DISEÑO DE SISTEMA ELECTRÓNICO JERÁRQUICO MEDIANTE COMANDOS PARA ROBOT HEXÁPODO.

JORGE MAURICIO ESCOBAR OLARTE.
JUAN DAVID SILVA ORTEGA.

BOGOTÁ D.C.
PONTIFICIA UNIVERSIDAD JAVERIANA.
FACULTAD DE INGENIERIA.
CARRERA DE INGENIERIA ELECTRÓNICA.
2011.

DISEÑO DE SISTEMA ELECTRÓNICO JERÁRQUICO MEDIANTE COMANDOS PARA ROBOT HEXÁPODO.

JORGE MAURICIO ESCOBAR OLARTE.
JUAN DAVID SILVA ORTEGA.
Trabajo de grado para optar al título de
Ingeniero Electrónico
DIRECTOR:
Ing. Eduardo Andrés Gerlein Reyes. M.Sc

BOGOTÁ D.C.
PONTIFICIA UNIVERSIDAD JAVERIANA.
FACULTAD DE INGENIERÍA.
CARRERA DE INGENIERÍA ELECTRÓNICA.
2011.

ARTÍCULO 23 DE LA RESOLUCIÓN No. 13 DE JUNIO DE 1946.

"La universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado.
Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque los trabajos no contengan ataques o polémicas puramente personales. Antes bien, que se vea en ellos el anhelo de buscar la verdad y la justicia".

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA ELECTRÓNICA

RECTOR MAGNÍFICO: R.P. JOAQUÍN SÁNCHEZ GARCÍA, S.J.
DECANO ACADÉMICO: Ing. FRANCISCO JAVIER REBOLLEDO MUÑOZ.
DECANO DEL MEDIO UNIVERSITARIO: P. SERGIO BERNAL, S.J
DIRECTOR DE CARRERA: Ing. JUAN MANUEL CRUZ BOHÓRQUEZ. M.Ed.
DIRECTOR DEL PROYECTO: Ing. EDUARDO ANDRÉS GERLEIN REYES. M.Sc

TABLA DE CONTENIDO.

1. INTRODUCCIÓN.

Actualmente, en los programas académicos dedicados al desarrollo de hardware y software es de vital importancia disponer de herramientas y plataformas apropiadas para lograr un buen desarrollo de las capacidades adquiridas en la ingeniería, a través de la teoría y complementarla con la experiencia. En el área de la robótica, disponer de una plataforma completa, la cual contenga todos los elementos necesarios para desarrollar una buena y completa practica, conlleva a una óptima familiarización entre el estudiante y el área.

Se pretende habilitar una plataforma robótica - en particular un robot hexápodo - con el fin de que sea utilizada en proyectos de investigación, laboratorios, trabajos de grado, etc.

En algunas investigaciones acerca de robótica móvil abarcan en su mayoría lo correspondiente [1] [2] [3] a plataformas móviles de locomoción fija, es decir tipo ruedas y orugas. Por lo tanto, para aplicaciones donde el tipo de terreno o superficie sea irregular o posea obstáculos estos tipos de robots quedan en términos de desplazamiento muy impedidos.

Con la necesidad de avanzar en el campo de la robótica móvil, se han realizado investigaciones desarrollando máquinas caminantes para resolver los problemas de locomoción que tenían los anteriores robots, por lo cual estos tipos de robots satisfacen las expectativas de movilidad en terrenos irregulares, ya que estos nuevos al tener extremidades logran tipos de movimientos coordinados y estructurados con el fin de evadir obstáculos y adaptarse a dichos terrenos. El proyecto BIGDOG[4] es una de las mejores muestras para referenciar lo anterior mencionado. Este robot cuadrúpedo fue creado por la empresa de ingeniería y robótica Boston Dynamics, en compañía de Foster-Miller, Jet Propulsion Laboratory y Harvard University Concord Field Station; financiado por Defense Advanced Research Projects Agency. Este proyecto tiene el fin de ser un transportador de carga para los soldados en situaciones donde el terreno presenta altas irregularidades como para introducir vehículos. [4]

Para el presente trabajo se utilizó una plataforma tipo hexápodo modelo Rugwalker Hexapod II de Lynxmotion Inc, adquirido por la Universidad Javeriana en el año 2006.

El proyecto consiste en elaborar un sistema electrónico jerárquico mediante comandos para un robot Hexápodo, por lo cual es necesario obtener un modelo matemático para examinar el comportamiento cinemático del robot.

Analizando aspectos como estabilidad estática y dinámica se podrá establecer una estrategia la cual será implementada en un software capaz de generar comandos de control a la plataforma con el fin de validar el modelo matemático y las rutinas programadas.

La plataforma cuenta con un hardware el cual implementa un protocolo de comunicación serial, software y documentación apropiada para que ser programado de manera sencilla y en un futuro, llevar a cabo diferentes tareas. También se procederá a validar la estrategia de movimiento de la plataforma por medio de la implementación de al menos una rutina de movimiento con el fin de certificar su correcto funcionamiento. Adicionalmente, se pondrá a disposición la información acerca de la caracterización y modelamiento del robot, en los manuales de usuario. Se pretende que el hardware implementado sea modular, facilitando de tal manera realizar labores de mantenimiento.

El trabajo desarrollado se dividió en cuatro etapas; una primera etapa en la que se describe los aspectos más destacados con los cuales se desarrollo el proyecto. Entre ellos se relaciona el algoritmo de *Fuzzy C-means* implementado para desarrollar la base del modelo matemático. En la segunda etapa se encuentra una descripción de la plataforma refiriendo sus principales características como tipo de material utilizado, actuadores, transmisores, entre otros.

La tercera etapa representa el análisis cinemático del robot. Se obtiene un modelo de las extremidades del robot con fines de diseño de la rutina de movimiento sobre las mismas, centrándose en el cumplimiento del objetivo especifico:

- Obtener un modelo de las extremidades del robot con fines de diseño de la rutina de movimiento sobre las mismas.

La cuarta parte se encuentra divida en dos secciones. La primera corresponde a la descripción física del hardware, describiendo su funcionalidad y la implementación que se le dio. En la segunda parte se tiene el análisis sobre el software diseñado, esta parte contempla lo relacionado al protocolo de comunicación.

En esta sección se realiza un enfoque de los siguientes objetivos específicos:

- Diseñar e implementar una rutina de movimiento basada en comandos para las extremidades.
- Diseñar e implementar un módulo maestro que incluya comunicaciones y abierto a futuras aplicaciones.

Por último, en la etapa de pruebas de calidad y desempeño, se describen y evidencian las validaciones realizadas con el fin de realizar una comparación sobre los resultados obtenidos a partir del modelo matemático.

En esta parte del proyecto se obtiene el siguiente objetivo específico

- Validar el sistema programado mediante la ejecución de al menos una estrategia de movimiento sobre la plataforma.

De igual manera, se adjunta un MANUAL de usuario con el fin de brindar toda la información necesaria para realizar trabajos de programación, diseño de rutinas de movimiento, sobre la plataforma.

2. OBJETIVOS.

2.1. Objetivo General.

Diseñar e Implementar un sistema electrónico jerárquico mediante comandos para un robot Hexápodo.

2.2. Objetivos Específicos.

1. Obtener un modelo de las extremidades del robot con fines de diseño de la rutina de movimiento sobre las mismas.
2. Diseñar e implementar una rutina de movimiento basada en comandos para las extremidades.
3. Diseñar e implementar un módulo maestro que incluya comunicaciones y abierto a futuras aplicaciones.
4. Validar el sistema programado mediante la ejecución de al menos una estrategia de movimiento sobre la plataforma.
5. Elaborar un documento manual para ayudar a futuros usuarios a programar la plataforma para aplicaciones particulares.

3. MARCO TEÓRICO.

3.1. ROBOTS HEXÁPODOS.

Según la R.A.E. (Real Academia Española) define la palabra Hexápodo como **"**adjetivo. *Zoología.* Dicho de un animal, y especialmente de un insecto: Que tiene seis patas. U. t. c. s. (usado también como sustantivo)"[5].

El hexápodo es una estructura animada o inanimada que consta de seis extremidades la cuales están ubicadas paralelamente entre ellas en un cuerpo, dotado de movimiento voluntario o controlado. Como ejemplo de seres animados hexápodos se encuentran alguna variedad de insectos como las hormigas; y como inanimados la representación de estos mismos seres en forma robótica.

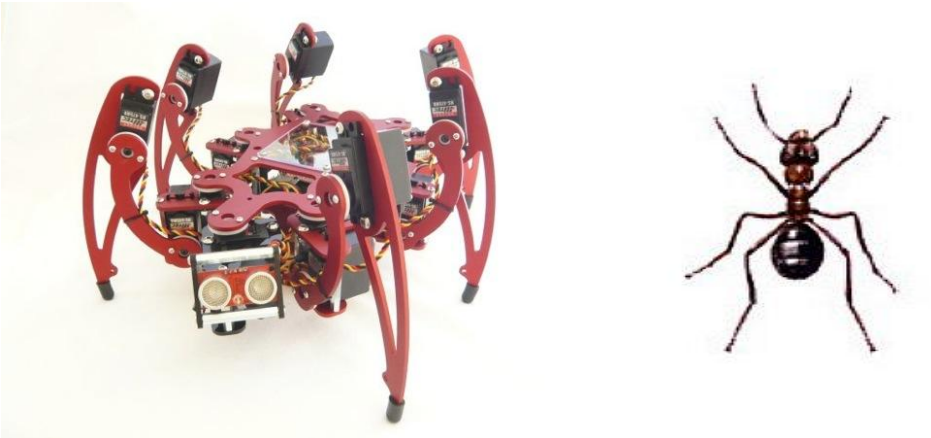


Figura 1.0. Representación animada e inanimada de una estructura Hexápodo. (Tomadas de [6] y [7])

En el desarrollo de la robótica móvil, los robots con extremidades han servido como alternativa ante los robots con ruedas, debido a su gran asequibilidad de movimiento en terrenos no estructurados [8].

El sistema de locomoción Hexápodo ofrece mayor versatilidad que los robots móviles con ruedas en dichas aplicaciones. La complejidad de estas estructuras depende en especial del grado de libertad que se tiene para el desarrollo del movimiento, el sincronismo entre las extremidades, y de las aplicaciones que se les desee [8].

La clasificación de los robots hexápodos está caracterizada según varios parámetros entre los cuales el más importante es la aplicación o uso, ya que estos tipos de robots son muy utilizados

en diferentes campos desde investigación, exploración de espacios hasta de carácter lúdico; y para cada tipo de aplicación su programación o estrategias de control varían.[8].

Aspectos como la programación, control, grados de libertad, morfología, limitaciones mecánicas, entre otras son necesarios de tener en cuenta a la hora de elaborar un robot hexápodo.

En la figura 1.1 se puede apreciar la proporción de trabajos en los que se implementan los robots con extremidades, en los cuales se puede evidenciar que la implementación de robots con seis extremidades es decir hexápodos es dominante con respecto a los demás.

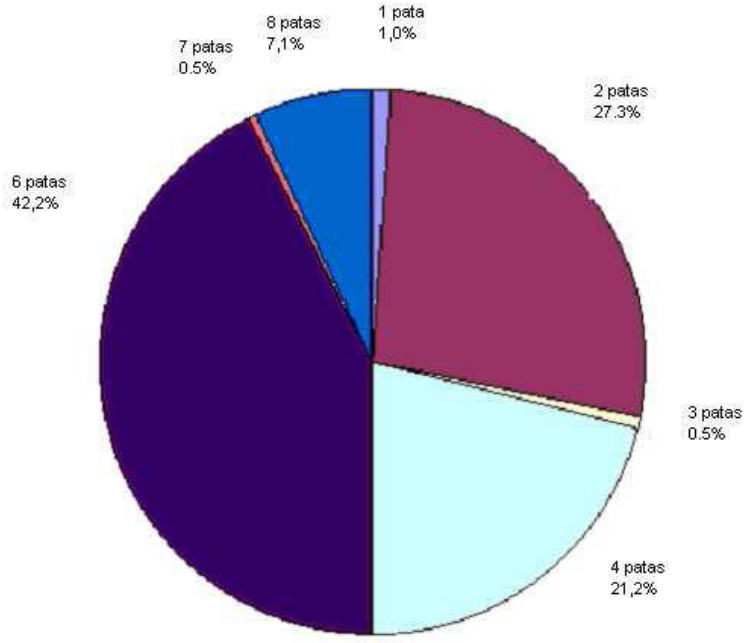


Figura 1.1. Proporción de trabajo con robots de diferentes numero de extremidades. (Tomada de [8])

3.1.1. HEXÁPODO SEGÚN GRADOS DE LIBERTAD.

La construcción de un hexápodo con solo un grado de libertad es la más sencilla de todas, ya que esto requiere pocos actuadores y sus extremidades serán completamente rígidas realizando solamente un tipo de movimiento efectuado por el actuador. Al hacer esto no requiere una programación de control tan compleja. La configuración habitual de estos tipos de robots es hacer que las patas centrales se encarguen de hacer subir o bajar al robot y las demás de la parte de avanzar o retroceder.

Figura 1.2. Hexápodo de un grado de libertad. (Tomado de [8])

Los hexápodos de dos grados de libertad son los más utilizados en aspectos lúdicos y de investigación. Éstos por tener dos grados de libertad requieren mayor número de actuadores comparados con los anteriores y necesitan una programación de control más compleja, ya que es necesaria una sincronización de coordinación entre sus movimientos.

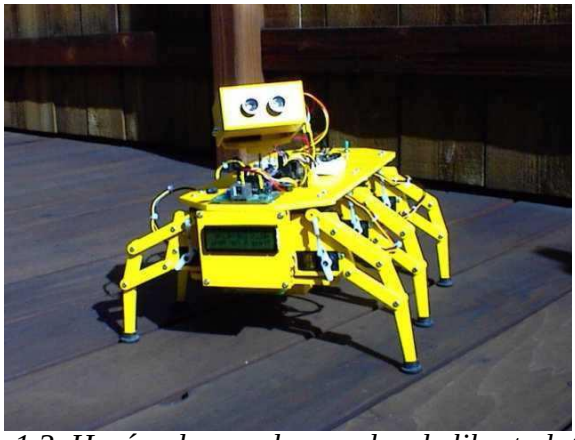


Figura 1.3. Hexápodo con dos grados de libertad. (Tomado de [8])

Existen también hexápodos con estructuras más complejas, esto conlleva a la implementación de mejores mecanismos de control con los cuales se obtienen un mayor grado de libertad. [8]

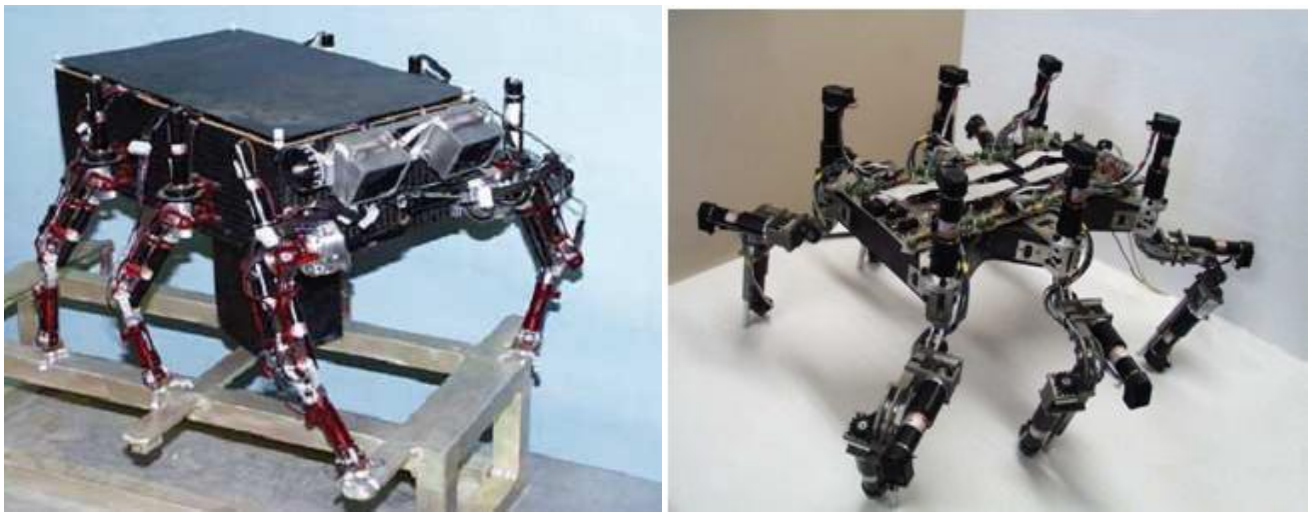


Figura 1.4. Hexápodos con más grados de libertad (estructuras complejas). (Tomado de [8])

3.1.2. CONFIGURACIÓN FÍSICA.

Los robots Hexápodos permiten un mayor desarrollo de los aspectos de planificación de movimiento. Dos conceptos importantes que se deben tener en cuenta a la hora de elaborar robots con extremidades es la estabilidad estática y estabilidad cinemática.

La estabilidad estática se refiere a la capacidad del robot de permanecer en equilibrio (sin caerse) cuando no está en movimiento, mientras que la estabilidad cinemática hace referencia a la capacidad de permanecer en equilibrio durante su movimiento, es decir se debe desarrollar una estrategia de movimiento para lograr un desplazamiento limpio (sin caerse) y que sus extremidades sean coordinadas.

Los robots hexápodos tienen ventaja en comparación a los robots con un número de extremidades inferior, debido a que existe una mayor cantidad de configuraciones disponibles en las que el sistema conserva un polígono de estabilidad definido en todas las fases de su protocolo de movimiento. [8] [9] [10]

Dentro de la configuración para el posicionamiento de las extremidades se tienen dos:

3.1.2.1. Configuración Bilateral.

Esta configuración presenta una simetría a lo largo del eje longitudinal del robot. Tiene ventaja a la hora de la programación de los movimientos, ya que la configuración física del robot aporta para el avance con movimientos paralelos, pero presenta inconvenientes en otro tipo de

movimientos, en especial en movimiento de giros. Esto se puede arreglar implementando en la programación de control una parte exclusivamente para el giro pero presenta complicaciones en el desarrollo del software ya que se debe realizar estrategias de movimiento más robustas y con mayor numero de estados [8].

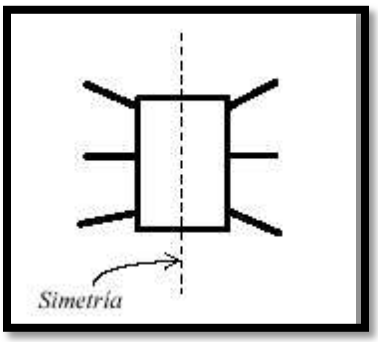


Figura 1.5. Configuración bilateral de las extremidades de un robot hexápodo. (Tomado de [8])

Este tipo de configuración es similar a la morfología de algunos insectos entre los cuales se puede encontrar el de la hormiga, debido a la disposición de las extremidades con su estructura física.

Figura 1.6 Estructura física de una Hormiga. (Tomado de [8])

3.1.2.2. Configuración Radial.

Esta configuración presenta una distribución de las extremidades en forma circular lo cual no presenta problemas de desplazamiento, ya que su movimiento en cualquier dirección es igual. Es decir que estos tipos de robots son holonómicos.

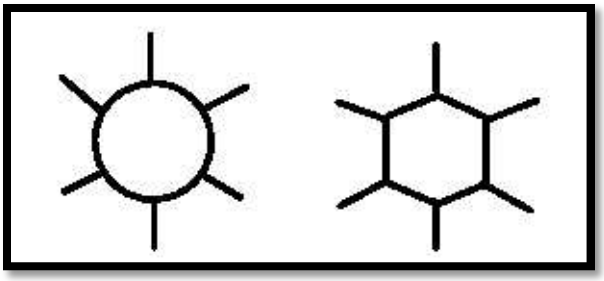


Figura 1.7. Configuración radial para un robot Hexápodo. (Tomado de [8])

Este tipo de configuración se asemeja a la morfología de una araña Loxosceles laeta, debido a la disposición de las extremidades entorno al cuerpo de la misma.

Figura 1.8 Estructura física de Araña Loxosceles laeta. (Tomado de [11])

3.1.3. MATERIALES DE LA ESTRUCTURA.

Entre los aspectos importantes en un robot se encuentran la estructura física y programación o control, ya que dependiendo de la aplicación (objetivo final, entorno de operación, entre otros) destinada para la estructura se deben tener en cuenta ciertos aspectos como el grado de resistencia de los materiales con los que se va a construir la estructura.

Entre los materiales más utilizados a la hora de construir plataformas robóticas se tienen: aluminio, metraquilato, fibra de carbono, acrílico, madera, PVC, acero y cobre. [8]

Dentro de los componentes que se deben tener presentes se encuentran los mecanismos de movimiento, los actuadores, transmisores, reductores, sensores etc.

Para los robots hexápodos los actuadores utilizados más comúnmente son los servomotores.

Figura 1.9 Plataformas Hexápodos desarrolladas con Aluminio y Madera. (Tomado de [58])

3.1.3.1. ACTUADOR.

Los Actuadores son los dispositivos encargados de generar el movimiento de los elementos móviles del robot según las órdenes dadas por la unidad de control; entre la diversidad de los actuadores, el servo motor es uno de los dispositivos más utilizados en la robótica.

Unas de las tantas razones que justifican el uso del servomotor y que lo hacen asequible al trabajo en esta área, son las dimensiones, consumo de energía, costo y sistema de control.

3.2. SERVOMOTOR.

El servomotor es un dispositivo electrónico que posee un eje controlado. Este puede llevarse a una posición específica y mantenerse dependiendo de la señal de control que se le ingrese en torno a un rango de operación.

El servomotor consta de un conversor de ancho de pulso a voltaje, un amplificador de error, una resistencia variable (potenciómetro) que está conectada al eje principal del motor, engranajes y un motor de corriente directa DC.

El tamaño de los servomotores dependiendo de su aplicación al igual que, el torque, la velocidad y el peso varían, permitiéndoles adaptarse a casi cualquier tipo de trabajo, son comúnmente usados en proyectos de robótica debido a su facilidad de control en la posición angular del eje de salida.[12][13]

Existen distintos tipos de servomotores, algunos de esos son:

- Servomotores de CC.
- Servomotores de AC.
- Servomotores de imanes permanentes.

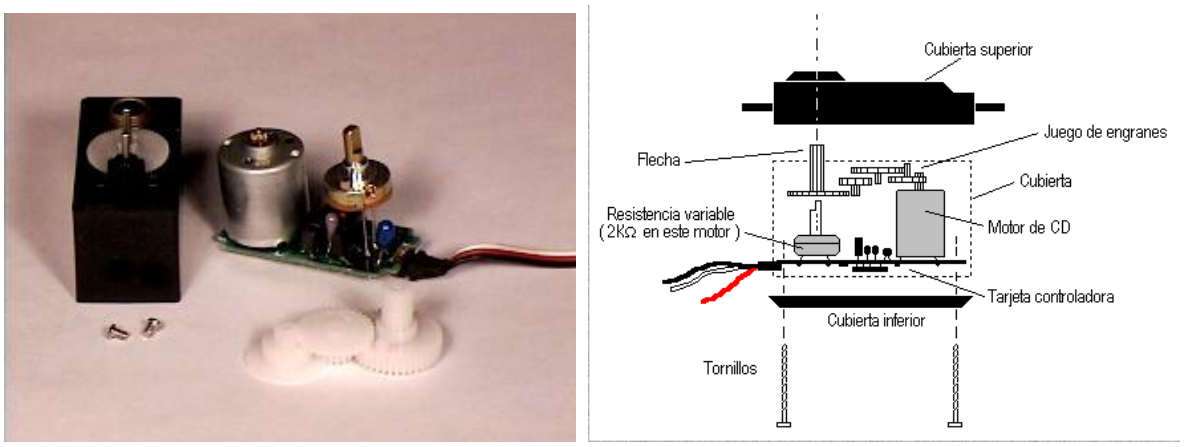


Figura 1.10. Componentes físicos de un Servo motor. (Tomado de [12])

3.2.1. FUNCIONAMIENTO.

Para la ubicación del servomotor en un punto deseado, éste utiliza un sistema de control interno que mediante una señal externa obtendrá el punto de referencia requerido.

La señal con la cual se realiza el control de posición es de tipo cuadrada con un periodo fijo. La variación del ancho de pulso de esta señal será la encargada de modificar el ángulo de posición, a medida que se incremente su ancho de pulso ubicará el motor en un ángulo mayor.

Un potenciómetro conectado por un costado directamente al eje principal del servo y por el otro al amplificador de error, permite supervisar el ángulo o estado actual del servomotor, ya que al rotar el servomotor el potenciómetro también lo hará y esto indicará un valor en voltaje en una de las entradas del amplificador.

Como la información de la ubicación que se desea está indicada por medio del ancho de pulso de la señal de control, para poderla comparar con la señal de voltaje producida por el potenciómetro se necesita un conversor de ancho de pulso-voltaje generando así su representación en una señal de voltaje que por medio del sistema diferencial del amplificador modificará la posición del eje de salida hasta que los valores se igualen y el servo pare en la posición indicada. [12][13][14][15]

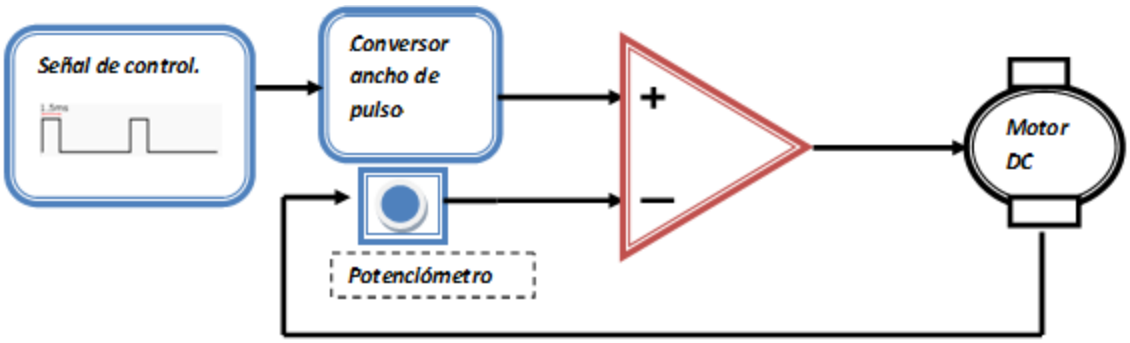


Figura 1.11 Diagrama de bloques funcional de un servomotor eléctrico.

3.2.1.1 Señal De Control.

La señal de control está parametrizada para los servos tipo *Standar* (*ver tabla 1.4*) los cuales se trabajaron en este proyecto. En la *Figura 1.12* se muestra la variación en la posición final con respecto al ancho de pulso.

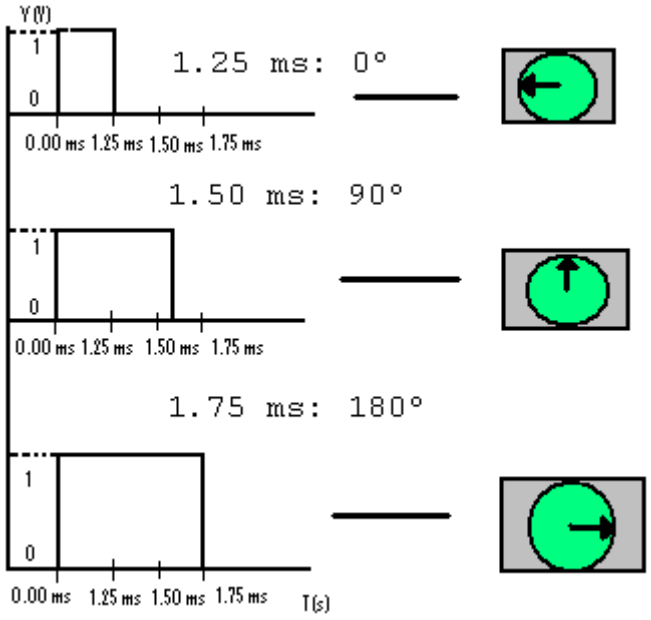


Figura 1.12. Variación de la posición de un servo motor con respecto al ancho de pulso.

En cuanto a los tiempos de ON y OFF de la señal de control, se debe tener en cuenta que la señal de OFF normalmente está alrededor de 20ms pero no es crítica, ésta podría variar aproximadamente entre 10ms y 30ms si se supera el tiempo superior el servo puede entrar en

modo de *Sleep* por lo cual su funcionamiento no será adecuado y su reacción se verá reflejado en un movimiento corto. El tiempo de ON varía entre 1 y 2 ms y es este pulso el que determina la posición del servo. El servo estándar dispone de 3 cables, 1 cable para fuente (Rojo o rayado) ,1 cable de tierra (Azul o Negro) y 1 cable de control (Amarillo o blanco).

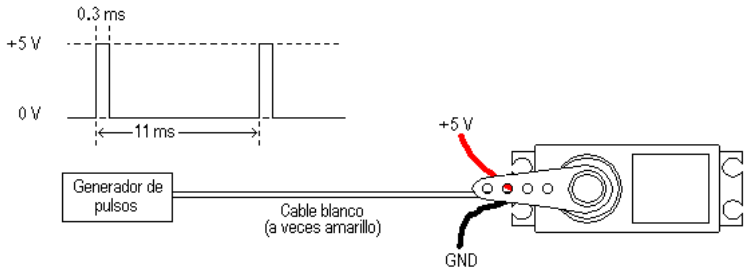


Figura 1.13. Configuración de cableado para el servo motor. (Tomado de [14])

La alimentación del servo varía conforme al fabricante, y para obtener un buen desempeño en el trabajo a realizar, es recomendable disponer de la hoja de características (*datasheet*) correspondiente al modelo y fabricante.

