UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO, CONSTRUCCIÓN Y CONTROL DE UN ROBOT HEXÁPODO

TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO MECATRÓNICO

PRESENTA:

MIGUEL ANGEL CÁRDENAS VERDUGO

DIRECTOR DE TESIS

DR. VÍCTOR JAVIER GONZÁLEZ VILLELA

CIUDAD UNIVERSITARIA, MÉXICO D. F., 2011

Índice

Índice de figuras

Índice de tablas

Índice de gráficas

Resumen

Un robot es un dispositivo reprogramable y multifuncional que utiliza movimientos variables programados para la realización de una variedad de tareas. Según esta definición del Instituto de Robótica de América (1979) se entiende que un robot es un dispositivo que puede ayudar al ser humano. Entre las principales funciones que se espera de un robot se encuentran: tareas que requieren alta precisión y rapidez, trabajos en ambientes peligrosos o desconocidos, operaciones molestas, repetitivas o que simplemente no son posibles para un ser humano.

Un robot móvil es un tipo de robot que puede desplazarse en el espacio. Dentro de la robótica móvil se encuentran los llamados “robots caminantes”, como su nombre lo indica, caminan como lo haría un animal o incluso un ser humano. Este tipo de robots no son tan comunes debido a la complejidad que su diseño, construcción y control implica.

El objetivo principal de este trabajo es comprobar que se puede tener un movimiento omnidireccional en un robot hexápodo con tres grados de libertad en cada pata. Dadas las necesidades para la experimentación con un robot hexápodo se construyó uno con 18 servomotores, piezas de acrílico cortadas con láser y diversos tornillos para la unión de las partes. Se programó una interfaz gráfica que simula y controla al modelo físico de forma inalámbrica.

Se pudo demostrar que con tres grados de libertad en cada pata se puede controlar al robot de forma omnidireccional. El modelo físico se comportó tal como se esperaba basándose en el simulador.

El estudio que se ha hecho sobre este tipo de robots es mínimo comparado con el que se hace sobre robots móviles con ruedas. Sin embargo, pueden ser los robots con patas la solución a problemas futuros por lo que no hay que dejarlos a un lado.

CAPÍTULO 1

Introducción

1.1 Inicios de la robótica

A lo largo de toda la historia, el hombre se ha sentido fascinado por máquinas y dispositivos capaces de imitar las funciones y los movimientos de los seres vivos. Los griegos tenían una palabra específica para denominar a estas máquinas: *automatos*. De esta palabra deriva la actual autómata: máquina que imita la figura y movimientos de un ser animado. Los mecanismos animados de Herón de Alejandría (85 d.C.) se movían a través de dispositivos hidráulicos, poleas y palancas y tenían fines eminentemente lúdicos.

La cultura árabe (siglos VIII a XV) heredó y difundió los conocimientos griegos, utilizándolos no sólo para realizar mecanismos destinados a la diversión, sino que les dio una aplicación práctica, introduciéndolos en la vida cotidiana de la realeza. Ejemplo de éstos son diversos sistemas dispensadores automáticos de agua para beber o lavarse. También de ese período son otros autómatas, de los que hasta la actualidad no han llegado más que referencias no suficientemente documentadas, como el *Hombre de hierro* de Alberto Magno (1204-1282) o la *Cabeza parlante* de Roger Bacon (1214-1294). Otro ejemplo relevante de aquella época fue *el Gallo de Estrasburgo* (1352). Éste, que es el autómata más antiguo que se conserva en la actualidad, formaba parte del reloj de la torre de la catedral de Estrasburgo y al dar las horas movía las alas y el pico.

Durante los siglos XV y XVI algunos de los más relevantes representantes del renacimiento se interesan también por los artefactos descritos y desarrollados por los griegos. Es conocido el *León mecánico* construido por Leonardo Da Vinci (1452-1519) para el rey Luis XII de Francia, que se abría el pecho con su garra y mostraba el escudo de armas del rey. En España es conocido el *Hombre de palo*, construido por Juanelo Turriano en el siglo XVI para el emperador Carlos V; este autómata con forma de monje, movía la cabeza, ojos, boca y brazos.

Durante los siglos XVII y XVIII se crearon artefactos mecánicos que tenían alguna de las características de los robots actuales. Estos dispositivos fueron creados en su mayoría por artesanos del gremio de la relojería. Su misión principal era la de entretener a las gentes de la corte y servir de atracción en las ferias. Estos autómatas representaban figuras humanas, animales o pueblos enteros. Son destacables entre otros el pato de Vaucanson y los muñecos de la familia Droz y de Mailladert.

Jacques Vaucanson (1709-1782), autor del primer telar mecánico, construyó varios muñecos animados, entre los que destaca un flautista capaz de tocar varias melodías y un pato (1738) capaz de graznar, beber, comer, digerir y evacuar la comida. El relojero suizo Pierre Jaquet Droz (1721-1790) y sus hijos Henri-Louis y Jaquet construyeron diversos muñecos capaces de escribir (1770), dibujar (1772) y tocas diversas melodías en un órgano (1773). Estos aún se conservan en el museo de Arte e Historia de Neuchâstel, Suiza. Contemporáneo de los relojeros franceses y suizos fue Henry Maillardet, quien construyó, entre otros, una muñeca capaz de dibujar y que aún se conserva en Filadelfia.

A finales del siglo XVIII y principios del XIX se desarrollaron algunas máquinas utilizadas fundamentalmente en la industria textil, entre las que destacan la hiladora giratoria de Hargreaves (1770), la hiladora mecánica de Crompton (1779), el telar mecánico de Cartwright (1785) y el telar de Jacquard (1801). Este último utilizaba una cinta perforada como un programa para las acciones de la máquina. Es a partir de este momento cuando se empiezan a utilizar dispositivos automáticos en la producción, dando paso a la automatización industrial.

La palabra robot fue usada por primera vez en el año de 1921, cuando el escritor checo Karel Capek (1890-1938) estrena en el teatro nacional de Praga su obra *Rossum's Universal Robot* (R. U. R.). Su origen es la palabra eslava *robota*, que se refiere al trabajo realizado de manera forzada. Los robots de R. U. R. eran máquinas androides fabricadas a partir de la "fórmula" obtenida por un brillante científico llamado Rossum. Estos robots servían a sus jefes humanos desarrollando todos los trabajos físicos, hasta que finalmente se rebelan contra sus dueños, destruyendo toda la vida humana, a excepción de uno de sus creadores, con la frustrada esperanza de que les enseñara a reproducirse.

El robot como máquina lleva un desarrollo independiente del término robot. Luego de los primeros autómatas, casi todos de aspecto humano, los progenitores más directos de los robots fueron los telemanipuladores. En 1948 R. C. Goertz del *Argonne National Laboratory* desarrolló, con el objetivo de manipular elementos radioactivos sin riesgo para el operador, el primer telemanipulador. Éste consistía en un dispositivo mecánico maestro-esclavo. El manipulador maestro, situado en la zona segura, era movido directamente por el operador, mientras que el esclavo, situado en contacto con los elementos radioactivos y unido mecánicamente al maestro, reproducía los movimientos de éste. El operador además de poder observar a través de un grueso cristal el resultado de sus acciones, sentía a través del dispositivo maestro, las fuerzas que el esclavo ejercía sobre el entorno.

Años más tarde, en 1954, Goertz hizo uso de la tecnología electrónica y del servocontrol sustituyendo la transmisión mecánica por otra eléctrica y desarrollando así el primer telemanipulador con servocontrol. La sustitución del operador por un programa de computadora que controlara los movimientos del manipulador dio paso al concepto de robot.

El desarrollo de robots móviles responde a la necesidad de extender el campo de aplicación de la robótica, restringido inicialmente al alcance de una estructura mecánica anclada en uno de sus extremos. Se trata también de incrementar la autonomía limitando todo lo posible la intervención humana.

Durante los años sesenta comenzaron a utilizarse vehículos autónomos en la industria, que eran guiados por cables bajo el suelo o mediante sensores ópticos para seguir líneas trazadas en la planta. Estas aplicaciones, hoy en día son comunes en muchos procesos de fabricación, se caracterizan por un entorno fuertemente estructurado para facilitar la automatización.

En los años setenta se vuelve a trabajar en el desarrollo de robots móviles dotados de una mayor autonomía. La mayor parte de las experiencias se desarrollan empleando plataformas que

soportan sistemas de visión. Sin embargo, el desarrollo tecnológico todavía no era el suficiente para lograr la navegación autónoma de forma eficiente. En los años ochenta el incremento espectacular de la capacidad computacional y el desarrollo de nuevos sensores, mecanismos y sistemas de control, permitió aumentar la autonomía.

En la Tabla 1 se muestra un diagrama con los diferentes tipos de robots.

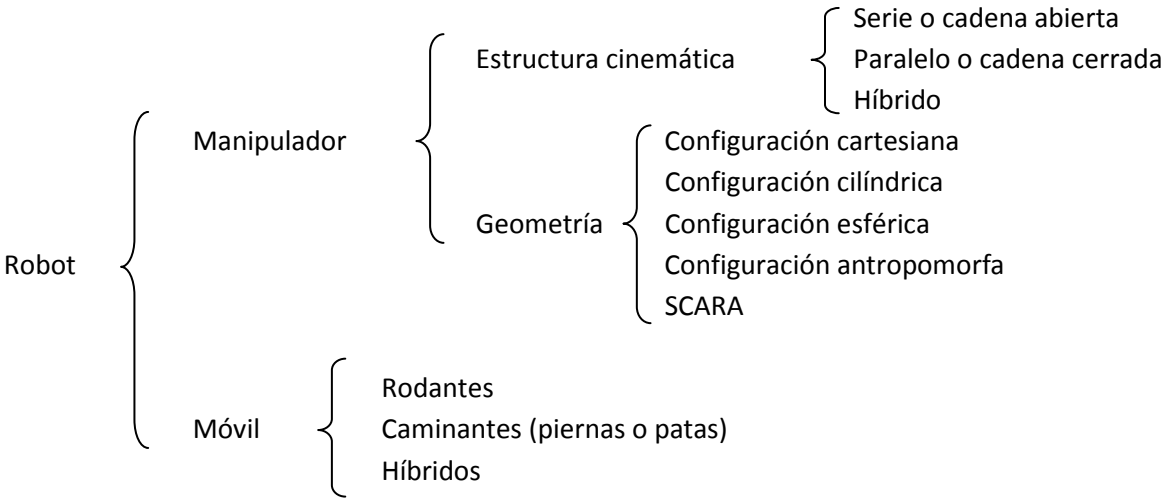


Tabla 1: Diferentes tipos de robots

1.2 Locomoción de un robot móvil

Un robot móvil necesita de un mecanismo de locomoción adecuado que le permita desplazarse en su entorno. Existe una gran variedad de formas de moverse y por lo tanto la decisión de la forma en que el robot se moverá es un aspecto importante del diseño del mismo. En el laboratorio, existen robots en los que se está investigando que pueden caminar, correr, deslizarse, patinar, nadar, volar y rodar. La mayoría de estos mecanismos de locomoción han sido inspirados por sus contrapartes biológicas [7].

Las tres formas más comunes para moverse de los robots móviles son mediante patas, ruedas y orugas [8].

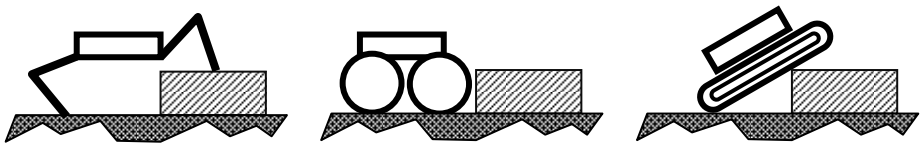


Figura 1: Desplazamiento de un robot móvil [11]

Los vehículos con ruedas son por mucho los más populares por varias razones, entre las que destacan: son mecánicamente más simples y fáciles de construir, la carga que pueden mover tomando en cuenta el peso-mecanismo es favorable. Los robots con patas y orugas requieren

generalmente una estructura más compleja y pesada que los que utilizan ruedas con las mismas capacidades de carga.

La principal desventaja que tienen los robots con ruedas es su pobre desempeño en terrenos irregulares. Como una regla, los vehículos con ruedas tienen problemas cuando la altura del obstáculo es mayor que el radio de la rueda [8]. Una solución muchas veces usada pero en ocasiones impráctica es simplemente utilizar ruedas más grandes que cualquier obstáculo. La eficiencia del uso de ruedas depende en gran medida de las características del terreno, particularmente lo plano y duro, mientras que la eficiencia del uso de piernas depende del peso de las patas y del cuerpo del robot. Es entendible que la naturaleza favorezca la locomoción mediante patas ya que deben de operar en terrenos no estructurados. Por ejemplo, en el caso de los insectos en el bosque la variación vertical de la altura del terreno es a veces mayor en magnitud que la altura misma del insecto. De la misma manera es entendible que casi todos los robots móviles industriales, al desplazarse por terrenos estructurados, utilicen alguna forma de locomoción con ruedas [7].

Para los robots que deben de operar en un ambiente natural, las orugas son en ocasiones utilizadas ya que permiten sortear grandes obstáculos. La principal desventaja que tiene esta forma de locomoción es su ineficiencia. La fricción que se genera en las orugas al girar disipa parte de la energía.

Los robots caminantes pueden potencialmente sobreponerse a mayores problemas del terreno que sus contrapartes con ruedas u orugas. Partiendo de que cada pierna debe de tener al menos 2 actuadores, el costo de construcción del robot es relativamente más elevado. Las patas, que pueden incluir varios grados de libertad, deben de ser capaces de sostener por completo el peso del robot, y en muchos casos deben de ser capaces de elevar y bajar al robot por lo que el consumo de energía es relativamente mayor a otros tipos de locomoción.

Una última ventaja de los robots con patas es el potencial que tienen para manipular objetos de su ambiente con gran habilidad. Un excelente ejemplo de esto es el escarabajo del estiércol, que es capaz de rodas una bola de estiércol mientras se desplaza utilizando sus patas delanteras.

Ambientes hostiles como el de la superficie de Marte impulsan mecanismos de locomoción cada vez más inusuales. Recientemente, para ambientes naturales se han hecho investigaciones en robots híbridos [7].

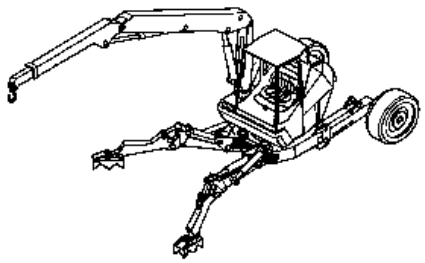


Figura 2: Robot híbrido RoboTrac [7]

CAPÍTULO 2

Robots caminantes

2.1 Inspiración biológica

Los insectos y las arañas cuando nacen son inmediatamente capaces de caminar. Para ellos, el problema del balance al caminar es relativamente simple. Los mamíferos de 4 patas tardan varios minutos luego de nacer para permanecer de pie y algunos más para empezar a caminar. Los humanos, con 2 piernas, no tienen posiciones estáticamente estables por lo que luego de nacer requieren meses para poder estar de pie y caminar, y aún más para aprender a saltar, correr, y permanecer en 1 pierna.

La locomoción mediante patas se caracteriza por una serie de puntos de contacto entre las extremidades y el terreno. Las ventajas de este tipo de locomoción incluyen la adaptabilidad y la maniobrabilidad en todo tipo de terreno. Debido a que sólo una serie de puntos de contacto es requerida, la calidad del terreno carece de importancia siempre y cuando se mantenga una adecuada adherencia.

