];
q7i4=[(-piezita1x-piezita1x2);(piezita1y+L5); L1+dinamixel2z;1];
q7i5=[-dinamixel2x;L5; L1;1];
q7i6=[rotula_x; L5; (L1+rotula_z);1];
qefe1=[-dinamixel2z;L5;L1];
qefe2=[0;L5;pie_z];
qtorso=[-0;48.87;L3+L2+L1+90.39;1];
A2r1 = [ eye(3), q1d1(1:3,1) ; [0 0 0 1]];
A01p1 =desfase*A2*A2r1;
A2r2 = [ eye(3), q1d2(1:3,1) ; [0 0 0 1]];
A01p2 = desfase*A2*A2r2;
A2r3 = [ eye(3), q1d3(1:3,1) ; [0 0 0 1]];
A01p3 = desfase*A2*A2r3;
A2r4 = [ eye(3), q1d4(1:3,1) ; [0 0 0 1]];
A01p4 = desfase*A2*A2r4;
A2r5 = [ eye(3), q1d5(1:3,1) ; [0 0 0 1]];
A01p5 = desfase*A2*A2r5;
A2r6 = [ eye(3), q1d6(1:3,1) ; [0 0 0 1]];
A01p6 = desfase*A2*A2r6;
A2r7 = [ eye(3), q1d7(1:3,1) ; [0 0 0 1]];
A01p7 = desfase*A2*A2r7;
A3r1 = [ eye(3), q2d1(1:3,1) ; [0 0 0 1]];
A02p1=desfase*A1*A2*A3r1;
A3r2 = [ eye(3), q2d2(1:3,1) ; [0 0 0 1]];
A02p2=desfase*A1*A2*A3r2;
A3r3 = [ eye(3), q2d3(1:3,1) ; [0 0 0 1]];
A02p3=desfase*A1*A2*A3r3;
A4r1 = [ eye(3), q3d1(1:3,1) ; [0 0 0 1]];
A03p1=desfase*A1*A2*A3*A4r1;
```

```
A4r2 = [ eye(3), q3d2(1:3,1) ; [0 0 0 1]];
A03p2=desfase*A1*A2*A3*A4r2;
A5r1 = [ eye(3), q4d1(1:3,1) ; [0 0 0 1]];
A04p1=desfase*A1*A2*A3*A4*A5r1;
A5r2 = [ eye(3), q4d2(1:3,1) ; [0 0 0 1]];
A04p2=desfase*A1*A2*A3*A4*A5r2;
A5r3 = [ eye(3), q4d3(1:3,1) ; [0 0 0 1]];
A04p3=desfase*A1*A2*A3*A4*A5r3;
A5r4 = [ eye(3), q4d4(1:3,1) ; [0 0 0 1]];
A04p4=desfase*A1*A2*A3*A4*A5r4;
A5r5 = [ eye(3), q4d5(1:3,1) ; [0 0 0 1]];
A04p5=desfase*A1*A2*A3*A4*A5r5;
A5r6= [ eye(3), q4d6(1:3,1) ; [0 0 0 1]];
A04p6=desfase*A1*A2*A3*A4*A5r6;
A6r1= [ eye(3), q5d1(1:3,1) ; [0 0 0 1]];
A05p1=desfase*A1*A2*A3*A4*A5*A6r1;
A7r1= [ eye(3), q6d1(1:3,1) ; [0 0 0 1]];
A06p1=desfase*A1*A2*A3*A4*A5*A6*A7r1;
A7r2= [ eye(3), q6d2(1:3,1) ; [0 0 0 1]];
A06p2=desfase*A1*A2*A3*A4*A5*A6*A7r2;
A13r1= [ eye(3), q12i1(1:3,1) ; [0 0 0 1]];
A012p1=desfase*A1*A2*A3*A4*A5*A6*A13r1;
A13r2= [ eye(3), q12i2(1:3,1) ; [0 0 0 1]];
A012p2=desfase*A1*A2*A3*A4*A5*A6*A12*A13r2;
A13r3= [ eye(3), q12i3(1:3,1) ; [0 0 0 1]];
A012p3=desfase*A1*A2*A3*A4*A5*A6*A12*A13r3;
A11r1= [ eye(3), q11i1(1:3,1) ; [0 0 0 1]];
A011p1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A11r1;
A11r2= [ eye(3), q11i2(1:3,1) ; [0 0 0 1]];
A011p2=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A11r2;
A11r3= [ eye(3), q11i3(1:3,1) ; [0 0 0 1]];
A011p3=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A11r3;
A11r4= [ eye(3), q11i4(1:3,1) ; [0 0 0 1]];
A011p4=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A11r4;
A11r5= [ eye(3), q11i5(1:3,1) ; [0 0 0 1]];
A011p5=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A11r5;
A11r6= [ eye(3), q11i6(1:3,1) ; [0 0 0 1]];
A011p6=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A11r6;
A10r1= [ eye(3), q10i1(1:3,1) ; [0 0 0 1]];
A010p1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A10r1;
A10r2= [ eye(3), q10i2(1:3,1) ; [0 0 0 1]];
A010p2=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A10r2;
A9r1= [ eye(3), q9i1(1:3,1) ; [0 0 0 1]];
A09p1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A9r1;
A8r1= [ eye(3), q8i1(1:3,1) ; [0 0 0 1]];
A08p1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8r1;
A8r2= [ eye(3), q8i2(1:3,1) ; [0 0 0 1]];
A08p2=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8r2;
A8r3= [ eye(3), q8i3(1:3,1) ; [0 0 0 1]];
A08p3=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8r3;
A8r4= [ eye(3), q8i4(1:3,1) ; [0 0 0 1]];
A08p4=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A8r4;
A72r1= [ eye(3), q7i1(1:3,1) ; [0 0 0 1]];
A07p1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A72r1;
```