3.3. ESPACIO O VOLUMEN DE TRABAJO.

El robot tiende a tener una geometría fija, y limitada. El espacio de trabajo es el límite de posiciones en espacio que el robot o sus componentes pueden alcanzar, es decir que este espacio depende de la estructura física de los elementos de movimiento del robot y de sus grados de libertad. [16]

Existen varios tipos de regiones de trabajo y dependiendo de la estructura del robot difieren, para eso es necesario analizar dependiendo del tipo del robot con que se trabaje.

3.3.1. Configuración Cartesiana.

Esos tipos de robots presentan espacios de trabajo regulares, espacios cúbicos. Pueden poseer de uno a tres ejes de referencia. Las posiciones de las articulaciones por lo general son ubicadas ortogonalmente a las coordenadas de la posición del efector final, haciendo que su análisis cinemático sea posible de analizar. Sus movimientos de un punto a otro punto son lineales.

Un ejemplo de esta configuración se puede observar en máquinas como el Agitador Orbital [17] , utilizado para realizar diagramas sobre materiales en grandes producciones o para la fabricación de piezas complejas con gran número de detalles.

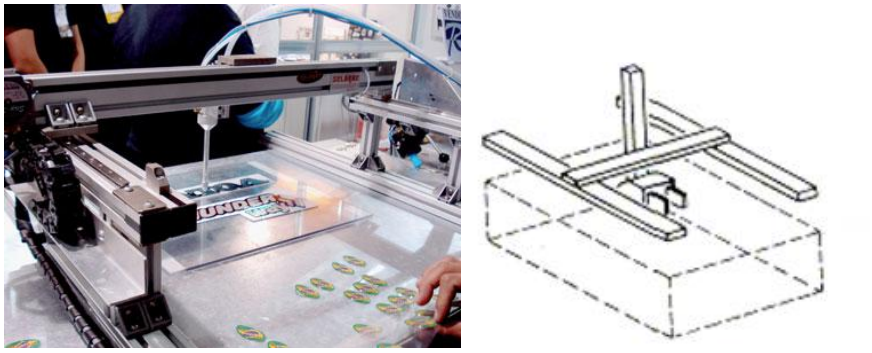


Figura 1.14 Volumen de trabajo cartesiano (izquierda), Agitador Orbital (derecha ejemplo robot cartesiano). (Tomado de [16] [17])

3.3.2. Configuración Cilíndrica.

El robot de configuración cilíndrica presenta un volumen de trabajo similar a un cilindro. (En algunas ocasiones este robot no tiene una rotación de 360°). Estos tipos de configuraciones se pueden observar en los robots SCARA (Selective Compliant Assembly Robot Arm o brazo de robot de montaje selectivamente compatible) utilizados en empresas del sector farmacéutico y de alimentos [18]

Figura 1.15 Volumen de trabajo cilíndrico (izquierda) .Robot SCARA (derecha ejemplo robot con volumen de trabajo cilíndrico) (Tomado de [16] [18])

3.3.3. Configuración Polar.

Los robots que poseen una configuración polar, los de brazo articulado presentan un volumen de trabajo irregular. Esta configuración se puede encontrar mayormente en brazos robóticos con más de 3 grados de libertad. [16]

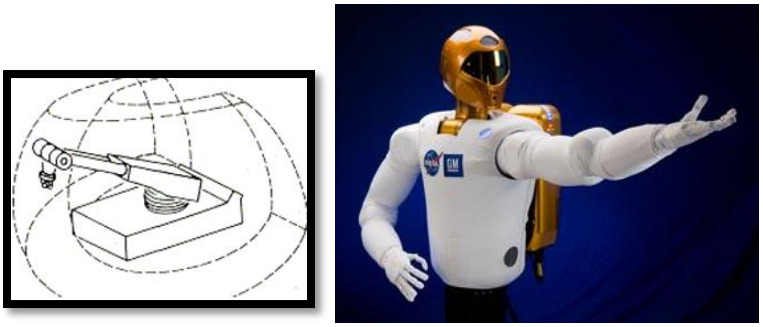


Figura 1.16 Volumen de trabajo Polar (izquierda). Brazo robótico de un androide con configuración de trabajo polar (derecha). (Tomada de [16] [19])

3.3.4. PRECISIÒN DE LOS MOVIMIENTOS.

La dependencia de la precisión de los movimientos se genera a partir de tres factores:

- Resolución espacial.
- Exactitud.
- Repetición.

3.3.4.1. Resolución espacial.

El menor incremento de movimiento mediante el cual el robot pueda fragmentar su volumen de trabajo, también se define como la distancia entre dos puntos contiguos cuya separación es normalmente de un milímetro o menos, esto depende de la clase del robot.

Esta resolución depende de dos factores: El o los sistemas que controlan la resolución y las inexactitudes mecánicas debidas a variaciones. Las inexactitudes mecánicas están estrechamente relacionadas con la calidad de los elementos que conforman las uniones, junturas y las articulaciones. Algunas de estas inexactitudes mecánicas pueden ser: El tamaño de los engranes,

las tensiones entre poleas, los escapes de fluidos, roce o fricción entre engranes, entre otros. [16][20]

3.3.4.2. Exactitud.

Hace referencia a aquella capacidad del robot para posicionar el extremo de su extremidad en un punto determinado dentro del volumen de trabajo, mediante esta, se mide la distancia entre la posición definida, y la posición real del actuador final del robot (Está relacionada directamente con la resolución espacial).

La exactitud en un robot se incrementa en la medida en que su extremidad se encuentra más cerca a su base. A medida que el brazo se disponga más lejos de su base, la exactitud se reduce, puesto que las inexactitudes mecánicas se incrementan proporcionalmente al ser extendida la extremidad. "Otro factor que afecta la exactitud es el peso de la carga"; en la medida en que la carga sea de mayor peso, tendrá como consecuencia la reducción de la exactitud, este peso también afecta la velocidad en los movimientos de la extremidad y a su vez hace variar la resistencia mecánica. [16][20]

3.3.4.3. Repetición.

Hace referencia a la capacidad del robot de regresar a su punto pre programado un "*x*" número de veces. Dicha magnitud determina el nivel de exactitud en la repetición de los movimientos del manipulador al momento de realizar una determinada tarea. [16][20]

3.4. CINEMÁTICA.

Uno de los temas importantes a la hora de desarrollar proyectos en el área de la robótica no solo es el hardware y software sino el estudio del movimiento, en especial si se está trabajando en robótica móvil. Por medio de la cinemática del robot se puede estudiar y analizar el movimiento del mismo con respecto a un eje de referencia, en especial describir las relaciones entre posición y orientación que hay entre los actuadores y la del efector final. [20][21][22]

Existen dos ramas fundamentales para analizar y resolver el movimiento del robot y es la cinemática directa y la indirecta.

- **Problema cinemática directo:** Determinar la posición y orientación del extremo del robot, con respecto a un sistema de coordenadas de referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot.
- **Problema cinemática inverso**: Determinar la configuración que deben adoptar las articulaciones del robot para alcanzar una posición y orientación del extremo conocidas.

En la clasificación de los robots, el análisis cinemático de los robots "seriales" es mucho más sencillo comparado con los robots "paralelos". La solución a la cinemática en el caso de los robots seriales se puede realizar por medio de métodos geométricos o por métodos sistemático como el de Denavit y Hartenberg. [20][21][22]

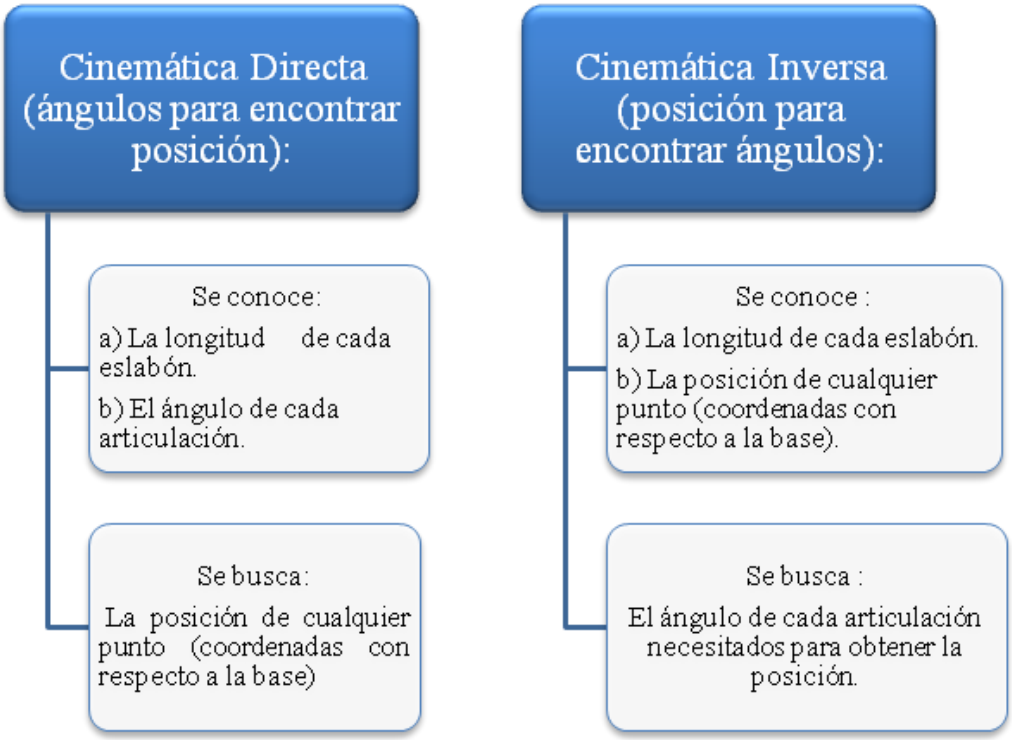


Tabla 1.0 Diagramación del problema cinemático para cinemática directa e inversa.

Para analizar el estudio de los métodos enunciados anteriormente es necesario tener presente un término clave que permite la descripción de éstos, que es la Matriz de Transformación Homogénea.

3.4.1. MATRIZ DE TRANSFORMACIÓN HOMOGÉNEA.

Es una matriz T que representa la transformación de un vector de coordenadas homogéneas de un sistema de coordenadas a otro.

La matriz T está compuesta de cuatro submatices:

R_{3x3} SubMatriz de Rotación.

R_{3x1} SubMatriz de Translación.

F_{1x3} SubMatriz de Perspectiva.

E_{1x1} SubMatriz de Escalado global.

$$\boldsymbol{T} = \begin{bmatrix} R_{3X3} & P_{3X1} \\ F_{1X3} & E_{1X1} \end{bmatrix}$$

Ecuación 1.0 Matriz de transformación homogénea.

En robótica generalmente la SubMatriz de perspectiva F es nula, y la SubMatriz de escalado global corresponde a la unidad. Por lo tanto la matriz de transformada homogénea quedaría como:

$$\boldsymbol{T} = \begin{bmatrix} R_{3X3} & P_{3X1} \\ 0 & 1 \end{bmatrix}$$

Ecuación 1.1 Matriz de transformación homogénea implementada en la robótica.

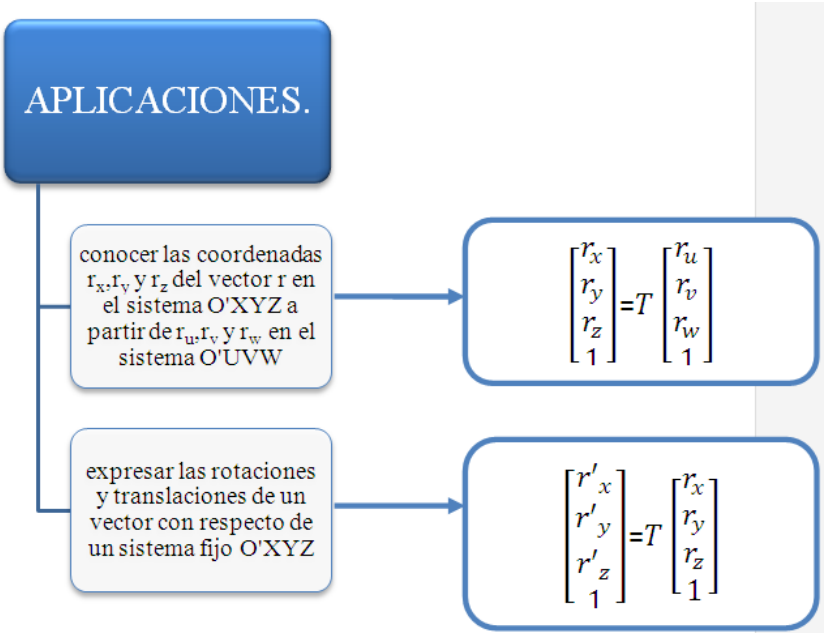


Tabla 1.1 Aplicaciones de la matriz de transformada homogénea.

3.4.2. CINEMÁTICA DIRECTA.

El problema cinemático directo consiste en determinar cuál es la posición y orientación del extremo final del robot a la través de su matriz de transformación homogénea **T**, con respecto a un sistema de coordenadas que se toma como referencia (por lo general es la base del robot), conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot. La matriz T está en función de los parámetros de las articulaciones del robot.
Una expresión general para un robot de *n* grados de libertad se tiene:

$$x = f_x(q_1, q_2, q_3, q_3, q_4, \dots, q_n)$$
$$y = f_y(q_1, q_2, q_3, q_3, q_4, \dots, q_n)$$
$$z = f_z(q_1, q_2, q_3, q_3, q_4, \dots, q_n)$$
$$\alpha = f_\alpha(q_1, q_2, q_3, q_3, q_4, \dots, q_n)$$
$$\beta = f_\beta(q_1, q_2, q_3, q_3, q_4, \dots, q_n)$$
$$\gamma = f_\gamma(q_1, q_2, q_3, q_3, q_4, \dots, q_n)$$

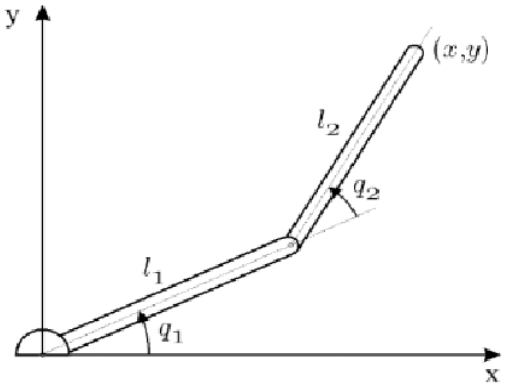
Ecuación 1.2 Expresión general para un robot de n grados de libertad.

Donde:
$q_{1\dots n}$ = Son las variables de las articulaciones.
x,y,z = Coordenadas de la posición del extremo del robot.
α,β,γ = ángulos de la orientación del extremo del robot. [22][24][25]

Cuando se está trabajando con robots de uno y dos grados de libertad se puede implementar el método geométrico, cuando se presentan mayores grados de libertad se puede resolver por el algoritmo de Denavit y Hartenberg. [232][24][25]

3.4.2.1. Método Geométrico.

Se puede obtener la posición y orientación del extremo del robot apoyándose en las relaciones geométricas. Cada relación articulación - eslabón corresponde a un grado de libertad.



$$x = l_1 \cos q_1 + l_2 \cos(q_1+q_2)$$
$$y = l_1 \operatorname{sen} q_1 + l_2 \operatorname{sen}(q_1+q_2)$$

Figura 1.17 Diagramación geométrica de un robot con dos grados de libertad. (Tomado de [22])

3.4.2.2 Método Denavit-Hartenberg

Denavit-Hartenberg propusieron un método matricial que por medio de un procedimiento sistemático permite describir la cinemática de una estructura articulada constituida por varias articulaciones, cada una de ellas con un solo grado de libertad. [22][23]

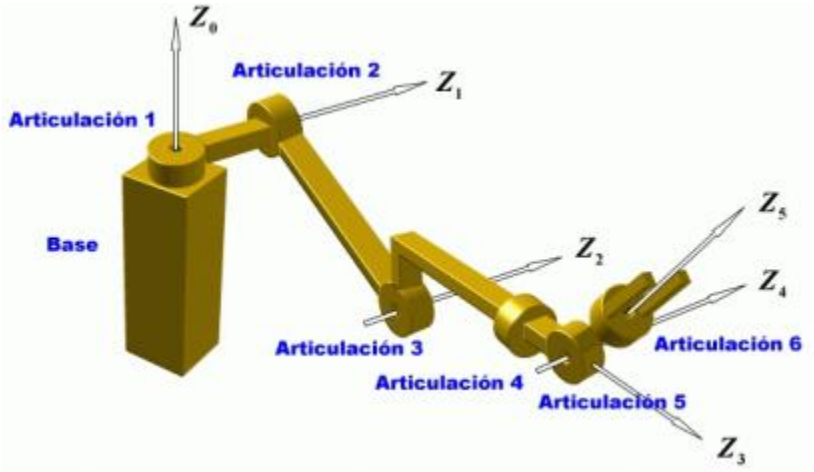


Figura 1.18 cadena articulada, cada articulación con su eje de referencia. (Tomado de [26])

Este método establece un sistema de coordenadas ligado a cada eslabón i de la cadena articulada, pudiéndose determinar a continuación las ecuaciones cinemáticas de la cadena completa.

Según la representación *D-H*, escogiendo los sistemas de coordenadas asociados para cada eslabón, será posible pasar de uno al siguiente mediante cuatro transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón.

Estas transformaciones básicas consisten en una sucesión de rotaciones y traslaciones que permitan relacionar el sistema de referencia del elemento *i* con el sistema del elemento *i*-1. Las transformaciones en cuestión son las siguientes:

- Rotación alrededor del eje Z_{i-1} un ángulo θ_i.
- Traslación a lo largo de Z_{i-1} una distancia d_i; vector d_i (0,0,d_i).
- Traslación a lo largo de X_i una distancia a_i; vector a_i (0,0,a_i).
- Rotación alrededor del eje X_i, un ángulo α_i.[22][25][26]

$$A_{i-1}^{i} = T(z,\theta_i) * T(0,0,d_i) * T(a_i,0,0) * T(x,\alpha_i)$$

Ecuación 1.3 matriz principal del algoritmo ***Denavit-Hartenberg***

Desarrollando la expresión:

$$A_{i-1}^{i} = \begin{bmatrix} cos\theta_i & -sen\theta_i & 0 & 0 \\ sen\theta_i & cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha_i & -sen\alpha_i & 0 \\ 0 & sen\alpha_i & cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 1.4 representación matricial del algoritmo ***Denavit-Hartenberg***

Por lo tanto se obtiene la expresión general de DH, donde $\theta_i, d_i, a_i, \alpha_i$ son los parámetros DH del eslabón *i*.

3.4.3. CINEMÁTICA INVERSA.

El objetivo del problema cinemático inverso consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot, ***q***=[*q1 q2 ... qn*]T, para que su extremo se posicione y oriente según una determinada localización espacial.

Se debe encontrar una solución de la forma:

$$q_k = f_k\,(x,y,z,\alpha,\beta,\gamma)$$

$$k = 1.....n$$

Ecuación 1.5 Solución general de la cinemática inversa.

Donde:

- $q_{1....n}$ = Son las variables de las articulaciones.
- x,y,z = Coordenadas de la posición del extremo del robot.
- α,β,γ = ángulos de la orientación del extremo del robot. [20][22][24][25][27]
- n = numero de grados de libertad.

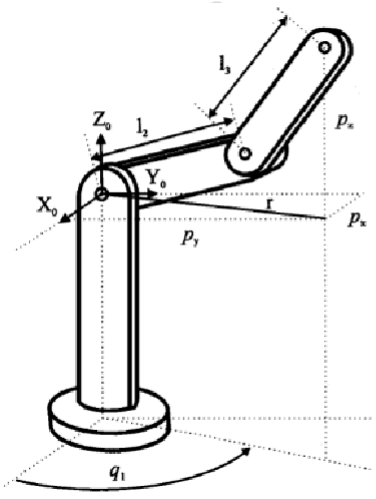
Este tipo de solución presenta las siguientes características:

- Posibilidad de resolución en tiempo real (seguimiento de trayectorias).
- Posibilidad de incluir restricciones que garanticen la mejor solución (por ejemplo, límite en los recorridos articulares).
- Posibilidad de simplificaciones.

Problema: el mecanismo de solución depende fuertemente de la configuración física del robot y con frecuencia la solución no es única. [27][28]

3.4.3.1. Método Geométrico.

Los métodos geométricos permiten obtener normalmente los valores de las primeras variables articulares, que son las que consiguen posicionar el robot. Para ello utilizan relaciones trigonometrías y geométricas sobre los elementos del robot. Se suele recurrir a la resolución de triángulos formados por los elementos y articulaciones del robot. Partiendo de un eje de coordenadas principal p_x, p_y, p_z y con relaciones trigonométricas con los eslabones se obtiene:



$$\mathbf{q_1} = \arctan\left(\frac{p_y}{p_x}\right)$$

$$\left.\begin{aligned} r^2 &= p_x^2 + p_y^2 \\ r^2 + p_z^2 &= l_2^2 + l_3^2 + 2l_2 l_3 \cos q_3 \end{aligned}\right\}$$

$$\cos q_3 = \frac{p_x^2 + p_y^2 + p_z^2 - l_2^2 - l_3^2}{2l_2 l_3}$$

$$\sin q_3 = \pm\sqrt{1 - \cos^2 q_3}$$

$$\mathbf{q_3} = \arctan\left(\frac{\pm\sqrt{1 - \cos^2 q_3}}{\cos q_3}\right)$$

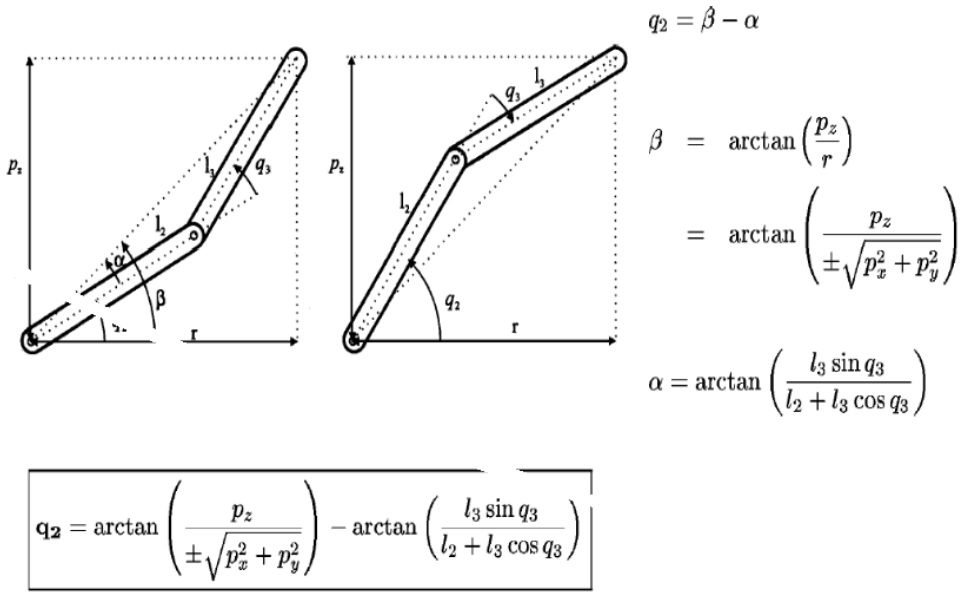


Figura 1.19 Robot con estructura planar con modelo cinemático inverso por métodos geométricos. (Tomado de [22])

3.5. POLÍGONO DE ESTABILIDAD.

Polígono de Estabilidad o polígono de apoyo es una figura formada por las extremidades que se encuentran apoyadas en la superficie y que soportan el peso del robot, "Independientemente del número de patas que estén realizando contacto con la superficie, la proyección del centro de gravedad debe estar dentro del área del polígono para que la plataforma mantenga la estabilidad dinámica" [29]

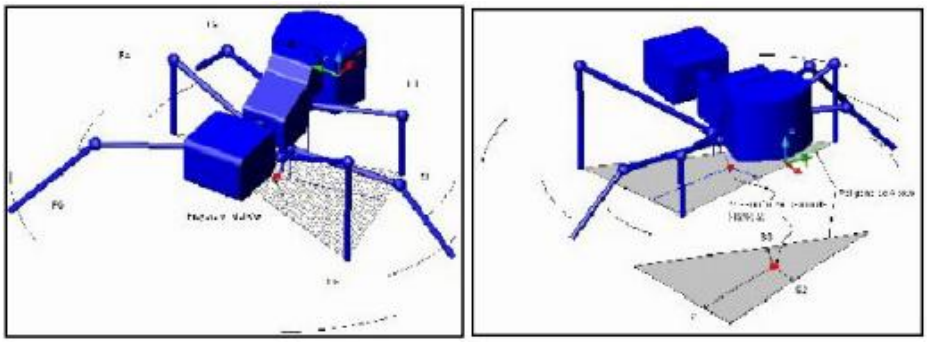


Figura 1.20 representación del polígono de estabilidad. (Tomado de [29]).

3.6. FUNDAMENTOS DE LÓGICA DIFUSA.

"El término "lógica difusa" puede interpretarse como un super conjunto de la tradicional lógica booleana extendida para manejar el concepto de "parcialmente verdadero (valores de verdad entre "absolutamente verdadero" y "absolutamente falso"). Fue presentada por Lofti Zadeh en los años 60 como un medio para modelar la incertidumbre del lenguaje natural.

Según Zadeh, no se debe considerar la teoría difusa como una simple teoría, sino que se debería considerar el proceso de *fusificación* como una metodología para generalizar cualquier teoría desde su versión ordinaria o discreta a una nueva versión continua o borrosa. Así, puede hablarse de "cálculo borroso", "ecuaciones diferenciales borrosas", "sistemas dinámicos borrosos", etc."[30], este tipo de conjunto, constituye una gran herramienta para representar el conocimiento humano.

También es considerada como una metodología capaz de proporcionar una manera sencilla de obtener conclusiones a partir de información de entrada incompleta o imprecisa.

TIPOS DE CONJUNTOS.

CONJUNTO CLASICO	CONJUNTO DIFUSO
• Restrictivos. • En estos conjuntos, un elemento pertenece o no pertenece al conjunto. • La pertenencia de un elemento x a un conjunto determinado "A" es descrita mediante la función de pertenencia $\boldsymbol{\mu_A}$. $\mu_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$ $\mu_A(x) : U \rightarrow \{0,1\}$	• No restrictivos. • En estos conjuntos, un elemento puede pertenecer en determinada proporción a uno o más conjuntos. • Para los conjuntos difusos, el **grado de pertenencia** indica cuando el elemento es similar al concepto representado por el conjunto. $\mu_A(x) : U \rightarrow [0,1]$

Tabla 1.2 Tipos de conjuntos en la lógica difusa.

Algoritmo de agrupamiento (*clustering*): Procedimiento o **medio de agrupación** que funciona mediante la inserción de una serie de vectores de datos con base en algún criterio específico, en este caso ese criterio es de **distancia**. Los algoritmos de *cluster* **permiten extraer representantes de un conjunto de datos**, estos algoritmos, separan o clasifican un conjunto de muestras en un determinado número de grupos, basándose en semejanzas.[31]

Métodos de agrupamiento (clustering) con función objetivo: Permiten una formulación más precisa conforme al criterio de clustering. Para cada uno de los clusters, la función objetivo permite analizar la calidad de las agrupaciones candidato. Uno de los métodos define su medida por medio de la distancia euclidiana y como función objetivo "la suma cuadrática de la distancia entre los puntos" centro de cluster o cluster prototipo, por ende, el objetivo del agrupamiento es minimizar dicha sumatoria.

Función de pertenencia tipo Ruspini: La suma de todos los valores de pertenecía de un dato es igual a la unidad.

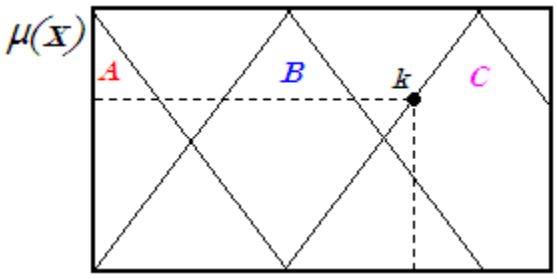


Figura 1.21 Funciones de pertenencia

El punto k cumple con la afirmación descrita anteriormente, de tal manera que:

$$\sum_{i=0}^{l} \mu_i(k) = 1$$

Ecuación 1.6 Función de pertenencia tipo Ruspini.

Algoritmo fuzzy c-means: El algoritmo fuzzy c-means es un método de agrupamiento difuso ampliamente usado en el reconocimiento de patrones, análisis de datos, procesamiento de

imágenes, modelado y optimización. En el agrupamiento difuso, cada objeto puede pertenecer a más de un conjunto, por tanto cada objeto tendrá un grado de pertenecía asociado a cada uno de ellos.

Se define J, denotada como la función objetivo y su resultado es una medida global entre datos y centroides, el objetivo es minimizarla.

$$\boldsymbol{J}(\boldsymbol{Z};\boldsymbol{V},\boldsymbol{U},\boldsymbol{A}) = \sum_{i=1}^{c}\sum_{j=1}^{N} \mu_{ij}^{m} d_{A_i}^{2}(\boldsymbol{z}_j, V_i)$$

Ecuación 1.7 Función objetivo de algoritmo Fuzzy C-means.

Z: Corresponde a la matriz de datos, dada por: $\boldsymbol{z}_k = [z_{1k}, z_{2k}, \dots, z_{nk}]^T \; k = 1, \dots, N$

N: Es el número de datos.