En los robots caminantes, un mínimo de 2 grados de libertad por pata es generalmente requerido para moverla y poder desplazarse. Es más común la adición de un tercer grado de libertad para maniobras más complejas [7]. La principal desventaja de agregar uniones y actuadores es que se necesita más energía para accionarlos, un control más complejo y se incrementa el peso del robot.

Los insectos, que son los seres con la locomoción más exitosa en la Tierra [7], son capaces de cruzar cualquier tipo de terreno con sus 6 patas, incluso de cabeza. Actualmente, la brecha entre las capacidades de un insecto y su contraparte artificial sigue siendo bastante amplia. Los insectos, a diferencia de la mayoría de los robots, combinan los grados de libertad con que pueden mover sus patas con estructuras pasivas como pelos microscópicos y superficies texturizadas con lo que incrementan la fuerza de adhesión en cada pata significativamente [7].

Las configuraciones de 6 patas han sido muy populares en la robótica móvil debido a su estabilidad estática durante la caminata, con ello el control se simplifica [7].

Los sistemas biológicos han demostrado exitosamente sus capacidades para desplazarse por todo tipo de terreno, por lo tanto podría ser deseable copiar sus mecanismos de locomoción. Sin embargo, replicarlo es extremadamente difícil por diversas razones. Para empezar, la complejidad mecánica es fácilmente almacenada por los sistemas biológicos a través de la réplica estructural. Adicionalmente, la célula permite en gran medida la miniaturización. Con un peso y un tamaño tan pequeño, los insectos pueden almacenar un nivel de robustez que nosotros no somos capaces de igualar con las técnicas actuales de fabricación humanas. Finalmente, los sistemas biológicos de almacenamiento de energía así como los sistemas de activación muscular e hidráulica poseen características en cuanto a torque, tiempo de respuesta y eficiencia que exceden por mucho a sus similares hechos por el hombre [7]. Queda claro que los robots caminantes deben de progresar mucho antes de que sean capaces de competir con sus equivalentes biológicos.

2.2 Reseña histórica

El sueño de crear vida artificial data de los antiguos griegos, árabes, Leonardo da Vinci, Pierre Jaquet-Droz y Nikola Tesla [10]. En nuestros tiempos estamos siendo testigos del surgimiento de máquinas inteligentes que pueden interactuar con el mundo casi tan bien como nosotros.

Las primeras máquinas caminantes en la historia fueron juguetes mecánicos. Dichos juguetes usualmente trabajaban gracias a resortes y engranes a donde las piernas eran sujetas. Las piernas seguían una secuencia con lo que al final de cada ciclo el juguete se movía hacia adelante. Estos juguetes pudieron considerarse la prueba de que máquinas caminantes más grandes podrían ser construidas [10].

La invención del motor de combustión interna fue el evento principal que hizo posible las máquinas caminantes. El primer intento de construir una máquina caminante fue hecho en Gran Bretaña en 1940 por A. C. Hutchison y F. S. Smith. Ambos construyeron una máquina con cuatro piernas controladas independientemente que caminaba utilizando el andar de un cuadrúpedo. El mismo concepto fue usado por General Electric para su llamado "*walking truck*" en los 1960's y fue diseñado por R. S. Mosher.

La investigación moderna en vehículos con piernas comenzó con la *University of Michigan y U. S. Army Tank-Automotive Center* (ATAC) en Warren, Michigan. La milicia de los Estados Unidos ha mantenido un constante interés en los vehículos de transporte para todo terreno desde la segunda guerra mundial. *Defense Advanced Research Projects Agency* (DARPA) ha invertido gran parte de su investigación en las máquinas caminantes.

La NASA y la armada estadounidense estuvieron interesadas en los problemas de movilidad en la Luna y otros planetas. Los posibles usos de éste tipo de máquinas era la exploración planetaria, una silla caminante para discapacitados y transporte militar.

Prototipos de máquinas caminantes han sido desarrolladas principalmente para la investigación en Japón y Estados Unidos (algunos grupos se encuentran en Rusia). En Europa la investigación en este campo ha sido relativamente baja hasta los últimos años.

A continuación se muestran algunos ejemplos representativos de robots caminantes desarrollados a lo largo de la historia.

1965- *The phony Pony*. R. McGhee. A veces conocido como "*Californian Horse*"

1973- *The Waseda University*. Uno de los robots más populares en ese tiempo fue Wabot debido a su semejanza a los humanos.

1974- un hexápodo construido por Petternella de la Universidad de Roma

1977- hexápodo de *Ohio State University* (OSU). Éste robot de 6 piernas pesaba cerca de 100 *kg*

1977- cuadrúpedo de Kyushu Institute of Technology.

1979- robot octópodo ReCUS (*Remotely Controlled Underwater Surveyor*) construido por el Centro técnico de investigación Komatsu en Japón. El robot medía 8 *m* de largo, 5 *m* de ancho, 6 *m* de altura y pesaba 29 toneladas.

1983- hexápodo de *Carnegie Mellon University* (CMU). Fue el primer robot caminante controlado por una computadora que transportaba a un hombre.

1983- robot hexápodo Odex I

1985- TITAN IV. *Tokyo Institute of Technology*, Aruku Norimono (vehículo caminante)

1989- Aquarobot, de *Robotics Laboratory*, *Port Harbor Research Institute* en Japón

1989- Genghis, robot hexápodo desarrollado por Grinell More, Rodney Brooks, y Colin Angle en *Massachusetts Institute of Technology*.

1994- robot octópodo construido en el Instituto de Robótica de la *Carnegie Mellon University* que ha sido empleado para la exploración de un volcán en Alaska.

1996- En la universidad tecnológica de Helsinki se ha desarrollado el MECANT-I, el cuál es un robot que lleva todo su sistema de control y energía abordo. Fue diseñado como un vehículo de prueba en exteriores para aplicaciones en el ambiente natural. MECANT-I es un robot caminante de 6 piernas completamente independiente que utiliza accionamientos hidráulicos. El peso total es de 1100 *kg*. Todas sus piernas son idénticas y con 3 grados de libertad en cada una. El cuerpo principal está construido de tubos de aluminio formando una estructura ligera y rígida. El vehículo es controlado remotamente por un operador utilizando 2 *joysticks*. El movimiento puede ser controlado omnidireccionalmente. El sistema de energía de MECANT-I es completamente auto suficiente, la potencia es generada por un motor de un aeroplano ultraligero de 38 *kW* de 2 cilindros y de enfriamiento por aire.

2005- robot hexápodo PTINTO diseñado para explorar el río Tinto en Huelva España. Cada pata es de tres grados de libertad y utiliza actuadores lineales [12].

2008- microrobot RoACH de 2.4 *g* de peso incluido el control electrónico y la batería. Puede desplazarse a 3 *cm/s*. Utiliza únicamente 2 actuadores que son cables que guardan memoria (flexinol, Dynalloy Inc.) [14].

2009- Dash es un robot hexápodo miniatura de 10 *cm*. Desarrollado en la universidad de Berkeley, California que utiliza un mecanismo y sólo un motor de corriente directa para recrear una marcha trípode y que puede desplazarse a una velocidad máxima de 1.5 *m/s* [13].

Durante los últimos 50 años los robots caminantes han demostrado ser de gran interés, sin embargo la investigación y desarrollo aún no se compara en cantidad con la que se realiza sobre

otros tipos de robots. Los trabajos más recientes están más enfocados a robots miniatura, al principio se pensaba más en que sirvieran de transporte por lo que su tamaño era mucho mayor.

2.3 Ventajas de los robots caminantes

Los robots caminantes tienen diversas ventajas sobre los rodantes entre las que se encuentran:

- Mayor movilidad
    - Pueden cambiar su dirección independientemente de la orientación en el cuerpo

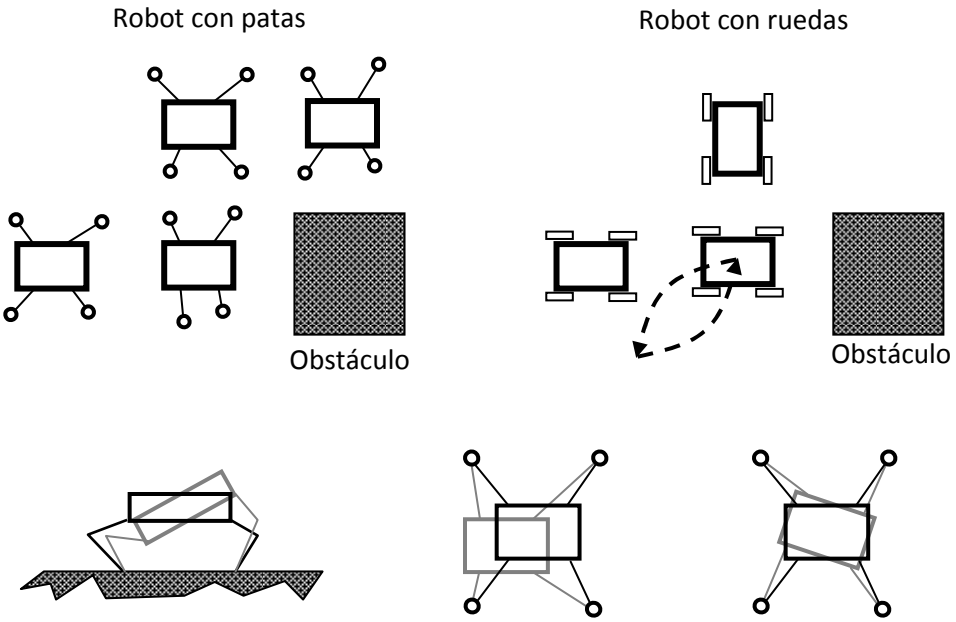


Figura 3: Movilidad de un robot caminante [11]

- Capacidad para sobrepasar obstáculos
    - Pueden superar algunos obstáculos simplemente pasando sobre ellos

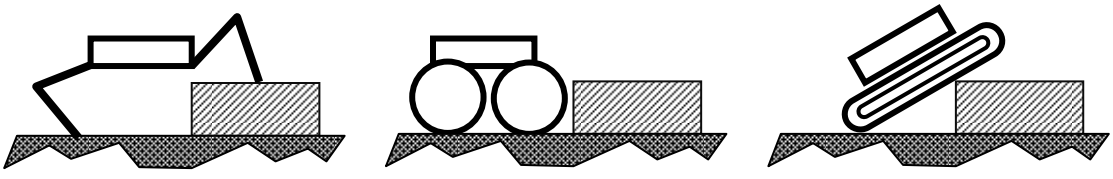


Figura 4: Obstáculos para un robot móvil [11]

- Suspensión activa
    - Cambiando la longitud de sus piernas es posible mantener nivelado el cuerpo sin importar las irregularidades del terreno

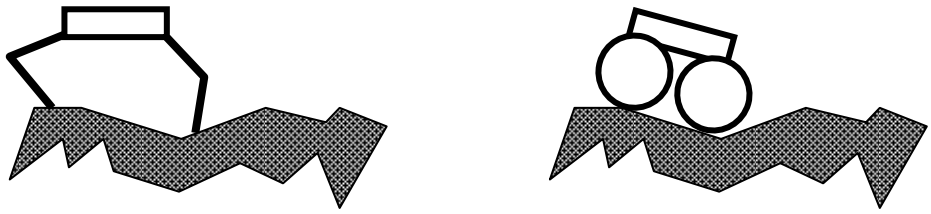


Figura 5: Suspensión activa de un robot caminante [11]

- Desplazamiento en terrenos irregulares
    - Pueden desplazarse en terreno arenoso, lodoso, rígido y suave además de que no necesitan terreno continuo para desplazarse

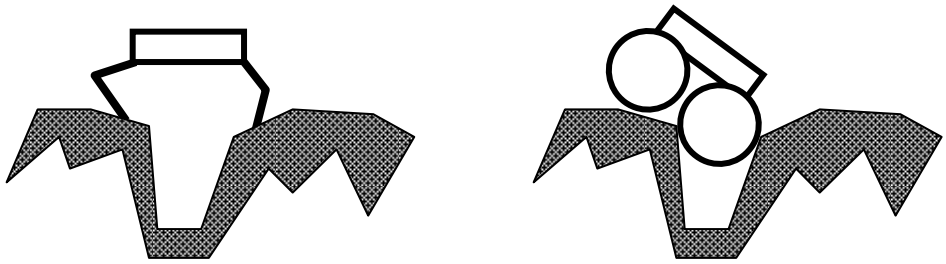


Figura 6: Desplazamiento en terrenos irregulares de un robot caminante y uno con ruedas [11]

- Menores deslizamientos
    - La pata se coloca y se quita del suelo de manera vertical por lo que se reducen los deslizamientos.

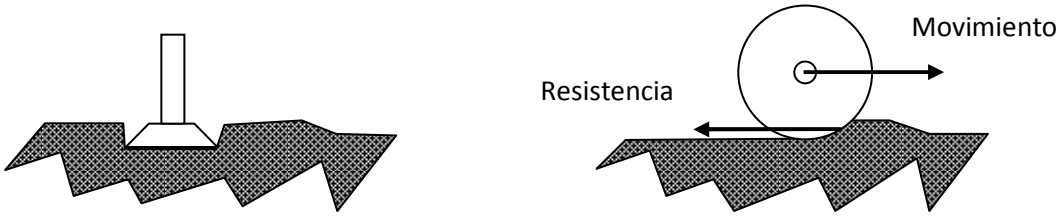


Figura 7: Deslizamientos en un robot móvil [11]

- Menor daño al terreno
    - Tienen menos contacto con la superficie por lo que producen menos daños.

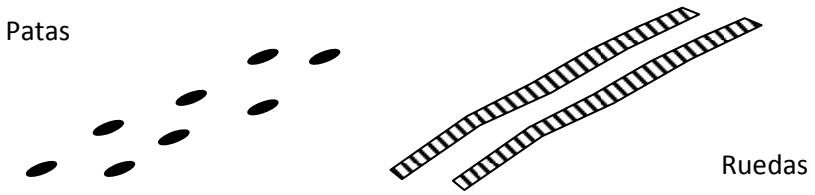


Figura 8: Daño ambiental de un robot móvil [11]

- Velocidad promedio (en terrenos irregulares)
    - Los sistemas con piernas son capaces de adaptarse muy bien a terrenos irregulares y mantener velocidades promedio similares sobre diversos tipos de terreno.

Figura 9: Velocidad promedio en terrenos irregulares de los robots móviles [11]

En ambientes estructurados o construidos por el hombre un robot con ruedas es la mejor opción ya que le permite desplazarse de manera rápida y fluida, sin embargo, para ambientes no estructurados donde el terreno es irregular la mejor opción es un robot caminante.

2.4 Consideraciones

La complejidad del robot dependerá tanto de la movilidad que se quiera en cada una de las extremidades como del número de éstas. Entre menos piernas tenga el robot menor será su estabilidad. Entre menos grados del libertad se tengan menor será su movilidad.

Otro aspecto importante que debe tenerse en cuenta es la elección de los actuadores, los cuales además de proporcionar la potencia para el movimiento de las piernas deben de soportar el peso del robot.

Por último pero no menos importante es el control de los movimientos. Hacer que un robot caminante se desplace consta de una serie de pasos sucesivos que se repite una y otra vez. El número de secuencias posibles depende del número de patas que tenga el robot.

Número de patas	Número de secuencias	Número de secuencias que potencialmente presentan estabilidad estática
1	1	0
2	2	0
3	6	0
4	26	6
5	150	114
6	1 082	1 030
7	9 366	9 295
8	94 586	94 493
9	1 091 670	1 091 552
10	14 174 522	14 174 376

Tabla 2: Resultados de las secuencias de Bessonov y Umnov [15]

Para que una secuencia de movimientos presente potencialmente estabilidad estática es necesario que cuando menos 3 patas estén en contacto con el suelo en todo momento, es por eso que cuando se tienen 1, 2 ó 3 patas no es posible una secuencia de movimientos con estabilidad estática.