```
A72r2= [ eye(3), q7i2(1:3,1) ; [0 0 0 1]];
A07p2=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A72r2;
A72r3= [ eye(3), q7i3(1:3,1) ; [0 0 0 1]];
A07p3=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A72r3;
A72r4= [ eye(3), q7i4(1:3,1) ; [0 0 0 1]];
A07p4=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A72r4;
A72r5= [ eye(3), q7i5(1:3,1) ; [0 0 0 1]];
A07p5=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A72r5;
A72r6= [ eye(3), q7i6(1:3,1) ; [0 0 0 1]];
A07p6=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A72r6;
Aefe1= [ eye(3), qefe1(1:3,1) ; [0 0 0 1]];
A0efe1=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A8*Aefe1;
Aefe2= [ eye(3), qefe2(1:3,1) ; [0 0 0 1]];
A0efe2=desfase*A1*A2*A3*A4*A5*A6*A12*A11*A10*A9*A7*A8*Aefe2;
Atorso= [ eye(3), qtorso(1:3,1) ; [0 0 0 1]];
Atorsof=desfase*A1*A2*A3*A4*A5*A6*Atorso;
dibujar_sistema_referencia_MTH(M00, 50, 3, '0');
dibujar_sistema_referencia_MTH(A01, 50, 3, '1');
dibujar_sistema_referencia_MTH(A02, 40, 3, '2');
dibujar_sistema_referencia_MTH(A03, 50, 3, '3');
dibujar_sistema_referencia_MTH(A04, 40, 3, '4');
dibujar_sistema_referencia_MTH(A05, 50, 3, '5');
dibujar_sistema_referencia_MTH(A06, 60, 3, '6');
dibujar_sistema_referencia_MTH(A07, 50, 3, '7');
dibujar_sistema_referencia_MTH(A08, 40, 3, '8');
dibujar_sistema_referencia_MTH(A09, 50, 3, '9');
dibujar_sistema_referencia_MTH(A010, 40, 3, '10');
dibujar_sistema_referencia_MTH(A011, 50, 3, '11');
dibujar_sistema_referencia_MTH(A012,60, 3, '12');
dibujar_sistema_referencia_MTH(Aefec,60, 3, 'F');
hold on
graf_centros_masa(centrosmasa)
graf_centros_masa(centromasat)
dibujar_linea(Cmtp,Cmt,colorlineavalido )
A0cad = MTHRotx(degtorad(-90));
A0cad1= MTHRotz(degtorad(180));
A0cad6= MTHtrasz(pie_z);
Rot0=M00*A0cad;
Rot0=A0cad6*Rot0;
dibujar_objeto_matlab_from_stl(bioloid.planta_pie_derecho,Rot0);
A0cad2 = MTHRotx(degtorad(180));
A0cad3= MTHRotz(degtorad(90));
A0cad4= MTHtrasz(-pie2_z);
A0cad5= MTHtrasx(pie2_x);
Rot0=M00*A0cad2*A0cad3;
Rot0=A0cad4*Rot0;
Rot0=A0cad5*Rot0;
dibujar_objeto_matlab_from_stl(bioloid.planta_pie1_derecho,Rot0);
A1cad = MTHRotx(degtorad(90));
A1cad1 = MTHRotz(degtorad(90));
A1cad2 = MTHRoty(degtorad(180));
Rot1=A01p1*A1cad*A1cad1*A1cad2;
dibujar_objeto_matlab_from_stl(bioloid.rotula_derecho,Rot1);
A1cad3 = MTHRotx(degtorad(-90));
```

```
A1cad4 = MTHRotz(degtorad(180));
Rot1=A01p2*A1cad3;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot1);
A1cad5 = MTHRoty(degtorad(180));
A1cad9 = MTHRotx(degtorad(180));
Rot1=A01p3*A1cad5;
Rot11=A01p7*A1cad5*A1cad9;
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot1);
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot11);
A1cad6 = MTHRoty(degtorad(90));
A1cad7 = MTHRotx(degtorad(180));
A1cad8 = MTHRotz(degtorad(180));
Rot1=A01p4*A1cad3*A1cad6*A1cad7*A1cad8;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot1);
A1cad8 = MTHRoty(degtorad(90));
Rot1=A01p5*A1cad3*A1cad8;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot1);
Rot1=A01p6*A1cad3;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot1);
A2cad0 = MTHRotz(degtorad(90));
Rot2=A02*A2cad0;
dibujar_objeto_matlab_from_stl(bioloid.perone,Rot2);
A2cad1 = MTHRotx(degtorad(90));
Rot2=A02p1*A2cad0*A2cad1;
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna2,Rot2);
Rot2=A02p2*A2cad1;
dibujar_objeto_matlab_from_stl(bioloid.piezita_brazo,Rot2);
A2cad2 = MTHRotz(degtorad(90));
A2cad3 = MTHRoty(degtorad(90));
Rot2=A02p3*A2cad2*A2cad3;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot2);
A3cad0 = MTHRotx(degtorad(-90));
Rot3=A03p1*A3cad0;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot3);
A3cad1 = MTHRotz(degtorad(-90));
Rot3=A03*A3cad1;
dibujar_objeto_matlab_from_stl(bioloid.rodilla,Rot3);
A3cad2 = MTHRoty(degtorad(180));
A3cad3 = MTHRotz(degtorad(-90));
Rot3=A03p2*A3cad2*A3cad3;
dibujar_objeto_matlab_from_stl(bioloid.perone,Rot3);
Rot4=A04p1*A3cad0;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot4);
A4cad1 = MTHRotx(degtorad(90));
A4cad2 = MTHRoty(degtorad(180));
Rot4=A04*A4cad1* A4cad2;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot4);
A4cad3 = MTHRotz(degtorad(90));
A4cad4 = MTHRoty(degtorad(-90));
Rot4=A04p2*A4cad3*A4cad4;
dibujar_objeto_matlab_from_stl(bioloid.rotula_derecho,Rot4);
Rot4=A04p3*A4cad4;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot4);
Rot4=A04p4*A4cad1*A4cad4;
```

```
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot4);
Rot4=A04p5*A1cad5;
Rot44=A04p6*A1cad5*A1cad9;
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot4);
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot44);
Rot5=A05p1*A4cad3;
dibujar_objeto_matlab_from_stl(bioloid.planta_pie1_derecho,Rot5);
Rot6=A06p1;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot6);
A6cad1 = MTHRotz(degtorad(180));
Rot6=A06p2*A6cad1*A3cad2;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot6);
A12cad1 = MTHRoty(degtorad(180));
Rot12=A012p1*A12cad1;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot12);
Rot12=A012p2;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot12);
Rot12=A012p3*A4cad3;
dibujar_objeto_matlab_from_stl(bioloid.planta_pie1_derecho,Rot12);
Rot11= A011p1*A1cad5;
Rot111=A011p2*A1cad5*A1cad9;
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot11);
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot111);
Rot11=A011p3*A4cad1*A4cad4;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot11);
Rot11= A011p4*A4cad4;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot11);
Rot11= A011p5*A4cad3*A4cad4;
dibujar_objeto_matlab_from_stl(bioloid.rotula_derecho,Rot11);
Rot11=A011*A4cad1* A4cad2;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot11);
Rot11=A011p6*A3cad0;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot11);
Rot10=A010p1*A3cad2*A3cad3;
dibujar_objeto_matlab_from_stl(bioloid.perone,Rot10);
Rot9=A010p2*A3cad1;
dibujar_objeto_matlab_from_stl(bioloid.rodilla,Rot9);
Rot9=A09p1*A3cad0;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot9);
Rot8=A08p1*A2cad2*A2cad3;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot8);
Rot8=A08p2*A2cad1;
dibujar_objeto_matlab_from_stl(bioloid.piezita_brazo,Rot8);
Rot2=A08p3*A2cad0*A2cad1;
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna2,Rot2);
rotiz1=MTHRotz(degtorad(180));
Rot8=A08p4*A2cad0;
dibujar_objeto_matlab_from_stl(bioloid.perone,Rot8);
Rot7=A07p1*A1cad3;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot7);
Rot7=A07p2*A1cad3*A1cad8;
dibujar_objeto_matlab_from_stl(bioloid.mariposa_dinamixel,Rot7);
Rot7=A08*A3cad2*A4cad1*A12cad1;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot7);
```