V: Es el vector de los prototipos de *cluster (*centros*)*

U: Corresponde a la matriz de partición difusa de Z.

A: Corresponde a la matriz de norma inducida, la cual generará una dependencia con respecto al cálculo de la distancia, para el algoritmo de Fuzzy C-Means es igual a la matriz identidad I.

i=1….c: Para todas las clases.

j=1….N: Para todos los datos.

d: Corresponde a la medida de distancia entre los datos y los centros de *cluster*. Esta medida, es afectada por la matriz de norma inducida (*para FCM esta matriz es I*).

Para implementar dicho algoritmo se hace necesario establecer determinados parámetros para acotar el número de ejecuciones del mismo:

- ✓ Establecer un número de *clusters* óptimo para generar el modelo (al terminar de ejecutar el algoritmo esta suposición se valida frente a las "figuras de merito" descritas más adelante).

- ✓ Inicializar la matriz de partición difusa "U" con valores aleatorios pero que cumplan con la condición.

$$\sum_{i=1}^{K} \mu_{ij} = 1$$

Ecuación 1.8 Matriz de partición difusa U.

- ✓ Establecer una condición de tolerancia (ε), cuando la diferencia entre la variación de las posiciones de los *clusters* sea menor a la tolerancia, el algoritmo se dejara de ejecutar.
- ✓ Establecer un numero de iteraciones máximo con el fin evitar un número de ejecuciones infinitas en caso de que el algoritmo no logre el margen de tolerancia (ε).

Una vez se complete la inicialización, se procede realizar la ejecución del mismo de la siguiente manera:

1. Se calculan los centros de *clúster* iníciales mediante la relación:

$$V_i = \frac{\sum_{j=1}^{N} (\mu_{ij})^q X_j}{\sum_{j=1}^{N} (\mu_{ij})^q}$$

Ecuación 1.9 Función de los centros de Clusters.

q: Es el grado de fusibidad, por simplicidad es igual a 2.

2. Se calculan las distancias que separan los datos a los cluster.

$$d^2(X_j, V_i) = \sum_{t=1}^{M} [X_j(t) - V_i(t)]^2$$

Ecuación 1.10 Función de la separación de los clusters.

3. Se actualizan los grados de pertenencia de la matriz de partición y los centros de cluster.

$$V_i^+ = \frac{\sum_{j=1}^{N} (\mu_{ij})^q X_j}{\sum_{j=1}^{N} (\mu_{ij})^q} \qquad \mu_{ij}^+ = \frac{\left[\frac{1}{d^2(X_j, V_i)}\right]^{\frac{1}{q-1}}}{\sum_{i=1}^{K}\left[\frac{1}{d^2(X_j, V_i)}\right]^{\frac{1}{q-1}}}$$

4. Se evalúa si se cumple la condición para la tolerancia "ε", si no se cumple, se repiten los pasos 2 y 3.

$$E_t = \sum_{i=1}^{K} \left\| V_i - V_i^* \right\|$$

$$\boldsymbol{E_t \leq \varepsilon}$$

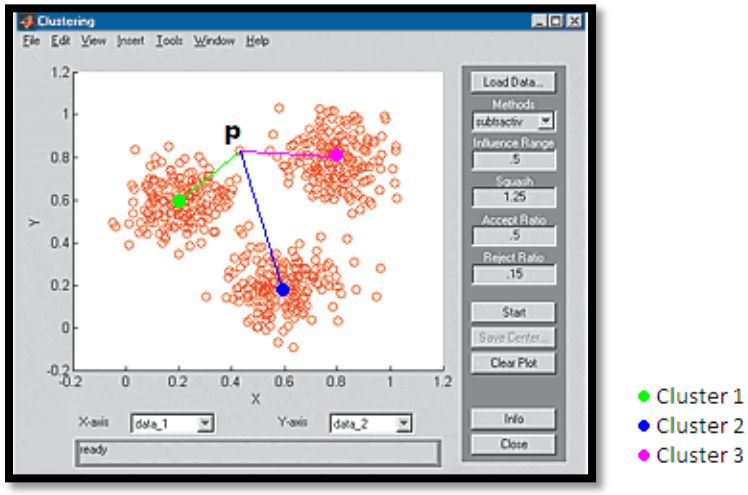


Figura 1.22 Implementación FCM MATLAB (Tomado de [32])

Como ejemplo, en la figura 1.22 se aprecia claramente que sobre la distribución de objetos se pueden generar tres grupos o clusters. A su vez, se aprecian los centros de cluster (verde, azul, morado), y un punto "P" dado, este objeto pertenece en cierta medida a cada uno de los grupos, a esta pertenencia se le asocia el termino grado de pertenencia y varia en el rango 0 a 1 para cada conjunto.

Limitaciones del Algoritmo:

- En primer lugar, los resultados del algoritmo dependen de la partición inicial, la cual es totalmente arbitraria.[33]

- el FCM no calcula el número c de conjuntos, ya que dicho número se fija en el primer paso del algoritmo. En algunos casos este dato puede ser conocido, pero en otros puede no estar disponible.[33]

- El uso de la distancia euclidiana lleva a que el FCM sólo detecte conjuntos hiperestésicos (esferas de n dimensiones). La detección de otras formas de conjuntos obligaría a utilizar definiciones distintas de distancia.[33]

4. DESCRIPCIÓN FÍSICA DE LA PLATAFORMA.

A continuación se presenta la tabla 1.3 en donde se describen las características generales del robot hexápodo con el que se trabajó.

TIPO DE ARQUITECTURA.	*Zoomórficas y Móviles.* (Móviles: Por su gran capacidad de desplazamiento. Por el diseño de movimiento implementado de forma teledirigida por un humano a través de programación. Zoomórficos: su sistema de locomoción hace referencia a algún tipo de insecto en este caso a los insectos tipo hexápodos como las hormigas.) (ANEXO Clasificación según arquitectura *Tabla 10.0)*
TIPO DE GENRACIÓN.	*Tercera generación. (*por su uso de un computador y un hardware con lenguaje de programación para su sistema de control)

	(ANEXO Clasificación según generación *Tabla 10.1.)*
ESTÁNDARES	*Tipo C según las dos asociaciones.* (por su programación de control de trayectoria fija y continua) (ANEXO Clasificación según estándares *Tabla 10.2)*
CLASIFICACIÓN DE ROBOTS CON EXTREMIDADES.	*Número de extremidades:* seis, ubicadas paralelamente entre ellas. *Medio geográfico donde realiza el movimiento:* Movimiento en superficies rígidas y planas. *Grados de libertad:* 2 grados, uno vertical y el otro circular. (ANEXO *Mapa conceptual 11.0 y grados de libertad 10.3.1)*
TIPO DE ACTUADOR.	Doce Servomotores eléctricos controlados por secuencias de pulsos. 6 servos con accionamiento directo a la extremidad y los otros 6 por medio de transmisiones. (*Ver Actuadores 9.2.3, Servomotor 3.2, ANEXO Accionamiento directo 10.3.2, Especificaciones eléctricas y físicas de los actuadores 4.1)*
TRANSMISORES.	Se utilizan de transmisiones del movimiento desde el actuador hasta la extremidad, rótulas de doble cabeza redonda de plástico con tornillo metálico. (ANEXO transmisores 10.2.4 y *figura 1.23. y 1.24*)

HARDWARE Y SOFTWARE IMPLEMENTADO.	Hardware Arduino con programación lenguaje Arduino basado en C++ y Java.
MOVILIDAD	La movilidad está definida para todas las extremidades igual de la siguiente manera: **Servo 1:** Servo superior de la extremidad encargado del movimiento rotacional o cilíndrico. (movimiento angular) *(ver Movilidad 3.3)* **Servo 2:** Servo inferior encargado del movimiento vertical. (movimiento lineal) (*ver Movilidad 9.3)*
ESPACIO O VOLUMEN DE TRABAJO.	*Configuración Cartesiana* para el servo superior encargado del movimiento lineal. *Configuración Cilíndrica* para el servo inferior encargado del movimiento angular. *(ver Espacio o volumen de trabajo 3.3)*
CONFIGURACIÓN FÍSICA.	*Configuración Bilateral:* Esta configuración presenta una simetría a lo largo del eje longitudinal del robot. *(ver Configuración física 3.1.2)*
MATERIAL DE LA EXTRUCTURA.	Láminas de PVC expandido y acrílico.

Tabla 1.3 Descripción física y clasificación de la plataforma.

4.1. PLATAFORMA HEXÁPODO Y MATERIALES.

El chasís de la plataforma lo conforman dos placas rectangulares de PVC-expandido, unidas mediante diez pilares metálicos formando así el cuerpo principal del hexápodo (Figura 1.25).
Cinco de las extremidades están construidas con PVC-expandido y una de acrílico, ya que fue necesario construir una de las extremidades faltantes en el proceso de adecuación de la plataforma.
Como transmisores de movimiento del actuador a la articulación se tiene por cada extremidad una rótula de bola de cabeza doble (figura 1.23)**.** A estas rótulas es necesario ensamblarles eslabones fijos ajustados a cada extremidad para realizar la conexión entre el transmisor y la extremidad (Figura 1.24).

Figura 1.23. Rótula de bola de doble cabeza ajustable.

Figura 1.24. Eslabones para conexión entre rótula y extremidad.

El cuerpo principal de la plataforma son dos láminas iguales y verticalmente paralelas entre sí, de 5 mm de espesor con una silueta simétrica, la cual se puede simplificar en su forma en geometría básica a partir de un rectángulo principal unido a otros tres rectángulos ubicados equidistantes (como se muestra en la figura 1.25), juntos a su vez con medio círculo. Estos salientes constituyen los soportes del cual se agarran los servomotores superiores, encargados del movimiento rotacional.

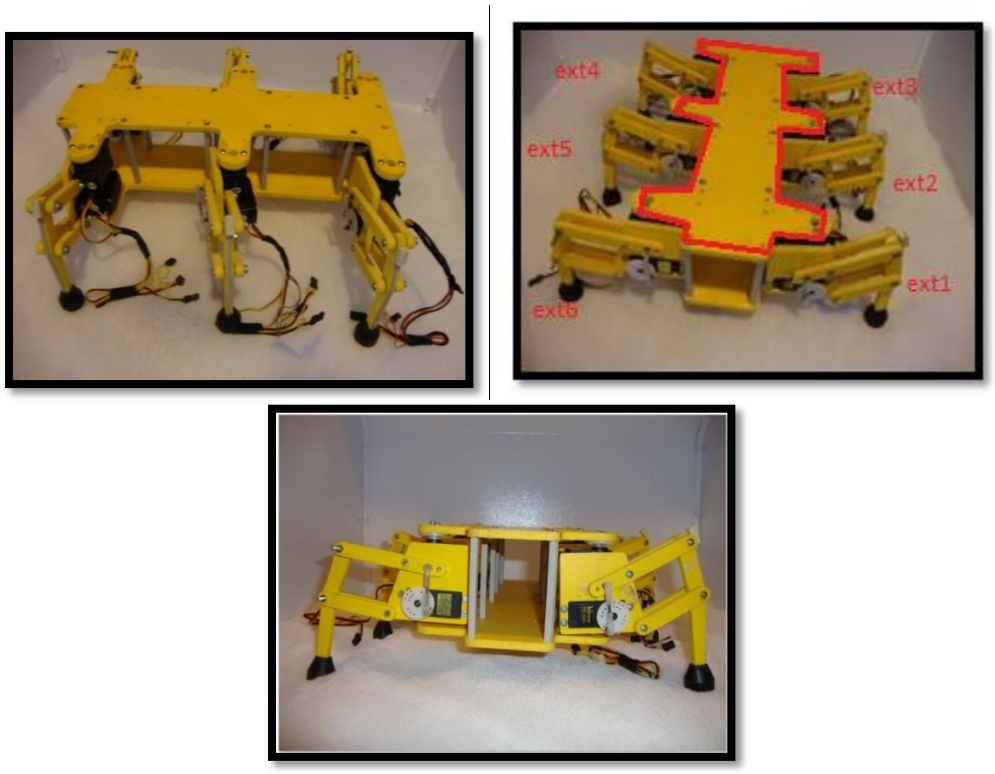


Figura 1.25.Estructura general de la plataforma Hexápodo con diferentes perspectivas.

Cada extremidad está compuesta por seis piezas fijas y dos servomotores uno encargado del movimiento rotacional y otro del movimiento vertical es decir dos grados de libertad por extremidad.

Al momento de diseñar la estrategia de movimiento de la plataforma es necesario tener en cuenta el espacio de trabajo de cada extremidad, en especial el relacionado al movimiento rotacional, ya que si sus rotaciones son muy amplias pueden ocasionar choques físicos entre las extremidades produciendo desgaste de las mismas y errores de estabilidad.

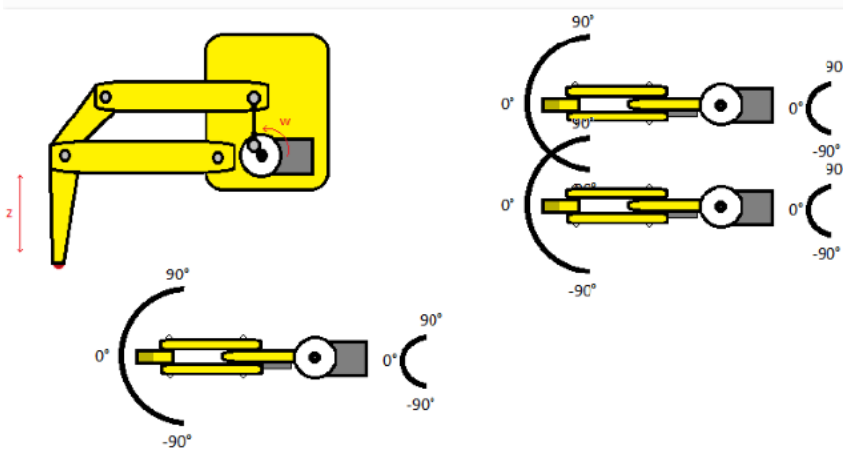


Figura 1.26. Extremidad de la plataforma hexápodo y su diagrama de movimiento tanto rotacional como vertical.

4.2 . ESPECIFICACIONES ELÉCTRICAS Y FÍSICAS DE LOS ACTUADORES.

Los actuadores son la fuente primordial del movimiento para la plataforma.

La plataforma está conformada por 12 servomotores:

- 9 Hitec Standar HS-300
- 2 Hobbico Command Standar Sport-servo CS-60
- 1 Towerpro SG5010º

Hitec HS-300 Standard Servo	***Hobbico Command Standar Sport-Servo Cs-60***	***TowerPro SG-5010 Stándar Servo***
Tipo de modulación: Análoga. *Torque:* 4.8 V: 3.02 Kg-cm. 6V: 3.67Kg-cm. *Velocidad:* 4.8V:0.19 s/60º 6V: 0.15 s/60º *Peso:* 47.1 g.	*Tipo de modulación:* Análoga. *Torque:* 4.8V=4.39Kg/cm 6V=76 5.47 kg/cm *Velocidad:* 4.8V:0.23 s/60º 6V: 0.18 s/60º *Peso:* 40g	*Tipo de modulación:* Análoga. *Torque:* 4.8V: 5.2 Kg/cm 6.0V:6.5 kg/cm *Velocidad:* 4.8V : 0.20seg / 60º 6.0V: 0.16seg / 60º *Peso:* 45g

Figura 1.27. Servo motor Hitec hs300 Standar (Tomado de [34])

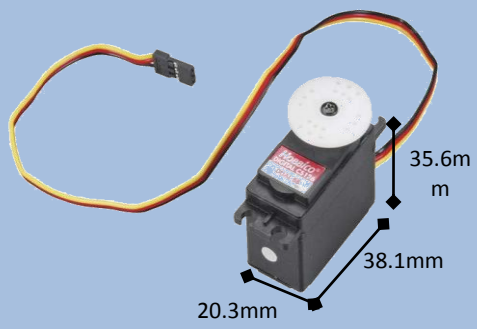


Figura 1.28. Servo motor Hobbico Command Standar Sport-Servo Cs-60 (Tomado de [35])

Figura 1.29.Servo motor Servo TowerPro SG-5010. (Tomado de [36])

Tabla 1.4 Especificaciones generales de los actuadores utilizados en la plataforma. (Tomado de [34] [35] [36])

4.3. ESTRATEGIA DE MOVIMIENTO.

La elección de una adecuada estrategia de movimiento para los robots móviles se encuentra inmersa en el análisis de su geometría, cinemática, estabilidad, velocidad, entre otros aspectos que afecten de manera directa o indirecta la plataforma.

Dependiendo de la cantidad de extremidades involucradas en el movimiento, es necesario escoger la estrategia apropiada, pues la implementación de la secuencia adecuada difiere mucho, tanto por la cantidad de extremidades como por la ubicación de las mismas.

Para robots hexápodos existen varios tipos de estrategias de movimiento, entre las cuales encontramos:

MONOPOD 1EM	BIPOD 2EM	TRIPOD 3EM
• Una extremidad por movimiento • Presenta beneficios de estabilidad de movimiento • Limitaciones de velocidad por el mayor cantidad de estados. • Adecuada para condiciones de baja energía de alimentación (batería)	• Dos extremidades por movimiento. • Presenta condiciones estables con respecto al movimiento. • Condiciones neutras de velocidad pero mayores que las de 1EM • Consumo moderado de energía eléctrica.	• Tres extremidades por movimiento. • Es necesesario diseñar una buena estrategia de movimiento ya que puede presentar problemas de estabilidad. • Adecuada en términos de deplazamiento y velocidad. • Propicio sobre el desarrollo de software. • Mayor consumo de energia.

Tabla 1.5. Características principales de las diferentes estrategias de movimiento.

La estrategia 3EM (tres extremidades por movimiento) también llamada "*tripoid gait",* es válida teniendo como prioridad la velocidad. Si la prioridad es la velocidad pero conservando el estado de la fuente de alimentación que llevaría incorporada el robot (baterías) la mejor estrategia de movimiento es la 2EM (dos extremidades por movimiento). La estrategia 1EM (una extremidad por movimiento) solo sería recomendable en condiciones de batería baja a las que en principio no debería de llegarse nunca, aunque esta presenta beneficios de estabilidad pero limitaciones de velocidad. [37]

Se optó por escoger la estrategia de movimiento 3EM, pues se desea obtener ganancia en velocidad. Como se está buscando obtener la menor cantidad de estados posibles para generar el avance, al trabajar con 1EM O 2EM el número de estados sería mayor y esto a su vez conlleva a la generación de un código de programación más denso, mayor consumo de memoria del hardware. Además, la estrategia 2EM tiene como prioridad características energéticas las cuales **no** están contempladas en el proyecto y la estrategia 1EM además de ser lenta está relacionada igualmente con características de ahorro energético, y este hecho tampoco se contempla.

Teniendo en cuenta los aspectos enunciados anteriormente y la Tabla 1.5 se determino que la mejor proporción entre velocidad, estabilidad y cantidad de estados es la estrategia 3EM.

Se presentan a continuación análisis sobre las estrategias 2EM y 3EM, a partir de las cuales se analizaron sus características teniendo en cuenta las consideraciones planteadas anteriormente.

MOVIMIENTO PLATAFORMA (BIPOD 2EM).

Figura 1.30. Estados de movimiento (estrategia 1, 2EM)

Para la descripción de la siguiente rutina, se dividen las extremidades en tres parejas (A: extremidades 1 y 6, B: extremidades 2 y 5, C: extremidades 3 y 4).

Descripción de estados:

E0: Estado en el cual la plataforma se encuentra en posición aleatoria.

En este estado la ubicación de extremidades no es pertinente, por lo tanto están en un orden no específico.

E1: Las tres parejas se mueven hacia atrás.

Las seis extremidades a la vez realizarán un movimiento rotacional hacia atrás.

E2: La pareja C se levanta mueve hacia adelante y baja.

Extremidades 3 y 4 se levantarán, realizando un movimiento rotacional hacia el frente y movimiento vertical hacia abajo donde permanecerán en reposo hasta el siguiente estado.

E3: La pareja B se levanta mueve hacia adelante y baja.

En este estado se repite lo del anterior estado pero con diferencia que en esta ocasión es la pareja B es decir extremidades 2 y 5.

E4: La pareja A se levanta mueve hacia adelante y baja.

Estado igual que los dos anteriores pero con movimiento de las extremidades 1 y 6.

Retorno a E1.

Las tres extremidades de igual manera realizarán movimiento hacia atrás, regresando a la posición del estado 1. Para así volver a repetir el ciclo de estados partiendo desde E1.

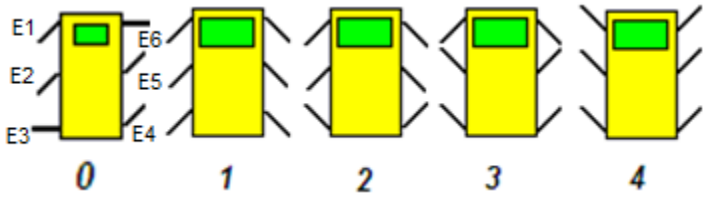


Figura 1.31. Secuencia de movimientos estrategia 1, 2EM.

Se puede observar que el movimiento que realizaría la plataforma con esta estrategia sería mediante el desplazamiento de pares de extremidades, ya que siempre el accionamiento será de dos extremidades a la vez. Por lo tanto, un paso de desplazamiento para la plataforma conllevaría la realización de cinco estados haciendo que el movimiento total no se logre con fluidez.

MOVIMIENTO PLATAFORMA (TRIPOD 3EM).

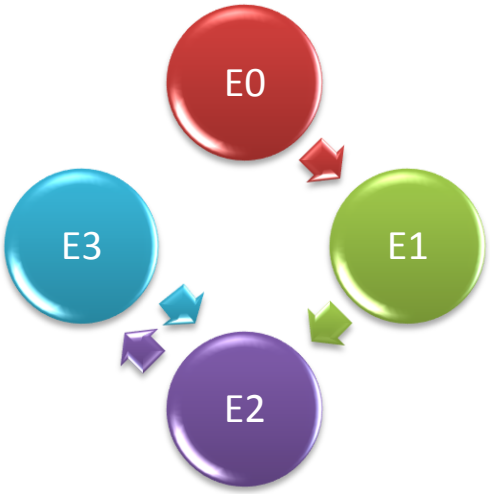


Figura 1.32. Estados de movimiento (estrategia 2, 3EM)

Descripción de estados:

Para la descripción de esta rutina se realiza dos grupos conformados por tres extremidades cada uno.

La clasificación de los grupos queda de la siguiente manera: Grupo 1: extremidad 1,3 y 5 Grupo 2: extremidad 2,4 y 6.

E0: Estado en el cual la plataforma se encuentra en una posición aleatoria.

En este estado se comienza con la misma premisa de la anterior estrategia. Una posición arbitraria de las extremidades.

E1: Estado en el cual un primer grupo de 3 extremidades (GRUPO 1) se movilizan hacia adelante al mismo tiempo, dando un primer "paso", sin que el otro trío se mueva.

Las extremidades del grupo 1: 1, 3 y 5 se levantan, realizan movimiento rotacional hacia adelante y finalizan con un movimiento vertical hacia abajo permaneciendo en reposo hasta el siguiente estado así logrando un semipaso para la plataforma.

E2: GRUPO 2 de extremidades se mueven adelante mientras que GRUPO1 impulsa hacia atrás.

Las extremidades 2, 4,6 realizan movimiento hacia delante mientras que las demás (extremidades 1, 3,5) realizan el impulso hacia atrás permitiendo así lograr un desplazamiento a la plataforma.

E3: GRUPO 1 de extremidades se mueve adelante y GRUPO 2 impulsa hacia atrás.

Realiza los mismos movimientos del estado anterior pero en sentido contrario.

En el Tabla 1.6 se puede observar el esquema de movimiento de las extremidades siendo:

- T0, T1, T2, Y T3 son los periodos de tiempo donde se activan cada una de las extremidades.
- CE equivale a cambio de estado.
- X equivale a posición aleatoria.
- N equivale a un estado de reposo (stop)
- + equivale a movimiento hacia adelante
- - equivale a movimiento hacia atrás.

EXTREMIDAD	T0	CE	T1	CE	T2	CE	T3
1	X		-		+		-
2	X		+		-		+
3	X		-		+		-
4	X		+		-		+
5	X		-		+		-
6	X		+		-		+

Tabla 1.6. Secuencia de movimiento para cada extremidad.

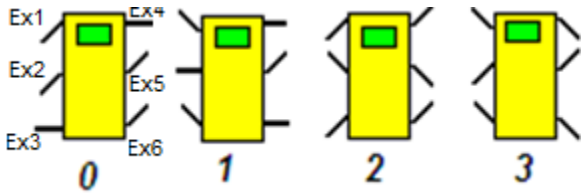


Figura 1.33. Secuencia de movimiento para 3EM.

Para la secuencia descrita se debe tener especial precaución al momento de manejar los ángulos de desplazamiento, en especial para el servomotor superior que genera el ángulo theta de la extremidad encargado del movimiento rotacional, con el fin de evitar un choque entre las extremidades.

4.3.1. Estabilidad estática.

En cuanto a la estabilidad, una de las características de esta rutina radica en el hecho de que el polígono de estabilidad se mantiene presente durante la ejecución de la rutina, pues siempre se encuentra apoyadas un grupo de tres extremidades.

En las figuras 1.34 y 1.35 se muestran los gráficos donde se evidencia el desplazamiento de la plataforma y un bosquejo del polígono de estabilidad determinado por las extremidades que estén en contacto con la superficie.

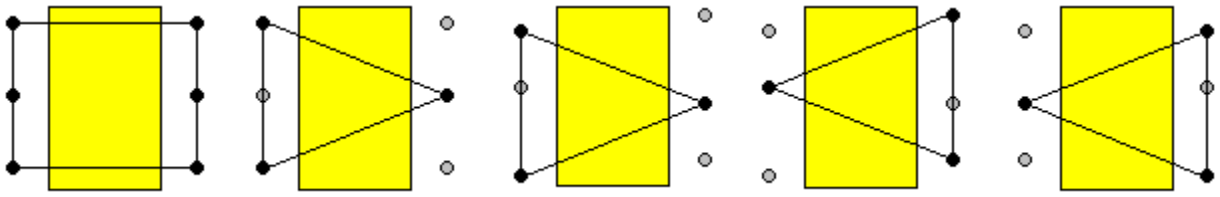
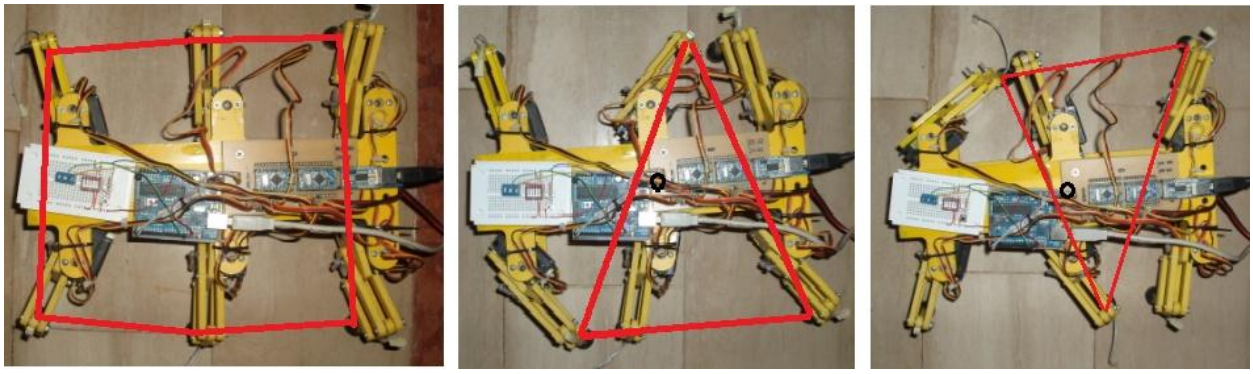


Figura 1.34. Análisis polígono de estabilidad.



Centro de masa.

Polígono de estabilidad.

Figura 1.35. Polígono de estabilidad sobre la plataforma.

Analizando las variaciones del polígono que se evidencian en la figura 1.35, se muestran los puntos críticos con respecto al rango de movimiento de las extremidades, se puede dar el caso en el que el uno de los segmentos del polígono se encuentre sobre el centro de masa de la plataforma, lo que conlleva a que el sistema pase a una estabilidad marginal, por tal razón se delimitan los ángulos de movimiento dentro de la rutina programada, garantizando la posición del centro de masa dentro del polígono. (Ver figura 1.35 y 1.36)

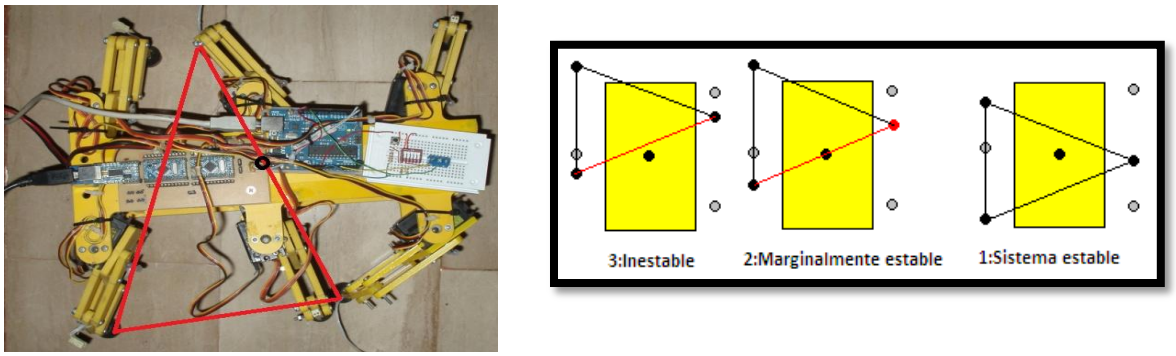


Figura 1.36 Análisis de las regiones del polígono de estabilidad.