Para lograr un movimiento omnidireccional del robot es necesario cuando menos 3 grados de libertad por pata lo que hace más compleja tanto la fabricación del robot como el control del mismo ya que con ello se necesitan 3 variables de control en cada pierna.

2.5 Objetivo

Lograr un movimiento omnidireccional en un robot hexápodo con tres grados de libertad en cada pata.

2.6 Alcance

Este trabajo tiene como meta el control cinemático omnidireccional del robot.

Todo el control requerido para lograr el movimiento omnidireccional se encuentra alojado en un programa de computadora diseñado especialmente para este trabajo que simula y muestra de manera tridimensional el modelo físico a escala. Toda la información necesaria para controlar el robot es enviada de forma inalámbrica durante la ejecución del programa, por lo que el modelo físico realiza todos los movimientos que efectúa el modelo virtual de forma aparentemente instantánea.

El simulador no tiene ninguna clase de realimentación por parte del robot. La única realimentación que existe en el modelo físico es el sistema de control que tiene cada servomotor.

2.7 Contribuciones

Las principales contribuciones de este trabajo son:

- Programa controlador de 18 servomotores con sólo un microcontrolador. Este programa puede ser fácilmente modificado para que dependiendo la marca y modelo del servomotor utilizado pueda manejar un mayor número de estos.
- Se muestran algunas herramientas y programas que pueden ser utilizados tanto para modelar como para simular un objeto.
- Una primera iteración en el ámbito de los robots hexápodos con un programa controlador que puede ser fácilmente modificado para futuras experimentaciones.

CAPÍTULO 3

Herramientas matemáticas

En este capítulo se verán algunas herramientas matemáticas: el operador de traslación y el operador de rotación. Ambas herramientas son muy utilizadas en el ámbito de la robótica para calcular la cinemática directa y en la programación gráfica para manipular puntos en el espacio. Ambos operadores se utilizaron para simular y controlar el robot.

3.1 Operador de traslación

Una traslación mueve un punto en el espacio una distancia finita en dirección de un vector dado. Con esta interpretación de lo que es una traslación lo único que se necesita para realizarla es un sistema de coordenadas en base al cual se llevará a cabo la transformación y un vector que indique tanto la dirección como la magnitud de la misma.

En la Figura 10 se puede ver gráficamente como el vector $\boldsymbol{u_1}$es trasladado por el vector $\boldsymbol{v}$

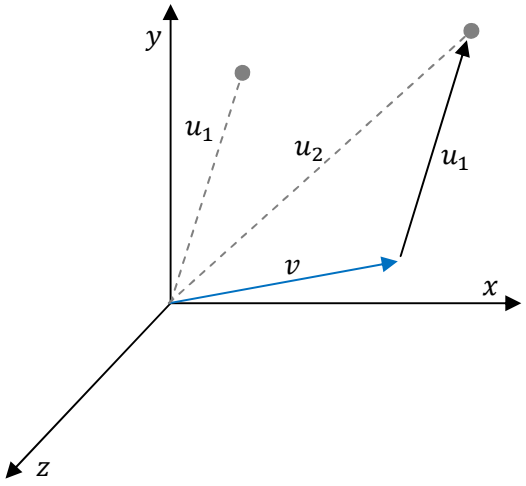


Figura 10: Traslación del vector $\boldsymbol{u_1}$

El resultado de la operación es un nuevo vector $\boldsymbol{u_2}$, que se calcula como

$$\boldsymbol{u_2 = u_1 + v}$$

Esta misma operación puede realizarse con un operador matricial

$$\boldsymbol{u_2 = D(v)u_1}$$

Donde $\boldsymbol{v}$ es el vector que indica la magnitud y dirección de la traslación, y $\boldsymbol{D}$ el operador de traslación

$$\boldsymbol{D}(\boldsymbol{v}) = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \boldsymbol{v_x} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \boldsymbol{v_y} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \boldsymbol{v_z} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Donde $\boldsymbol{v_x}$, $\boldsymbol{v_y}$ y $\boldsymbol{v_z}$ son los componentes del vector $\boldsymbol{v}$ en el sistema coordenado.

3.2 Operador de rotación

La rotación de un punto en el espacio puede verse como un movimiento angular sobre un eje. Lo que se necesita para poder llevar a cabo dicha transformación es la dirección del eje de rotación y la magnitud angular de la propia rotación.

En la Figura 11 se puede ver gráficamente como el vector $\boldsymbol{u_1}$es rotado alrededor del vector $\boldsymbol{v}$, que en este caso en particular tiene la misma dirección del vector $\boldsymbol{z}$ del sistema de coordenadas.

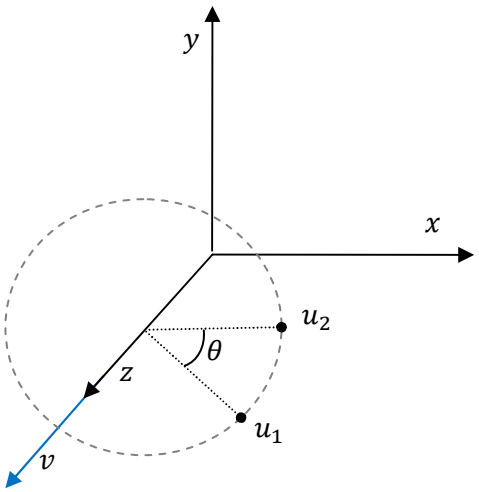


Figura 11: Rotación del vector u_1

El resultado de la rotación es un nuevo vector $\boldsymbol{u_2}$ que se calcula como

$$\boldsymbol{u_2} = \boldsymbol{R_v}(\boldsymbol{\theta})\, \boldsymbol{u_1}$$

Donde $\boldsymbol{R_v}$ es el operador de rotación aplicado al vector $\boldsymbol{u_1}$, y $\boldsymbol{\theta}$ es la magnitud de la rotación.

Las rotaciones más utilizadas son las realizadas sobre los ejes coordenados.

$$\boldsymbol{R_x(\theta)} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{cos(\theta)} & \boldsymbol{-sin(\theta)} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{sin(\theta)} & \boldsymbol{cos(\theta)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$$\boldsymbol{R_y(\theta)} = \begin{bmatrix} \boldsymbol{cos(\theta)} & \mathbf{0} & \boldsymbol{sin(\theta)} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{-sin(\theta)} & \mathbf{0} & \boldsymbol{cos(\theta)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$$\boldsymbol{R_z(\theta)} = \begin{bmatrix} \boldsymbol{cos(\theta)} & \boldsymbol{-sin(\theta)} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{sin(\theta)} & \boldsymbol{cos(\theta)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

CAPÍTULO 4

Diseño

En éste capítulo se muestra el controlador de servomotores, características de la comunicación por medio de Zigbee, los programas y herramientas utilizadas para implementar un simulador y controlador tridimensional.

4.1 Controlador de servomotores

Uno de los primeros pasos que se dieron al iniciar el proyecto fue el diseño de un controlador que fuera capaz de manejar hasta 18 señales diferentes.

El servomotor elegido es de la marca Hitec (Figura 12) modelo HS-485HB cuyo rango de operación va de 0° ($0.6\ ms$) a 180° ($2.4\ ms$). La relación entre orientación y señal de control es lineal.

Figura 12: Servomotor utilizado Hitec HS-485HB

	A 4.8 *volts*	**A 6.0 *volts***
Velocidad operativa	0.22 *s* / 60°	0.18 *s* / 60°
Par de salida	4.8 *kg*cm*	6.0 *kg*cm*
Peso	45 *g*	
Tamaño	39.8 x 19.8 x 38.0 *mm*	

Tabla 3: Especificaciones del servomotor HS-485HB

El microcontrolador utilizado para manejar los 18 servomotores fue un PIC18F4550 trabajando con una frecuencia de oscilación de $48\ MHz$. Este microcontrolador trabaja en modo esclavo con un microcontrolador maestro PIC18F452 mediante protocolo i^2C. A su vez el microcontrolador maestro tiene comunicación con la computadora por medio de Zigbee (protocolo RS232).

Se decidió utilizar 2 microcontroladores para que el microcontrolador maestro al no tener que mantener las posiciones de los 18 servomotores sea capaz en futuras aplicaciones de utilizar los recursos de los que dispone para otras acciones como pueden ser la lectura de sensores y con ello informar a la computadora o directamente para tomar decisiones sobre el comportamiento del robot.

Idealmente un servomotor mantiene su posición mediante una señal de control de $50\ Hz$, por motivos de diseño se utilizó una señal de control de $44.4\ Hz$

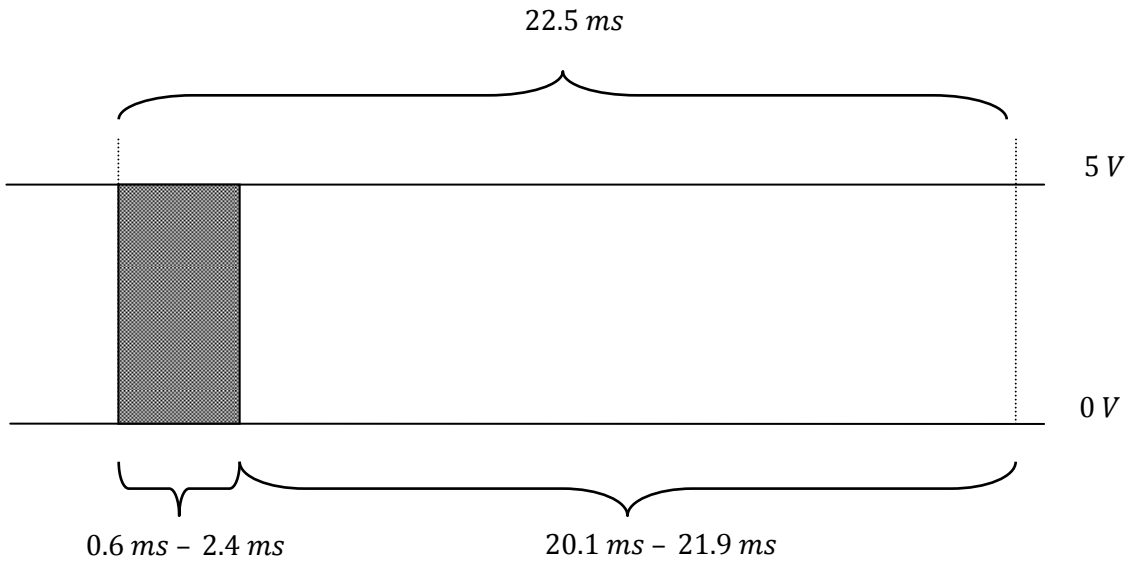


Figura 13: Señal de control utilizada para controlar cada servomotor

Una vez que sabemos cómo controlar la posición de 1 servomotor el siguiente paso es lograr controlar 18. Para esto se utilizaron las interrupciones por desbordamiento del TIMER0 y el TIMER3 con que cuenta el microcontrolador.

El TIMER0 y el TIMER3 pueden ser utilizados como contadores de 16 bits. Este contador se incrementa con cada ciclo máquina ejecutado por el microcontrolador.

Con un oscilador de $48\ MHz$ cada ciclo máquina ($c.m.$) tarda $83.33\ ns$.

Por lo tanto, 2^{16} ó $65536\ c.m.$ es lo que tarda cada TIMER para que ocurra el desbordamiento o lo que es lo mismo, ocurre cada $5.46\ ms$.

Cada TIMER se encarga de controlar 9 servomotores de forma sucesiva de tal manera que cuando termina con el noveno servomotor y comienza nuevamente con el primero han transcurrido $22.5\ ms$ lo que corresponde a una señal de control de $44.4\ Hz$.

Cada señal de control necesita de al menos un ancho de pulso de $2.4\ ms$. Para fines prácticos se ocupó uno de $2.5\ ms$.

$$Intervalo = \frac{22.5\ ms}{9\ servos} = 2.5\ ms/servo$$

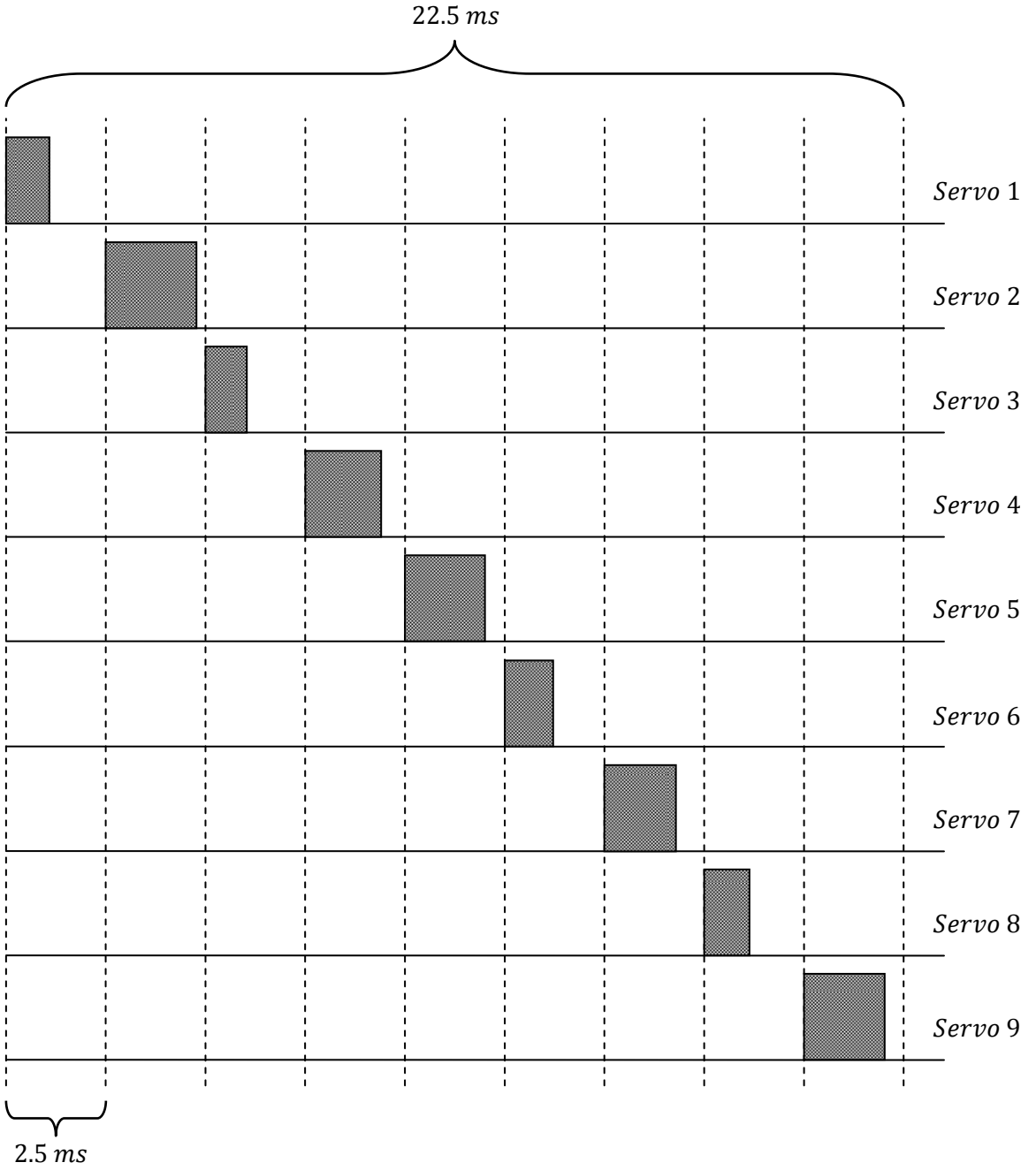
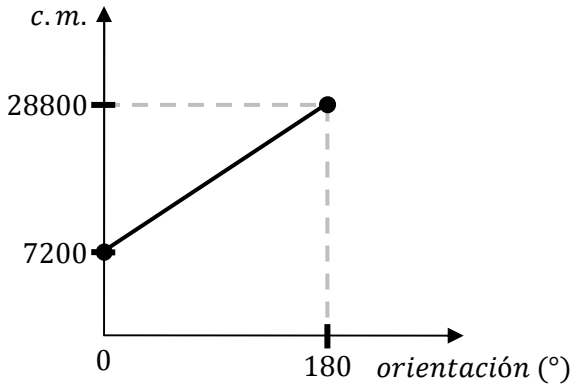


Figura 14: Señales generadas con cada TIMER que sirven para controlar 9 servomotores

Debido a que la relación entre la orientación del servomotor y señal de control es lineal, es posible obtener una ecuación de la recta en la que se relaciona la orientación y los $c.m.$

$$0.6\ ms \rightarrow 0° \rightarrow 7200\ c.m. \qquad 2.4\ ms \rightarrow 180° \rightarrow 28800\ c.m.$$



$$c.m. = 120 * G + 7200$$

Donde:

$c.m. \rightarrow ciclos\ máquina$

$G \rightarrow orientación\ deseada\ (°)$

Figura 15: Relación que existe entre los ciclos máquina que dura la señal de control y la orientación del servomotor

El número de ciclos máquina que debe de permanecer en estado alto (1 lógico), cada uno de los 18 servomotores, es calculado en un programa de computadora que luego es enviado al controlador vía Zigbee.