```
Rot7=A07p3*A1cad5;
Rot77=A07p4*A1cad5*A1cad9;
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot7);
dibujar_objeto_matlab_from_stl(bioloid.piezita_pierna,Rot77);
Rot7= A07p5*A1cad3*A2cad3;
dibujar_objeto_matlab_from_stl(bioloid.servomotor,Rot7);
Rot7=A07p6*A1cad*A1cad1*A1cad2;
dibujar_objeto_matlab_from_stl(bioloid.rotula_derecho,Rot7);
Rot7=A0efe1*A12cad1*A4cad3;
dibujar_objeto_matlab_from_stl(bioloid.planta_pie1_derecho,Rot7);
Rotf=A0efe2*rotiz1*A1cad3;
dibujar_objeto_matlab_from_stl(bioloid.planta_pie_derecho,Rotf);
Rottor=Atorsof*A4cad1*A2cad3 ;
dibujar_objeto_matlab_from_stl(bioloid.torso,Rottor);
titulo1=('coordenadas de centro de masa: ');
titulo3=('coordenadas efector final: ');
titulo2=num2str(Cmt);
titulo4=num2str(coorefec);
tituloes=('  ');
titulo=[titulo1 titulo2  tituloes titulo3 titulo4];
title(titulo)
grid on;
axis ([-200,350,-200,350,0,410]);
view(56,32);
camlight(40,20); lighting phong;
end



% funcion que se enccarga de cargar las piezas de un archivo stl a una
% estructura de matlab.
% Autor Cristian David Villate Martinez

function bioloid = Bioloid_piezas()
color=[0.5 0.5 0.5];
color2=[0 0 0];
bioloid.planta_pie_derecho=fun_stl2matlab('planta_pie.stl',color,0);
bioloid.planta_pie1_derecho=fun_stl2matlab('muslo.stl',color,0);
bioloid.rotula_derecho=fun_stl2matlab('rotula.stl',color,0);
bioloid.servomotor=fun_stl2matlab('AX-12A.stl',color2,0);
bioloid.piezita_pierna=fun_stl2matlab('piezita_pierna.stl',color,0);
bioloid.mariposa_dinamixel=fun_stl2matlab('wheel AX-12A.stl',color2,0);
bioloid.perone=fun_stl2matlab('perone.stl',color,0);
bioloid.piezita_pierna2=fun_stl2matlab('piezita_pierna2.stl',color,0);
bioloid.piezita_brazo=fun_stl2matlab('piezita_brazo.stl',color,0);
bioloid.rodilla=fun_stl2matlab('rodilla.stl',color,0);
bioloid.torso=fun_stl2matlab('bioloid_torso_dos.stl',color,0);

% programa que genera una caminata en el modelo real del robot
% Autor Cristian David Villate Martinez

CargarLibreriasDynamixel
movimiento0=zeros(1,9);
movimiento22=[-10 15 20 -15 0 0 -70 50 0];
```

```
movimiento3=[-12 0 0 -1 0 0 -47 46 0];
movimiento1=[-14 0 47 -46 0 0 0 1 0];
movimiento2=[-10 -13 20 -15 0 0 -70 50 0];
M1=Trayectoria_Robot_Bioloid(movimiento0,movimiento22);
M2=Trayectoria_Robot_Bioloid(movimiento22,movimiento3);
trayectorias1=[M1;M2];
M1=Trayectoria_Robot_Bioloid_prue(movimiento1,movimiento2);
M2=Trayectoria_Robot_Bioloid_prue(movimiento2,movimiento3);
trayectorias2=[M1;M2];
M1=Trayectoria_Robot_Bioloid(movimiento1,movimiento22);
M2=Trayectoria_Robot_Bioloid(movimiento22,movimiento3);
trayectorias3=[M1;M2];
[tam1, basura]=size(trayectorias1);
[tam2, basura]=size(trayectorias2);
[tam3, basura]=size(trayectorias3);
sentido_dinamixel=[-1 1 -1 1 -1 -1 -1 -1 -1 1 1 -1];
%dato 3 inicial 152.1994
posiciones_iniciales_dinamixel=[149.8534 149.2669 153 155.1320 148.9736 104.9853
148.3871 152.1994 148.0938 144.5748 152.7859 192.6686];
paso_articular=1;
qp=linspace(8,8,18);
ids_piernas = [15 17 13 11 9 7 16 18 14 12 10 8];
ids_brazos = [1 3 5 2 4 6];
vecid=[ids_brazos ids_piernas];
posiciones_iniciales_brazos=[60.1173 111.7302 87.0968 244.2815 177.7126
221.1144];
for i=1:tam1
    [Cmtp,movimiento]=Bioloid_Calculos_centroid(trayectorias1(i,:));
    centroidest(i,:)=Cmtp;
     trayectoria1_dinamixel(i,:)=
(trayectorias1(i,:).*sentido_dinamixel*paso_articular)+posiciones_iniciales_dina
mixel;
end
trayectoria1_dinamixel=round(trayectoria1_dinamixel);
posiciones_iniciales_dinamixel=trayectoria1_dinamixel(tam1,:);
trayectorias2=[trayectorias2(1:tam2,7:12) trayectorias2(1:tam2,1:6)]*-1;
trayectoriasres=trayectorias2(1,:);
for i=1:tam2
    [Cmtp,movimiento]=Bioloid_Calculos_dos_centroid(trayectorias2(i,:));
    centroidest2(i,:)=Cmtp;
    trayectoria2_dinamixel(i,:)=((trayectorias2(i,:)-
trayectoriasres).*sentido_dinamixel*paso_articular)+posiciones_iniciales_dinamix
el;
end
trayectoriasres3=trayectorias3(1,:);
posiciones_iniciales_dinamixel=trayectoria2_dinamixel(tam2,:);
for i=1:tam3
    [Cmtp,movimiento]=Bioloid_Calculos_centroid(trayectorias3(i,:));
    centroidest3(i,:)=Cmtp;
    trayectoria3_dinamixel(i,:)=((trayectorias3(i,:)-
trayectoriasres3).*sentido_dinamixel*paso_articular)+posiciones_iniciales_dinami
xel;
end
% secuencia
```