A continuación se grafica el **área de trabajo** (figura 1.34) en la que el robot puede ejecutar sus movimientos, se considera esta área como angular.

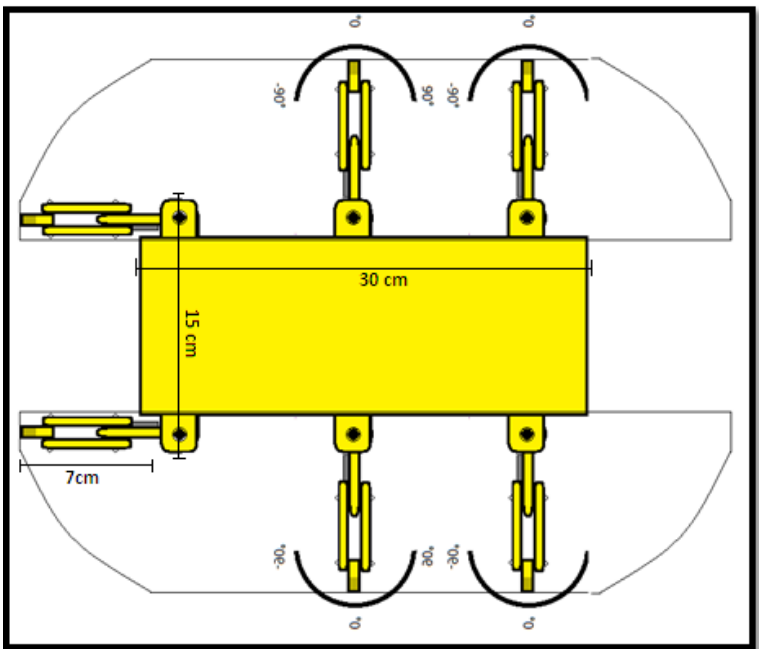


Figura 1.37 Área de trabajo de la plataforma Hexápodo.

5. DESCRIPCIÓN CINEMÁTICA DE LA PLATAFORMA.

En principio se optó por desarrollar el análisis de la cinemática de la plataforma por medio del método de cinemática directa, ya que analizando la estructura del robot hexápodo se encontró que por su estructura y grados de libertad éste se puede tratar como robot serial, por lo tanto es posible examinar su cinemática por métodos geométricos, teniendo que el principal argumento para realizar este método es ejecutarlo a estructuras con movimientos menores a dos grados de libertad.

5.1. MODELO GEOMETRICO.

Con base en la teoría analizada entre los posibles métodos para calcular la cinemática, los métodos geométricos son óptimos a la hora de realizar este análisis sobre sistemas con pocos grados de libertad. Debido a que la plataforma cuenta con dos grados de libertad por extremidad, se utilizó este método.

Se comenzó por generar un modelo matemático mediante métodos geométricos dado a que el movimiento en las extremidades se asociaba a un modelo en coordenadas cilíndricas. Por tal razón, se generó el siguiente modelo basado en las propiedades geométricas de los componentes de las extremidades:

Theta (θ): Corresponde a la posición angular de las extremidades del robot. Este parámetro puede variar entre +90° y -90° con un valor absoluto de 180° de movimiento, se debe aclarar que este movimiento presenta restricciones en el rango, debido a que de ser habilitado en su totalidad, dos extremidades podrían impactar entre sí y ocasionar daños a los mecanismos internos de los servo motores.

Zeta (z): Tomando como origen el centro del rango, ese parámetro toma valores entre –2,7 cm y 1,7 cm debido a que la misma mecánica del robot impide que se desplace fuera de estos límites.

$$\frac{y_{out}}{b} = \sin(\theta_{out}) \, ; b = 5cm$$

$$y_{out} = b * \sin(\theta_{out})$$

$$S = r * \theta = 1cm * \theta$$

$$S_{out} = r_{out} * \theta_{out} = 2cm * \theta_{out}$$

$$S_{out} = S$$

$$1cm * \theta = 2cm * \theta_{out}$$

$$\theta_{out} = \frac{\theta}{2}$$

$$y_{out} = b * \sin\left(\frac{\theta}{2}\right)$$

$$y_{out} = 5cm * \sin\left(\frac{\theta}{2}\right)$$

$$\phi_{out} = \phi_{servo}$$

$$\begin{bmatrix} \phi_{out} \\ y_{out} \end{bmatrix} = \begin{bmatrix} \phi_{servo} \\ 5cm * \sin\left(\frac{\theta}{2}\right) \end{bmatrix}$$

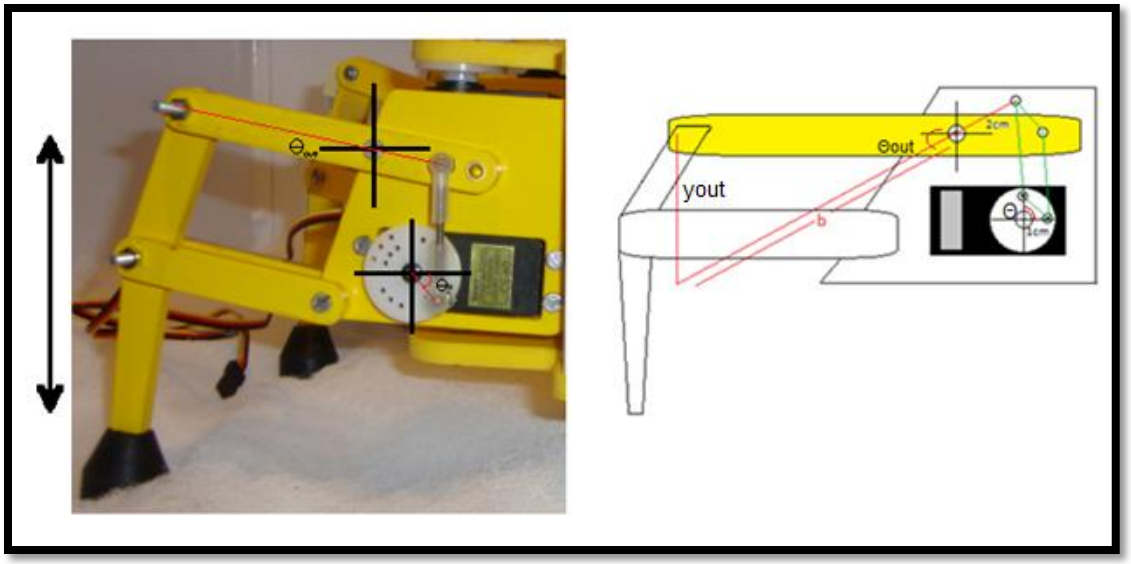


Figura. 1.38. Extremidad del robot hexápodo y análisis geométrico de ésta.

Con el resultado anterior se predice el siguiente comportamiento realizando variaciones en la posición del servo motor.

Grados	yout(cm)
90	-3,53353391
85	-3,37795104
80	-3,21393805
75	-3,04380715
70	-2,86788218
65	-2,68649804
60	-2,5
55	-2,30874307
50	-2,11309131
45	-1,91341716
40	-1,71010072
35	-1,503529
30	-1,29409523
25	-1,08219807
20	-0,86824089
15	-0,65263096
10	-0,43577871
5	-0,21809694
0	0

Tabla 1.7. Datos obtenidos para el análisis del modelo geométrico.

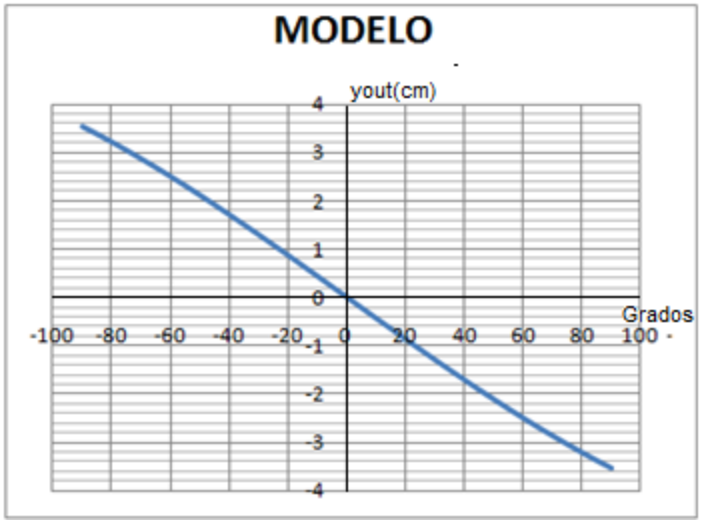


Figura 1.39. Gráfica del modelo obtenido de la cinemática directa método geométrico.

A continuación se procede a validar el modelo, realizando diferentes mediciones sobre el rango de movimiento de la extremidad y almacenando los datos para su posterior análisis (para la toma de datos se realizó dos veces el mismo procedimiento y se promediaron las muestras), obteniendo el siguiente comportamiento. Es necesario aclarar que la caracterización encontrada se halla a partir de la plataforma libre de carga.

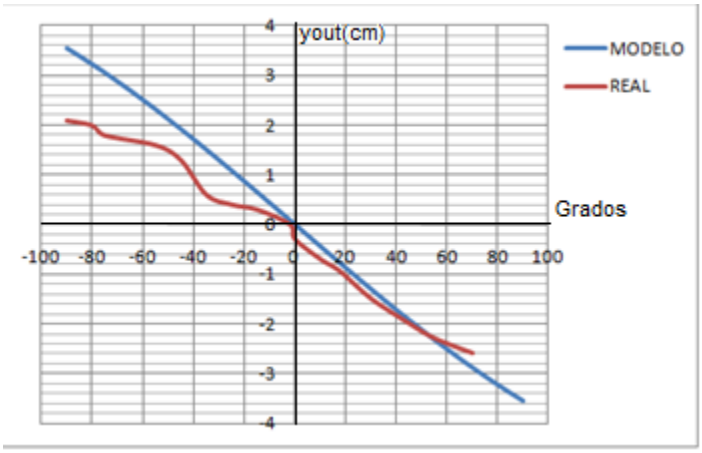


Figura 1.40. Comparación del modelo obtenido y datos tomados arbitrariamente con mediciones

Con los resultados, se evidenció un comportamiento altamente no lineal, por tal motivo se procedió a analizar con mayor detalle la naturaleza de dicho comportamiento de las extremidades.

En primer lugar se caracterizó el movimiento de las extremidades, variando la posición del piñón del servo y midiendo el desplazamiento alrededor del eje zeta, obteniendo los siguientes resultados:

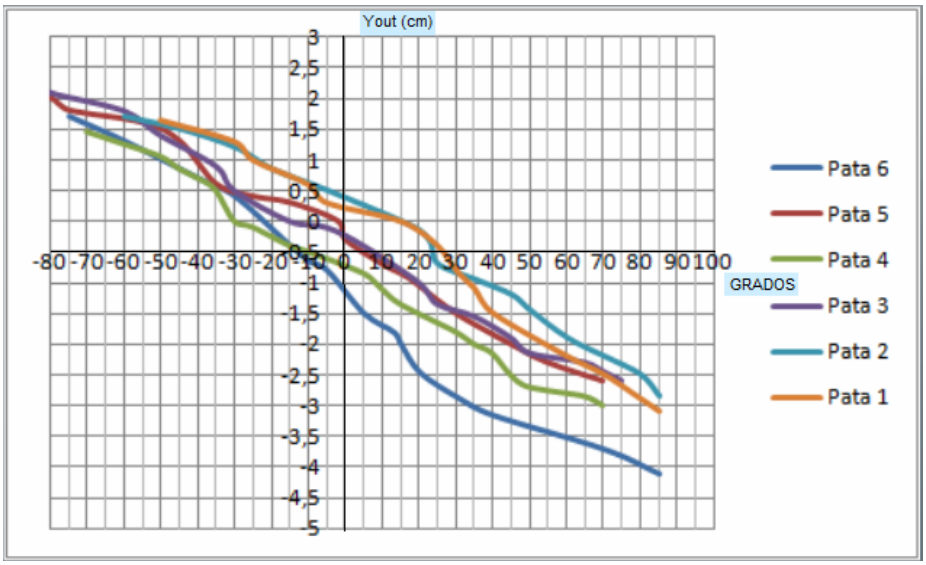


Figura 1.41. Comportamiento de las diferentes extremidades realizando mediciones aleatorias de la posición zeta.

Se comprueban los comportamientos no lineales que anulan los modelos geométricos. Analizando los resultados obtenidos por medio de la experimentación se atribuyó estos comportamientos a distintos fenómenos que afectan a las extremidades:

- ✓ En las regiones extremas del rango de movimiento, la posición de la rótula en el servo obliga a la "extensión plástica" a ejercer una fuerza ocasionando una tensión que fatiga el material y a la vez amortigua el movimiento sobre la sección superior de la extremidad, este fenómeno conlleva a una reducción del movimiento final referente al eje zeta de la extremidad.
- ✓ La rótula se encaja en una cavidad dispuesta por la sección plástica, esta configuración causa un movimiento giratorio no deseado de la sección sobre la rotula debido a la fatiga que presenta el material, teniendo como consecuencia la variación de la posición inicial

de referencia para generar el modelo, es decir, que las propiedades geométricas que describen el pentágono utilizado para realizar la caracterización, no se conservan.

- ✓ Debido a que el torque necesario para mover la extremidad de un punto al otro lo genera el servo, se graduó la "tensión" sobre los tornillos con el fin de que el dicho torque fuese suficiente para generar el movimiento de las secciones de forma aceptable y no forzada. Por otra parte, esta graduación genera un grado de flacidez sobre la posición final de la extremidad de la pata ocasionando un juego de la extremidad en torno al eje zeta sin presentarse una variación significativa en el giro del servo motor.

- ✓ La posición del piñón de salida del servo motor no es paralela al eje de la extremidad y a medida que esta sube o baja, esta diferencia aumenta o disminuye este fenómeno afecta la geometría del sistema.

5.2. IMPLEMENTACIÓN DEL ALGORITMO FUZZY C-MEANS PARA GENERACIÓN DEL MODELO MATEMATICO.

Debido a la curva características que describe el movimiento de la extremidad sobre el eje zeta, se plantea la opción de implementar un modelo que tenga como objetivo generar una familia de rectas asociadas a regiones lineales dentro de la gráfica. Por este motivo, se optó por generar el modelo mediante el algoritmo *Fuzzy c-means*, utilizado cuando los modelos matemáticos no son precisos, con el fin de localizar *centroides* que sirvan para generar rectas tangentes a ellos y así construir un modelo matemático del sistema.

Para este trabajo, la importancia de implementar dicho algoritmo, radica en el hecho de encontrar los centros de *cluster*, pues hallar estos centros implica localizar el punto medio donde las distancias del centro al dato son las menores.

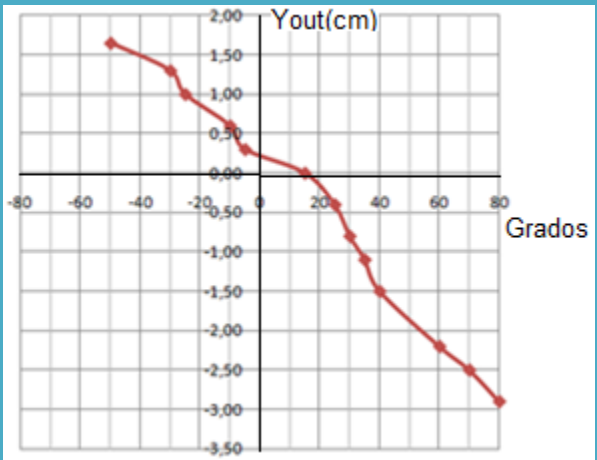
Se generarán distintos modelos a partir del incremento del número de *clusters*, con el fin de obtener una cantidad óptima de los mismos con los que se pueda obtener un modelo que cubra todo el rango de variaciones requerido.

Para la implementación de dicho algoritmo se diseño un script en Matlab (ver anexo) el cual ejecuta los pasos para el algoritmo de Fuzzy C-means.

Una vez generado esto el algoritmo debe arrojar el centro de los clusters por donde deberán pasar rectas tangentes a estos puntos y así poder obtener el modelo.

EXTREMIDAD 1:

Curva Experimental



Número de clusters:3

Centros de cluster:

Θ=-35.08 y_{out} = 1.32cm

Θ=31.86 y_{out} =-0.91cm

Θ= 65 y_{out} =-2.35cm

Número de clusters:5

Centros de cluster:

Θ=-35.08 y_{out} = 1.32cm

Θ=14.98 y_{out} = 0cm

Θ= 27.48 y_{out} =-0.59cm

Θ=37.53 y_{out} =-1.30cm

Θ=65.01 y_{out} =-1.35cm

Número de clusters:6

Centros de cluster:

Θ=-50 y_{out} = 1.65cm

Θ=-27.5 y_{out} = 1.15cm

Θ= -10 y_{out} =0.6cm

Θ=-5 y_{out} =0.3cm

Θ=19.75 y_{out} =-0.19cm

Θ=35.18 y_{out} =-1.14cm

Θ=65.01 y_{out} =-2.35cm

Modelo Matemático (Función a trozos)

f = -0.0175*t+0.775 ,-50,-30

f = -0.06*t-0.5 ,-30,-25

f = -0.02666*t+0.3333 ,-25,-10

f = -0.06*t ,-10,-5

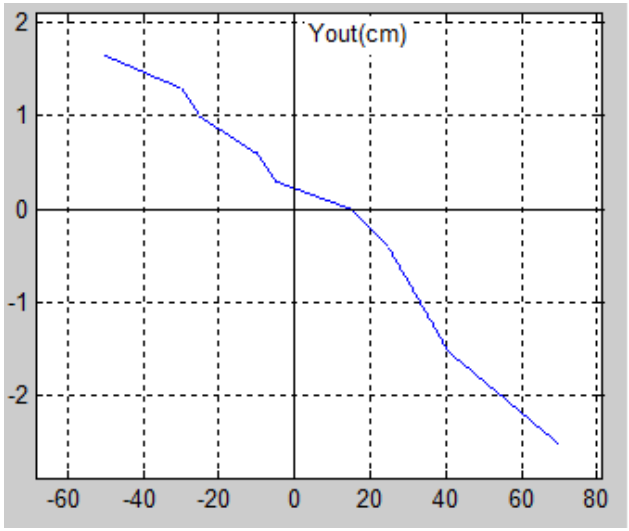
f = -0.0150*t+0.22500 ,-5,15

f = -0.040002*t+0.6 ,15,25

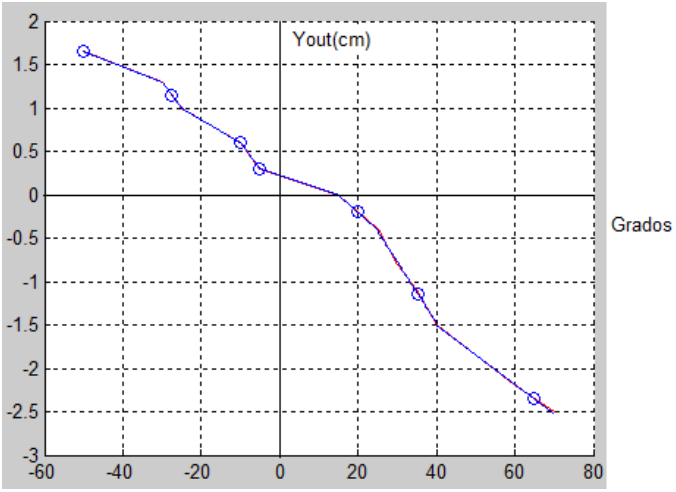
f = -0.070531*t+1.341 ,25,40

f = -0.034*t-0.1405 ,40,70

Gráfica Modelo Matemático

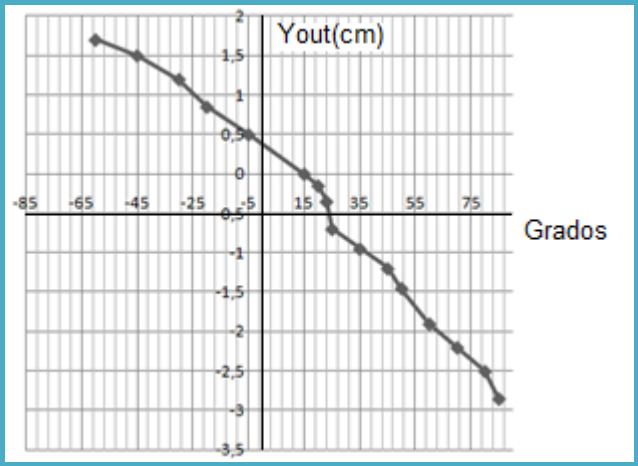


SUPERPOSICIÓN ENTRE MODELOS



EXTREMIDAD 2:

Curva Experimental



Número de clusters:3 **Centros de cluster:**

Θ=73.44 $y_{out} = -2.34cm$

Θ= 30.70 $y_{out} = -0.71cm$

Θ= -44.81 $y_{out} = 1.45cm$

Número de clusters:5 **Centros de cluster:**

Θ= -44.95 $y_{out} = 1.461cm$

Θ= 20.79 $y_{out} = -0.30cm$

Θ= 39.82 $y_{out} = -1.07cm$

Θ= 55.53 $y_{out} = -1.69cm$

Θ= 79.15 $y_{out} = 2.54cm$

Número de clusters:7 **Centros de cluster:**

Θ=-52.66 y_{out} = 1-60

Θ=-25.07 y_{out} =1.03

Θ=-5 y_{out} = 0.5

Θ=20.83 y_{out} = -0.30

Θ=39.83 y_{out} = -1.07

Θ=55.54 y_{out} =-1.69

Θ=79.15 y_{out} = -2.55

Modelo Matemático (Función a trozos)

f = -0.0141*t+0.85959, -60,-30

f = -0.03507*t+0.1478, -30,-20

f = -0.02436*t+0.3651, -20, 20

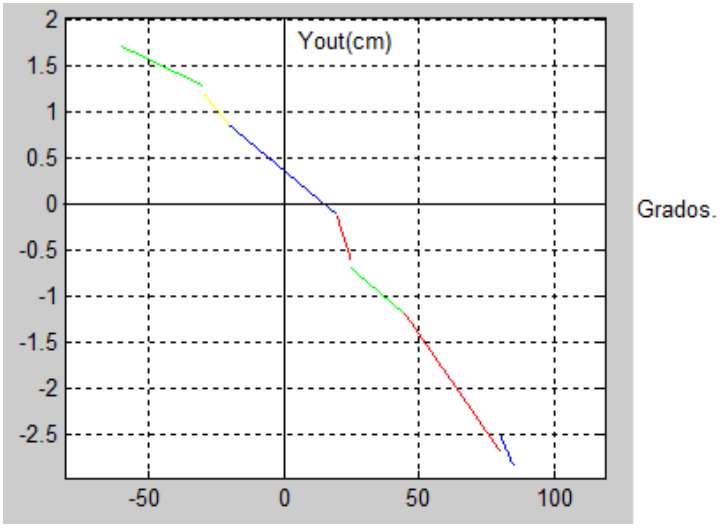
f = -0.09412*t+1.7324, 20, 25

f = -0.02499*t-0.0752, 25, 45

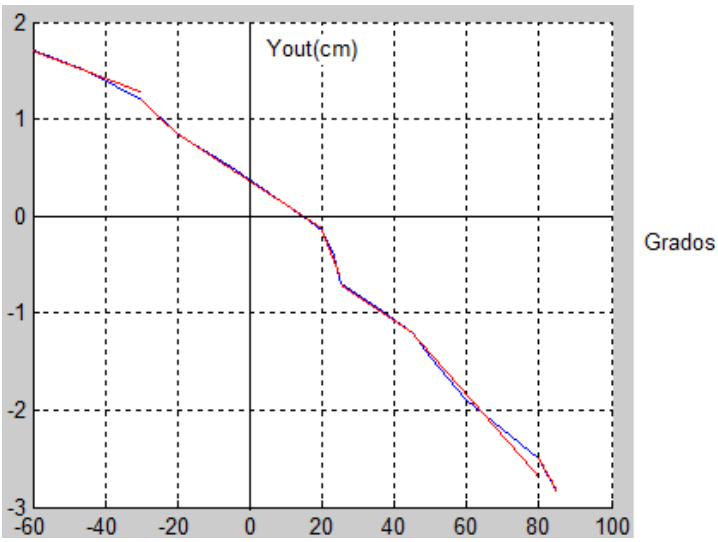
f = -0.04242*t+0.7075, 45, 80

f = -0.06800*t+2.94 , 80, 85

Gráfica Modelo Matemático

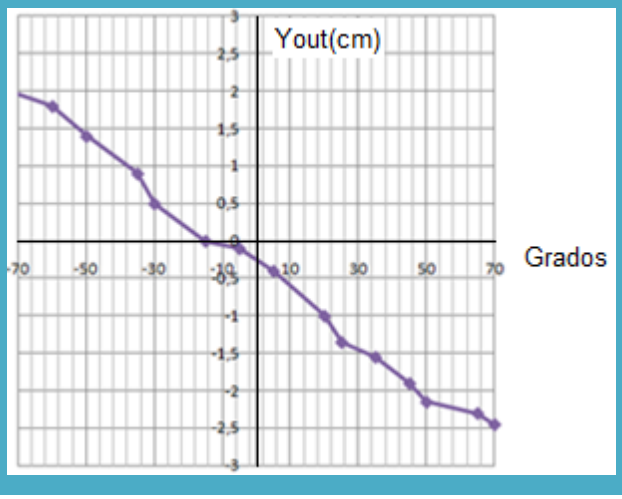


SUPERPOSICIÓN ENTRE MODELOS



EXTREMIDAD 3:

Curva Experimental



Número de clusters:3 **Centros de cluster:**

Θ=-64.66 $y_{out} = 1.772$

Θ=34.54 $y_{out} = -1.57$

Θ=-69.47 $y_{out} = -2.44$

Número de clusters:5 **Centros de cluster:**

Θ=-82.5 $y_{out} = 2.2$

Θ=-55.01 $y_{out} = 1.6$

Θ=-31.05 $y_{out} = 0.64$

Θ=34.73 $y_{out} = -1.58$

Θ=69.51 $y_{out} = -2.44$

Número de clusters:9 **Centros de cluster:**

Θ=-82.5 $y_{out} = 2.2$

Θ=-60 $y_{out} = 1.8$

Θ=-50 $y_{out} = 1.4$

Θ=-32.5 $y_{out} = 0.7$

Θ=-14.97 $y_{out} = 0$

Θ=4.97 $y_{out} = -0.4$

Θ=23.92 $y_{out} = -1.21$

Θ=46.22 $y_{out} = -1.97$

Θ=70.01 $y_{out} = -2.45$

Modelo Matemático(Función a trozos)

f = -0.040*t-1.1 ,-85,-80

f = -0.015*t+0.9 ,-80,-60

f = -0.040*t-0.6 ,-60,-35

f = -0.080*t-1.9 ,-35,-30

f = -0.032*t-0.467,-30,-15

f = -0.020*t-0.299,-15, 5

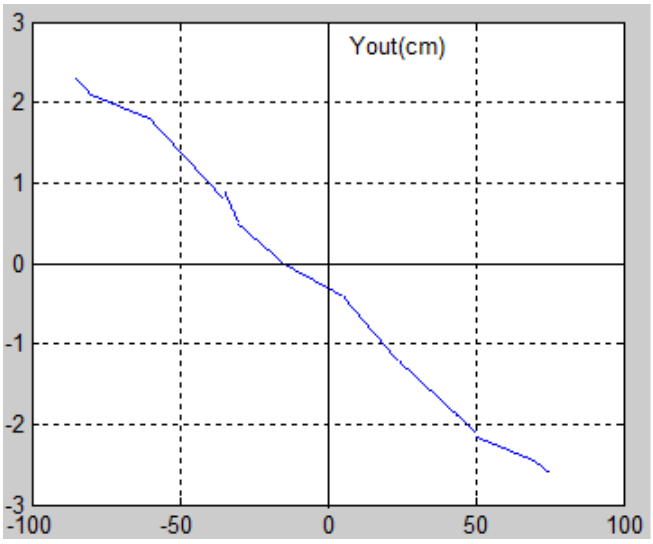
f = -0.043*t-0.186, 5, 24

f= -0.034*t-0.38 , 24, 50

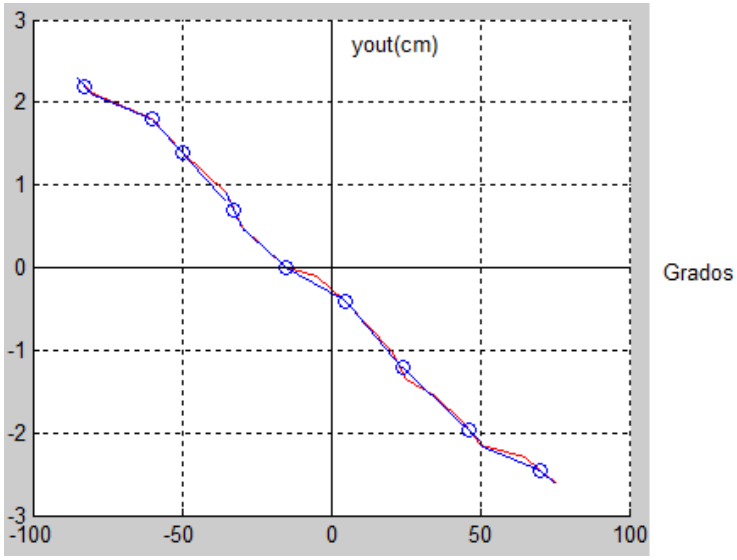
f=-0.015*t-1.400 , 50, 70

f=-0.030*t-0.35 , 70, 75

Gráfica Modelo Matemático

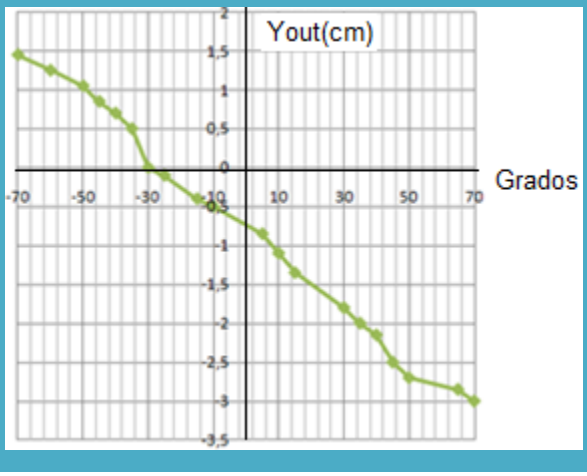


SUPERPOSICIÓN ENTRE MODELOS



EXTREMIDAD 4:

Curva Experimental



Número de clusters:3 **Centros de cluster:**

Θ=-46.09 $y_{out} = 0.788$

Θ=39.5 $y_{out} = -2.2$

Θ=67.38 $y_{out} = -2.92$

Número de clusters:5 **Centros de cluster:**

Θ=-61.34 $y_{out} = 1.27$

Θ=-34.41 $y_{out} = 0.36$

Θ=12.39 $y_{out} = -1.22$

Θ=40.03 $y_{out} = -2.22$

Θ=67.42 $y_{out} = -2.92$

Número de clusters:7 **Centros de cluster:**

Θ=-65.06 $y_{out} = 1.35$

Θ=-45.16 $y_{out} = 0.87$

Θ=-29.89 $y_{out} = 0.12$

Θ=-12.51 $y_{out} = 0.45$

Θ=12.48 $y_{out} = -1.22$

Θ=40.05 $y_{out} = -2.23$

Θ=67.42 $y_{out} = -2.92$

Modelo Matemático(Función a trozos)

f=-0.019*xa+0.0799,-70,-50

f=-0.037*xb-0.8128,-50,-35

f=-0.074*xc-2.1027,-35,-30.5

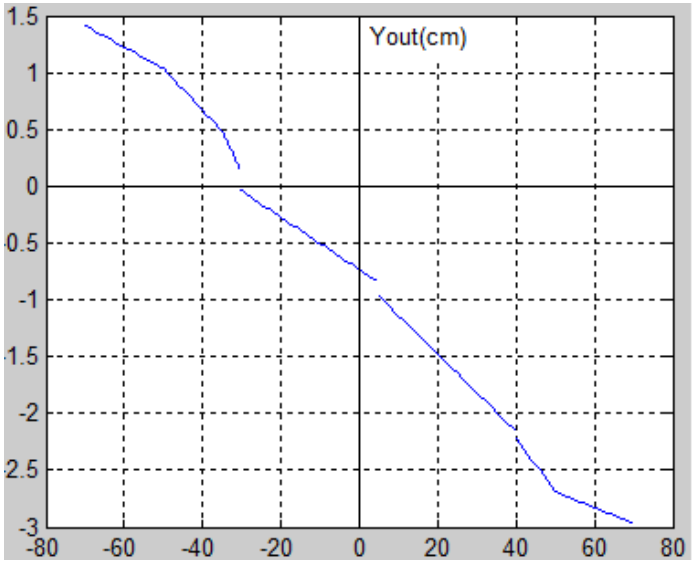
f=-0.023*xd-0.73 ,-30.5,5

f=-0.034*xe-0.798 ,5,40

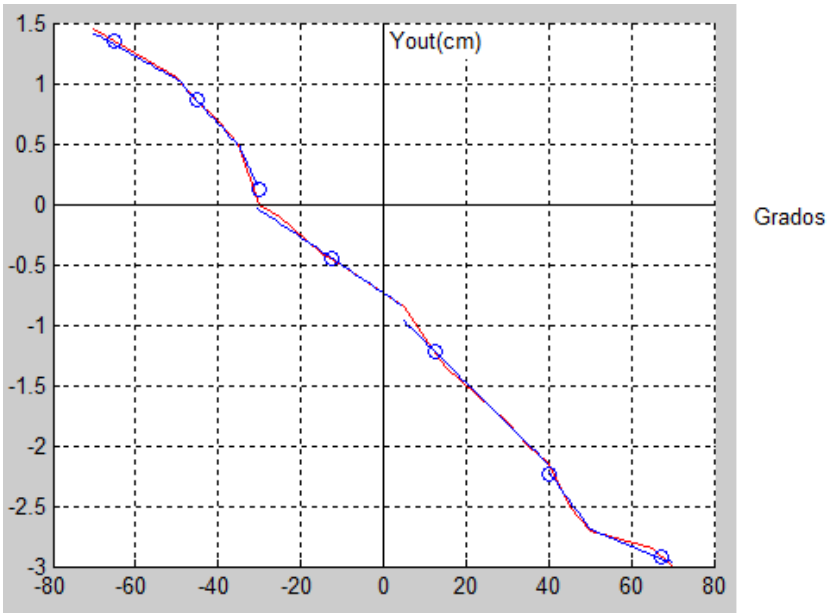
f=-0.047*xf-0.338 ,40,50

f=-0.014*xg-1.99 ,50,70

Gráfica Modelo Matemático

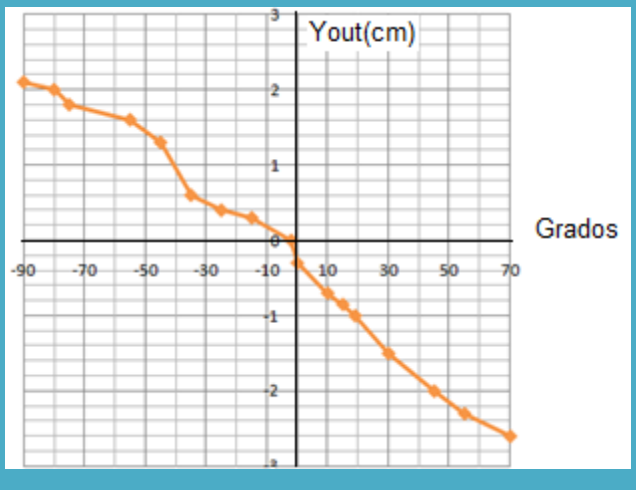


SUPERPOSICIÓN ENTRE MODELOS



EXTREMIDAD 5:

Curva Experimental



Número de clusters:3 **Centros de cluster:**

Θ=-81.6 y_{out} = 1.96

Θ=-40.12 y_{out} = 0.97

Θ=52.2 y_{out} = -2.16

Número de clusters:5 **Centros de cluster:**

Θ=-81.68 y_{out} = 1.97

Θ=-49.75 y_{out} = 1.44

Θ=-24.78 y_{out} = 0.42

Θ=20.59 y_{out} = -1.09

Θ=57.1 y_{out} = -2.31

Número de clusters:7 **Centros de cluster:**

Θ=-90 y_{out} = 2.1

Θ=77.49 y_{out} = 1.9

Θ=-50.06 y_{out} = 1.45

Θ=-30.37 y_{out} = 0.50

Θ=-15.06 y_{out} = 0.30

Θ=20.61 y_{out} = -1.09

Θ=57.11 y_{out} = -2.31

Modelo Matemático(Función a trozos)

f=-0.010*t+1.2 ,-90,-80

f=-0.040*t-1.212048193, -80,-75

f=-0.014*t+0.747473937, -75,-50

f=-0.047*t-0.93244898, -50,-35

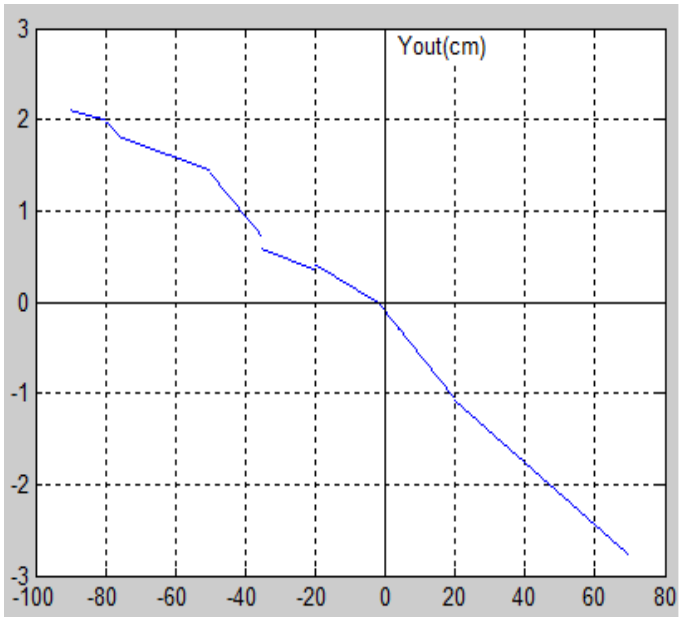
f=-0.015*t+0.05583181, -35,-20

f=-0.023*t-0.04778626, -20,-2

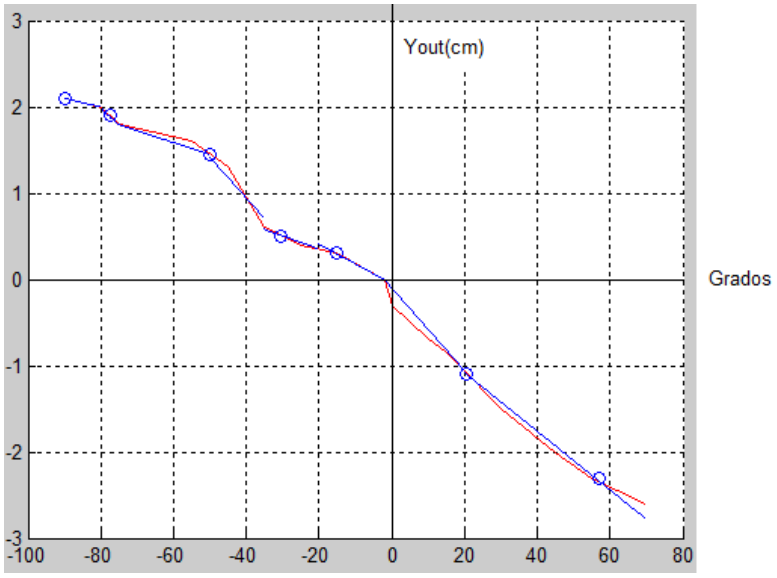
f=-0.048*t-0.096417514, -2,20

f=-0.034*t-0.396399341, 20,70

Gráfica Modelo Matemático

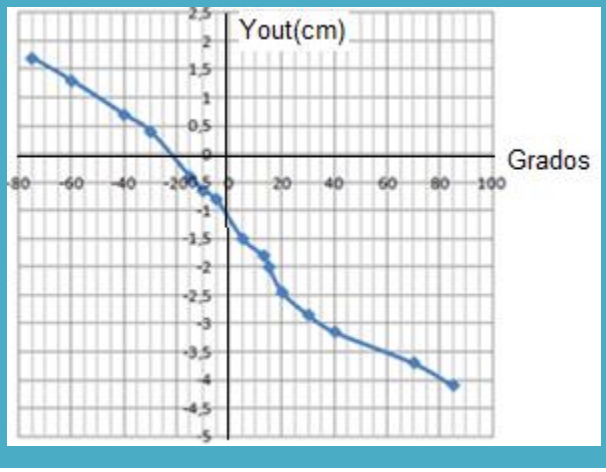


SUPERPOSICIÓN ENTRE MODELOS



EXTREMIDAD 6:

Curva Experimental



Número de clusters:3

Centros de cluster:

Θ=-67.54 y_{out} = 1.50

Θ=-34.23 y_{out} = 0.51

Θ=66.81 y_{out} = -3.68

Número de clusters:5

Centros de cluster:

Θ=-67.56 y_{out} = 1.502

Θ=-35.03 y_{out} = 0.55

Θ=-12.35 y_{out} = -0.53

Θ=27.24 y_{out} = -2.66

Θ=77.5 y_{out} = -3.9

Modelo Matemático(Función a trozos)

f=-0.029*t-0.465156154,-75,-35

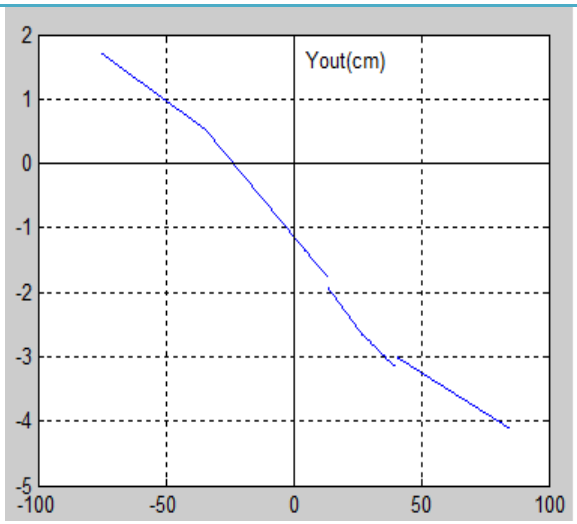
f=-0.048*t-1.118095238,-35,13

f=-0.054*t-1.194448093,13,27

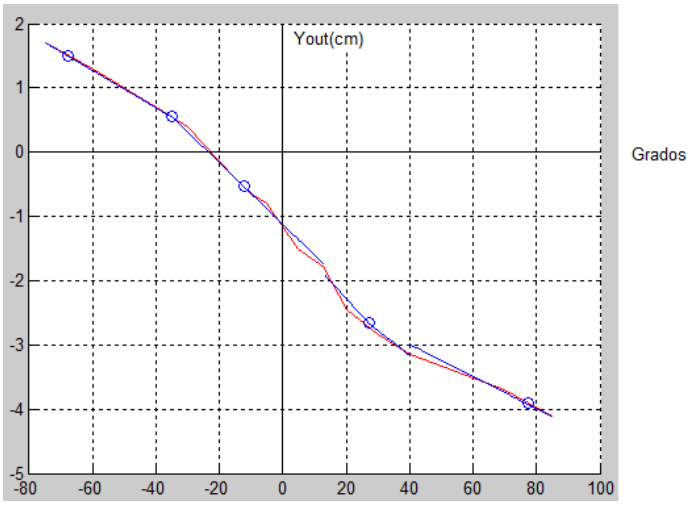
f=-0.039*t-1.606692913,27,40

f=-0.025*t-1.987942698,40,85

Gráfica Modelo Matemático



SUPERPOSICIÓN ENTRE MODELOS



5.2.1. Validación del modelo obtenido mediante Fuzzy C-Means.

Para realizar la validación del modelo se varió las posiciones angulares de las extremidades y se midió el desplazamiento vertical de las mismas, para ser comparado posteriormente con los datos obtenidos a partir de los modelos, así mismo se obtiene una relación del error sobre el modelo.

Se ejecutaron pruebas de validación sobre el modelo generado con el fin de corroborar la veracidad de las funciones a trozos.

Para validar el modelo obtenido, se procedió a realizar variaciones sobre la posición angular de los servo-motores para cada extremidad y se midió el valor del desplazamiento vertical, estos datos fueron comparados con los obtenidos mediante el modelo.

EXTREMIDAD 1:

Modelo		**Experimental**	**Error**
θ (º)	**y_{out} (cm)**	**y_{out} (cm)**	**Δy_{out} (cm)**
35	-1.13	-1.1	0.03
45	-1.67	-1.70	0.03
-45	1.56	1.50	0.06
30	-0,77	-0,70	0.07

Tabla 1.8 validación modelo para extremidad 1.

EXTREMIDAD 2:

Modelo		**Experimental**	**Error**
θ (º)	**y_{out} (cm)**	**y_{out} (cm)**	**Δy_{out} (cm)**
20	-0,15	-0,15	0
35	-0,94	-0,95	0.01
50	-1,42	-1,63	0.21
-25	1.02	1.25	0.23

Tabla 1.9 validación modelo para extremidad 2

EXTREMIDAD 3:

Modelo		**Experimental**	**Error**
θ(º)	**y_{out}(cm)**	**y_{out}(cm)**	**Δy_{out}(cm)**
45	-1,91	-1.9	0.01
25	-1,23	-1.35	0.12
-30	0,49	0.5	0.01
-50	1,4	1.4	0

Tabla 1.10 validación modelo para extremidad 3.

EXTREMIDAD 4:

Modelo		Experimental	Error
θ(°)	y_{out}(cm)	y_{out}(cm)	Δy_{out}(cm)
-30	0.4	0	0.4
50	-2.68	-2,7	0.02
-35	0,49	0.55	0.06
-65	1.3	1.2	0.1

Tabla 1.11 validación modelo para extremidad 4.

EXTREMIDAD 5:

Modelo		Experimental	Error
θ(°)	$y_{out}(cm)$	y_{out}(cm)	Δy_{out}(cm)
-55	1,53	1.6	0.07
-40	0,93	0.75	0.18
20	-1.08	-1.22	0.14
-35	0,7	0.6	0.1

Tabla 1.12 validación modelo para extremidad 5.

EXTREMIDAD 6:

Modelo		Experimental	Error
θ(°)	y_{out}(cm)	y_{out} (cm)	Δy_{out} (cm)
-40	0,6	0.5	0.1
-70	1.56	1.5	0.06
15	-2	-1.9	0.01
20	-2,27	-2.6	0.33

Tabla 1.13 validación modelo para extremidad 6.

6. HARDWARE Y SOFTWARE.

6.1. HARDWARE IMPLEMENTADO.

El hardware con el que se trabajó para el diseño de los módulos con la jerarquía Maestro - Esclavo del robot fueron las placas ARDUINO UNO (Módulo Maestro del sistema) y

ARDUINO MINI (se implementaron dos placas para utilizarlas como Módulos esclavos, cada uno con la función de controlar el movimiento de tres extremidades).

1. **Compatibilidad:** Debido a que el área del diseño y programación de procesadores es demasiado amplia, es necesario implementar componentes altamente compatibles, puesto que se pretende dar libertad en temas como la implementación de software de diseño y programación para micro controladores, en ese punto.

2. **Complejidad:** Se dispuso pensando en el usuario final de la plataforma, puesto que no se pretende enfatizar la implementación del robot en un área específica sino que se pretende dejarla abierta para su uso en varias áreas de la carrera.
    - Programación de micro controladores (DISPRO).
    - Modelamiento de robots.
    - Diseño e implementación de sistemas de control.

3. **Disponibilidad:** Se pretende entregar un sistema que asegure una continuidad operacional durante un período de tiempo. Por esta razón, los componentes deben funcionar de manera correcta para asegurar su correcto funcionamiento.

4. **Costo y Empaquetado:** Debido a que la plataforma estará a disposición de un gran número de estudiantes, es vital implementar en ella, software asequible y fácil de conseguir, puesto que si se presenta algún accidente con algún módulo o parte de la tarjeta, sea sencillo y económico repararla.

5. **Escalabilidad:** De igual forma, se presenta un hardware amplio, idóneo de ser implementado en diversas aplicaciones fuera de la plataforma hexápoda.

La arquitectura que se propuso para el desarrollo del proyecto está dispuesta de la siguiente manera:

1. Micro controlador maestro. Este controla:
    - Esclavos X2.
    - Puertos disponibles para adicionar hardware (cámaras, solenoides, actuadores, etc.)
2. Micro controlador esclavo. Este controla:
    - 3 Extremidades (6 servos).
    - 3 Interruptores de fin de carrera (opcional-disponibles)
    - 6 Puertos I/O disponibles para agregar hardware adicional.

- **Hardware:** El sistema Arduino propone implementar un hardware ya ensamblado, con placas diseñadas con componentes sobre un montaje superficial.
- **Espacio y diseño**: Los micro controladores con los que se trabajaron eran de gran tamaño, por lo tanto el circuito impreso diseñado fue de gran dimensión haciendo que a la plataforma le aportara peso y para algunas extremidades obstrucción de movimiento. Por el contrario las placas Arduinos son de menor tamaño y peso y estéticamente son mejores.
- **Memoria:** Las memorias física de los Arduinos tanto Flash, SRAM y EEPROM son superior que la de los micro controladores utilizados logrando así poder grabar mayor cantidad de código. [38]
- **Comunicación:** Se implementó la comunicación entre los módulos Arduino vía USART RX/TX, pues se encontraron características adecuadas dentro de la librería SERIAL, como la función "flush" la cual refresca los buffers para la trasmisión de datos.
- **Investigación:** Una de las metas de este proyecto es habilitar la plataforma para entregarla como insumo para la investigación en la Universidad, desarrollo o aprendizaje para la Facultad de Ingeniería Electrónica en la Universidad Javeriana

6.2. DESCRIPCIÓN DE LAS TARJETAS DE DESARROLLO ARDUINO.

"Arduino es una plataforma de hardware de código abierto, basada en una sencilla placa con entradas y salidas, analógicas y digitales, en un entorno de desarrollo que está basado en el lenguaje de programación Processing. Es un dispositivo que conecta el mundo físico con el mundo virtual, o el mundo analógico con el digital" [39]

Sus creadores son el zaragozano David Cuartielles, ingeniero electrónico y docente de la Universidad de Mälmo, Suecia y Massimo Banzi, italiano, diseñador y desarrollador Web. El proyecto fue concebido en Italia en el año 2005". [39][40][41]

6.2.1. Arduino Uno.

"El Arduino Uno es una placa de diseño de múltiples aplicaciones con un microcontrolador base ATmega328. Este posee 14 entradas/salidas digitales, de las cuales 6 se pueden implementar como salidas PWM. Además, también posee 6 entradas análogas, un cristal de 16Mhz como oscilador principal, conexión USB, entrada de poder para conexiones de toma para alimentación, ICSP header, y un pulsador de reset". [42]

La placa Arduino Uno no utiliza el adaptador USB-SERIAL, En vez de eso, el microcontrolador utiliza la ATMEGA8U2 programado como un conversor USB-SERIAL.
La ATmega328 tiene 32 KB de memoria flash, posee 2 KB de SRAM y 1 KB de EEPROM

Cada uno de los 14 pines del Arduino Uno pueden ser usados como entrada o salida. Estos pines operan con un valor de voltaje máximo de 5v y 40mA y poseen una resistencia de pull-up (por defecto) de 20-50 kOhms. [42]

ESPECIFICACIONES TÉCNICAS (Tarjeta Arduino UNO).

Microcontrolador	ATMEGA 328
Voltaje de operación.	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límites)	6-20V
Pines Digitales I/O	14 (de los cuales se dividen 6 para salida PWM)
Pines de entrada Análogos	6
Corriente DC por pin I/O	40 mA
Corriente DC por pin 3.3V	50 mA
Memoria Flash	32 KB (ATmega328) los cuales 0.5 KB usado para bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidad de reloj	16 MH

Cuadro 1.14. Especificaciones técnicas del Arduino Uno. (Tomado de [42])

Figura 1.42. Placa Arduino Uno implementada como Maestro del sistema Hexápodo (tomado de [42])

6.2.2. Arduino Mini.

"El Arduino Mini es una placa con un ATMEGA168 como microcontrolador, cuenta con 14 puertos digitales ya sean entradas o salidas de los cuales 6 se pueden utilizar como salidas PWM, quedando las otras 8 como entradas análogas. De oscilador principal posee un cristal de 16 MHz" [43]

Por su tamaño, no posee un hardware directo para ser programado, por esta razón debe utilizar un adaptador mini USB especializado.

ESPECIFICACIONES TÉCNICAS.

Microcontrolador	ATMEGA168
Voltaje de operación.	5V
Voltaje de entrada	7-9 V
Pines Digitales I/O	14 (de los cuales se dividen 6 para salida PWM)
Pines de entrada Análogos	8
Corriente DC por pin I/O	40 mA
Memoria Flash	16 KB los cuales 2 KB usado para bootloader
SRAM	1 KB
EEPROM	512 bytes
Velocidad de reloj	16 MHz

Cuadro 1.15. Especificaciones técnicas del Arduino Mini. (Tomado de [43])

Cada uno de sus 14 pines se utilizan como entrada o salida, estos operan a un voltaje de 5 V de los cuales están dispuestos a proveer o recibir un máximo de 40 mA y tienen una resistencia interna de pull-up (desconectada por defecto) de 20-50 kOhms. De igual manera los pines 3, 5, 6, 9, 10 y 11 se pueden manejar como salida PWM. [43]

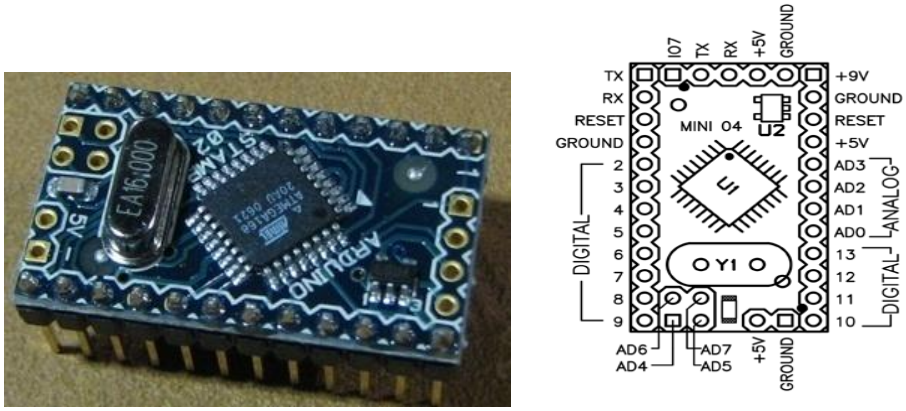


Figura 1.43. Placa Arduino Mini implementada como Esclavo del sistema Hexápodo. (Tomado de [43])

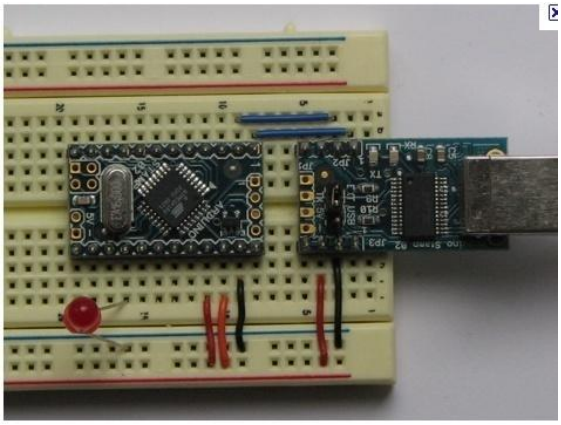


Figura 1.44. Circuito para la Placa Arduino Mini con el adaptador USB para su programación (tomado de [43]).

6.3. HARDWARE ADICIONAL.

Para este proyecto, se diseñó y ensambló una tarjeta madre sobre la cual se montaron los módulos Arduino Mini, el adaptador de programación para los Arduino Mini y se dispuso de pines hembra para implementar las demás características de las tarjetas para futuras aplicaciones. Además se montó una *protoboard* sobre la cual se adicionó el circuito encargado del accionamiento para la secuencia de movimiento.

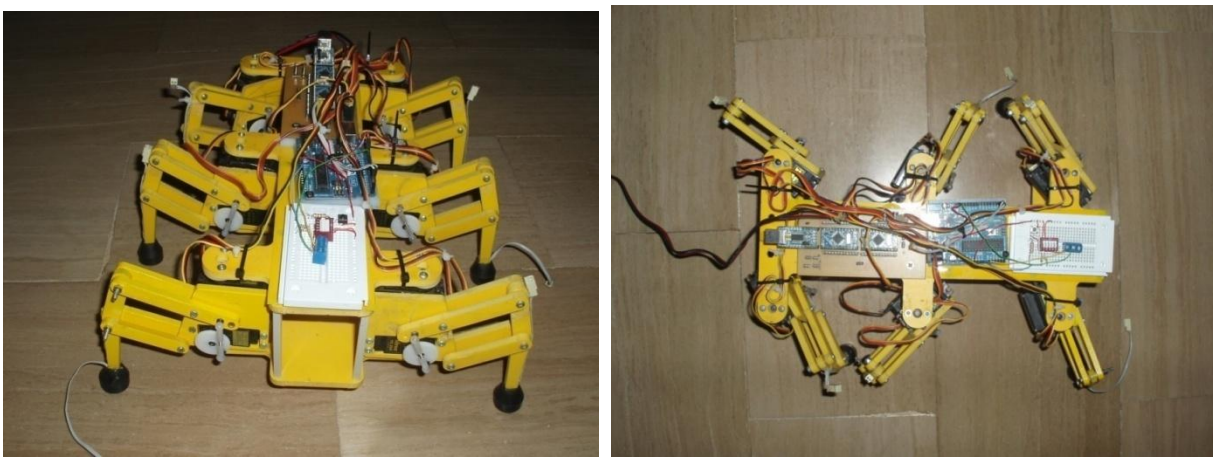


Figura 1.45. Plataforma Hexápodo con sus respectivos módulos.. (Estructura completa)

Las características principales que dispone la tarjeta madre son:

- Capacidad de conexión de dos módulos Arduino mini.
- Conectores disponibles para 12 servomotores.
- Conexión para Adaptador USB.
- *Jumpers* intercambiables para configurar los diferentes modos de programación:
    - ✓ Maestro-Esclavos.
    - ✓ Esclavo1-PC.
    - ✓ Esclavo2-PC.