A continuación se muestra el proceso que lleva a cabo el microcontrolador para mantener la orientación de cada servomotor.

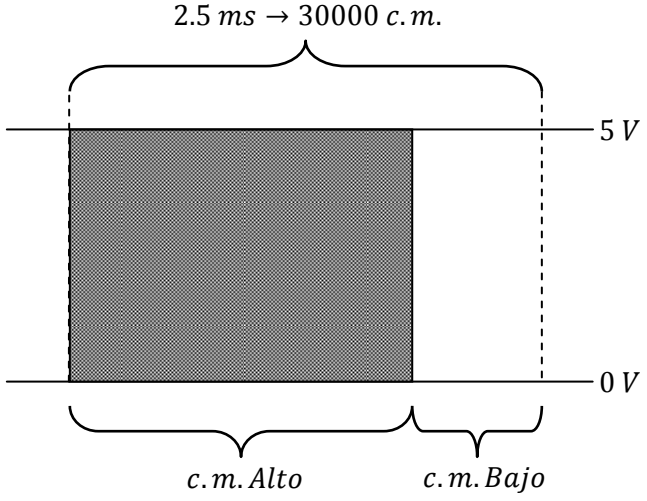


Figura 16: Señal de control generada por el TIMER para controlar cada servomotor

Suponiendo que se desean $X°$:

1. Primero se calcula el número de ciclos máquina que debe permanecer en estado alto:

$$c.m. = 120 * X + 7200 = Y$$

2. Tomando en cuenta que el contador del TIMER es de 16 bits, se fija un valor inicial tal que desde ese punto hasta que ocurra el desbordamiento transcurran "Y" ciclos máquina.

$$valor\ inicial\ del\ contador = 65536 - Y$$

3. En este punto se pone la señal en un valor alto (1 lógico) y comienza a contar el TIMER

4. Una vez que se ha desbordado el contador del TIMER se pone la señal en un valor bajo (0 lógico) y se calcula el número de ciclos máquina que debe de permanecer en estado bajo hasta completar el intervalo de $2.5\ ms$ que le corresponde a cada servomotor.

$$valor\ inicial\ del\ contador = 30000 - Y$$

   30000 es el número de ciclos máquina que debe durar el intervalo completo.

5. En el momento en que se vuelve a desbordar el contador con los nuevos parámetros es momento de repetir todos los pasos para la generar la siguiente señal de control y así cuando se termina de generar la 9ª señal de control también han transcurrido $22.5\ ms$ desde que se comenzó a generar la 1ª por lo que es momento de comenzar todo de nuevo.

4.2 Zigbee

Zigbee es un protocolo de comunicaciones inalámbrico basado en el estándar de comunicaciones para redes inalámbricas IEEE_802.15.4. Creado por *Zigbee Alliance*, una organización, teóricamente sin ánimo de lucro, de más de 200 grandes empresas (destacan Mitsubishi, Honeywell, Philips, Invensys, entre otras), muchas de ellas fabricantes de semiconductores.

Zigbee permite que dispositivos electrónicos de bajo consumo puedan realizar sus comunicaciones inalámbricas. Es especialmente útil para redes de sensores en entornos industriales, médicos y, sobre todo, domóticos.

Las comunicaciones Zigbee se realizan en la banda libre de 2.4 *GHz*. A diferencia de bluetooth, este protocolo no utiliza FHSS (Frequency hooping), sino que realiza las comunicaciones a través de una única frecuencia, es decir, de un canal. Normalmente puede escogerse un canal de entre 16 posibles. El alcance depende de la potencia de transmisión del dispositivo y del tipo de antenas utilizadas (cerámicas, dipolos, etc.) el alcance normal con antena dipolo en línea vista es de aproximadamente 100 *m* y en interiores de unos 30 *m* (versión de MaxStream de 1 *mW* de potencia). La velocidad de transmisión de datos de una red Zigbee es de hasta 256 *kbps*. Una red Zigbee la pueden formar, teóricamente, hasta 65535 equipos, es decir, el protocolo está preparado para poder controlar en la misma red esta cantidad enorme de dispositivos.

La comunicación entre el robot y la computadora se hizo de manera inalámbrica por medio de Zigbee.

Se utilizó protocolo Zigbee por varias razones:

- El envío y recepción de información se realiza de la misma manera que por RS232 y el microcontrolador utilizado tiene incorporado un periférico que facilita dicha comunicación.
- La velocidad de transferencia que se alcanza con Zigbee es más que suficiente para el control del robot.
- La facilidad de crear redes en este protocolo deja toda una gama de posibilidades para trabajos futuros.
- El consumo de energía puede reducirse en gran medida con la implementación de algunas funciones propias como lo es la “suspensión de labores”, durante la cual el módulo permanece inactivo cierto tiempo luego del cual se comunica con los demás módulos para obtener o enviar información.
- Uso de bandas de radio libres y sin necesidad de licencias.
- A futuro se espera que sea una tecnología inalámbrica de muy bajo costo.

Los módulos XBee pueden funcionar de 2 formas: modo AT (transparente) y API. Se utilizó modo API por algunas ventajas que ofrece como lo son:

- Información direccionada
- Se crean redes automáticamente
- El módulo transmisor detecta cuando el módulo receptor recibe la información o en su defecto detecta la ausencia del receptor.

Para utilizar el modo API debe de seguirse un protocolo específico para enviar información.

Frame Fields		Offset	Example	Description
Start Delimiter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x16	
Frame-specific Data	Frame Type	3	0x10	
	Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
	64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following addresses are also supported: 0x0000000000000000 – Reserved 64-bit address for the coordinator 0x000000000000FFFF – Broadcast address
		6	0x13	
		7	0xA2	
		8	0x00	
		9	0x40	
		10	0x0A	
		11	0x01	
		LSB 12	0x27	
	16-bit Destination Network Address	MSB 13	0xFF	Set to the 16-bit address of the destination device, if known. Set to 0xFFFE if the address is unknown, or if sending a broadcast
		LSB 14	0xFE	
	Broadcast Radius	15	0x00	Sets maximum number of hops a broadcast transmission can occur. If set to 0, the broadcast radius will be set to the maximum hops value.
	Options	16	0x00	Bit field of supported transmission options. Supported values include the following: 0x20 - Enable APS encryption (if EE=1) 0x40 – Use the extended transmission timeout for this destination. Enabling APS encryption decreases the maximum number of RF payload bytes by 4 (below the value reported by NP). Setting the extended timeout bit causes the stack to set the extended transmission timeout for the destination address. All unused and unsupported bits must be set to 0.
	RF Data	17	0x54	Data that is sent to the destination device
		18	0x78	
		19	0x44	
		20	0x61	
		21	0x74	
		22	0x61	
		23	0x30	
		24	0x41	
Checksum		25	0x13	0xFF – the 8 bit sum of bytes from offset 3 to this byte.

Tabla 4: Protocolo para el envío de datos utilizando Zigbee

También existe un protocolo para la recepción de datos.

Frame Fields		Offset	Example	Description
Start Delimiter		0	0x7E	
Length		MSB 1		Number of bytes between the length and the checksum
		LSB 2		
Frame-specific Data	Frame Type	3	0x90	
	64-bit Source Address	MSB 4	0x00	64-bit address of sender. Set to 0xFFFFFFFFFFFFFFFF (unknown 64-bit address) if the sender's 64-bit address is unknown
		5	0x13	
		6	0xA2	
		7	0x00	
		8	0x40	
		9	0x52	
		10	0x2B	
		LSB 11	0xAA	
	16-bit Source Network Address	MSB 12	0x7D	16-bit address of sender
		LSB 13	0x84	
	Receive Options	14	0x01	0x01 – Packet Acknowledged 0x02 – Packet was a broadcast packet 0x20 – Packet encrypted with APS encryption 0x40 – Packet was sent from an end device
	Received Data	15	0x52	Received RF data
		16	0x78	
		17	0x44	
		18	0x61	
		19	0x74	
		20	0x61	
Checksum		21	0x0D	0xFF – the 8 bit sum of bytes from offset 3 to this byte.

Tabla 5: Protocolo para la recepción de datos mediante Zigbee

El microcontrolador maestro (PIC18F452) es el encargado de decodificar la cadena recibida y obtener la información útil para luego enviarla al microcontrolador esclavo (PIC18F4550).

4.3 Blender

Blender es un programa que puede ser utilizado para modelar, texturizar, simular, animar y crear aplicaciones tridimensionales, incluyendo videojuegos, películas animadas y efectos especiales.

El programa fue inicialmente distribuido de forma gratuita pero sin el código fuente, con un manual disponible para la venta, aunque posteriormente pasó a ser software libre. Actualmente es compatible con todas las versiones de Windows, Mac OS X, Linux, Solaris, FreeBSD e IRIX.

El 18 de febrero de 2010 se estrenó el primer largometraje animado realizado íntegramente con software libre, usando Blender como principal herramienta. Se utilizó en la película Spider Man 2 para hacer una previsualización de escenas.

El modelo virtual (Figura 17) que se utilizó para la simulación fue creado en Blender 2.49a.

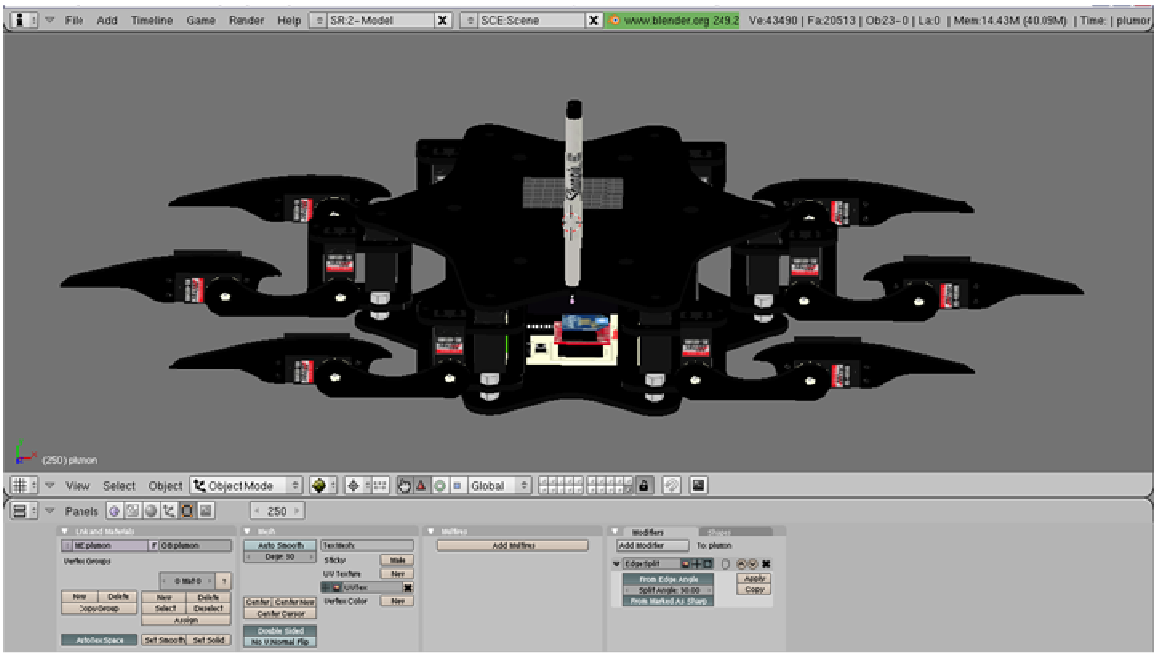


Figura 17: Imagen tomada del programa Blender con el modelo completamente diseñado

4.4 XNA

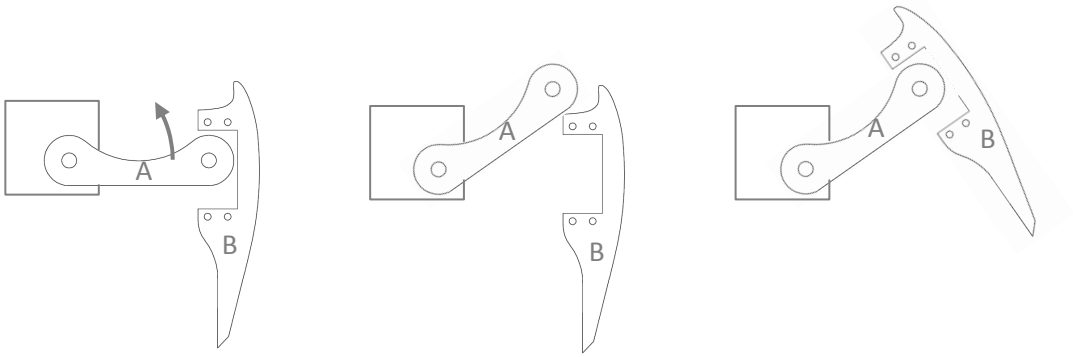
XNA es un conjunto de herramientas con un entorno de ejecución administrado proporcionado por Microsoft que facilita el desarrollo de videojuegos.

Utilizando las diferentes herramientas con que cuenta XNA es posible importar, graficar y manipular cada una de las partes que componen el modelo virtual creado previamente en Blender.

4.5 Simulador

Para controlar y simular al robot en su totalidad de diseñó una aplicación en C# que trabaja con XNA y el modelo realizado en Blender.

Los modelos que se importan traen consigo toda la información necesaria para poder graficarlos correctamente como lo son la traslación y rotación respecto al origen. Sin embargo, las partes que componen al robot no mantienen entre sí ninguna clase de unión, por lo que al realizar una traslación o una rotación de una pieza se deben de calcular las transformaciones al resto de las piezas en la cadena cinemática.



Rotar la pieza A del modelo

La pieza A rota independientemente de las demás piezas, se debe calcular la transformación de la pieza B

Se aplica a la pieza B la transformación calculada

Figura 18: Procedimiento para graficar de forma adecuada las diferentes partes del modelo en XNA

Otro punto importante es que cada una de las piezas tiene su propio centro y es en ese punto donde se aplicarán todas las transformaciones, por lo que se recomienda elegirlo de forma que nos facilite los cálculos posteriores.