```
for i=1:tam1
q=[posiciones_iniciales_brazos trayectoria1_dinamixel(i,:)];
Data = mover_n_robot_servos_pv( vecid, q, qp);
end
n=1;
while(n<4)
for i=1:tam2
q=[posiciones_iniciales_brazos trayectoria2_dinamixel(i,:)];
Data = mover_n_robot_servos_pv2( vecid, q, qp);
end
for i=1:tam3
q=[posiciones_iniciales_brazos trayectoria3_dinamixel(i,:)];
Data = mover_n_robot_servos_pv3( vecid, q, qp);
end
n=n+1;
end
trayecfin=[trayectoria1_dinamixel;trayectoria2_dinamixel;trayectoria3_dinamixel]
;
plot(trayecfin)
caminata.trayecfin=trayecfin;
caminata.trayectoria1_dinamixel=trayectoria1_dinamixel;
caminata.trayectoria2_dinamixel=trayectoria2_dinamixel;
caminata.trayectoria3_dinamixel=trayectoria3_dinamixel;
caminata.trayectorias1=trayectorias1;
caminata.trayectorias2=trayectorias2;
caminata.trayectorias3=trayectorias3;
caminata.centroidest=centroidest;
caminata.centroidest2=centroidest2;
caminata.centroidest3=centroidest3;
caminata.valido=1;
save('datos_caminata10.mat','caminata');
DescargarLibreriasDynamixel

% carga la libreria dinamixel

Variables_Globales
ERRBIT_VOLTAGE     = 1;
ERRBIT_ANGLE       = 2;
ERRBIT_OVERHEAT    = 4;
ERRBIT_RANGE       = 8;
ERRBIT_CHECKSUM    = 16;
ERRBIT_OVERLOAD    = 32;
ERRBIT_INSTRUCTION = 64;

COMM_TXSUCCESS     = 0;
COMM_RXSUCCESS     = 1;
COMM_TXFAIL        = 2;
COMM_RXFAIL        = 3;
COMM_TXERROR       = 4;
COMM_RXWAITING     = 5;
COMM_RXTIMEOUT     = 6;
COMM_RXCORRUPT     = 7;
```

```
%%%%%%

% P_TORQUE_ENABLE = 24;
 P_TORQUE_ENABLE = 24;
P_LED = 25;
P_GOAL_POSITION = 30;
P_GOAL_VELOCITY = 32;
P_PRESENT_POSITION = 36;
P_PRESENT_VELOCITY = 38;
P_PRESENT_LOAD = 40;
DEFAULT_PORTNUM = 19%7;%4; % com3
DEFAULT_BAUDNUM = 1; % 1mbps
P_Moving = 46;

%%%%%
KVrgAX12 = 0.575;    % Esta constante se calcula dependiendo del voltaje de
                     % alimentación, se puede usar el programa:
                     % prueba_velocidad_motores para calcular su valor
KVgrAX12 = 1/KVrgAX12;

loadlibrary('dynamixel','dynamixel.h');%loadlibrary('dynamixel','dxl_matlab.h');
%libfunctions('dynamixel');

res = calllib('dynamixel','dxl_initialize',DEFAULT_PORTNUM,DEFAULT_BAUDNUM);
if res == 1
    disp('Succeed to open USB2Dynamixel!');
else
    disp('Failed to open USB2Dynamixel!');
    break;
end

%Close Device
calllib('dynamixel','dxl_terminate');
unloadlibrary('dynamixel');


function dibujar_linea( p0, p1, color )
% vectores que contienen las componentes de los puntos a dibujar
x = [ p0(1), p1(1)];
y = [ p0(2), p1(2)];
z = [ p0(3), p1(3)];
% Dibujo de las lineas
plot3(x,y,z, 'Color', color ,'LineWidth',4);


% dibujar_objeto_matlab_from_stl.m
% Esta funcion dibuja un objeto importado de STL segun la
% la MTH A
% Ejemplo:
% clear, close all, clc
% objeto = fun_stl2matlab('fig3d1.stl', [1,0,0],0);
% A = [ 1,  0,          0,         30; ...
%       0,  cos(pi/6), -sin(pi/6), 20; ...
```

```
%       0,  sin(pi/6),  cos(pi/6), 50; ...
%       0,  0,          0,         1];
% dibujar_objeto_matlab_from_stl(objeto,A);
% view(40,20), camlight(40,20);, lighting phong;


function dibujar_objeto_matlab_from_stl(objeto,A)

for i=1:objeto.n_faces
    for j=1:3
        p = [ objeto.x(j,i); objeto.y(j,i); objeto.z(j,i); 1];
        pa = A*p;
        objeto_A.x(j,i) = pa(1);
        objeto_A.y(j,i) = pa(2);
        objeto_A.z(j,i) = pa(3);
    end
end
p=patch(objeto_A.x,objeto_A.y,objeto_A.z,objeto.tcolor);
set(p, 'EdgeColor', 'none' );
set(p, 'FaceAlpha',0.3 );



% dibujar_sistema_referencia_MTH.m
% Esta función dibuja un sistema de referencia rotado segun
% la matriz de ttransformación homogenea MYH. L es la longuitud,
% G el grosor y el texto
%
% Ejemplo:
% MTH = eye(4);
% dibujar_sistema_referencia_MTH(MTH, 10, 5, '1');
% grid on, view(50,30);
function dibujar_sistema_referencia_MTH(MTH, L, G, Subindicice)

% Puntos finales de los ejes x, y, z
pfx = MTH*[L;0;0;1];    % Punto final eje x
pfy = MTH*[0;L;0;1];    % Punto final eje y
pfz = MTH*[0;0;L;1];    % Punto final eje z
% Dibuja los ejes
line( [MTH(1,4), pfx(1)], [MTH(2,4), pfx(2)],  [MTH(3,4), pfx(3)],'LineWidth',G,
'Color', 'r' );
line( [MTH(1,4), pfy(1)], [MTH(2,4), pfy(2)],  [MTH(3,4), pfy(3)],'LineWidth',G,
'Color', 'g' );
line( [MTH(1,4), pfz(1)], [MTH(2,4), pfz(2)],  [MTH(3,4), pfz(3)],'LineWidth',G,
'Color', 'b' );
% Dibuja los titulos de los ejes coordenados
text( pfx(1),pfx(2),pfx(3),strcat(' x_',Subindicice) );
text( pfy(1),pfy(2),pfy(3),strcat(' y_',Subindicice) );
text( pfz(1),pfz(2),pfz(3),strcat(' z_',Subindicice) );