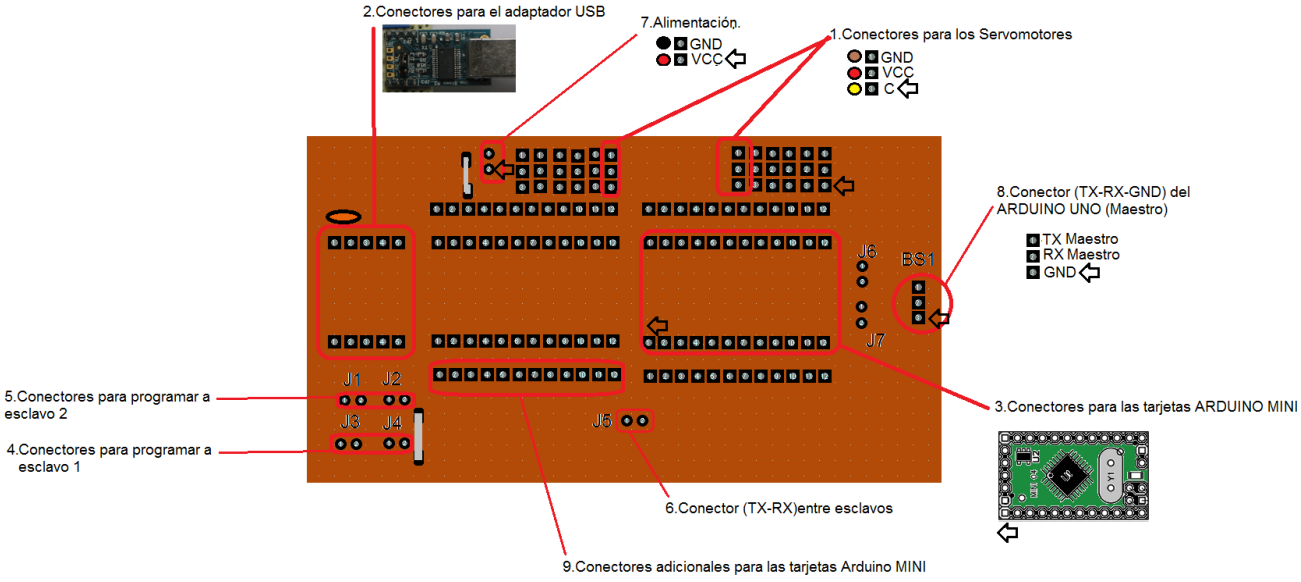


Figura 1.46 Conexiones tarjeta madre.

En el cuadro 1.16 está la descripción de la configuración de los *jumpers* utilizados en la tarjeta.

Se dispone de siete *jumpers* los cuales están distribuidos de la siguiente manera:

- Los *jumpers* J1 y J2 para la comunicación de programación entre esclavo 1 y computador, cabe aclarar que para un correcto uso es necesario tener los dos *jumpers* conectados y los demás desconectados es decir que los únicos *jumpers* que debe tener la tarjeta son los J1 y J2.
- Los *jumpers* J3 y J4 hacen la misma función y se debe tener el mismo procedimiento de conexión con respecto a los J1 y J2 pero estos son para la configuración de programación del esclavo 2.
- El *Jumper* J5 es la conexión directa para la comunicación TX-RX entre los esclavos. Cabe resaltar que si este jumper no se acciona no habrá comunicación entre esclavos y por lo tanto no habrá con el maestro y no se ejecutara la rutina de movimiento.
- Los *Jumpers* J6 y J7 hacen la conexión de comunicación directa entre el maestro y los esclavos.

	J1	**J2**	**J3**	**J4**	**J5**	**J6**	**J7**	**BS1**
PC-E1			X	X				
PC-E2	X	X						
Maestro-E1-E2					X	X	X	X

Cuadro 1.16 Configuración de los jumpers para la comunicación.

6.3.1. Configuración para ejecutar rutina de movimiento.

A continuación se presenta el esquema de conexiones. Configurando el sistema de esta manera y programando los módulos con los programas adjuntos, se logrará que el sistema ejecute una secuencia de pasos. Al oprimir el interruptor S1 una sola vez el robot dará un paso, al oprimir S2 éste ejecutará pasos indefinidamente hasta que el interruptor se abra, al oprimir S3 la plataforma ejecutará los pasos a una velocidad menor.

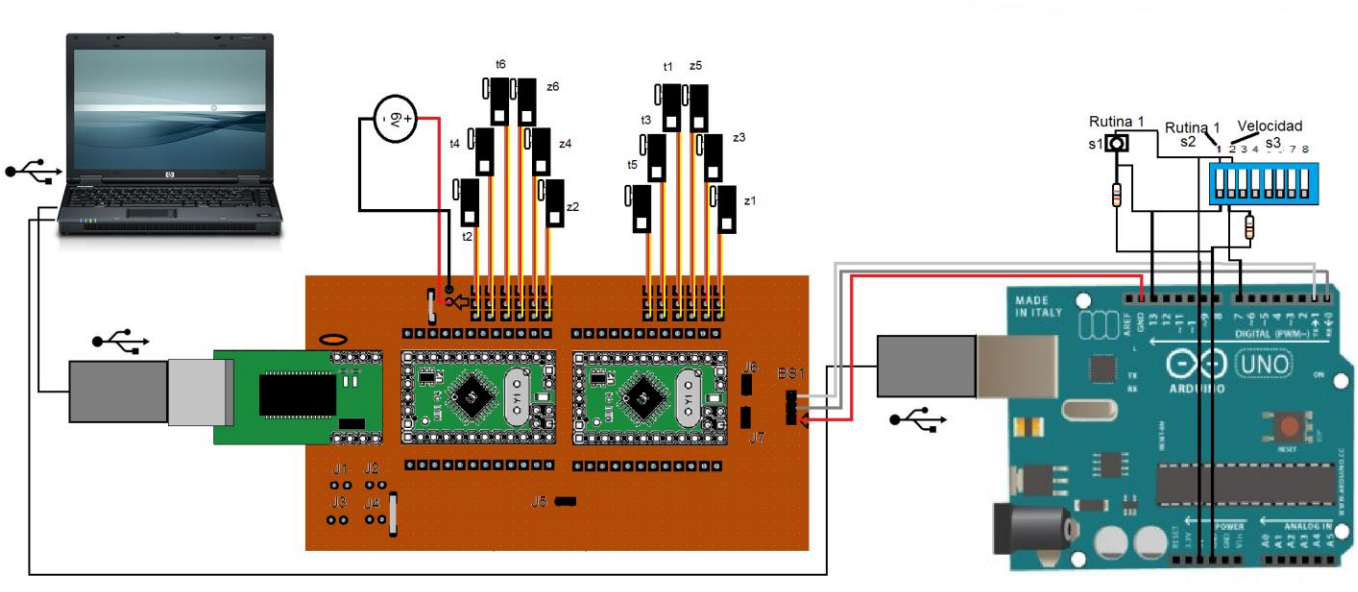


Figura 1.47 Conexiones maestro esclavo.

6.4. SOFTWARE.

Se propone una arquitectura jerárquica maestro-esclavo, encargada de generar e interpretar comandos sobre la plataforma junto con los servomotores, siendo la responsable de ejecutar las rutinas a diseñar. La arquitectura que se propone es tipo cíclica, siendo el maestro el encargado de dar las órdenes o comandos que generarán una reacción por parte de los esclavos que accionarán los servomotores encargados del movimiento sobre las extremidades.

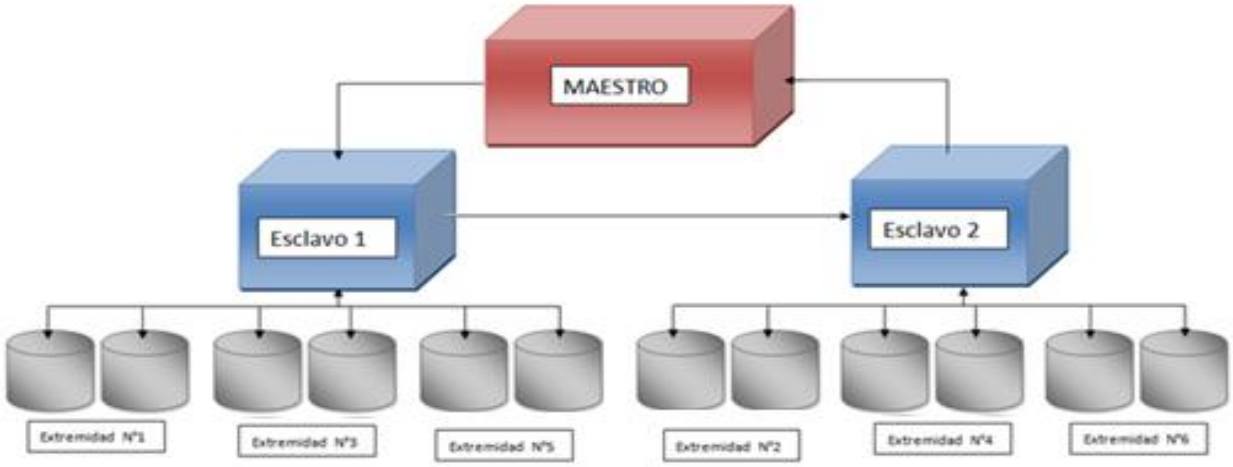


Figura 1.48. Arquitectura maestro-esclavo

La disposición de órdenes entre el maestro y los esclavos se organizará mediante el siguiente ciclo.

Figura 1.49.Diagrama de órdenes para el maestro.

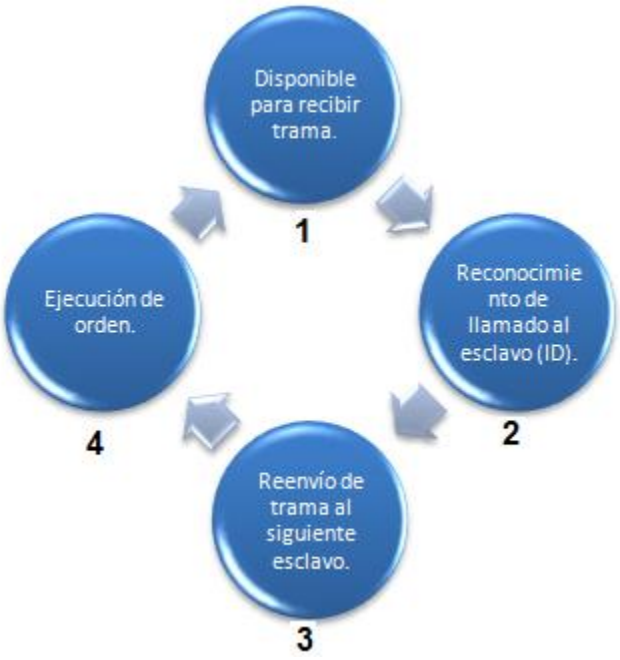
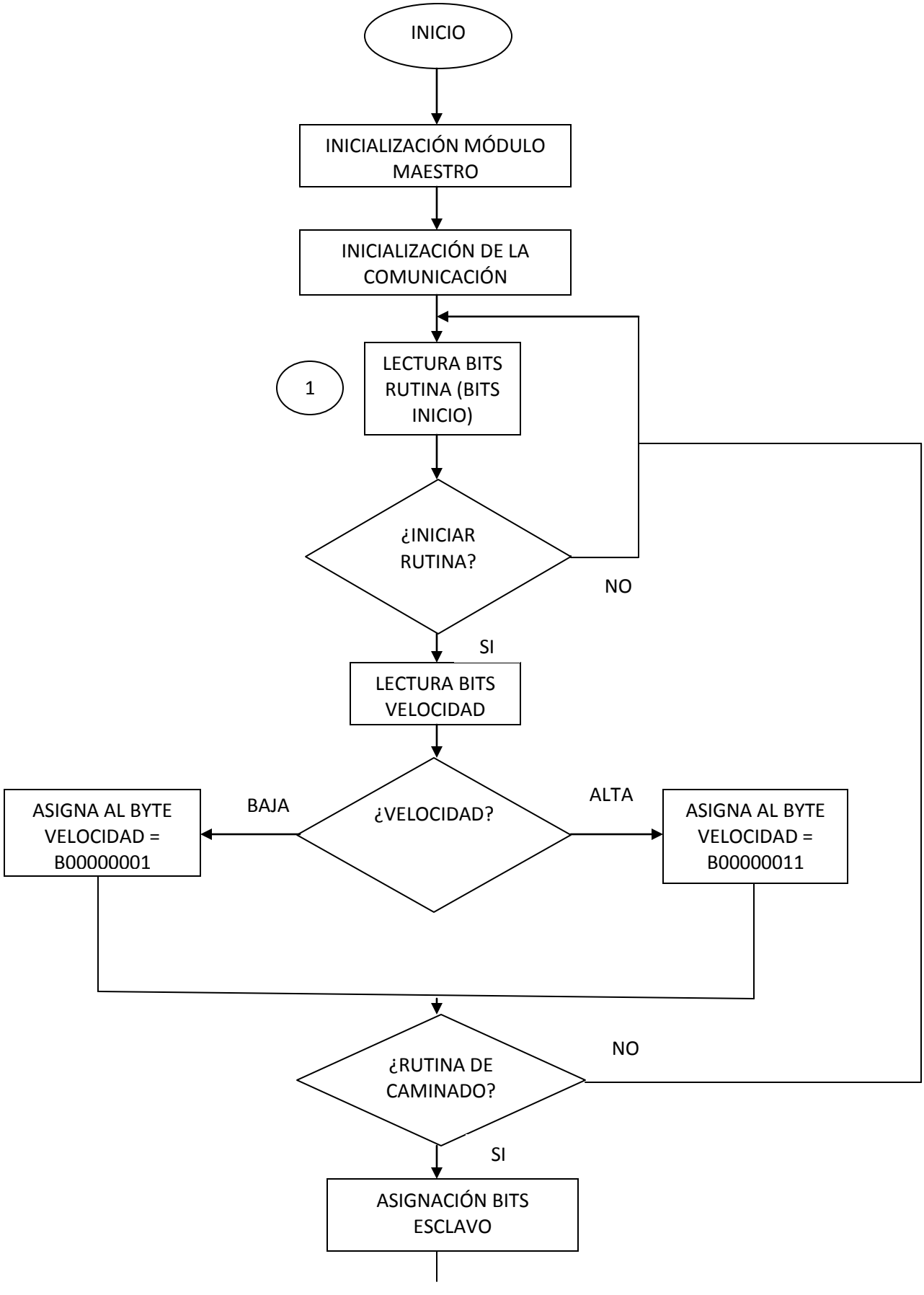


Figura 1.50. Diagrama de órdenes para el esclavo.


INICIO
INICIALIZACIÓN MÓDULO MAESTRO
INICIALIZACIÓN DE LA COMUNICACIÓN
1
LECTURA BITS RUTINA (BITS INICIO)
¿INICIAR RUTINA?
NO
SI
LECTURA BITS VELOCIDAD
¿VELOCIDAD?
BAJA
ALTA
ASIGNA AL BYTE VELOCIDAD = B00000001
ASIGNA AL BYTE VELOCIDAD = B00000011
¿RUTINA DE CAMINADO?
NO
SI
ASIGNACIÓN BITS ESCLAVO

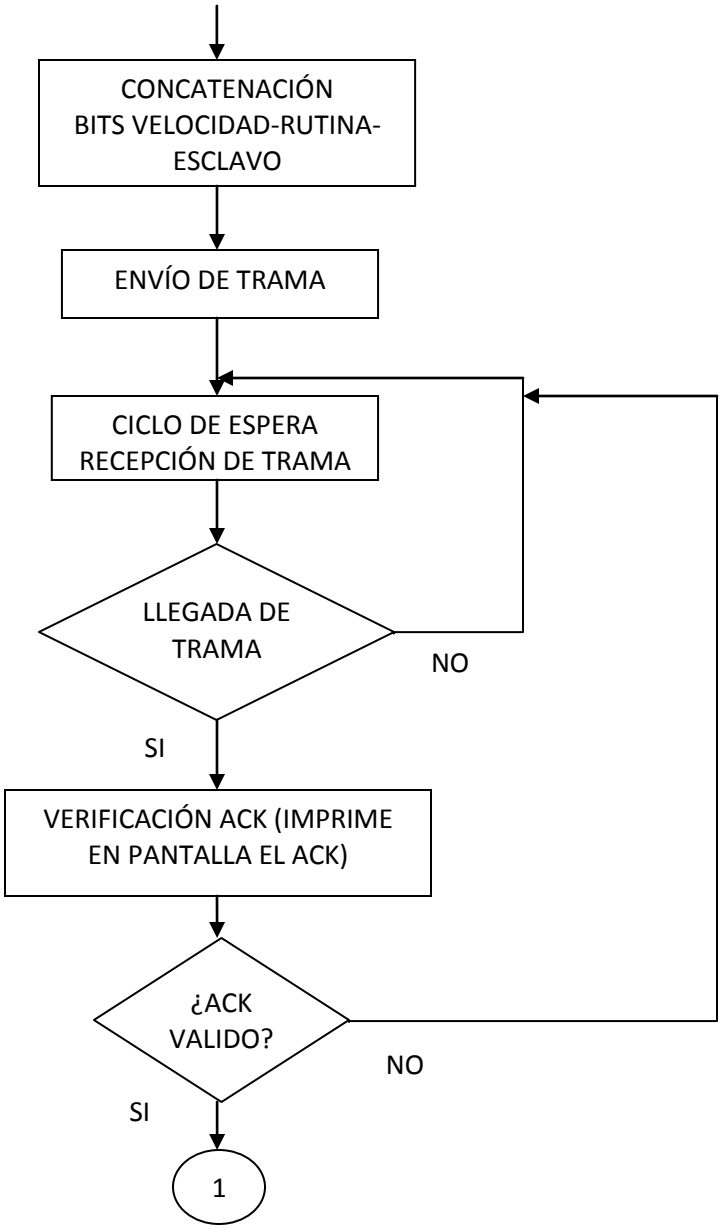
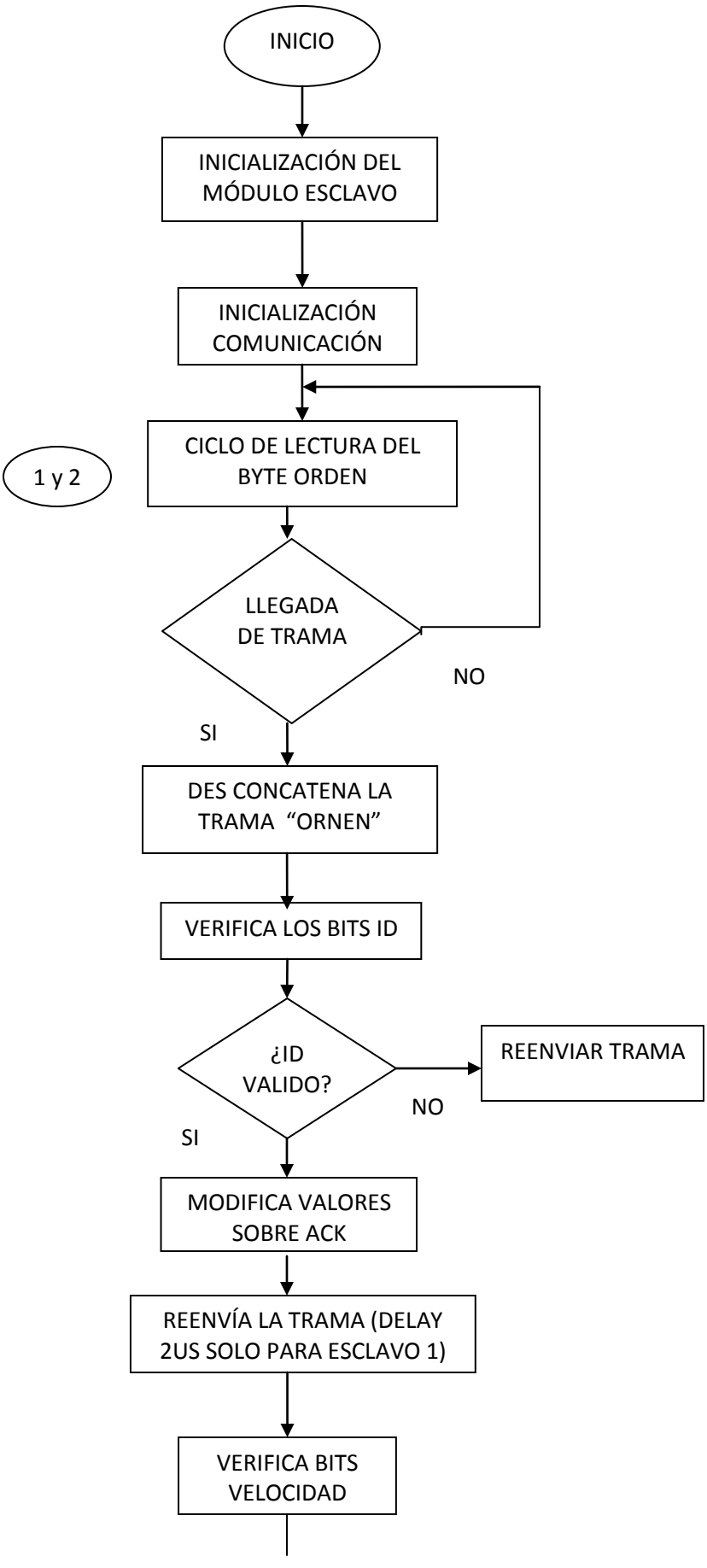


Diagrama de flujo 1.0. Software para el módulo Maestro.


INICIO
INICIALIZACIÓN DEL MÓDULO ESCLAVO
INICIALIZACIÓN COMUNICACIÓN
1 y 2
CICLO DE LECTURA DEL BYTE ORDEN
LLEGADA DE TRAMA
NO
SI
DES CONCATENA LA TRAMA "ORNEN"
VERIFICA LOS BITS ID
¿ID VALIDO?
REENVIAR TRAMA
NO
SI
MODIFICA VALORES SOBRE ACK
REENVÍA LA TRAMA (DELAY 2US SOLO PARA ESCLAVO 1)
VERIFICA BITS VELOCIDAD

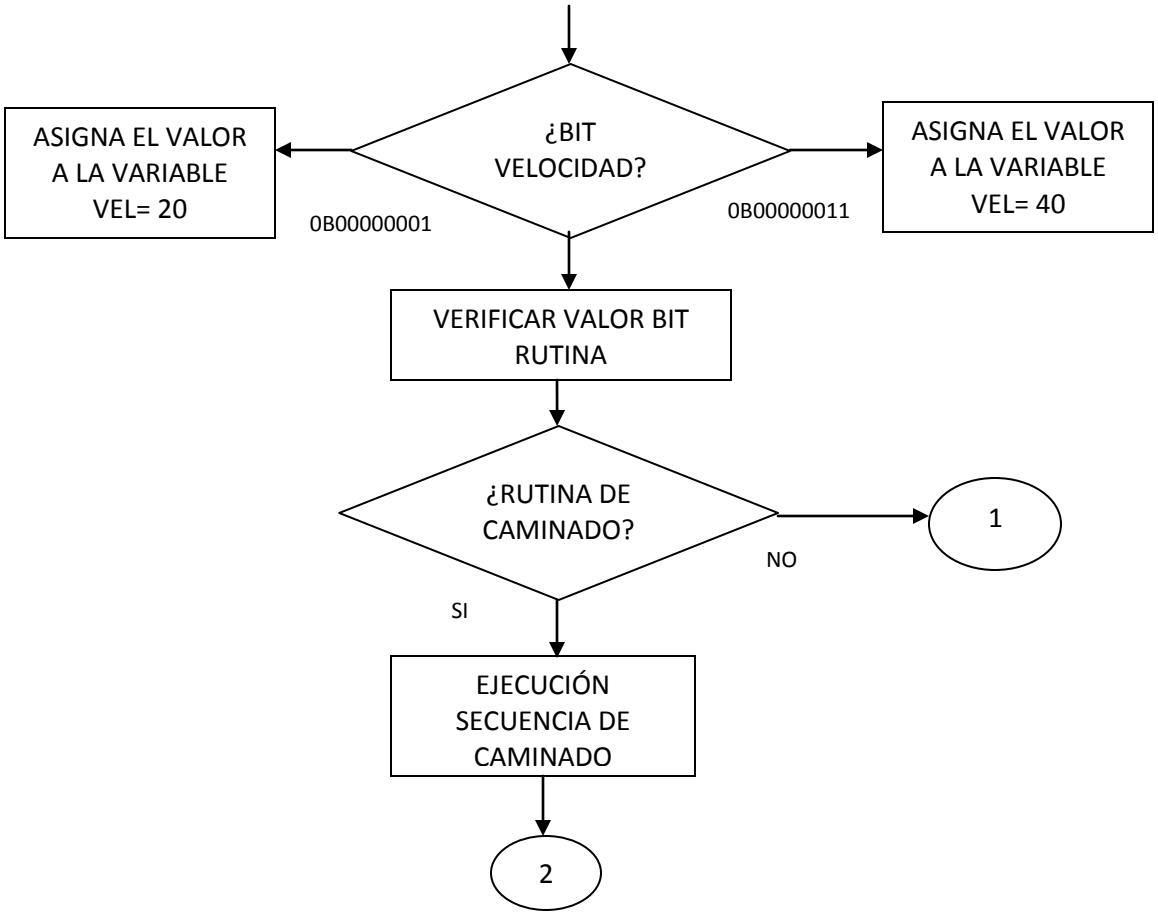


Diagrama de flujo 1.1. Software para los módulos Esclavos.

FUNCIONES MAESTRO.
Modificar Bits de rutina.
Modificar bits de velocidad.
Modificar Bits de ID.
Concatenar el Byte de comando.

FUNCIONES ESCLAVOS.
Modificar bits de ACK.
Envió señal de control hacia los servos motores.
Ejecuta secuencia de movimiento.

Cuadro 1.17. Funciones de los módulos Maestro y Esclavos.

La arquitectura del sistema mostrada en la figura 1.48 no tiene en cuenta la habilitación del hardware que tendrá a su disposición la plataforma, pues la estrategia de movimiento inicial no necesitará de puertos extras. Se pretende dejar habilitado puertos para implementar la plataforma en diversas aplicaciones futuras.

6.5. PROTOCOLO DE COMUNICACIÓN.

Para realizar la comunicación entre los módulos ARDUINOUNO y ARDUINOMINI, se manejaron los puertos de trasmisión y recepción de cada una de las tarjetas (vía USART), dispuestos de la siguiente forma:

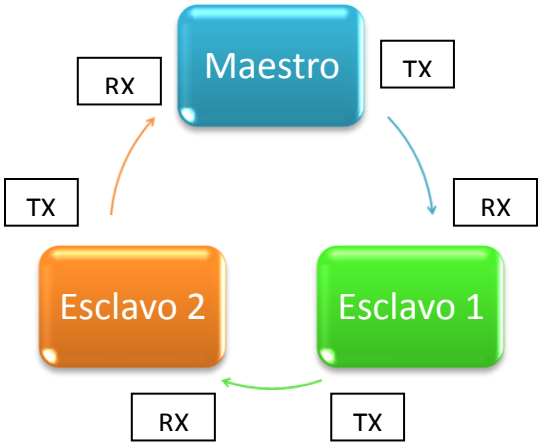


Figura 1.51 Diagrama de bloques protocolo de comunicación, Red en anillo

La figura 1.51 muestra una comunicación en formación de "anillo", mediante esta arquitectura se evita una posible colisión de datos y facilita el seguimiento de los mensajes sobre la red.

La trama de orden se estructura de la siguiente manera:

B7	B6	B5	B4	B3	B2	B1	B0
VELOCIDAD	VELOCIDAD	ID ESC	ID ESC	ID ESC	ID ESC	RUTINA	RUTINA

Cuadro 1.18 trama de orden de la estructura.

Rutina: Para el presente trabajo, se contempla la programación de una única rutina de caminado, pero se deja a disposición de futuras aplicaciones (2-bits) para seleccionar rutinas.

Esclavo: Para que el maestro reconozca que el mensaje fue recibido por cada uno de los esclavos se disponen de 4 bits correspondientes a la identificación-esclavo (ID ESC), cuando el maestro envíe el mensaje por la red de comunicación hacia el primer esclavo, este identificará si le corresponde ejecutar la orden proveniente, de ser así el esclavo modificará el ID ESC y lo enviará al esclavo 2 que realizará lo mismo. Una vez llegue el mensaje, el maestro identificará

por medio de la comparación de la orden enviada y recibida, si el mensaje fue entendido y ejecutado.

DE	PARA	ID ESC 3	ID ESC 2	ID ESC 1	ID ESC 0
Maestro	Esclavo1	1	0	0	X
Esclavo1	Maestro	1	1	0	X
Maestro	Esclavo2	1	0	1	X
Esclavo2	Maestro	0	1	1	X
Maestro	Esclavo 1 y 2	1	1	1	X
Esclavos 1 y 2	Maestro	1	1	0	X

Cuadro 1.19 rutinas de comunicación entre esclavo y maestro.

Velocidad: Estos 2 bits se encuentran dispuestos para que el usuario pueda seleccionar la velocidad sobre el movimiento de la plataforma, disponibles como velocidad BAJA y ALTA.

6.6. SINCRONIZACIÓN DEL SISTEMA.

Los módulos Arduino disponen de relojes a 16 MHz, esto equivale a 62.5*n*s por ciclo de reloj. Debido a la arquitectura de la comunicación la sincronización entre esclavos se realizó de la siguiente manera:

Al momento en el que Esclavo 1 transmite la trama (proveniente del maestro) a Esclavo 2 este espera un tiempo muerto equivalente a 32 instrucciones (este número corresponde a la diferencia entre el inicio de la ejecución de la rutina del esclavo 2 con respecto al esclavo 1), puesto que cuando la trama llega al esclavo 2 éste debe realizar las mismas verificaciones con respecto al ID que el esclavo 1, por tal razón se le ingresa un retardo al esclavo 1 con el fin de que ambos esclavos inicien la ejecución del movimiento a la vez. Por otra parte, es de aclarar que este retardo entre esclavos equivaldría a una diferencia de 2 µs entre ejecuciones. Este tiempo es despreciable en comparación al tiempo que tarda en variar un grado la extremidad, este tiempo es de aproximadamente 0,017s = 17000 µs. Por tanto la asincronía entre extremidades equivaldría a 0.01% de diferencia en grados aproximadamente en caso de no ingresar el delay de 2 µs.