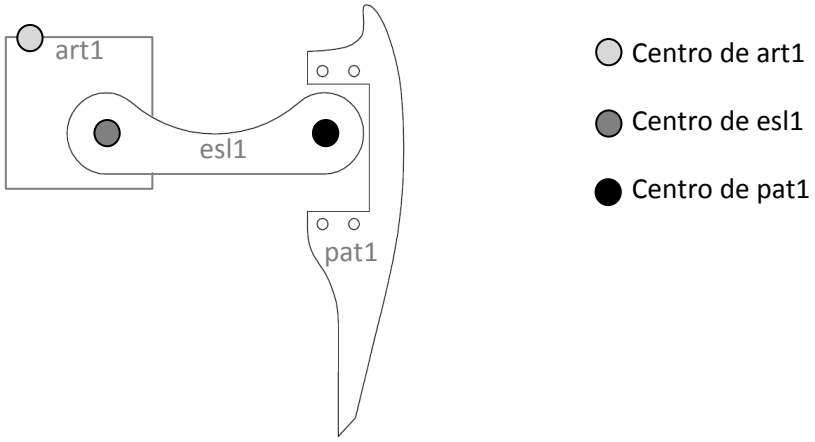


Figura 19: Cada pieza del modelo tiene un centro donde se aplican todas las transformaciones

Tanto el simulador como el controlador pueden ser fácilmente adecuados a modelos de diferentes dimensiones por lo que se convierte en una herramienta bastante flexible para simular robots.

La figura 20 muestra una imagen tomada directamente del simulador

Figura 20: Imagen tomada del simulador en XNA

CAPÍTULO 5

Construcción

En este capítulo se muestran las piezas utilizadas para la construcción del robot así como diagramas que muestran el ensamblado de cada una de sus partes. También se indican algunos ajustes que deben de realizarse una vez que el robot se encuentra completamente ensamblado.

5.1 Construcción de las piezas

La construcción del robot se hizo casi en su totalidad de piezas de acrílico de 4 *mm* transparente, que luego fue pintado con esmalte acrílico negro.

Las piezas de acrílico se cortaron con láser. Se optó por este método gracias a la velocidad, limpieza y precisión de los cortes así como por el precio de éstos.

También se utilizaron espaciadores cilíndricos de aluminio cortados a la medida y tornillos de diferentes dimensiones.

Las Figuras 21,22 y 23 muestran las piezas utilizadas así como el número de cada una de ellas. Las piezas no están a escala real.

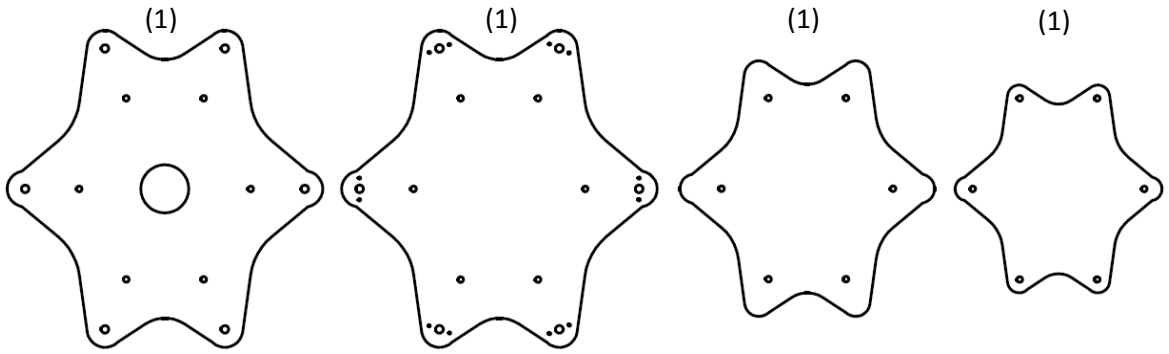


Figura 21: Piezas utilizadas para formar el cuerpo del robot

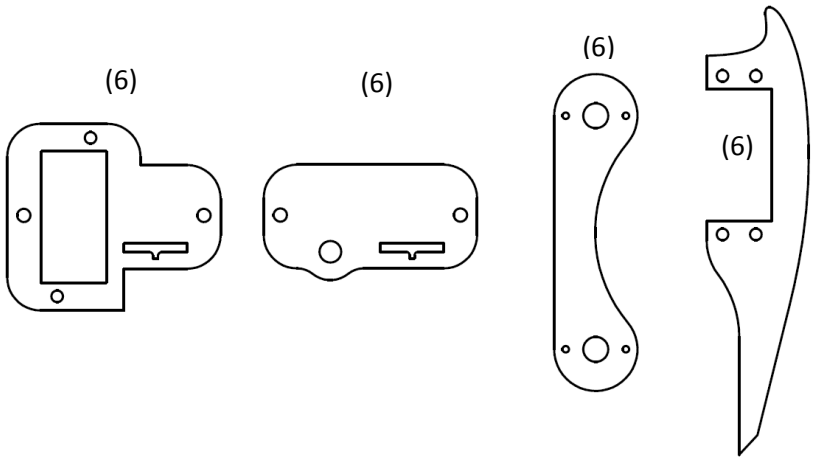


Figura 22: Piezas utilizadas para formar las extremidades del robot

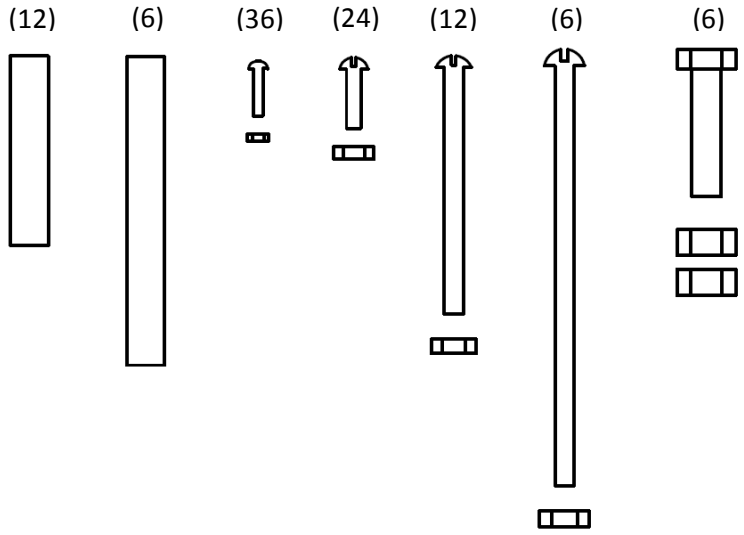


Figura 23: Espaciadores y tornillos utilizados para unir las diferentes piezas

5.2 Conexiones

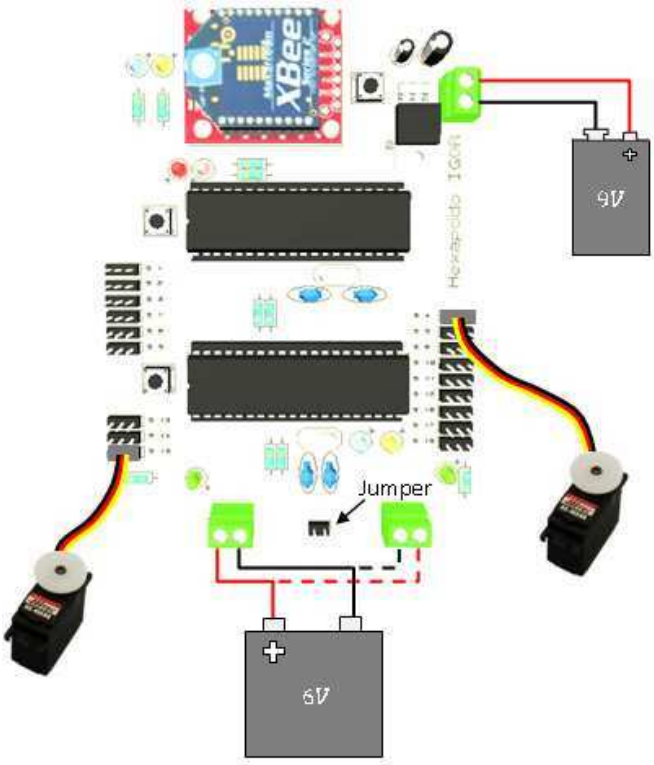


Figura 24: Diagrama de conexiones

Todos los servos del lado izquierdo se conectan de la misma forma en que está conectado el servo izquierdo en la Figura 24. Todos los servos del lado derecho se conectan como lo muestra el servo derecho en la Figura 24.

Se pueden alimentar de forma independiente los servomotores del lado izquierdo y del derecho utilizando ambas terminales. Si se coloca el Jumper se crea un puente entre ambas terminales.

Antes de ensamblar el robot es importante que todos los servos se encuentren con la orientación correcta, esto se hace enviando los datos desde el simulador con una posición de referencia fácilmente reproducible. En la Figura 25 se muestra una configuración fácilmente reproducible.

Figura 25: Vista frontal y superior del robot con una configuración fácilmente reproducible

Una vez que se tienen todos los servos con las orientaciones correspondientes a la posición previamente establecida se procede al ensamblaje tratando de que todas las piezas queden como se definió en el simulador y evitando rotar los motores antes de que sean completamente ensamblados.

5.3 Ensamble de una pata

Figura 26: Ensamble completo de una pata

Tres patas deben de ensamblarse como se muestra en la Figura 26, las otras tres se arman en espejo.

5.4 Ensamble de la parte superior del cuerpo

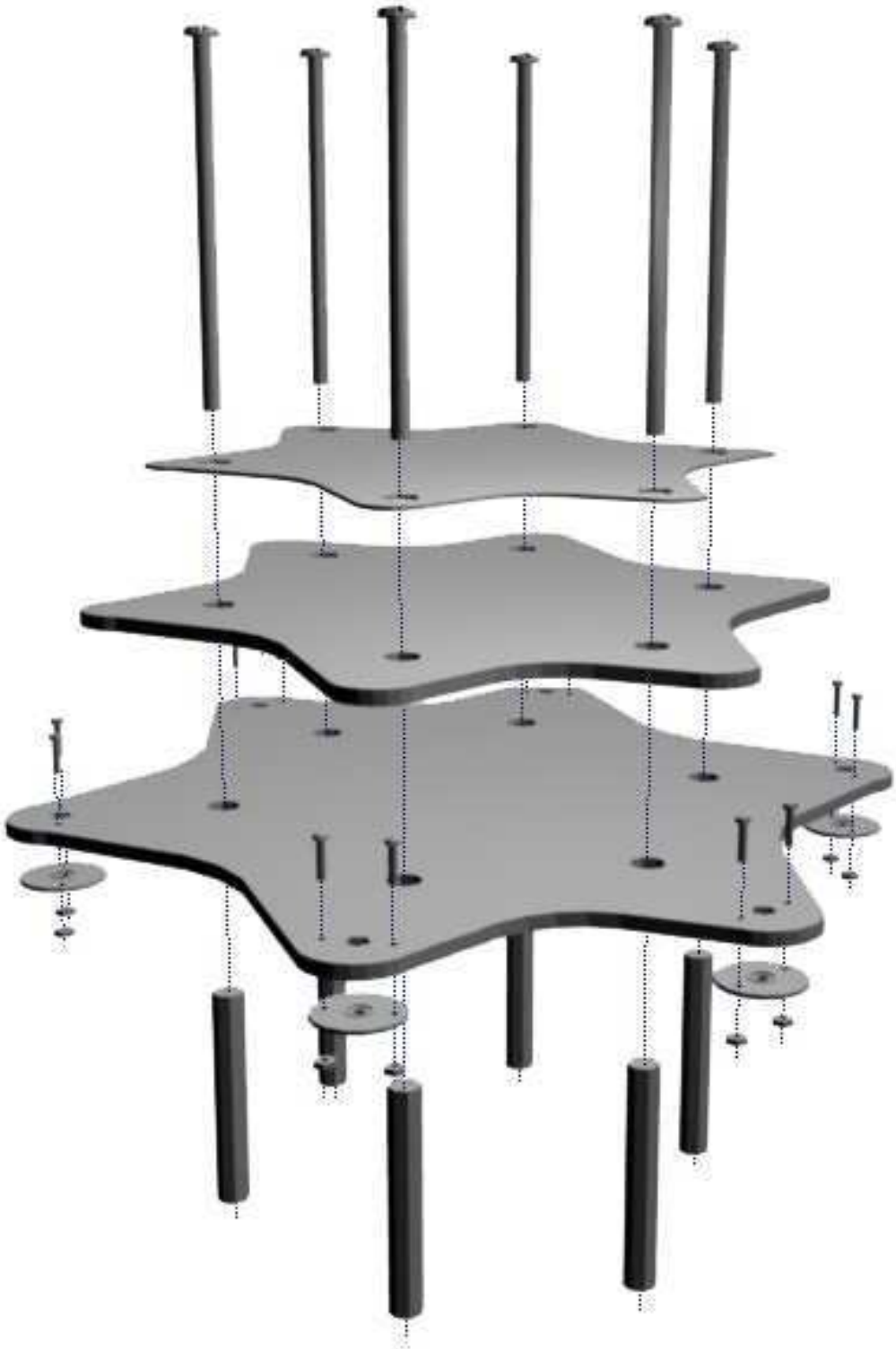


Figura 27: Ensamble de la parte superior del cuerpo

5.5 Unión de la pata al cuerpo

Figura 28: Unión de la pata al cuerpo

5.6 Ajustes

Una vez que se tiene el robot ensamblado lo más probable es que todas las orientaciones difieran ligeramente de las deseadas, y esto no sólo se debe al error humano sino que también es ocasionado por la unión que existe entre el eje del motor y el acoplamiento. En la Figura 29 se ilustra el problema.

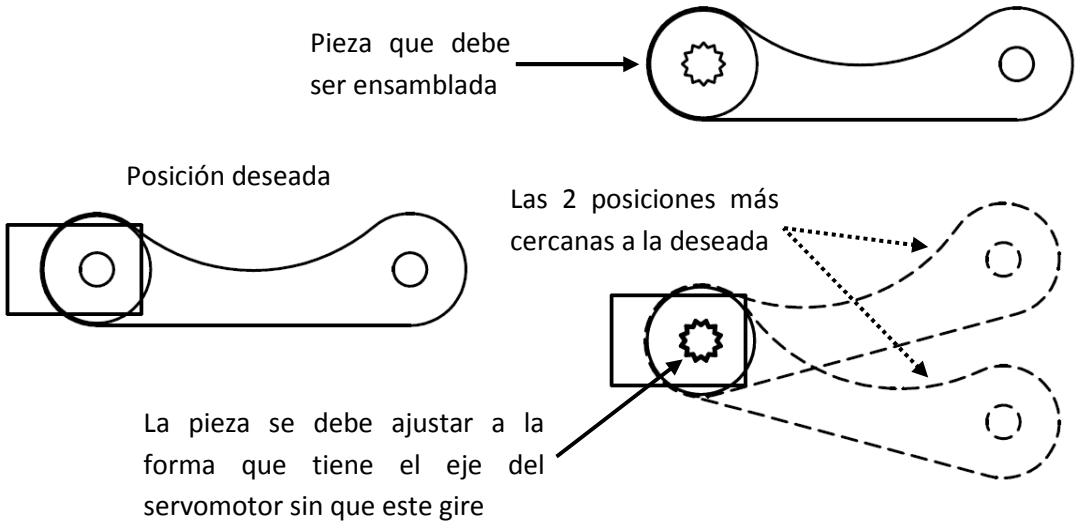


Figura 29: Problema de orientación durante el ensamblado de las piezas

Como se observa en la Figura 29, debido a la unión dentada que existe entre el eje del servomotor y el acoplamiento, la orientación resultante de la pieza esta discretizada, por lo que es imposible ensamblar la pieza como lo muestra el simulador sin girar el eje del servomotor.