function[] =ejes(A, t, subindice)
xp=A(:,1);
yp=A(:,2);
zp=A(:,3);
```

```
r=A(:,4);
p0 = [0, 0, 0];
for i=1:3
   xc(i)=(xp(i)*t)+r(i);
   yc(i)=(yp(i)*t)+r(i);
   zc(i)=(zp(i)*t)+r(i);
   p0(i)=p0(i)+r(i);
end
x = [ p0(1), xc(1)];
y = [ p0(2), xc(2)];
z = [ p0(3), xc(3)];
color=[1 0 0];
% Dibujo de las lineas
plot3(x,y,z, 'Color', color ,'LineWidth',2);
text(x(2),y(2),z(2),strcat('X_',subindice))
hold on
x = [ p0(1), yc(1)];
y = [ p0(2), yc(2)];
z = [ p0(3), yc(3)];
color=[0 1 0];
plot3(x,y,z, 'Color', color ,'LineWidth',2);
text(x(2),y(2),z(2),strcat('Y_',subindice))
hold on
x = [ p0(1), zc(1)];
y = [ p0(2), zc(2)];
z = [ p0(3), zc(3)];
color=[0 0 1];
plot3(x,y,z, 'Color', color ,'LineWidth',2);
text(x(2),y(2),z(2),strcat('Z_',subindice))
end
```

```
% fprimer_choque.m
%
% esta funcion se encarga en generar todos los datos antes de el primer choque.
%
% M es la matriz que contiene la trayectoria
% q vector articular
% iteracion es el numero de ciclos antes de existir un choque
% articulaciones es el numero de articulaciones totales
%
```

```
% salida
%
% iteracionguar es el numero de ciclos antes de existir un choque
% primer_choque es una variable que simboliza la existencia de un primer
% choque
% angulosf es el vector articular final de la trayectoria antes de existir
% un choqe
% qprimier vector articular antes de el primer choque
% q_anterior es el vector articular del punto anterior de la matriz antes
% de existir un choque
% Msin_choque es la matriz antes de existir un choque
%
% ejemplo
% [iteracionguar primer_choque qprimer
Msin_choque]=fprimer_choque(M,q,iteracion,articulaciones)
%
% Autor Cristian David Villate Martinez
%
function [iteracionguar primer_choque angulosf qprimer q_anterior
Msin_choque]=fprimer_choque(M,q,iteracion,articulaciones)
 iteracionguar=iteracion;
% datos cuando existen nueve articulaciones
 if(articulaciones==9)
  Msin_choque=M([1:iteracionguar],[1:articulaciones+3]);
  primer_choque=1;
   qprimer=q;
   anterior=Msin_choque(iteracionguar-1,:);
   angulosf=Msin_choque(iteracionguar,:);
   q_anterior=[anterior(1:6),anterior(9:11)];
 else
% datos cuando existen doce articulaciones
  Msin_choque=M([1:iteracionguar],[1:articulaciones]);
  primer_choque=1;
  q_anterior=Msin_choque(iteracionguar-1,:);
   qprimer=q;
   angulosf=qprimer
 end
end
```

```
function graf_centros_masa(centrosmasa)
for i=1:centrosmasa.sistemas
 if(centrosmasa.sistemas==1)
     valor='F';
 else
  valor=num2str(i);
 end
plot3(centrosmasa.xm(i),centrosmasa.ym(i),centrosmasa.zm(i),'*',
'Color',centrosmasa.color ,'LineWidth',5);
text(centrosmasa.xm(i),centrosmasa.ym(i),centrosmasa.zm(i),strcat('Cm_',valor))
hold on
```

```
end

%id: Vector con los id de los motores
% Ejemplo
% CargarLibreriasDynamixel;
% PresentPos=leer_posicion_motores_id([101 105])
% DescargarLibreriasDynamixel;


function PresentPos = leer_posicion_motores_id(id)

Variables_Globales;
nmotores = length(id);

 %% PresentPos  lee  la posicion con la instruccion 36 = Present Position
    for i=1:nmotores
        PresentPos(i) =
int32(calllib('dynamixel','dxl_read_word',int32(id(i)),36));
        %pause(0.01)
    end
    PresentPos = double(PresentPos)*(300/1023);
end

% mat_tras_screw.m
% Función que calcula la MTH originada por el desplazamiento de un screw
function A=mat_tras_screw(si,s0i,angulo,t)
sx=si(1);sy=si(2);,sz=si(3);
s0x=s0i(1);s0y=s0i(2);,s0z=s0i(3);
a11=((sx^2-1)*(1-cos(angulo)))+1;
a12=sx*sy*(1-cos(angulo))-sz*sin(angulo);
a13=sx*sz*(1-cos(angulo))+sy*sin(angulo);
a21=sy*sx*(1-cos(angulo))+sz*sin(angulo);
a22=(sy^2-1)*(1-cos(angulo))+1;
a23=sy*sz*(1-cos(angulo))-sx*sin(angulo);
a31=sz*sx*(1-cos(angulo))-sy*sin(angulo);
a32=sz*sy*(1-cos(angulo))+sx*sin(angulo);
a33=(sz^2-1)*(1-cos(angulo))+1;
a14=t*sx-s0x*(a11-1)-s0y*a12-s0z*a13;
a24=t*sy-s0x*a21-s0y*(a22-1)-s0z*a23;
a34=t*sz-s0x*a31-s0y*a32-s0z*(a33-1);
a41=0;
a42=0;
a43=0;
a44=1;
A=[a11 a12 a13 a14; a21 a22 a23 a24; a31 a32 a33 a34 ; a41 a42 a43 a44];

% mover_n_robot_servos_pv.m
% Example:
%
% CargarLibreriasDynamixel
% qp=[100,50]                       %% velocidad
% q=[90,90]                         %% posicion
% vecid = [101, 105]                  %% id motores
```

```
% Data = mover_n_robot_servos_pv( vecid, q, qp)     %% llamado de la funcion
% plot(Data.Vec_time, Data.Vec_position);
% DescargarLibreriasDynamixel

function Data = mover_n_robot_servos_pv( vecid, q, qp)

Variables_Globales;

nmotores = length(vecid);
if ~((nmotores == length(q)) && (nmotores== length(qp)))
    error('El tamano de los vectores es diferente');
end



qm = int32(q*(1023/300));
qpm = int32(qp*(KVgrAX12));

%Broadcast ID
calllib('dynamixel','dxl_set_txpacket_id', 254);

%Length is 14
%That handles position and speed for n dynamixels
%%(L + 1) * N + 4 (L: Data length for each Dynamixel actuator, N: The
%%number of Dynamixel actuators)
tamano_arreglo = (4+1)*nmotores+4;
calllib('dynamixel','dxl_set_txpacket_length',tamano_arreglo);

%SyncWrite instruction
calllib('dynamixel','dxl_set_txpacket_instruction',131);