7. PRUEBAS DE CALIDAD Y DESEMPEÑO.

Para validar la eficiencia sobre la estrategia de movimiento, se propone a través de mediciones de velocidad de la plataforma, realizar una comparación entre la ejecución de caminado con velocidad baja y alta.

1. Se posiciona la plataforma sobre un punto de referencia inicial sobre el extremo frontal del robot, a partir de este punto se toman las medidas de distancia.
2. Se ejecuta la rutina de movimiento, variando el número de pasos a ejecutar.
3. Se toma el tiempo desde el inicio de la rutina hasta que la plataforma quede en reposo.
4. Se mide la distancia entre el punto de referencia y la posición final del robot.
5. Se repiten los anteriores pasos, después se varia la velocidad y se ejecutan la rutina nombrada anteriormente.

tiempo(s)	distancia(cm)	velocidad(cm/s)
3,6	13,5	3,75
3,5	12,3	3,51
3,31	14	4,23
3,17	13	4,10
6,2	27	4,35
6,8	25	3,68
6,81	28	4,11
6,69	24	3,59
17,08	49	2,87
13,82	46	3,33
10,45	35	3,35
7,5	26	3,47

tiempo(s)	distancia(cm)	velocidad(cm/s)
1,82	12,5	6,87
1,74	13	7,48
3,43	26,5	7,73
3,63	27,5	7,58
7,28	48	6,60
7,29	48,5	6,65

Cuadro 1.20 Muestreo de velocidad (Izq. Desplazamiento lento – Der desplazamiento rápido).

Una vez obtenidos los datos, se promedian y se realiza una comparación entre los datos obtenidos para los dos tipos de velocidades.

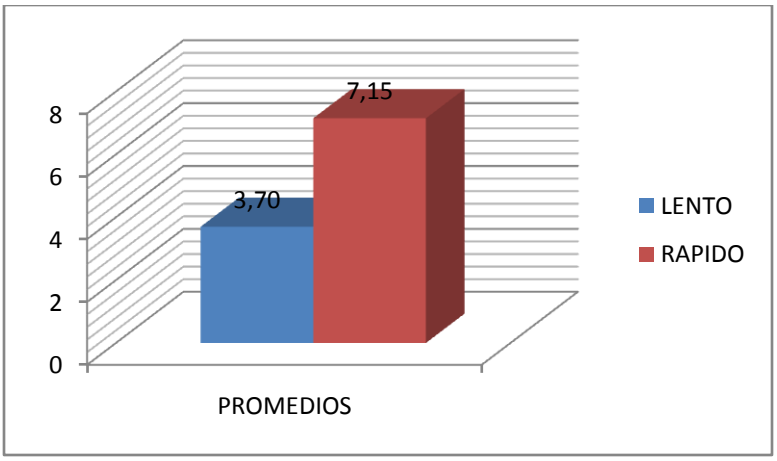


Figura 1.52 Comparación velocidad de ejecución de rutina.

Analizando la gráfica obtenida, se evidencia un incremento de velocidad equivalente al 51,75%. Por otra parte se evidencia un desvío del robot al momento de realizar el avance, éste se presenta debido a la variación de las fuerzas ejercidas por parte de los servos, al no ser iguales los servomotores nuevos ejercen un torque mayor a la de los demás, desviando a la plataforma de su curso.

Para la sección de calidad, se diseñaron los siguientes casos de prueba a cumplir, esto con el fin de verificar la funcionalidad del anillo de comunicación

Pruebas funcionales:

NOMBRE DEL CASO	ESTADO
CP_01 Envío de trama para velocidad: Alta, Ambos esclavos, Secuencia caminado.	✓
CP_02 Envío de trama para velocidad: Alta, Esclavo 1, Secuencia caminado.	✓
CP_03 Envío de trama para velocidad: Alta, Esclavo 2, Secuencia caminado.	✓
CP_04 Interrupción en el anillo de comunicación (esclavo 1 a esclavo 2).	✓
CP_05 Envío trama indefinida.	✓

El soporte del cumplimiento de los casos de prueba anteriormente descritos se obtuvo mediante video (VER ARCHIVOS ANEXOS).

8. COSTOS Y FUENTES DE FINANCIACIÓN.

Para llevar a cabo este proyecto la plataforma fue suministrada por la Pontificia Universidad Javeriana y se hicieron los siguientes gastos.

• Reparación de una de las extremidades de la plataforma realizando la fabricación.	
MATERIALES	
Nombre	**Precio**
Lámina de acrílico	$ 15,000
Diseño y corte de la extremidad faltante	$ 35,000
Dos servomotores	$ 50,000
Rótula plástica	$ 15,000
Tornillos	$ 1,000
Spray amarillo	$ 10,000
Amarres plásticos	$ 5,000
Cinta adhesiva doble cara	$ 6,000

• Adquisición Arduinos	
MATERIALES	
Nombre	**Precio**
Arduino Uno	$ 85,000
Arduino Mini (2)	$ 100,000
Adaptador	$ 25,000
Dos Cables USB	$ 20,000

MATERIALES	
Nombre	**Precio**
Fabricación diseño impreso tarjeta madre	$ 18,000
Componentes electrónicos(proto board pequeña, baquela, soporte plástico	$ 40,000
para la tarjeta, jumpers, deep switch, pulsadores, resistencias, condensador,	
cable, cinta termoencogible, molex, regleta hembra y macho, soldadura)	
Servomotores nuevos	$ 75,000
Desplazamientos	$ 100,000
Fuente De alimentación	$ 100,000

Estación de soldadura	$ 200,000
Computador portátil	$1,850,000
DVM	$ 80,000
TOTAL	**$2,780,000**

9. CONCLUSIONES.

Como la plataforma se encontraba en un estado de abandono y deterioro debido a la falta de mantenimiento y uso, esto tuvo como consecuencia la pérdida de las propiedades mecánicas de la misma junto con el hecho de que al faltarle una extremidad y al ser su principal material (PVC expandido) difícil de conseguir se opto por la búsqueda de un material con características similares al PVC y así ensamblar la extremidad faltante a partir de un material diferente (acrílico).

La elaboración de un modelo adecuado a las necesidades del sistema no radica únicamente en el análisis matemático, las propiedades de los materiales, desgaste e interacción de las mismas, representan variaciones significativas a la hora de caracterizar el sistema, agregando un componente extra de complejidad al momento de modelar la plataforma, toda variación presentada sobre la configuración de las partes en las extremidades, implican un cambio en la caracterización de las mismas.

Como la plataforma contaba con nueve servos de una marca con referencia descontinuada, debido a que los restantes (doce servos en total) se encontraban extraviados al momento de realizar este trabajo se vio la necesidad de adquirir dos nuevos servos de diferente marca. Se considera de vital importancia hacer que la brecha de diferencias entre componentes sea mínima, es decir, si se requiere un número *n* de componentes de una sola clase, en aras de realizar un trabajo en conjunto (servomotores del Hexápodo) es de preferir que estos componentes sean de una misma marca.

Con base en lo enunciado anteriormente se encontró que el principal problema de tener servos diferentes y unos con menos tiempo de uso, radica en que el torque de salida es desigual teniendo como consecuencia pérdidas del movimiento total de la plataforma.

El uso de los módulos ARDUINO con fines pedagógicos fortalece las habilidades de los estudiantes al momento de desarrollar soluciones de software, también promueven las buenas prácticas de programación.

La implementación tanto en software como hardware de la sección de comunicación para los módulos Arduino es versátil con base a nuestra experiencia, haciendo que en el desarrollo del software se realice de manera más sencilla.

Las pruebas de testing elaboradas y ejecutadas son de vital importancia ya que por medio de estas se puede validar el correcto funcionamiento del desarrollo de la plataforma tanto en software como hardware, mostrando que la arquitectura propuesta para la comunicación entre maestro y esclavos y la estrategia de movimiento diseñada compaginan en buena medida y tiene como resultado un movimiento fluido y continuo de la plataforma.

Es necesario aclarar que para realizar las pruebas y la validación de la estrategia de movimiento, la variación de tipos de superficie presenta inconvenientes ya que al posicionar la plataforma en una superficie con alto grado de fricción como lo es una alfombra, esta necesita realizar mayor esfuerzo y por las limitaciones de los servos, el movimiento no es fluido y presenta atascamientos en algunas extremidades lo que podría tener como consecuencia nuevas averías.

La disposición de los servos nuevos en una de las extremidades (extremidad 6) sobre un extremo de la plataforma genera una acción de fuerza no proporcional a las demás generando un desvío sobre la ruta de movimiento haciendo que su desplazamiento tenga tendencia diagonal en su ruta.

La implementación de la plataforma hexápoda (al tener significativas limitaciones sobre el torque generado por parte de los servos) sobre labores de investigación, se proyecta de una manera extremadamente limitada debido al desgaste de la misma, a pesar de que el sistema fue sometido a labores de reparación, se requiere reemplazar en gran medida los servomotores de piñones plásticos por unos de mejores características con el fin de rescatar las propiedades sobre el movimiento de las extremidades, por otra parte los costes de realizar un cambio de este tipo sugieren que adquirir una plataforma más actual, tendría un costo menor y mejores beneficios.

A la hora de implementar el algoritmo de Fuzzy C-Means se debe tener presente sus limitaciones entre las que se encuentran, la necesidad de que un usuario con conocimiento sobre la plataforma inicialice el algoritmo y determine si el número y la posición final de los clusters obtenidos son satisfactorios para general el modelo, esto debido a que la matriz de pertenencia se inicializa de manera aleatoria ocasionando que por cada ejecución del algoritmo la posición final de los clusters pueda variar.

Por otra parte debido a que el algoritmo Fuzzy C-Means tiene como finalidad reducir la función objetivo basada en la distancia euclidiana lo que conlleva a la generación de clusters circulares en el espacio bidimensional (θ vs yout), y los datos obtenidos a partir de la poción de la extremidad no se distribuyen en conjuntos que puedan ser aproximados a una figura circular si no que su distribución podría ser agrupada mediante clusters elipsoidales abriendo la posibilidad de implementar algoritmos como el de Gustafson Kessel capaz de agrupar dato con distribuciones elipsoidales.

10. ANEXOS Y COMPLEMENTOS.

10.1. ROBÓTICA.

Por siglos el hombre ha elaborado maquinas para realizar todo tipo de funciones, en especial para realizar labores tediosas o que requieran de una gran cantidad de esfuerzo o de gran precisión facilitando la vida y el trabajo de nuestra especie.

Con el paso de la historia y el desarrollo de la tecnología, se ha podido apreciar el progreso y evolución en la elaboración de las maquinas, desde la edad antigua con las civilizaciones egipcias y griegas las cuales desarrollaban e implementaban sistemas mecánicos e hidráulicos para construir estatuas y templos, pasando a la principal era que fuese la precursora de la robótica actual donde sucedió la revolución industrial en la que se impulso la elaboración y utilización de estos agentes mecánicos.

Solo para el año de 1917 el termino ROBOT se empezó a implementar con la obra "Rossum's Universal Robots" del escritor checo Karel Capek que utilizo la palabra *'Robota'* que tiene traducción en checo como "trabajador forzado", esta palabra la utilizo para describir a maquinas en forma de humanoide.

Entre las definiciones más generales de robótica se encuentra con la que la describe como "Ciencia o rama de la tecnología, que estudia el diseño y construcción de máquinas capaces de desempeñar tareas realizadas por el ser humano o que requieren del uso de inteligencia". [44]

La definición de robótica se implemento por primera vez por el escritor de ciencia ficción Isaac Asimov donde en sus obras expuso conceptos y leyes fundamentales de la robótica.

En el libro *Robótica Industrial* se define la robótica como una "ciencia aplicada que surge de la combinación de la tecnología de las máquinas-herramientas y de la informática". [45] [46]

Una máquina-herramienta se define como una máquina que efectúa cualquier trabajo manual, y la informática como la ciencia del tratamiento automático y racional de la información. Uniendo ambos conceptos la robótica surge al automatizar de manera racional las máquinas-herramienta, es decir al permitir que un programa informático controle las operaciones que antes realizaba un operario. [45]

Actualmente hay muchas aplicaciones en robótica como robótica industrial, robótica de servicio, robótica inteligente, robótica de exploración, Robótica bélica, etc... [47]

Se puede encontrar múltiples definiciones de robot pero conservando la esencia de su función como:

- Manipulador programable multifuncional, diseñado para mover piezas, herramientas, dispositivos especiales mediante movimientos variados, programados para la ejecución de diversas tareas. [48]
- Máquina controlada por ordenador y programada para moverse, manipular objetos y realizar trabajos a la vez que interacciona con su entorno. [48]
- Aparato automático que realiza funciones normalmente ejecutadas por los hombres. [48]
- Manipulador automático o servo controlado, reprogramable, polivalente, capaz de posicionar y orientar piezas, útiles o dispositivos especiales, siguiendo trayectorias variables reprogramables, para la ejecución de tareas variadas.[48]
- La definición adoptada por el Instituto Norteamericano de Robótica RIA (ROBOT INSTITUTE OF AMERICA) aceptada internacionalmente para Robot es: Manipulador multifuncional y reprogramable, diseñado para mover materiales, piezas, herramientas o dispositivos especiales, mediante movimientos programados y variables que permiten llevar a cabo diversas tareas. [48],[49]

10.1.1. CLASIFICACIÓN DE LAS ESTRUCTURAS ROBOTICAS.

Para analizar y estudiar la diversidad de robots es necesario establecer clases o grupos, esta clasificación depende de varios aspectos. A la hora de trabajar o diseñar robots la potencia del software del controlador es uno de las principales estructuras que definen al robot, ya que establece la utilidad y limitaciones del diseño de este.

Para generar las clasificaciones existentes se tuvieron en cuenta los siguientes aspectos, su generación, su nivel de inteligencia, su nivel de control, y a su nivel de lenguaje de programación.

Estas clasificaciones reflejan la potencia del software en el controlador, en particular, la sofisticada interacción de los sensores y partes mecánicas.

10.1.1.1. Arquitectura.

TIPOS DE ROBOT SEGÚN SU ARQUITECTURA		
Poli articulados: Robots de muy diversas formas, con pocos grados de libertad. Generalmente son robots de uso industrial para uso de trasportar materiales de cuidado.	**Móviles:** Robots con gran capacidad de desplazamiento. La forma de movimiento de estos es teledirigida por el ser humano, con secuencias de movimiento establecidas o bien se guían por la información recibida de su entorno a través de sus sensores. Los robots móviles están provistos de extremidades, orugas y/o ruedas que los capacitan para desplazarse de acuerdo a su programación.	**Androides:** Son robots que intentan reproducir total o parcialmente la forma y el comportamiento cinemático del ser humano.
Zoomórficos: caracterizados principalmente por sus sistemas de locomoción donde su objetivo principal es la imitación de diversos seres vivos incluido el ser humano.	**Médicos:** Los robots médicos son, fundamentalmente, prótesis para disminuidos físicos que se adaptan al cuerpo y están dotados de potentes sistemas de mando. Con ellos se logra suplir las extremidades o incluso órganos de los seres humanos	**Híbridos:** Estos robots corresponden a aquellos de difícil clasificación cuya estructura resulta de una combinación de las expuestas anteriormente.

Cuadro 10.0. Clasificación de los Robots según su arquitectura. (Tomado de [51])

10.1.1.2. Generación.

TIPOS DE ROBOT SEGÚN SU GENERACIÓN	
1ª Generación	El sistema de control usado en esta generación de robots es sistemas a lazo abierto, está basado en la "paradas fijas" mecánicamente. Estos dispositivos por lo general se les conocen como "esclavo" mecánico, ya que el hombre es él quien provee mediante su intervención directa el control de movimiento. Como ejemplo de esta primera etapa están los mecanismos de relojería que mueven las cajas musicales o los juguetes de cuerda.
2ª Generación	El movimiento se controla a través de una secuencia numérica almacenada en disco o cinta magnética. Por lo general, este tipo de robots se utiliza en la industria automotriz y son de gran tamaño. Estos dispositivos actúan automáticamente sin necesidad de la participación de algún operario ejecutando movimientos repetitivos en el tiempo, que obedecen a lógicas combinatorias, secuenciales, programadores paso a paso, neumáticos o Controladores Lógicos Programables. Un aspecto muy importante está constituido por la facilidad reprogramación que convierte a estos Robots en unidades "versátiles" cuyo campo de aplicación no sólo se encuentra en la manipulación de materiales sino en todo los procesos de manufactura, como por ejemplo: en el estampado en frío y

	en caliente asistiendo a las máquinas-herramientas para la carga y descarga de piezas. En la inyección de termoplásticos y metales no ferrosos, en los procesos de soldadura a punto y continúa en tareas de pintado y reemplazando con ventaja algunas operaciones de máquinas convencionales.
3ª Generación	Utilizan las computadoras para su control y tienen cierta percepción de su entorno a través del uso de sensores, atreves de estos ellos pueden llegar a elegir la mejor forma de realizar su objetivo teniendo en cuenta el ambiente o los medios que los circunda. Con esta generación se inicia la era de los robots inteligentes y aparecen los lenguajes de programación para escribir los programas de control.
4ª Generación	Se trata de robots altamente inteligentes con más y mejores extensiones sensoriales, para entender sus acciones y captar el mundo que los rodea. Incorporan conceptos "modélicos" de conducta.
5ª Generación	Actualmente en desarrollo. Esta nueva generación de robots basará su acción principalmente en modelos conductuales establecidos.

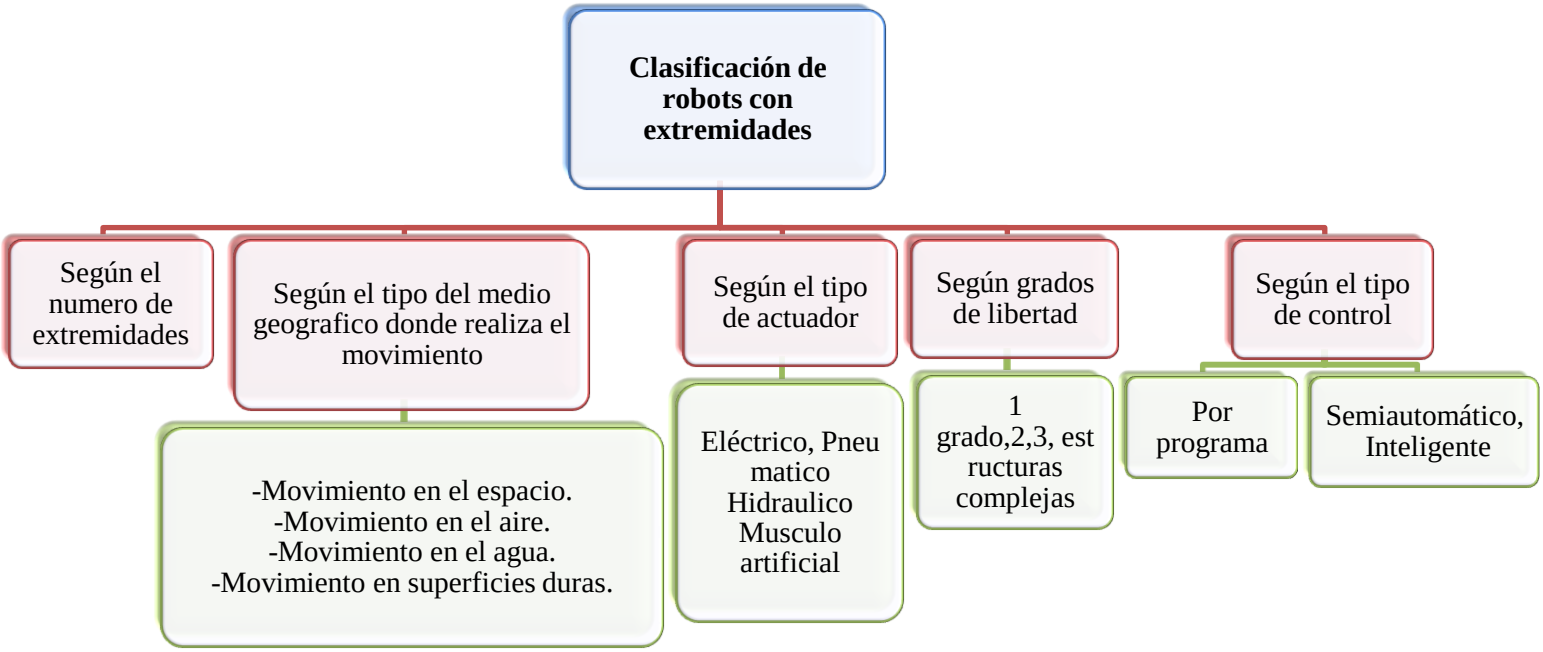
Cuadro 10.1. Clasificación de los Robots según su generación. [46] [52]

10.1.1.3. Estándares.

Estándar Entidades:	*Asociación Francesa de Robótica Industrial (AFRI)*	Asociación de Robots Japonesa (JIRA)
TIPO A:	Manipulador con control manual o telemando.	Dispositivos de manejo manual, controlados por una persona.
TIPO B:	Manipulador automático con ciclos pre ajustados; regulación mediante fines de carrera o topes; control por PLC; hacinamiento neumático, eléctrico o hidráulico.	Robots de secuencia arreglada.
TIPO C:	Robot programable con trayectoria continua o punto a punto. Carece de conocimiento sobre su entorno.	Robots de secuencia variable, donde un operador puede modificar la secuencia fácilmente.
TIPO D:	Robot capaz de adquirir datos de su entorno, re adaptando su tarea en función de éstos.	Robots regeneradores, donde el operador humano conduce el robot a través de la tarea.
TIPO E:		Robots de control numérico, donde el operador alimenta la programación del movimiento, hasta que se enseñe manualmente la tarea.

Cuadro 10.2. Clasificación de los Robots según estándares [46] [52]

10.1.1.4. Clasificación software y hardware.



Mapa Conceptual 10.0 Clasificación de los Robots según software y hardware.

10.2. ESTRUCTURA MECÁNICA.

Todo robot o maquina está compuesto por dos partes importantes: software y hardware. Donde el hardware del robot es su estructura mecánica, esta es la encargada de formar la constitución física del robot, el cual está compuesto por un grupo de piezas o eslabones que están unidos mediante articulaciones que permiten el movimiento entre dos de estos elementos consecutivos. Los componentes principales del robot se pueden clasificar en: estructura mecánica, transmisiones, sistema de accionamiento, sistema sensorial, sistema de control y elementos terminales. (No todos los robots se encuentran en esta caracterización, pero la estructura general de un robot o una maquina es la nombrada anteriormente).

10.2.1. Articulaciones.

Es común encontrar dos tipos de estas: giratoria y prismática. La unión prismática o junta deslizante, da la capacidad a un eslabón determinado de moverse o deslizarse en línea recta sobre otro, mientras que la unión giratorio para el caso de tener un grado de libertad hace el papel de

bisagra entre eslabones, esa tipo de articulación permite que dos o más articulaciones puedan combinarse estrechamente.

10.2.2. Eslabones.

Con objeto de lograr la respuesta más rápida posible para un movimiento dado y un sistema de accionamiento, los eslabones que forman las estructura deben de mantenerse lo más ligeros posibles. Los eslabones deben ser también tan rígidos como sea posible. En la práctica hay que considerar muchos otros factores tales como el coste, las necesidades para alojar los accionadores, árboles de transmisión y cajas de engranaje, el comportamiento vibracional, el comportamiento no elástico tal como el pandeo y la necesidad de alcanzar un espacio de trabajo determinado.

10.2.3. Transmisiones.

Son los elementos encargados de transmitir el movimiento desde los actuadores hasta las articulaciones.

10.2.4. Reductores.

Son los encargados de adaptar el par y la velocidad de la salida del actuadora los valores adecuados para el movimiento de los elementos del robot.

10.3. MOVILIDAD.

La movilidad de un robot esta predispuesta en la estructura física de este, en especial por el tipo de articulaciones (para los robots con extremidades) que posea en la juntura de sus eslabones. El tipo correcto de articulación define como se puede mover un eslabón en relación con el otro.

Hay dos tipos de movilidad lineal y angular, la lineal puede ser horizontal o vertical, y la angular formando ángulos.

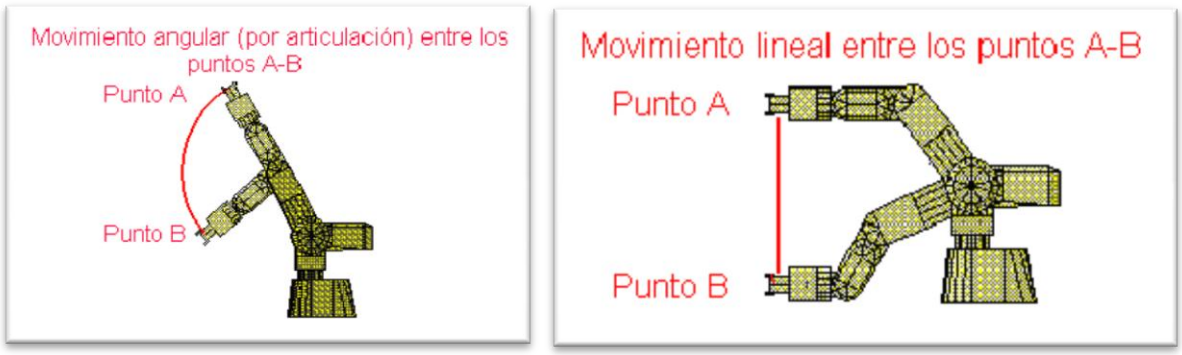


Figura 10.0 tipos de movilidad: movilidad lineal (derecha) movilidad angular (izquierda). (Tomado de [14])

10.3.1. Grados de libertad.

Cada uno de los movimientos independientes que puede realizar cada articulación con respecto a la anterior.

Para posicionar y orientar alguna extremidad o la estructura general del robot se necesita cierta cantidad de movimientos, por lo general el número de grados de libertad de un robot está determinado por el número de articulaciones que este posee.

10.3.2. Accionamiento directo.

Es cuando el eje del actuador se conecta directamente a la carga o articulación, sin la utilización de un reductor intermedio.

Ventajas del accionamiento directo:

- Posicionamiento rápido y preciso (evita rozamiento y juego de las transmisiones).
- Mejor Controlabilidad del sistema.
- Simplificación del sistema mecánico al eliminarse el reductor.