Para solucionar este problema se necesita realizar un ajuste en el simulador que modifique al modelo físico, pero que mantenga en la misma posición al modelo virtual.

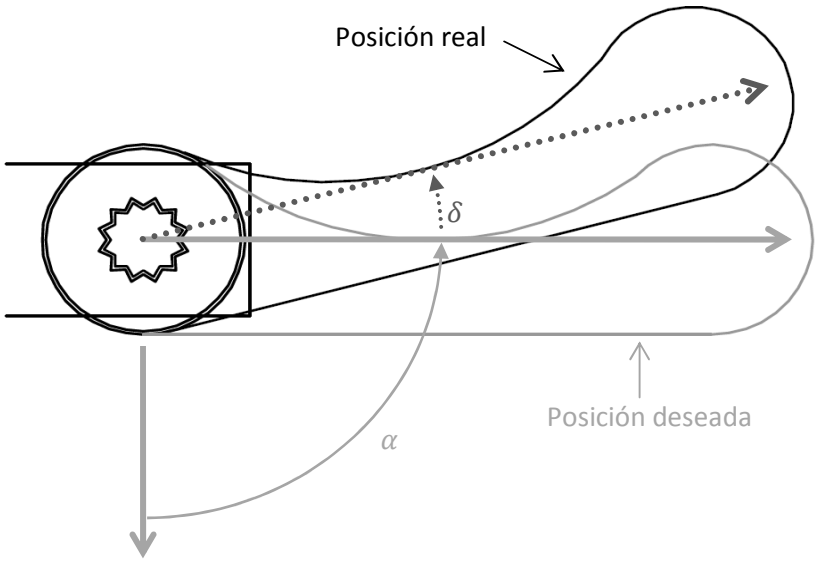


Figura 30: Diferencia entre la posición real y la posición deseada al momento de ensamblar las piezas

Se puede observar en la Figura 30 que la orientación de la pieza ensamblada difiere de la deseada en δ grados, se requiere un ajuste lineal del dato mandado por parte del simulador. En la Tabla 6 se muestra la solución a este problema.

	Orientación en el simulador	**Orientación que se manda al robot**	**Orientación obtenida en el robot**	
Sin ajuste				
	α	α	$\alpha + \delta$	La pieza se encuentra δ grados por encima de lo que debería
Con ajuste				
	α	$\alpha - \delta$	$(\alpha + \delta) - \delta = \alpha$	Al mandar una orientación con δ grados menos, el servomotor se posicionara δ grados menos que antes.

Tabla 6: Ajuste necesario para compensar las diferencias en la orientación de cada pieza durante el ensamblado

CAPÍTULO 6

Control

En este capítulo se muestra la cinemática directa y la cinemática inversa de una pata del robot. Métodos para calcular la posición de cada una de las patas para controlar en conjunto el cuerpo del robot. Los tipos de marcha o locomoción que se pueden llevar a cabo así como la generación de las trayectorias en cada una de las patas para lograr un movimiento omnidireccional. También se explican cada una de las funciones del control de Xbox 360 al momento de controlar remotamente el robot.

6.1 Cinemática directa de una pata.

Por medio de transformaciones homogéneas es posible determinar el punto donde se encuentra el extremo de la pata conociendo las dimensiones de la misma así como la rotación de cada una de sus articulaciones. A esto es lo que se conoce como cinemática directa.

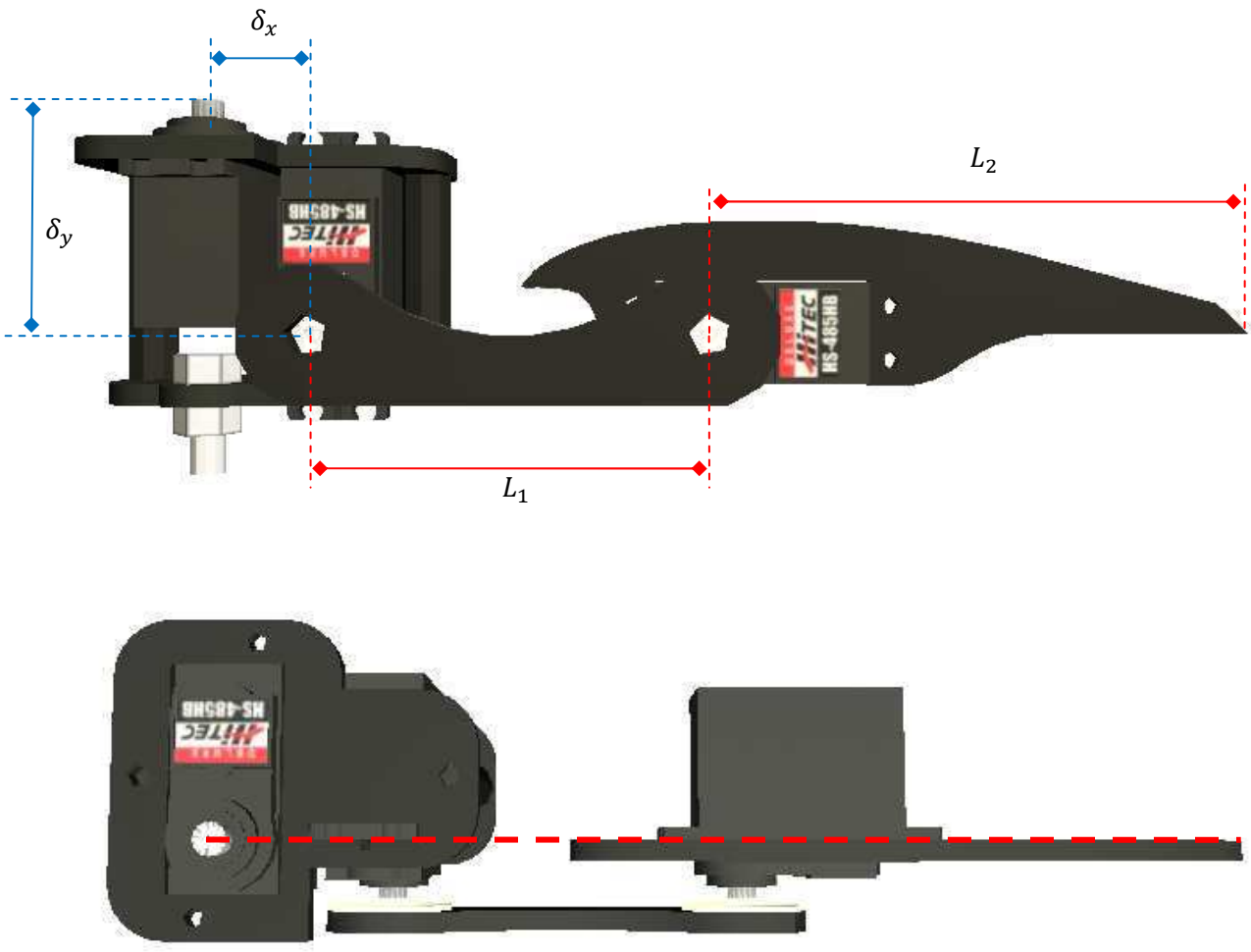


Figura 31: Dimensiones de una pata

Figura 32: Posición en el espacio de una pata respecto al sistema de coordenadas x,y,z

Modelo Simplificado

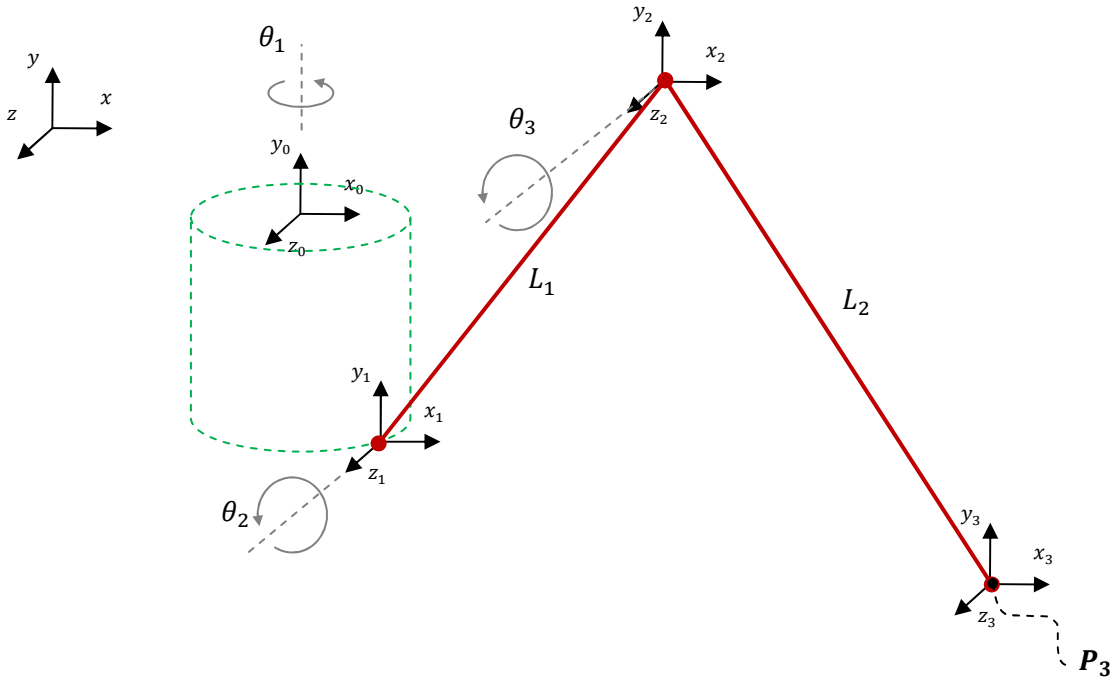


Figura 33: Modelo simplificado de una pata

Rotación en el eje $\boldsymbol{y_0}$:

$$\boldsymbol{T_0^{01}} = \begin{bmatrix} \boldsymbol{cos(\theta_1)} & \boldsymbol{0} & \boldsymbol{sin(\theta_1)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{-sin(\theta_1)} & \boldsymbol{0} & \boldsymbol{cos(\theta_1)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

Traslación en los ejes $\boldsymbol{x_0}$ y $\boldsymbol{y_0}$:

$$\boldsymbol{T_{01}^1} = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\delta_x} \\ \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{-\delta_y} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

$$\boldsymbol{T_0^1 = T_0^{01} * T_{01}^1} = \begin{bmatrix} \boldsymbol{cos(\theta_1)} & \boldsymbol{0} & \boldsymbol{sin(\theta_1)} & \boldsymbol{\delta_x cos(\theta_1)} \\ \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{-\delta_y} \\ \boldsymbol{-sin(\theta_1)} & \boldsymbol{0} & \boldsymbol{cos(\theta_1)} & \boldsymbol{-\delta_x sin(\theta_1)} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

Rotación en el eje $\boldsymbol{z_1}$:

$$\boldsymbol{T_1^{12}} = \begin{bmatrix} \boldsymbol{cos(\theta_2)} & \boldsymbol{-sin(\theta_2)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{sin(\theta_2)} & \boldsymbol{cos(\theta_2)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

Traslación en el eje $\boldsymbol{x_1}$:

$$\boldsymbol{T_{12}^2} = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{L_1} \\ \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

$$\boldsymbol{T_1^2 = T_1^{12} * T_{12}^2} = \begin{bmatrix} \boldsymbol{cos(\theta_2)} & \boldsymbol{-sin(\theta_2)} & \boldsymbol{0} & \boldsymbol{L_1 cos(\theta_2)} \\ \boldsymbol{sin(\theta_2)} & \boldsymbol{cos(\theta_2)} & \boldsymbol{0} & \boldsymbol{L_1 sin(\theta_2)} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

Rotación en el eje $\boldsymbol{z_2}$:

$$\boldsymbol{T_2^{23}} = \begin{bmatrix} \boldsymbol{cos(\theta_3)} & \boldsymbol{-sin(\theta_3)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{sin(\theta_3)} & \boldsymbol{cos(\theta_3)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

Traslación en el eje $\boldsymbol{x_2}$: $$\boldsymbol{T_{23}^3 = \begin{bmatrix} 1 & 0 & 0 & L_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

$$\boldsymbol{T_2^3 = T_2^{23} * T_{23}^3 = \begin{bmatrix} cos(\theta_3) & -sin(\theta_3) & 0 & L_2 cos(\theta_3) \\ sin(\theta_3) & cos(\theta_3) & 0 & L_2 sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

La cinemática directa de la pata se calcula como

$$\boldsymbol{P = T_0^3 * P_3}$$

En donde $\boldsymbol{T_0^3 = T_0^1 * T_1^2 * T_2^3}$ y $\boldsymbol{P_3}$ son las coordenadas del extremo de la pata en base al sistema de referencia 3. Puesto que la cinemática directa se realizó colocando el sistema de referencia 3

En el extremo de la pata, el punto $\boldsymbol{P_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}$

Por lo tanto,

$$\boldsymbol{P = \begin{bmatrix} \cos(\theta_1)\,(\delta_x + L_1 cos(\theta_2) + L_2 \cos(\theta_2 + \theta_3)) \\ -\delta_y + L_1 sin(\theta_2) + L_2 sin(\theta_2 + \theta_3) \\ -sin(\theta_1)(\delta_x + L_1 cos(\theta_2) + L_2 cos(\theta_2 + \theta_3)) \\ 1 \end{bmatrix}}$$

Sin embargo, el punto $\boldsymbol{P}$ esta visto desde el sistema de referencia $\boldsymbol{x_0, y_0, z_0}$

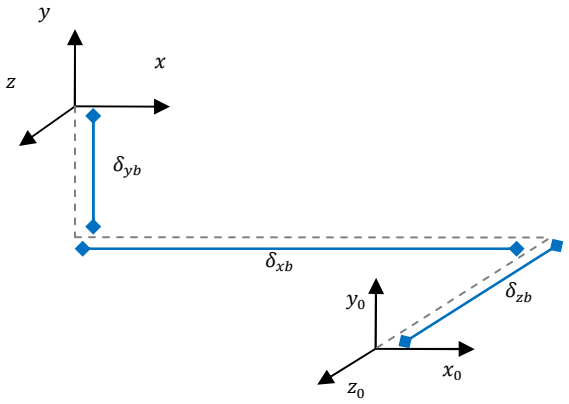


Figura 34: Traslación del sistema de referencia de la pata

Por lo tanto, el punto $\boldsymbol{P}$ visto desde el origen queda como:

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{\cos(\theta_1)\,(\delta_x + L_1 cos(\theta_2) + L_2 \cos(\theta_2 + \theta_3)) + \delta_{xb}} \\ \boldsymbol{-\delta_y + L_1 sin(\theta_2) + L_2 sin(\theta_2 + \theta_3) + \delta_{yb}} \\ \boldsymbol{-sin(\theta_1)(\delta_x + L_1 cos(\theta_2) + L_2 cos(\theta_2 + \theta_3)) + \delta_{zb}} \\ \boldsymbol{1} \end{bmatrix}$$

6.2 Cinemática inversa de una pata

La cinemática inversa es utilizada para encontrar las rotaciones que deben de realizarse en cada una de las articulaciones para colocar el extremo de la pata en un punto específico.

Existen diversos métodos para calcular la cinemática inversa. Debido a que se desea encontrar la solución a un problema de sólo 3 grados de libertad el método más simple es el geométrico.

La cinemática inversa se calculó en relación al origen del robot (sistema de coordenadas $\boldsymbol{x, y, z}$) y no depende de la posición ni orientación del robot en el espacio.