%Starting address (goal position)
calllib('dynamixel','dxl_set_txpacket_parameter',0, 30);

%length of data to write to each dynamixel
%We're writing position and speed = 4 bytes
calllib('dynamixel','dxl_set_txpacket_parameter',1, 4);

%Parameters for syncwrite
% id | position | speed
%ID

for im=1:nmotores
    %Parameters for syncwrite dynamixel id = im
    calllib('dynamixel','dxl_set_txpacket_parameter',(im-1)*5+2,vecid(im) );
    %position
    lowByte = calllib('dynamixel','dxl_get_lowbyte', int32(qm(im)));
    highByte = calllib('dynamixel','dxl_get_highbyte', int32(qm(im)));
    calllib('dynamixel','dxl_set_txpacket_parameter',int32((im-1)*5+3),
lowByte);
    calllib('dynamixel','dxl_set_txpacket_parameter',int32((im-1)*5+4),
highByte);
```

```
    %Speed
    lowByte = calllib('dynamixel','dxl_get_lowbyte', int32(qpm(im)));
    highByte = calllib('dynamixel','dxl_get_highbyte', int32(qpm(im)));
    calllib('dynamixel','dxl_set_txpacket_parameter',int32((im-1)*5+5),
lowByte);
    calllib('dynamixel','dxl_set_txpacket_parameter',int32((im-1)*5+6),
highByte);
end

%transmit
calllib('dynamixel','dxl_txrx_packet');

% Inicializa Contador
i = 0;
Data.Vec_position = [];
Data.Vec_time = [];
Moving = 1;
tic;
while (Moving == 1)
    %Read Present position
    for m=1:nmotores
        PresentPos(m) =
int32(calllib('dynamixel','dxl_read_word',vecid(m),P_PRESENT_POSITION));
    end
    CommStatus = int32(calllib('dynamixel','dxl_get_result'));
    t = toc;
    if CommStatus == COMM_RXSUCCESS
        i = i+1;

        Data.Vec_position(i,:) = double(PresentPos)*(300/1023);
        Data.Vec_time(i,1) = t;

        % Detecta la condicion que todos los motores lleguen a la meta
        nOK = 1;
        %14 hiba
        for m=1:nmotores
            nOK= nOK*(abs(PresentPos(m) - qm(m)) <= 14);
        end

        % si se cumplen todas las condiciones
        if nOK
            Moving = 0;
            %disp('Finish the first movement');
        end
    else
        PrintCommStatus(CommStatus);
        break;
    end

end


% MTHRotx.m
```

```
%
% Crea una MTH con una rotación respecto al eje x

function MTH = MTHRotx(angulo)

MTH = [1,         0,              0              ,0 ;...
       0,         cos(angulo),    -sin(angulo)   ,0 ;...
       0,         sin(angulo),    cos(angulo)    ,0 ;...
       0,         0,              0              ,1];


% MTHRoty.m
%
% Crea una MTH con una rotación respecto al eje y

function MTH = MTHRoty(angulo)

MTH = [cos(angulo),         0,         sin(angulo)     ,0 ;...
       0,                   1,         0               ,0 ;...
       -sin(angulo),        0,         cos(angulo)     ,0 ;...
       0,                   0,         0               ,1];

% MTHRotz.m
%
% Crea una MTH con una rotación respecto al eje z

function MTH = MTHRotz(angulo)

MTH = [cos(angulo),       -sin(angulo),    0   ,0 ;...
        sin(angulo),      cos(angulo),     0   ,0 ;...
        0,                0,               1   ,0 ;...
        0,                0,               0   ,1];




% MTHtrasx.m
%
% Crea una MTH con una traslación respecto al eje x

function MTH = MTHtrasx(distancia)

MTH = eye(4);
MTH(1,4) = distancia;

% MTHtrasy.m
%
% Crea una MTH con una traslación respecto al eje y

function MTH = MTHtrasy(distancia)
```

```
MTH = eye(4);
MTH(2,4) = distancia;

% MTHtrasz.m
%
% Crea una MTH con una traslación respecto al eje z

function MTH = MTHtrasz(distancia)

MTH = eye(4);
MTH(3,4) = distancia;

% selec_articulacion_excluida.m
%
% esta funcion se encarga de generar un vector que de forma aleatoria
dependiendo
% las articulaciones y los pasos este genera un avance tanto negativo o
% positivo dependiendo de las articulaciones que llegaron al limite
% n es el numero de articulaciones
% paso es el avance que se desea
% tamafin es la cantidad de total de articulaciones
% eliminapos es el vector de articulaciones que llegaron al limite
% signos es el vector que contiene el signo entre el limite final e inicial
% de las articulaciones
% ejemplo
%
% n=6;
% paso=1;
% tamafin=9;
% eliminapos=[5 6 9];
% signos=[-1 1 1 -1 1 1 -1 1 1];
% deltaq=selec_articulacion_excluida(n,paso,tamafin,eliminapos,signos)
%
% Autor Cristian David Villate Martinez
function deltaq=selec_articulacion_excluida(n,paso,tamafin,eliminapos,signos)
% selecion de articulacion aleatoria
art=fix(rand(1)*n)+1;
if(art==n+1)
    art=n;
end
% seleccion de la articulacion cuando existen articulaciones que llegaron
% al limite
if(length(eliminapos)>0)
    sumexclu=1;
    sumaart=1;
 % dato de validacion de llegada
    llego=0;
 % recorriendo el vector para encontrar las articulaciones que llegaron al
 % limite
    for i=1:tamafin
     % detectado articulaciones que llegaron al limite
        if(eliminapos(sumexclu)==i)
```

```
            sumexclu=sumexclu+1;
            if(sumexclu>length(eliminapos))
              sumexclu=length(eliminapos);
            end
        else
     % obtención de la articulación
         if(sumaart==art)&&(llego==0)
            art=i;
            llego=1;
        end
            sumaart=sumaart+1;
        end

    end
end
% creacion del vector articular de paso teniendo encuenta el paso y el
% signo
deltaq=zeros(1,tamafin);
deltaq(art)=signos(art)*paso;



% signos_pasos.m
%
% esta funcion se encarga de generar un vector que dependiendo la distancia
% entre el punto final e inicial determina el signo del paso que se
% encuentra entre los dos.
% qini es el punto inicial
% qfin es el punto final
% articulaciones es el numero de articulaciones
% ejemplo
%
% qini=[0 0 0 0 0 0 0 0 0];
% qfin=[-15 10 30 -20 0 0 -50 73 0];
% articulaciones=9;
% signos=signos_pasos(qini,qfin,articulaciones)
%
% Autor Cristian David Villate Martinez
function signos=signos_pasos(qini,qfin,articulaciones)
% aisignacion de signos
for i=1:articulaciones
    if(qini(i)<=qfin(i))
        signos(i)=1;
    else
        signos(i)=-1;
    end
end
end