10.4. SCRIPTS.

10.4.1. SCRIPT ARDUINO UNO. (MAESTRO)

```
const int adelante=13; //Se asignan los puertos de la tarjeta, con el alias indicado
const int atras=12;
const int vvl=7;
byte x;//Se declaran las variables mediante las cuales se hará la manipulación de bits para la lógica del programa.
byte x1;
byte x2;
byte x3;
byte x4;
byte x5;
byte x6;
byte x7;
byte x8;
byte ack;//Se declara el byte ACK mediante el cual el maestro hará reconocimiento de si la orden fue leída sobre la trama recibida por parte de los esclavos
byte r0;
byte r1;
byte r2;
byte r3;
byte a;
byte b;
byte mandar;//Corresponde a la trama de envío dada por el maestro a los esclavos
byte orden_recibida;
byte VELOCIDAD;
byte rutina;
byte esclavo;
byte token;
void setup()
```

```
{
  Serial.begin(2400);//Se inicializa la comunicación serial vía usart
  pinMode(vvl, INPUT);//Se declaran las entradas al sistema
  pinMode(adelante, INPUT);
  pinMode(atras, INPUT);
}
void loop (){
  inicio:
  a=digitalRead(adelante);
  if(a==1) //El sistema inicia la ejecución si el usuario presiona el interruptor de avance
  {
     x=digitalRead(vvl);
     if (x ==1 )  //velocidad ALTA //El sistema lee si la opción de velocidad asignada por el usuario mediante el switch
        {
         VELOCIDAD=B00000011;//el sistema asigna el byte de velocidad
        }
    if (x==0)        // velocidad BAJA//El sistema lee si la opción de velocidad asignada por el usuario mediante el switch
       {
        VELOCIDAD=B00000001; //el sistema asigna el byte de velocidad
       }
    if(a==1)
    {
    rutina=B00000011; //el sistema asigna el byte de rutina
    }
      esclavo=B00001110;   //ambos//El usuario indica a quien se dirige el maestro, habilitando la línea de código al remover los caracteres "//"
    //esclavo=B00001000; //esclavo-1
    //esclavo=B00001010; //esclavo-2
```

```
x1=bitRead(rutina,0);
x2=bitRead(rutina,1);//El sistema ensambla la orden a enviar
x3=bitRead(esclavo,0);
x4=bitRead(esclavo,1);
x5=bitRead(esclavo,2);
x6=bitRead(esclavo,3);
x7=bitRead(VELOCIDAD,0);
x8=bitRead(VELOCIDAD,1);
bitWrite(mandar,0,x1);
bitWrite(mandar,1,x2);
bitWrite(mandar,2,x3);
bitWrite(mandar,3,x4);
bitWrite(mandar,4,x5);
bitWrite(mandar,5,x6);
bitWrite(mandar,6,x7);
bitWrite(mandar,7,x8);
Serial.write(mandar);//Se envía la orden del maestro a los esclavos
delay(20);
//Serial.println(mandar,BIN);//Al habilitar estas líneas y conectar el Maestro al PC se puede validar la trama que se está enviando.
ESPERA://En esta sección el sistema lee la trama recibida por parte del esclavo y verifica el ACK de la misma , si este no corresponde a lo enviado, el sistema entra en un ciclo de espera hasta que le llegue la trama correcta
if(true)
{
orden_recibida=Serial.read();
r0=bitRead(orden_recibida,2);
r1=bitRead(orden_recibida,3);
r2=bitRead(orden_recibida,4);
r3=bitRead(orden_recibida,5);
bitWrite(ack,0,r0);
bitWrite(ack,1,r1);
```

```
bitWrite(ack,2,r2);
bitWrite(ack,3,r3);
//Serial.print("MANDAR");
//Serial.print(mandar,BIN);
Serial.print("ACK");
Serial.print(ack,BIN);
 if(esclavo==B00001110)
{
   if(ack==B00001100)
    {
     goto inicio;
    }
    else
    {
      goto ESPERA;
    }
}
 if(esclavo==B00001000)
 {
  if(ack==B00000010)
    {
     goto inicio;
    }
   else
  {
    goto ESPERA;
  }
 }
if(esclavo==B00001010)
 {
  if(ack==B00000110 || ack==B00001100)
```

```
        {
         goto inicio;
        }
       else
      {
        goto ESPERA;
      }
     }
  else
  {
    Serial.flush();
    goto ESPERA;
  }
    }
  }
}
```

10.4.2. SCRIPTS ARDUINOS MINI.

ESCLAVO 1:

```
int orden ; //Se declaran las variables
int   x=0   ;
byte  esclavo;
byte  rutina;
byte  velocidad;
int vel;
int r0;
int r1;
int r2;
int r3;
int r4;
int r5;
int r6;
```

```
int r7;
const byte t1 =10;//Se asignan los puertos con los alias a implementar en el código
const byte t3 =11;
const byte t5 =12;
const byte  z =13;
int prueba;
void setup()
{
  Serial.begin(2400);//Se inicializa la comunicación.
  pinMode (t1,OUTPUT);//Se asignan los puertos de salida para controlar los servos
  pinMode (t3,OUTPUT);
  pinMode (t5,OUTPUT);
  pinMode (z,OUTPUT);
}

void loop()
{ espera:
  if (Serial.available() > 0)
  {orden=Serial.read();// De haber datos de entrada se lee la orden de entrada
  Serial.flush();//Se limpia el buffer de entrada
  //Serial.println(orden);//Al habilitar esta linea y tener el Arduino en comunicación con el PC es posible verificar la trama "orden"
  if(true)
  {
    mensaje://En esta sección se particiona la orden de entrada con el fin de ingresar los bits correspondientes para la lógica del sistema
     r0=bitRead(orden,0);
    r1=bitRead(orden,1);
    r2=bitRead(orden,2);
    r3=bitRead(orden,3);
    r4=bitRead(orden,4);
    r5=bitRead(orden,5);
```

```
    r6=bitRead(orden,6);
    r7=bitRead(orden,7);
    bitWrite(rutina,0,r0);
    bitWrite(rutina,1,r1);
    bitWrite(esclavo,0,r2);
    bitWrite(esclavo,1,r3);
    bitWrite(esclavo,2,r4);
    bitWrite(esclavo,3,r5);
    bitWrite(velocidad,0,r6);
    bitWrite(velocidad,1,r7);
//SECUENCIA DE CAMINADO
    //Serial.println(r0);//Al habilitar este bloque, se puede verificar mediante comunicación con el pc, los bits que conforman la orden
    //Serial.println(r1);
    //Serial.println(r2);
    //Serial.println(r3);
    //Serial.println(r4);
    //Serial.println(r5);
    //Serial.println(r6);
    //Serial.println(r7);
      if(esclavo==B00001000)//En este bloque el  esclavo identifica si la orden es para él, de ser así, modifica los bits que luego el maestro entenderá como ACK
    { bitWrite(orden,2,0);
      bitWrite(orden,3,1);
      bitWrite(orden,4,0);
      bitWrite(orden,5,0);
      Serial.write(orden);//Envía la orden al siguiente esclavo
      delayMicroseconds(2);//Realiza un delay para sincronizar el inicio de la ejecución de la rutina con el siguiente esclavo
      goto ejecución;
    }
   if(esclavo==B00001110)
```

```
 { bitWrite(orden,2,1);
   bitWrite(orden,3,0);
   bitWrite(orden,4,0);
   bitWrite(orden,5,1);
   Serial.write(orden);
   delayMicroseconds(2);
   goto ejecución;
 }
Else //Si la orden recibida no es para él, entonces no la altera y la envía al siguiente esclavo
 {
  Serial.write(orden);
  goto  fin;
 }
   ejecucion://En esta sección el sistema identifica los parámetros enviados para asignar las
variable que manipulara la velocidad de la plataforma y la identificación de la rutina a ejecutar.
   if(velocidad==1)
   {
     vel=20;
   }
   if(velocidad==3)
   {
     vel=40;
   }
   if(rutina==3)
   {

 //-----------------MOVIMIENTO1-------------------------//
    x=0;
    do
     { //Z  Z 1-3-5 abajo
      //PORTB=%00000001
```

```
     digitalWrite (z,1);
     digitalWrite(t1,0);
     digitalWrite(t3,0);
     digitalWrite(t5,0);
     delayMicroseconds(1200);
     digitalWrite (z,0);
     digitalWrite(t1,0);
     digitalWrite(t3,0);
     digitalWrite(t5,0);
     delay(20);
     x=x+1;
    }while (x<vel);
//---------------------------------------------//
//=============================================//
//----------------- MOVIMIENTO 2-----------------------//
    x=0;
    do //'Theta  atras 1-3-5
    {
     digitalWrite (z,0);
     digitalWrite(t1,1);
     digitalWrite(t3,1);
     digitalWrite(t5,1);
     delayMicroseconds(1550);
     digitalWrite (z,0);
     digitalWrite(t1,0);
     digitalWrite(t3,1);
     digitalWrite(t5,1);
     delayMicroseconds(100);
     digitalWrite (z,0);
     digitalWrite(t1,0);
     digitalWrite(t3,1);
```

```
    digitalWrite(t5,0);
    delayMicroseconds(200);
    digitalWrite (z,0);
    digitalWrite(t1,0);
    digitalWrite(t3,0);
    digitalWrite(t5,0);
    delay(20);
    x=x+1;
   }while(x<vel);
//-----------------------------------------------//
//===============================================//
//----------------- MOVIMIENTO 3-----------------------//
    x=0;
    do //'Z 1-3-5 arriba
    {
     digitalWrite (z,1);
     digitalWrite(t1,0);
     digitalWrite(t3,0);
     digitalWrite(t5,0);
     delayMicroseconds(3000);
     digitalWrite (z,0);
     digitalWrite(t1,0);
     digitalWrite(t3,0);
     digitalWrite(t5,0);
     delay(20);
     x=x+1;
    }while(x<vel);
//-----------------------------------------------//
//===============================================//
//----------------- MOVIMIENTO 4-----------------------//
    x=0;
```

```
    do //'Theta adelante 1-3-5
    {
       digitalWrite (z,0);
       digitalWrite(t1,1);
       digitalWrite(t3,1);
       digitalWrite(t5,1);
       delayMicroseconds(1000);
       digitalWrite (z,0);
       digitalWrite(t1,0);
       digitalWrite(t3,1);
       digitalWrite(t5,1);
       delayMicroseconds(50);
       digitalWrite (z,0);
       digitalWrite(t1,0);
       digitalWrite(t3,1);
       digitalWrite(t5,0);
       delayMicroseconds(350);
       digitalWrite (z,0);
       digitalWrite(t1,0);
       digitalWrite(t3,0);
       digitalWrite(t5,0);
       delay(20);
       x=x+1;
    }while(x<vel);
 //-------------------------------------------//
  fin:
  {
   }
 }
 }
}
```

```
goto espera;
}
```

ESCLAVO 2:

```
int orden ; //E código para el esclavo 2 se particiona de igual manera que para el esclavo 1
int   x=0   ;
byte  esclavo;
byte  rutina;
byte  velocidad;
int vel;
int r0;
int r1;
int r2;
int r3;
int r4;
int r5;
int r6;
int r7;
const byte t2 =10;
const byte t4 =11;
const byte t6 =12;
const byte  z =13;
int prueba;
void setup()
{
  Serial.begin(2400);
  pinMode (t2,OUTPUT);
  pinMode (t4,OUTPUT);
  pinMode (t6,OUTPUT);
  pinMode (z,OUTPUT);
```

```
}

void loop()
{ espera:
  if (Serial.available() > 0)
  {orden=Serial.read();
  //Serial.println(orden);
  if(true)
  {
    mensaje:
    //orden=Serial.read();
    r0=bitRead(orden,0);
    r1=bitRead(orden,1);
    r2=bitRead(orden,2);
    r3=bitRead(orden,3);
    r4=bitRead(orden,4);
    r5=bitRead(orden,5);
    r6=bitRead(orden,6);
    r7=bitRead(orden,7);
    bitWrite(rutina,0,r0);
    bitWrite(rutina,1,r1);
    bitWrite(velocidad,0,r6);
    bitWrite(velocidad,1,r7);
    bitWrite(esclavo,0,r2);
    bitWrite(esclavo,1,r3);
    bitWrite(esclavo,2,r4);
    bitWrite(esclavo,3,r5);
    //SECUENCIA DE CAMINADO
    // Serial.println(r0);
    // Serial.println(r1);
    //Serial.println(r2);
```

```
 //Serial.println(r3);
 //Serial.println(r4);
 //Serial.println(r5);
 //Serial.println(r6);
 //Serial.println(r7);
 if(esclavo==B00001010)
 { bitWrite(orden,2,0);
   bitWrite(orden,3,1);
   bitWrite(orden,4,1);
   bitWrite(orden,5,0);
   Serial.write(orden);
   goto ejecución;

 }
 if(esclavo==B00001001)
 { bitWrite(orden,2,0);
   bitWrite(orden,3,1);
   bitWrite(orden,4,1);
   bitWrite(orden,5,0);
   Serial.write(orden);
   goto ejecución;
 }
else
 {
  Serial.write(orden);

  goto  fin;
 }
   ejecución:
   if(velocidad==1)
   {
```

```
   vel=20;
  }
  if(velocidad==3)
  {
   vel=40;
  }
if(rutina==B00000011)

{

//-----------------MOV1-------------------------//
   x=0;
   do
    {
     digitalWrite (z,1);
     digitalWrite(t2,0);
     digitalWrite(t4,0);
     digitalWrite(t6,0);
     delayMicroseconds(3000);
     digitalWrite (z,0);
     digitalWrite(t2,0);
     digitalWrite(t4,0);
     digitalWrite(t6,0);
     delay(20);
     x=x+1;
    }while (x<vel);
//-----------------------------------------------//
//===============================================//
//-----------------MOV2-------------------------//
    x=0;
    do //'Theta  adelante  25° y 30°
```

```
    {
     digitalWrite (z,0);
     digitalWrite(t2,1);
     digitalWrite(t4,1);
     digitalWrite(t6,1);
     delayMicroseconds(1400);
     digitalWrite (z,0);
     digitalWrite(t2,0);
     digitalWrite(t4,1);
     digitalWrite(t6,1);
     delayMicroseconds(200);
     digitalWrite (z,0);
     digitalWrite(t2,0);
     digitalWrite(t4,0);
     digitalWrite(t6,0);
     delay(20);
     x=x+1;
    }while(x<vel);
//-----------------------------------------------//
//===============================================//
//-----------------MOV3------------------------//
     x=0;
     do //'Z 1-3-5 arriba
     {
      digitalWrite (z,1);
      digitalWrite(t2,0);
      digitalWrite(t4,0);
      digitalWrite(t6,0);
      delayMicroseconds(1000);
      digitalWrite (z,0);
      digitalWrite(t2,0);
```

```
      digitalWrite(t4,0);
      digitalWrite(t6,0);
      delay(20);
      x=x+1;
     }while(x<vel);
//-----------------------------------------------//
//===============================================//
//-----------------MOV4------------------------//
     x=0;
     do //'Theta atras -30°
     {
        digitalWrite (z,0);
        digitalWrite(t2,1);
        digitalWrite(t4,1);
        digitalWrite(t6,1);
        delayMicroseconds(1050);
        digitalWrite (z,0);
        digitalWrite(t2,1);
        digitalWrite(t4,1);
        digitalWrite(t6,0);
        delayMicroseconds(150);
        digitalWrite (z,0);
        digitalWrite(t2,1);
        digitalWrite(t4,0);
        digitalWrite(t6,0);
        delayMicroseconds(700);
        digitalWrite (z,0);
        digitalWrite(t2,0);
        digitalWrite(t4,0);
        digitalWrite(t6,0);
        delay(20);
```

```
            x=x+1;
          }while(x<vel);
      //-----------------------------------------//
        fin:
        {
        }

      }
      }
    }
  goto espera;
  }
```

10.4.3. SCRIPT ALGORITMO FCM MATLAB.

```
clear all
close all
clc
load x1; %SE CARGAN LOS DATOS DE ENTRADA OBTENIDOS MEDIANTE LA VARIACION DEL SERVO.
load y1;
%xydata=load('pata2.m');
datosy=y6;
datosx=x6;

hold on
figure(1); plot(datosx,datosy), grid on ;
a=input('Digite el numero de clusters =');% se declaran los parámetros pertinentes para inicializar el algoritmo
h=a+1;
m=2;
e=0.0000001;
i=3;%Numero de filas de la matriz U
j=1000;  %Numero de columnas de la matriz U
Uactual=0;
d(i,j)=0;
Matriz_Uact(i,j)=0;% Se llena la matriz actualizada con ceros.
matriz_U(i,j)=0;
datosx(j)=0;
```

```
datosy(j)=0;
iteraciones=0;
sumaR=0;
diferencia=1;
denominadorx=0;
denominadory=0;
for i=1:h;
for j=1:1000;
        matriz_U(i,j)=rand; %Se escribe en la matriz U valores
aleatorios entre 0 y 1
    end
end

for j=1:1000;
    suma_ruspini=0;
for c=1:h;
    suma_ruspini=suma_ruspini+matriz_U(c,j);%Se determina el factor que
dividirá a los elementos de las columnas
end                                          % para garantizar una
partición tipo ruspini.
    for c=1:h;
          matriz_U(c,j)=matriz_U(c,j)/suma_ruspini;% se sobre escriben
los valores de la matriz U.
    end
end


%3)
while (diferencia > e) %Se plantea un lazo, por medio del cual el
algoritmo se ejecutara hasta que la diferencia entre la matriz U y su
valor anterior sea inferior a 'e' la tolerancia en la terminación.
%%Centros de Clusters
Sumnumeradorx=0;
Sumnumeradory=0;
Denominador=0;

for i=1:h;
    for j=1:1000;
     Sumnumeradorx= matriz_U(i,j)^2*datosx(j)+  Sumnumeradorx;
     Sumnumeradory= matriz_U(i,j)^2*datosy(j)+  Sumnumeradory;
     Denominador=matriz_U(i,j)^2 +Denominador;
    end
    vx(i)= Sumnumeradorx/Denominador;%Se determinan las coordenadas de
los centros de cluster
```

```
70.       vy(i)= Sumnumeradory/Denominador;
71.       m(i)=vx(i)/vy(i);
72.       b(i)=vy(i)-m(i)*vx(i)
73.         Sumnumeradorx=0;
74.         Sumnumeradory=0;
75.         Denominador=0;
76.   end
77.
78.   Ccluster=0;
79.
80.   for i=1:h
81.       Ccluster(i,1) = vx(i);
82.       Ccluster(i,2)= vy(i);
83.   end
84.
85.   Ccluster;
86.   %Distancias de los datos con respecto a los clusters
87.
88.
89.   for j=1:1000;
90.
91.       for i=1:h;
92.       djkx=0;
93.       djky=0;
94.       djkx=(datosx(j)-vx(i));
95.       djky=(datosy(j)-vy(i));
96.       djk(j,i)=[djkx djky]*[djkx djky]';
97.      end
98.
99.   end
100.  djkx=0;
101.  djky=0;
102.  d=djk';
103.  %%Actualización de la matriz de partición
104.  nsuma=0;
105.  sumaruspini=0;
106.  iteracion=0;
107.  suma=0;
108.
109.  for k=1:1000
110.   for i=1:h
111.      for j=1:h
112.      suma=(d(i,k))^2/(d(j,k))^2;
```

```
    end
    Matriz_Uact(i,k)=1/(suma);
 end
end
%Uact ruspini
for j=1:1000;
    sumaR=0;
for c=1:h;
    sumaR=sumaR+Matriz_Uact(c,j);%Se determina el factor que dividirá a los elementos de las columnas
end                                          % para garantizar una partición tipo ruspini.
   for c=1:h;
          Matriz_Uact(c,j)=Matriz_Uact(c,j)/sumaR;% se reescriben los valores.
    end
end


diferencia=max(max(Matriz_Uact - matriz_U))% determina ||difU||

iteraciones=iteraciones+1% nos muestra el numero de iteraciones realizadas
matriz_U=Matriz_Uact;% actualiza la matriz U
end
%t = -45:0.1:-30
%f = -0.057*t-1.721635884
%plot(t,f,'color','g')

 hold on
  plot(vx,vy,'O','color','r')

  figure(2)
  plot(Matriz_Uact'),grid on


```

11. BIBLIOGRAFÍA Y FUENTES DE INFORMACIÓN.

[1] Villalba Herrera, L. Castañeda Hernández, L. Ruiz Escobar, J. Navas Barajas, D. (abril 2002), *Robot Rastreador Megatron* [en línea]. [Investigación organizada por el grupo de investigación "Semillero de Robótica"]. Bucaramanga, Colombia, Universidad Pontificia Bolivariana, Ingeniería Electrónica. disponible en: <http://www.slideshare.net/LuisnxT/megatron-presentation> (23 de marzo 2011).

[2] The MIT Artificial Intelligence Laboratory. MIT University, 9 de febrero de 1998. [en línea], disponible en: <http://www.ai.mit.edu/projects/mobile-robots/> (15 de febrero del 2011)

[3] Rebah Bouaiachi, I. (Diciembre 2009) *Plataforma Genérica Para Desarrollo Con Robots Móviles* [en línea]. [Proyecto fin de carrera]. Madrid, España, Universidad Autónoma De Madrid, Ingeniería Electrónica., disponible en: <http://arantxa.ii.uam.es/~jms/pfcsteleco/lecturas/20091221IsmailRebah.pdf> (2 de abril del 2011)

[4] Boston Dynamics. *BigDog - The Most Advanced Rough-Terrain Robot on Earth* [en línea], Abril 2009. disponible en:<http://www.bostondynamics.com/robot_bigdog.html>. (15 de enero 2011)

[5] R.A.E. *Definición de Hexápodo* disponible en: <http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=hexapodo>

[6] "MSR-H101 Hexapod kit, ni en tus peores pesadillas*"* (figura 1) disponible en: http://es.engadget.com/category/robots/page/27/

[7] "Desinsectación: Plagas comunes urbanas e industriales." (figura 1) Disponible en: http://www.controld.es/esp/desinsectacion.htm

[8] Martínez De Oraá, N. Fernández Gómez, P. (2003), *Robot Hexápodo* [proyecto][en línea], Catalunya, España. Escuela *Universitaria Politécnica de Vilanova* i la Geltrú, Ingeniería Técnica En Electrónica Industrial, disponible en: < http://e-md.upc.edu/diposit/material/26459/26459.pdf> (29 de abril 20011)

[9] **Herrero Reder N. (2002).** *Robot Hexápodo basado en 68HC11* [proyecto fin de carrera] [en línea] Málaga, España. Universidad de Málaga. Ingeniería de Telecomunicaciones, disponible en: <http://webpersonal.uma.es/~iherrero/index.php?option=com_content&task=view&id=23&Itemid=44> ()

[10] Gorrostieta, E. Morales, C. Solano J. Vargas, E. (enero 2011). *Diseñando un Robot Caminante de Seis Patas*. [proyecto] [en línea], Monterrey, México. Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Querétaro disponible en: <http://www.mecatronica.net/emilio/papers/amrob00.PDF> ()

[11] *"Mordedura de arañas del rincón",* (figura 1.8), disponible en: <http://centinela66.wordpress.com/2010/02/16/mordedura-de-aranas-de-rincon/>

[12] Todo Robot. (2010), *El servomotor* [en línea], disponible en: <http://www.todorobot.com.ar/documentos/servomotor.pdf> (10 de abril 2011)

[13] X-Robotics. (2008), *Servos* [en línea], disponible en: <http://www.x-robotics.com/motorizacion.htm> (10 de abril 2011)

[14] LABORATORIO DE ELECTRONICA. (Agosto 2005), *El servomotor*, Escuela Politécnica Superior de Albacete. [en línea], disponible en: <http://www.info-ab.uclm.es/labelec/Solar/electronica/index.htm> ( )

[15] Pérez C. (2007), *Automatización Industrial: Descripción y manejo del servomotor de prácticas* [en línea], disponible en: <http://isa.umh.es/asignaturas/ai/practicas/p05.pdf> (10 de abril 2011)

[16] González Fernández, V. *Control automatizado y Robótica para el área de Tecnología: Estructura de un robot industrial,* (2002). [en línea]. Disponible en: <http://cfievalladolid2.net/tecno/cyr_01/robotica/index.htm>

[17] *"Agitador Orbital:* Robot cartesiano Seluque® modelo Mobil Master 1.200" [en línea], disponible en: <http://seluque.com.br/espanol/robo.htm>

[18] OMROM Automatización Industrial. *"Robots Scara"* [en línea]. Disponible en: <http://industrial.omron.es/es/products/catalogue/motion_and_drives/robots/scara_robots/default.html>

[19] Markowitz R. "*Twitteará* androide desde transbordador de la NASA", Milenio online [en línea]. México. (Agosto 4 2010), pp 1. Disponible en: < http://impreso.milenio.com/node/8810538>

[20] CUCEI, CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIAS. [en línea], Programa Interactivo Tutorial Sobre Robótica. Abril 2003. Guadalajara, México., disponible en: <http://proton.ucting.udg.mx/robotica/r166/r66/r66.htm> (7 de abril del 2011)

[21] Cárdenas Gutiérrez, M. *"Algoritmo para la Solución de la Cinemática Directa de robots Paralelos Planares 3RRR Destinados a Aplicaciones en tiempo real" [en línea],* en Convención de Telecomunicaciones e Informática CONVETEL, Guatemala y El salvador, Universidad de Sonsonate, pp 9 disponible en: <http://ewh.ieee.org/r9/el_salvador/convetel/descargas/cinematica_directa_robots.pdf> (28 de abril del 2011)

[22] M.C. Miguel de J. Ramírez C. *Cinemática del Robot Industrial: Automatización de Sistemas de Manufactura* [diapositiva en línea], 34 diapositivas (byn – col), disponible en: <http://www.angelfire.com/extreme/greynosom/archivos/Cinematica_Robot.pdf> (28 de abril 2011)

[23] Sánchez, O. *"Cinemática inversa de los manipuladores"* [en línea]. Disponible en: <http://omarsanchez.net/Documents/Cinem%C3%A1ticainversadelosmanipuladores.pdf>

[24] Vivas Venegas, C. "Geometría y Cinemática, control de programación de robots" [en línea], disponible en: <http://www.esi2.us.es/~vivas/ayr2iaei/CIN_ROB.pdf>

[25] "*Cinemática Directa*. Concepto teórico" [en línea], disponible en: <http://www.aurova.ua.es/robolab/EJS4/PRR_Suficiencia_Intro_2.html>

[26] Cortés Parejo, J. (marzo 2008). *"La representación Denavit-Hartenberg"*. [en línea], Disponible en:
<http://personal.us.es/jcortes/Material/Material_archivos/Articulos%20PDF/RepresentDH.pdf>

[27] X-Robotics. (2008), *Cinemática inversa* [en línea], disponible en: < http://www.x-robotics.com/cinematica.htm> (29 de abril 2011)

[28] *"Cinemática inversa"*, [en línea] disponible en: <http://proton.ucting.udg.mx/robotica/r166/r83/r83.htm>

[29] Vargas Soto, J. *Diseño un Robot Hexápodo Tipo Hormiga*. En*: 8vo. Congreso Mexicano de Robótica, (*19 *y 20 de Octubre del 2006), P.80-85.*

[30] Introducción a la Lógica Difusa, Tomás Arredondo Vidal (26/6/09) , (10/05/20119

[31] Clustering Agrupamientos – Aprendizaje automático, disponible en: <http://es.scribd.com/doc/13735717>/, (10/05/2011).

[32] MathWorks. “Fuzzy clustering”. [en línea]. disponible en: <http://www.mathworks.com/products/fuzzylogic/description4.html>

[33] López García, S. *Algoritmo De Agrupamiento Genético Borroso Basado En El Algoritmo De Las C-Medias Borroso*, Junio 2001.

[34] Servo Data Base. *Hitec HS-300 - Standard Servo*. [en línea], disponible en : <http://www.servodatabase.com/servo/hitec/hs-300>. (1 de febrero 2011)

[35] Hobbico. *Servo Specifications and Applications*, [en línea], disponible en: <http://www.hobbico.com/radioaccys/hcam1000.html>. (1 de febrero del 2011)

[36] Himodel. *Servo TowerPro SG-5010 tamaño estándar con rodamiento,* [en línea], disponible en:: <www.himodel.es/product.php?productid=16230 >. (1 de febrero del 2011.)

[37] Comella Casado, I. (abril 2008), *Diseño del algoritmo de control de un robot Hexápodo* [Proyecto de grado] [en línea], Lérida España, Departamento de Informática e Ingeniería Industrial de la Universidad de Lleida, disponible en: < http://robotica.udl.cat/tfc/pdf/ivancomella.pdf> (29 de abril 2011)

[38] ATMEL, *“ATmega48PA/88PA/168PA/328 features”* [en linea], disponible en: <http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf>

[39] Arduino. 25 de marzo 2006, [en línea], disponible en: < http://arduino.cc/>. (5 de abril del 2011)

[40] “Proyecto Arduino: ¿Qué es Arduino?”, [en línea], disponible en: <http://www.proyectoarduino.com.ar/?page_id=7>

[41] Todo Electrónica. Enero 1987, [en línea], disponible en: <http://todoelectronica.com/%C3%82%C2%BFque-arduino-p-13506.html>. (5 de abril del 2011)

[42] Arduino. Arduino Uno. 25 de marzo 2006, [en línea], disponible en: <http://arduino.cc/en/Main/ArduinoBoardUno>. (5 de abril del 2011)

[43] Arduino. Arduino Mini. 25 de marzo 2006, [en línea], disponible en: <http://www.arduino.cc/en/Main/ArduinoBoardMini>. (5 de abril del 2011)

[44] *"Robótica"* (2007), [en línea] disponible en: <http://robotica.wordpress.com/about/>

[45] Moreno Torres, A. (31 de julio 2003), *Pucho Bot: Robot Cuadrúpedo*. [En línea]. Madrid, España. Disponible en: < http://www.iearobotics.com/personal/andres/proyectos/pucho/documentacion/capitulo1.pdf >. (8 de abril del 2011)

[46] Dr. Sánchez Pérez, O. (*26 de marzo 2007)*, *Robótica.* [Base de datos en línea]. Huelva, España. Disponible en: <http://www.uhu.es/omar_sanchez/> (7 de abril del 2011).

[47] *"¿Qué es la Robótica?"*, (2011). [En línea], disponible en: <http://www.roboticspot.com/robotica.php>

[48] DE FELICE, J. *"Que es la Robótica"* [en línea], disponible es: <http://robothumano.galeon.com/productos774285.html>, recuperado: (8 de abril 2011)

[49] DEI, Departamento de electrónica e informática. *Inteligencia Artificial* [en línea], disponible en: <http://www.dei.uc.edu.py/tai2002/IA/>. (10 de abril 2011)

[50] Robot ¿una maquina cualquiera? (abril 2005), *Clasificación de los robots.* [En línea] disponible en:

<http://www.educa.madrid.org/web/ies.alpajes.aranjuez/Web_robotica/tiposrobots.htm> (8 de abril del 2011)

[51] *I, Robot - Isaac Asimov,* (2004) [en línea], disponible en: < http://www.cristalab.com/blog/i-robot-isaac-asimov-c637l/> (8 de abril del 2011)

[52] Ramos J. González R. Meléndez S. *ROBOTEC tecnología robótica Clasificación de la robótica* [en línea], disponible en: <http://robotec11.tripod.com/id4.html> (8 de abril del 2011)

[53] Sánchez, Espinoza, A. López Cárdelo, C. *Diseño de un Robot hexápodo hardware y software de control* [trabajo de grado], Catalunya, España.
Escuela *Universitaria Politécnica de Vilanova* i la Geltrú, Robótica industrial, disponible en: < http://e-md.upc.edu/diposit/material/26484/26484.pdf>, (9 de abril 2011)

[54] Saavedra, T. y Forero, M. (2006), *Estrategia de movimiento para un robot hexápodo en terrenos accidentados* [trabajo de grado], Bogotá, Pontificia Universidad Javeriana, Ingeniería Electrónica.

[55] Qingsheng L, Baoling H, Xin M, Kun W, Xingguang D. (2006), *A FWN-Based Distributed Hierarchical System for Hexapod Bio-Robot Control*. En 6th World Congress on Intelligent Control and Automation, 21-23 de junio 2006, Dalian China. pp 8943-8947.

[56] Coronel Lemus, M. *Simulación de sistema difuso para el control de velocidad de un motor* CD. En: 3er. Congreso de Computo AGECOMP,P.25-32.

[57] Álvaro A. Orozco G, *Organización de espigas usando agrupamiento fuzzy c-means, Scientia et Technica* Año XI No 28 Octubre de 2005

[58] "Robotica y Electronica", (20079, [en línea], disponible en: < http://roboticayelectronica.blogspot.com/2007/12/rh-1-robot-hexpodo.html>