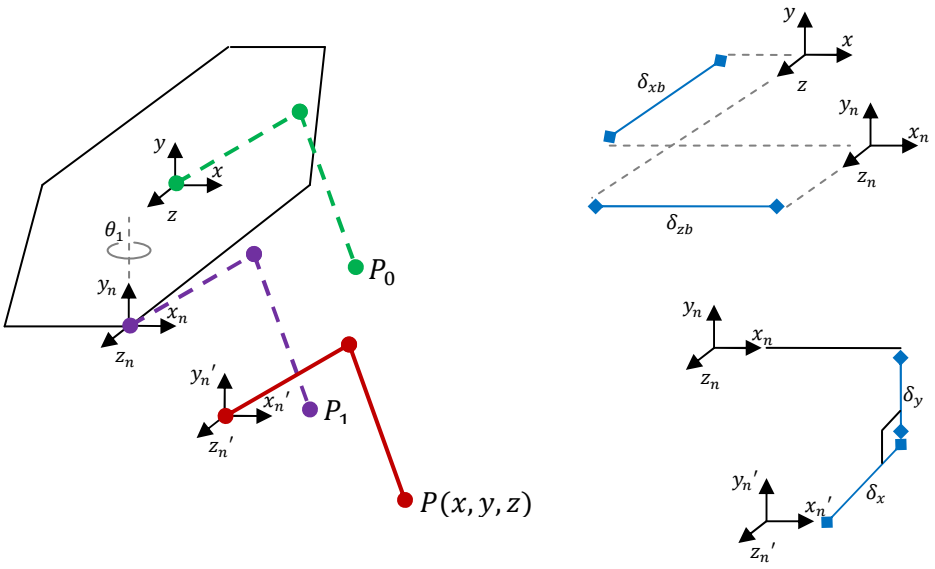


Figura 35: Diferentes sistemas de referencia utilizados para la cinemática inversa

$$\boldsymbol{P_1 = P(x - \delta_x cos(\theta_1),\quad y + \delta_y,\quad z - \delta_x \sin(\theta_1))}$$

$$\boldsymbol{P_0 = P(x - \delta_x cos(\theta_1) - \delta_{xb},\quad y + \delta_y,\quad z - \delta_x \sin(\theta_1) - \delta_{zb})}$$

$$\boldsymbol{x^* = x - \delta_x cos(\theta_1) - \delta_{xb},\quad y^* = y + \delta_y,\quad z^* = z - \delta_x \sin(\theta_1) - \delta_{zb}}$$

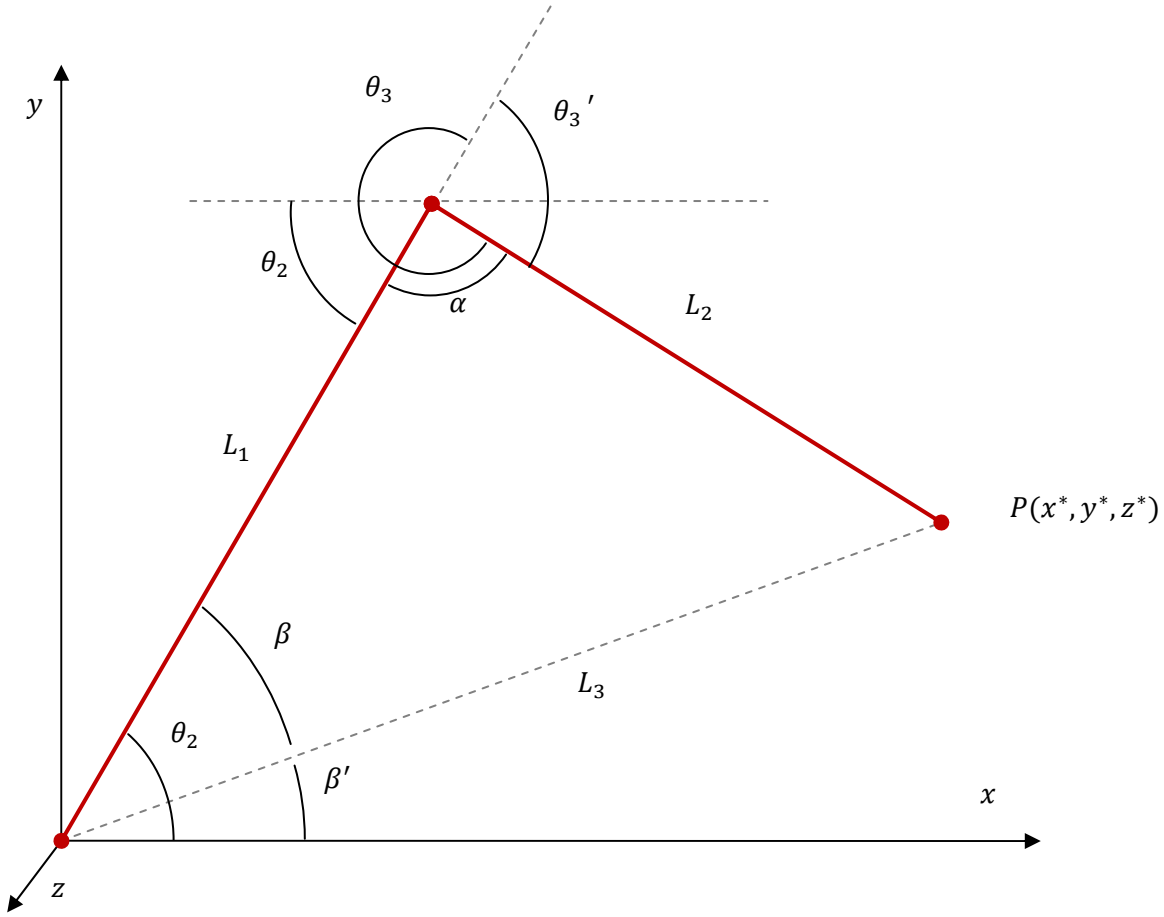


Figura 36: Cinemática inversa de una pata en el plano. Cálculo de θ_2 y θ_3

$$L_3 = \sqrt{x^{*2} + y^{*2} + z^{*2}}$$

$$\beta' = tan^{-1}\left(\frac{y^*}{\sqrt{x^{*2} + z^{*2}}}\right)$$

Utilizando la ley de cosenos

$$\alpha = cos^{-1}\left(\frac{{L_1}^2 + {L_2}^2 - {L_3}^2}{2L_1L_2}\right)$$

$$\beta = cos^{-1}\left(\frac{{L_1}^2 + {L_3}^2 - {L_2}^2}{2L_1L_3}\right)$$

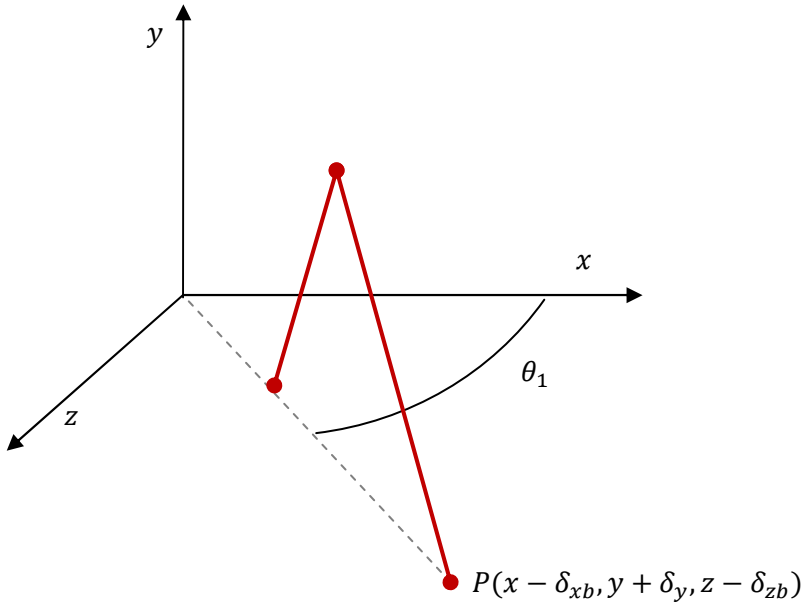


Figura 37: Cálculo de θ_1

$$\boldsymbol{\theta_1 = tan^{-1}\left(\frac{z-\delta_{zb}}{x-\delta_{xb}}\right)}$$

$$\boldsymbol{\theta_2 = \beta + \beta'}$$

$$\boldsymbol{\theta_3 = 180 + \alpha, \qquad \theta_3{}' = 180 - \alpha}$$

6.3 Posición de reposo

La posición de reposo es la que mantiene el robot mientras no tenga traslaciones ni rotaciones en su cuerpo y además tenga todas las patas en contacto con el suelo. Esta posición se construye a partir de rotaciones en cada una de sus articulaciones previamente definidas. En la Tabla 7 se muestran las rotaciones seleccionadas para la posición de reposo.

Número de pata	Articulación (°)	Eslabón (°)	Pata (°)
1	-45.0	45.0	-132.0
2	45.0	-45.0	132.0
3	-45.0	45.0	-132.0
4	45.0	-45.0	132.0
5	-45.0	45.0	-132.0
6	45.0	-45.0	132.0

Tabla 7: Orientación de cada pieza para la posición de reposo

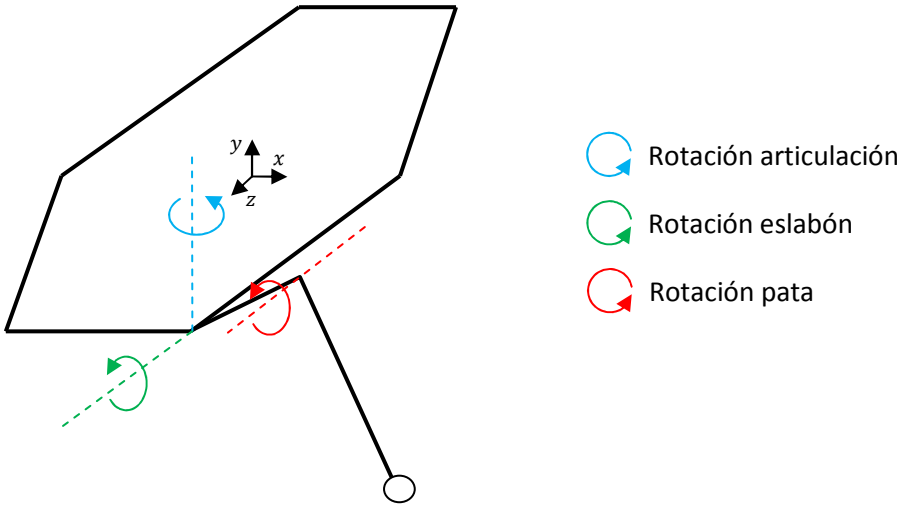


Figura 38: Diferentes rotaciones de una extremidad

A partir de la posición de reposo es posible determinar ciertas constantes que serán útiles para calcular los movimientos del robot.

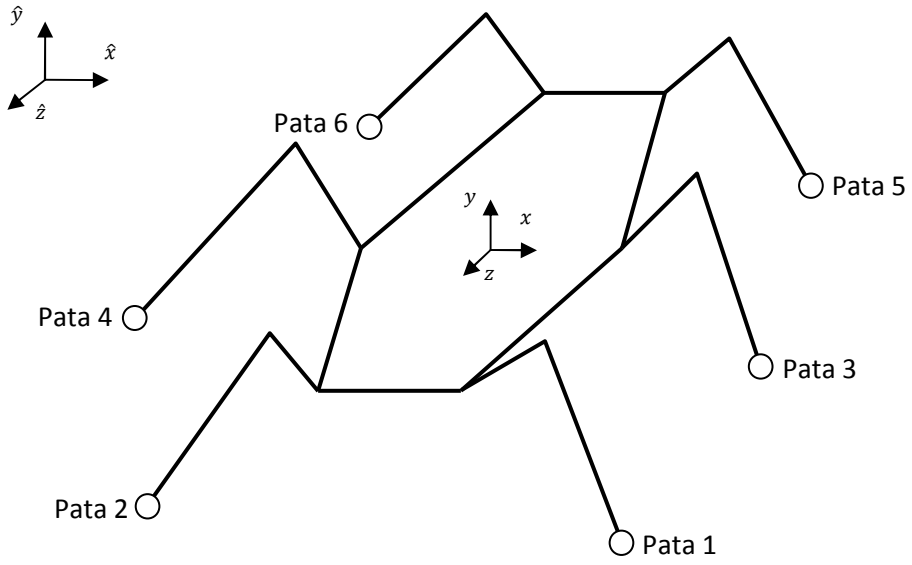


Figura 39: Modelo simplificado del robot con un nombre para cada extremidad

6.4 Movimientos de traslación y rotación del cuerpo del robot

El movimiento omnidireccional del cuerpo se logra gracias a los 3 grados de libertad que tiene cada pata del robot. Al hablar de movimiento omnidireccional se entiende que el cuerpo puede moverse en cualquier dirección de forma instantánea.

Una vez que se tiene la cinemática inversa del robot, la cual se compone de un modelo para cada pata, todo el control del robot se hace manipulando los puntos que se alcanzan con cada pata.

Ejemplo de un movimiento de traslación del cuerpo en el eje x

- Posición en la que el cuerpo del robot no tiene traslaciones ni rotaciones

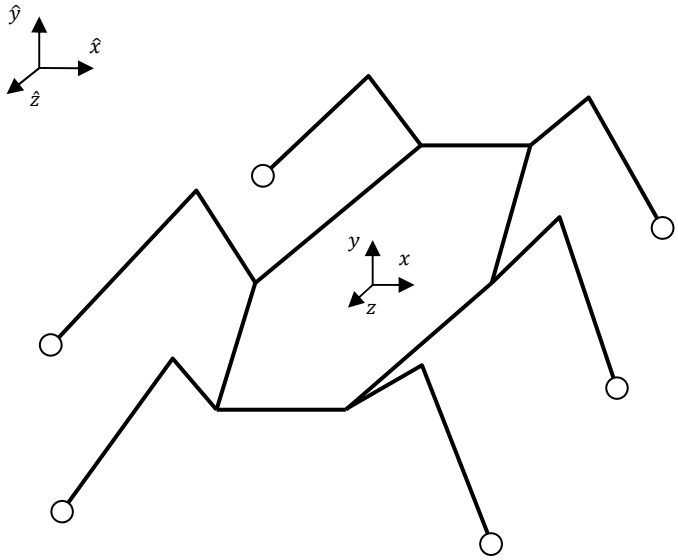


Figura 40: Robot en posición de reposo

- Se aplica una transformación (traslación en x) a los puntos finales de cada pata

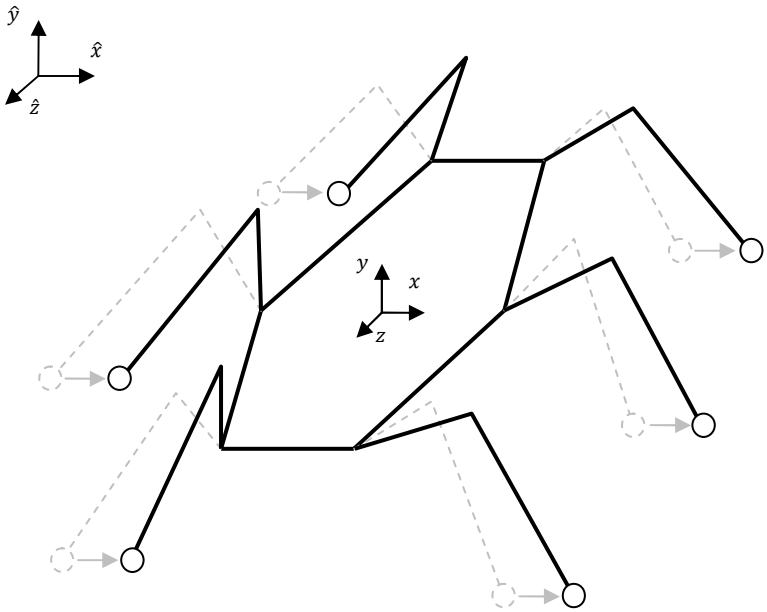


Figura 41: Traslación en x de todos los puntos finales

- Los puntos de contacto con el suelo se trasladan en relación al origen del robot $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, sin embargo, en relación al origen del espacio $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ el que se traslada es el cuerpo del robot y lo hace en sentido opuesto a la transformación de los puntos finales de cada pata