% Trayectoria_Robot_Bioloid.m
%
% esta funcion se encarga de generar una matriz de trayectorias de un
% punto inicial a un punto final por unos puntos no validos
```

```
% qini es el punto inicial
% qfin es el punto final
%
% ejemplo
%
% qini=[0 0 0 0 0 0 0 0 0];
% qfin=[-15 10 30 -20 0 0 -50 73 0];
% M=Trayectoria_Robot_Bioloid(qini,qfin);
%
% Autor Cristian David Villate Martinez
function M=Trayectoria_Robot_Bioloid(qini,qfin)
%datos basicos
NUMERO_ITERACIONES = 2000;
q_anterior=qini;
% actualiza el vector de coordenadas articulares
q=qini;
funciona=1;
iteracion=0;
iteracion_total=0;
articulaciones=length(qini);
pasos=1;
pasos_aux=pasos;
choque_estatico=0;
primer_choque=0;
primer_choque_total=0;
choque_encerrado=0;
% aisignacion de signos
 signos=signos_pasos(qini,qfin,articulaciones);
%ciclo generador de trayectorias
while(funciona==1)
    iteracion=iteracion+1;
    iteracion_total=iteracion_total+1;
    % superacion de iteraciones maximas
    if(iteracion_total>=NUMERO_ITERACIONES)
    % datos basicos de reinicio
        q=qini;
        q_anterior=qini;
         M=[];
       Msin_choque=Msin_choque_max;
        funciona=1;
         iteracion=1;
        iteracion_total=1;
        choque_estatico=0;
        primer_choque=0;
        choque_encerrado=0;
     % aisignacion de signos
        signos=signos_pasos(q,qfin,articulaciones);
    end
% avance cuando existe llegadas y choques igual a las articulaciones
if(choque_estatico==1)
    % evalucación de avanze articular
    if(q(poschoque)~=articulacion_choq)
      % aisignacion de signos
      signos=signos_pasos(q,qfin,articulaciones);
```

```
      choque_estatico=0;
      % pocición y cantidad de articulaciónes iguales
      eliminapos=find(q==qfin);
    else
      % simulación de no existencia de articulaciónes
      % iguales
 eliminapos=[];
    end
else
  % pocición y cantidad de articulaciones iguales
    eliminapos=find(q==qfin);
end
% numero de articulaciones que no an llegado al limite
   articulai=articulaciones-length(eliminapos);
% seleccion de articulacion

deltaq=selec_articulacion_excluida(articulai,pasos,articulaciones,eliminapos,sig
nos);
    q=q+deltaq;
% cálculos cinematicos del Robot Bioloid
[angulosf, movimiento]=Bioloid_Calculos(q);
% movimiento no valido
    if (movimiento==0)
% datos basicos de evaluacion de movimiento no valido
        q=q_anterior;
             contador_choque=0;
             iteracion_choque=0;
             contador_llegada=0;
             % ciclo de evaluacion de choques y llegadas articulares

while((iteracion_choque==(contador_choque+contador_llegada))&&(iteracion_choque<
articulaciones))
                qchoque=q;
                sumachoque=zeros(1,articulaciones);

sumachoque(iteracion_choque+1)=sumachoque(iteracion_choque+1)+(pasos*signos(iter
acion_choque+1));
                qchoque=qchoque+sumachoque;
                % cálculos cinematicos del Robot Bioloid
                [angulosf, movimiento]=Bioloid_Calculos(qchoque);
            % contador de movimientos no validos
            if (movimiento==0)
                contador_choque=contador_choque+1;
            else
             % contador de llegadas articulares
                if(q_anterior(iteracion_choque+1)==qfin(iteracion_choque+1))
                    contador_llegada=contador_llegada+1;
                 end
            end
            iteracion_choque=iteracion_choque+1;
            end
            % detección de movimientos donde no puede existir avances
             if(contador_choque==articulaciones)
            % asignación de datos a antes de la existencia de un choque
```

```
                M=Msin_choque;
                q=qprimer;
                q_anterior=q_anterioruno;
                iteracion=iteracionguar;
                angulosf=angulosfuno;
                primer_choque=0;
                choque_encerrado=1;
             % detección de la articulación con mayor trayectoria restante
                [maxima_art,pos]=max(abs(qfin-q));
             % cambio de sentido de avance excepto la articulación con
             % mayor trayectoria
                for i=1:articulaciones
                    if(i~=pos)
                     signos(i)=signos(i)*-1;
                    end
                end
             end
            % detección de choques donde existen avances mas lejos de los
            % limites finales
            if((contador_choque+contador_llegada)==articulaciones)
                % detección de la articulación con mayor trayectoria restante
                  [maxima_art,poschoque]=max(abs(qfin-q));
                 articulacion_choq=q(poschoque);
                % activacion de choques donde existen avances mas lejos de
                % los limites finales
                 choque_estatico=1;
            end
        % asignación de los datos del primer choque
        if iteracion>1
            iteracion = iteracion-1;
            if (primer_choque==0)&&(iteracion>1)
        % obtención de la primera trayectoria antes de existir un choque en
        % todo el proceso
             if(primer_choque_total==0)
         [iteracionguar primer_choque angulosfuno qprimer q_anterioruno
Msin_choque]=fprimer_choque(M,q,iteracion,articulaciones);
         primer_choque_total=1;
        % trayectoria maxima antes de existir un choque
         Msin_choque_max=Msin_choque;
         qprimer_max=qprimer;
            else
         % determinación de la cantida de angulos que llegaron tanto actuales
como
         % de la matriz maxima
                tamchoque_uno=length(find(qfin==q));
                tamchoque_ant=length(find(qfin==qprimer_max));
         % nueva asignación de una trayectoria maxima antes de la
         % existencia de un choque
              if(tamchoque_uno>tamchoque_ant)
              [iteracionguar primer_choque angulosfuno qprimer q_anterioruno
Msin_choque]=fprimer_choque(M,q,iteracion,articulaciones);
               Msin_choque_max=Msin_choque;
               qprimer_max=qprimer;
              end
```

```
            end
           end
        end
    end
  % movimiento transitorio
    if(movimiento==2)
     contador_mov2=1;
     valido_mov2=0;
     pasos_aux=pasos;
  % ciclo de deteccion de movimiento fuera del rango de los movimientos
  % transitorios
     while (valido_mov2==0)
       qchoque=q_anterior;
       sumachoque=zeros(1,articulaciones);

sumachoque(contador_mov2)=sumachoque(contador_mov2)+(pasos_aux*signos(contador_m
ov2));
       qchoque=qchoque+sumachoque;
        % cálculos cinematicos del Robot Bioloid
       [angulosf, movimiento]=Bioloid_Calculos(qchoque);
       % obtención de el numero de articulaciones que llegaron al limite
       eliminapos=find(qchoque==qfin);
       tam2=length(eliminapos);
       [maxima_art_mov2,poschoque_mov2]=max(abs(qchoque));
       % deteccion de cambio de movimiento de transitorio a valido

if(movimiento==1)&&((q_anterior(contador_mov2)~=qfin(contador_mov2)))||(tam2==ar
ticulaciones)
           valido_mov2=1;
           q=qchoque;
       end

if(movimiento==0)&&(maxima_art_mov2>abs(q_anterior(poschoque_mov2)))&&(iteracion
>1)
          M=Msin_choque;
           q=qprimer;
           q_anterior=q_anterioruno;
           iteracion=iteracionguar;
           angulosf=angulosfuno;
           primer_choque=0;
           valido_mov2=-1;
       end
       contador_mov2=contador_mov2+1;
         % aumento de los pasos cuando los movimientos articulares son
       % inferiores al movimiento transitorio
       if(contador_mov2>articulaciones)
        pasos_aux=pasos_aux+1;
         contador_mov2=1;
       end
     end
    end
 % cambio de sentido cuando pasa de un movimientos donde no puede existir
avances
  if(choque_encerrado==1)
```

```
      if(q(pos)~=qprimer(pos))
 % aisignacion de signos
 signos=signos_pasos(q,qfin,articulaciones);
choque_encerrado=0;
      end
  end
% cambio de vector articular precente a anterior
    q_anterior = q;
% matriz de trayectoria actualizada
    M(iteracion,:)=angulosf;
    % solucion final de la trayectorias
    if(sum(abs(q-qfin))<=0.1)
   % cálculos cinematicos del Robot Bioloid
  [angulosf, movimiento]=Bioloid_Calculos(qini);
        M=[angulosf;M];
         M=fix(M);
        funciona=2;
    end
end
% cambio de coordenadas articulares de coordenadas cartecianas
final=length(M);
for i=1:final
    [angulosf,coorefec, movimiento]=Bioloid_Calculos_mod(M(i,:));
    coordendas_totales(i,:)=coorefec;
end
  coordendas_totales_mod=fix(coordendas_totales);
  coordendas_totales_mod_dos=coordendas_totales_mod;
  Mmod=M;
% reduccion de coordenadas
for cont_uno=1:final
    cont_dos=1;
    salida=1;
    while(salida==1)
        if((cont_dos)<=final)

if(coordendas_totales_mod(cont_uno,:)==coordendas_totales_mod_dos(cont_dos,:))

coordendas_totales_mod_dos=[coordendas_totales_mod_dos([1:cont_dos],:);coordenda
s_totales_mod([(cont_uno+1):final],:)];
              Mmod=[Mmod([1:cont_dos],:);M([(cont_uno+1):final],:)];
              salida=0;
            else
              cont_dos=cont_dos+1;
            end

        else
          salida=0;
        end
    end
end
M=Mmod;
disp('LLEGO A LA SOLUCION')
end
```

```
% Variables_Globales
global ERRBIT_VOLTAGE
global ERRBIT_ANGLE
global ERRBIT_OVERHEAT
global ERRBIT_RANGE
global ERRBIT_CHECKSUM
global ERRBIT_OVERLOAD
global ERRBIT_INSTRUCTION

global COMM_TXSUCCESS
global COMM_RXSUCCESS
global COMM_TXFAIL
global COMM_RXFAIL
global COMM_TXERROR
global COMM_RXWAITING
global COMM_RXTIMEOUT
global COMM_RXCORRUPT


%%%%%

global P_TORQUE_ENABLE
global P_LED
global P_GOAL_POSITION
global P_GOAL_VELOCITY
global P_PRESENT_POSITION
global P_PRESENT_VELOCITY      %Valor Max 400 equivalente a 44RPM
global P_PRESENT_LOAD
global DEFAULT_PORTNUM
global DEFAULT_BAUDNUM
global P_Moving

%%%% Variables calculadas para la velocidad

global KVrgAX12 % Esta constante se calcula dependiendo del voltaje de
                % alimentación, se puede usar el programa:
                % prueba_velocidad_motores para calcular su valor

global KVgrAX12 % Es el inverso de KVrgAX12

% video de marcha del robot bioloid
% Autor Cristian David Villate Martinez
Matriz1=eye(4);
Matriz2=[1 0 -2.22044604925031e-16 57.3018246119400;0 1 0 78;2.22044604925031e-
16 0 1 -0.0759617536599628;0 0 0 1];
Matriz3=[1 0 -2.22044604925031e-16 114.603649223880;0 1 0 0;2.22044604925031e-16
0 1 -0.0759617536599628;0 0 0 1];
movimiento0=zeros(1,9);
movimiento1=[-14 0 47 -46 0 0 0 1 0];
movimiento2=[-5 -10 20 -15 0 0 -70 50 0];
movimiento3=[-12 0 0 -1 0 0 -47 46 0];
movimiento22=[-5 10 20 -15 0 0 -70 50 0];
M1=Trayectoria_Robot_Bioloid(movimiento1,movimiento22);
```

```
M2=Trayectoria_Robot_Bioloid(movimiento22,movimiento0);
trayectorias3=[M1;M2];
M1=Trayectoria_Robot_Bioloid_prue(movimiento1,movimiento2);
M2=Trayectoria_Robot_Bioloid_prue(movimiento2,movimiento3);
trayectorias2=[M1;M2];
M1=Trayectoria_Robot_Bioloid(movimiento0,movimiento22);
M2=Trayectoria_Robot_Bioloid(movimiento22,movimiento3);
trayectorias1=[M1;M2];
fin=length(trayectorias1);
load('bioloid_piezas_final.mat');
 fig = figure; set(fig, 'DoubleBuffer', 'On');
 mov = avifile('Bioloid_marcha_total_3D_16_10_16.avi');
for i=1:fin
    angulos=trayectorias1(i,:);
     Bioloid_graficas_derecho(angulos,Matriz1,bioloid)
    pause(0.0000001)
     F = getframe(fig);
    mov = addframe(mov, F);
     clf
end
fin=length(trayectorias2);
for i=1:fin
    angulos=trayectorias2(i,:);
     Bioloid_graficas_izquierdo(angulos,Matriz2,bioloid)
      pause(0.0000001)
     F = getframe(fig);
    mov = addframe(mov, F);
     clf
end
fin=length(trayectorias3);
for i=1:fin
    angulos=trayectorias3(i,:);
     Bioloid_graficas_derecho(angulos,Matriz3,bioloid)
      pause(0.0000001)
     F = getframe(fig);
    mov = addframe(mov, F);
     clf
end
mov = close(mov);
```