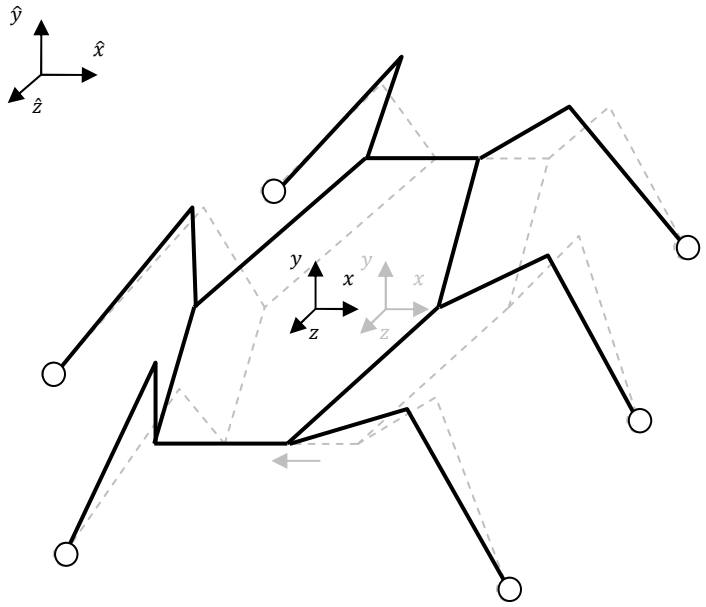


Figura 42: Traslación del cuerpo en $\boldsymbol{x}$

Ejemplo de un movimiento de rotación del cuerpo en el eje $\boldsymbol{z}$

- Posición en la que el cuerpo del robot no tiene traslaciones ni rotaciones

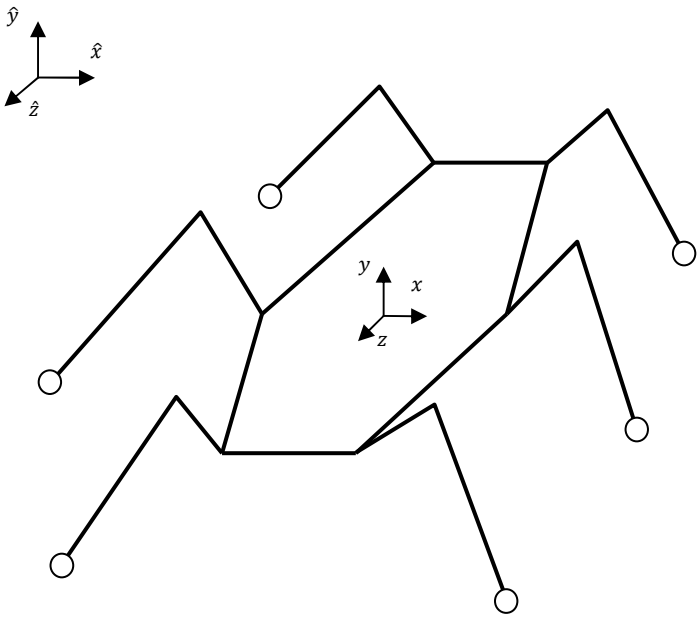


Figura 43: Posición en reposo del robot

- Se aplica una transformación (rotación en $\boldsymbol{z}$) a los puntos finales de cada pata

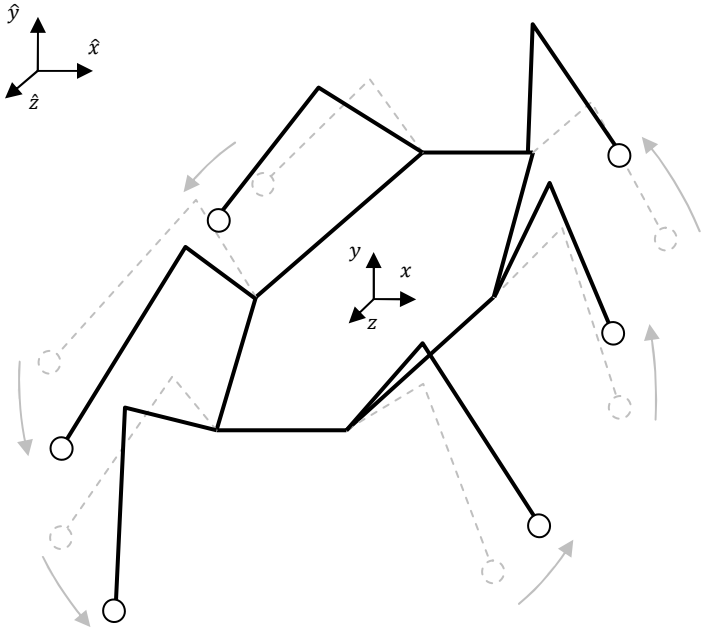


Figura 44: Rotación en z de todos los puntos finales

- Los puntos de contacto con el suelo se rotan en relación al origen del robot ($\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$), sin embargo, en relación al origen del espacio ($\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}$) el que rota es el cuerpo del robot y lo hace en sentido opuesto a la transformación de los puntos finales de cada pata

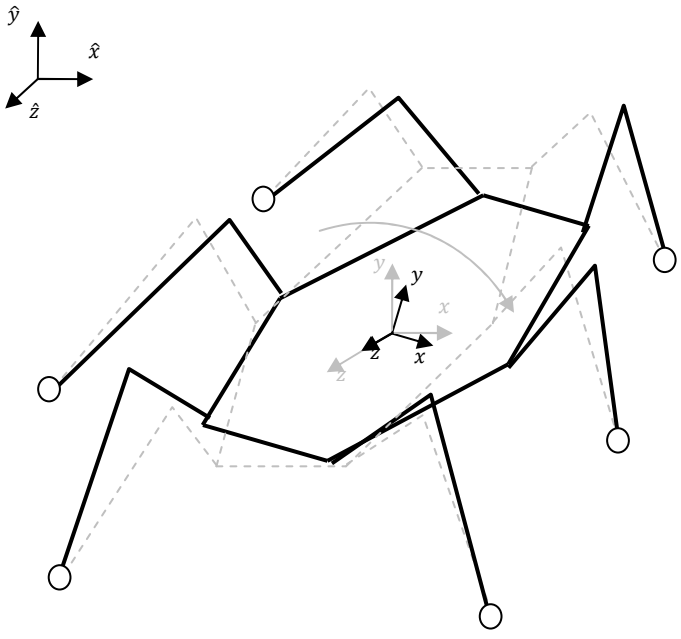


Figura 45: Rotación en z del cuerpo del robot

Una vez que se tiene claro el concepto de lo que se debe hacer, ya sea en una traslación o en una rotación, se puede diseñar un método que lo realice y arroje los resultados pertinentes.

Las traslaciones y rotaciones del cuerpo pueden realizarse en cualquier momento, no es necesario que el robot se encuentre en posición de reposo.

Existen 7 variables que controlan los movimientos de traslación y rotación del cuerpo del robot dentro del programa:

1. Traslación en X (*traslaciónCuerpoVector.X*)
2. Traslación en Y (*traslaciónCuerpoVector.Y*)
3. Traslación en Z (*traslaciónCuerpoVector.Z*)
4. Rotación en X (*dA*)
5. Rotación en Y (*dB*)
6. Rotación en Z (*dG*)
7. Centro de rotaciones (*centro*)

El centro de rotaciones indica el punto que servirá de pivote para las rotaciones $X, Y\ y\ Z$.

En la Figura 46 se muestra un ejemplo de una rotación en Z con el centro de rotaciones en el origen. En la Figura 47 una rotación en Z en con el centro de rotaciones desplazado sobre el eje Y.

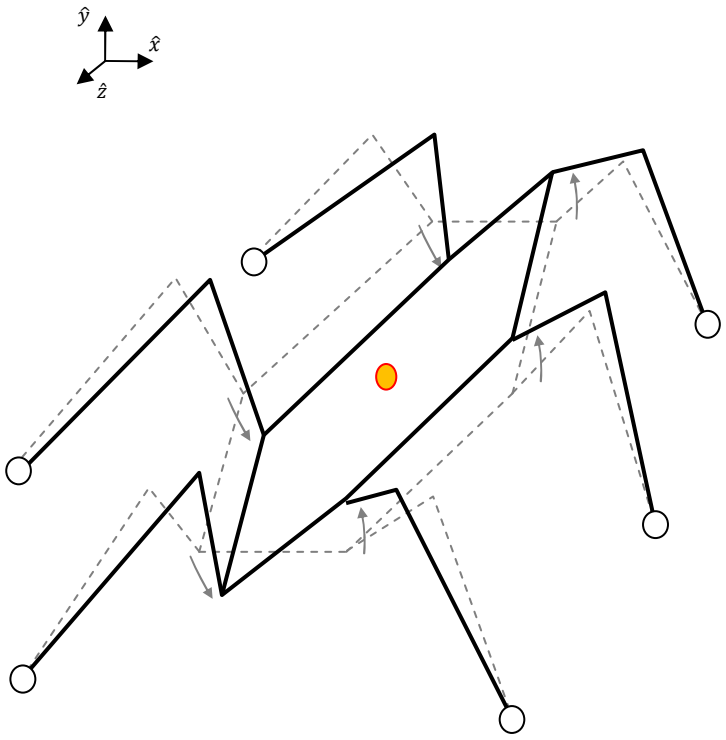


Figura 46: Rotación en Z del cuerpo del robot con el centro de rotaciones en el origen

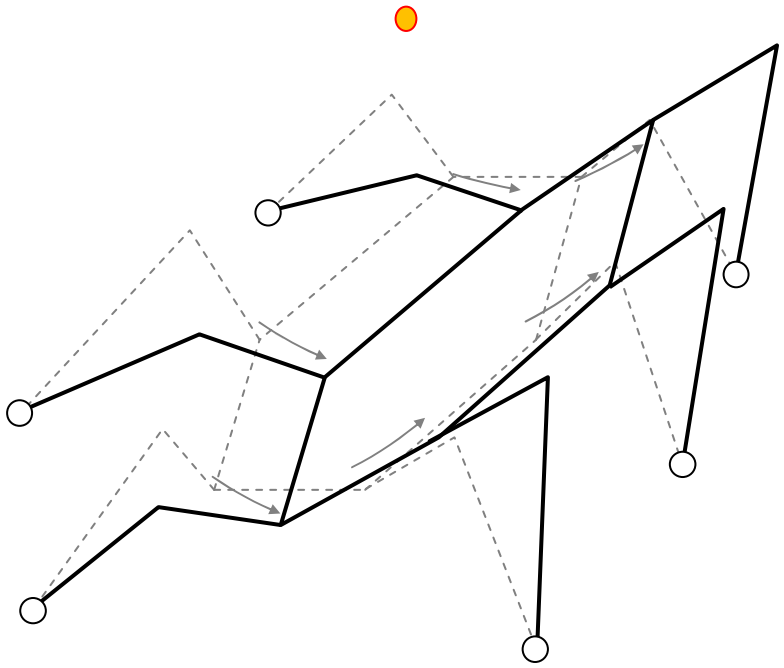


Figura 47: Rotación en Z del cuerpo del robot con el centro de rotaciones desplazado sobre el eje Y

En la Figura 48 se muestra el método utilizado para realizar movimientos de traslación y rotación en el cuerpo del robot.

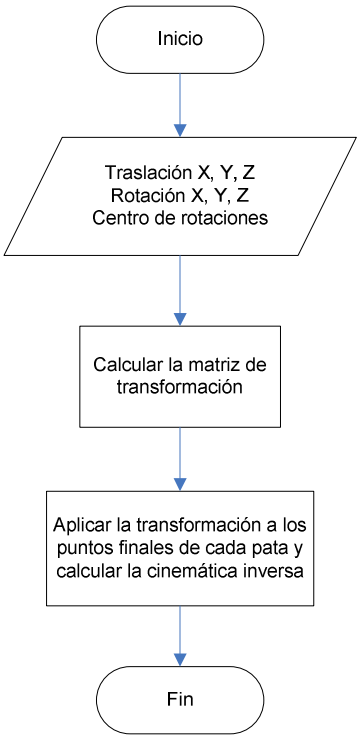


Figura 48 Método para realizar traslaciones y rotaciones del cuerpo del robot

Los puntos finales a los que se les deben aplicar la transformación son los puntos en los que se encuentran las patas cuando no hay traslación ni rotación en el cuerpo.

6.5 Tipos de marcha o locomoción

Anteriormente se analizó las traslaciones y rotaciones del cuerpo del robot (los puntos finales de cada pata en relación al espacio permanecen constantes por lo que el único movimiento aparente es del cuerpo del robot). Ahora es momento de los movimientos que permiten que el robot se desplace a cualquier posición de manera instantánea en el espacio.

Para que el robot pueda desplazarse, ya sea una traslación, rotación o incluso una traslación y rotación simultánea, cada una de las patas debe de seguir una trayectoria.

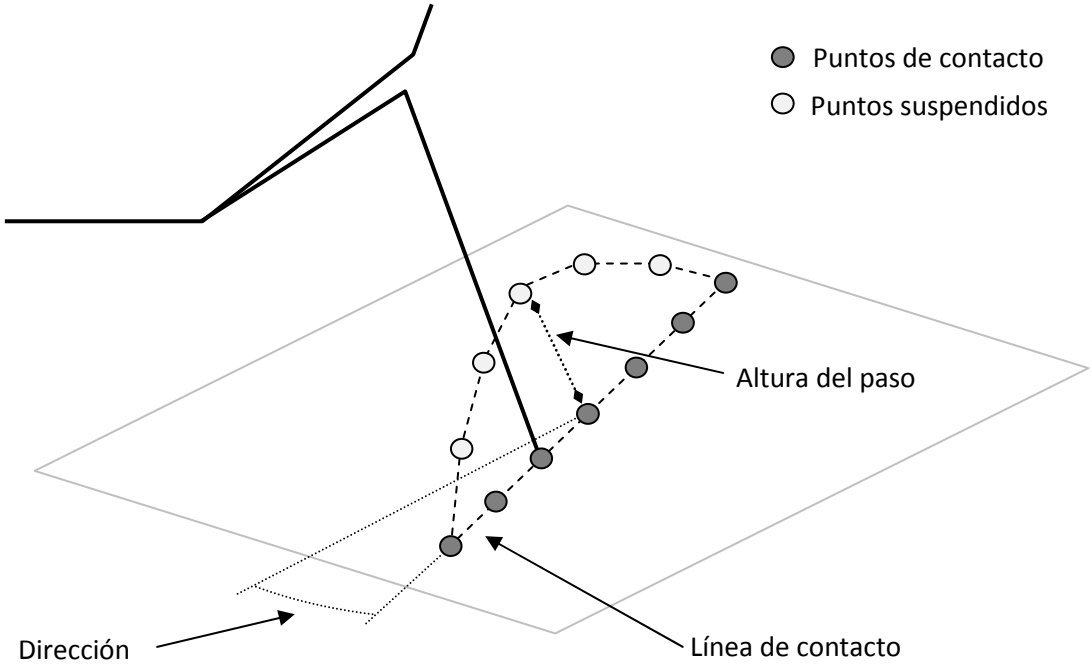


Figura 49: Trayectoria que debe de seguir una pata para que el robot se desplace

L_c - Línea de contacto – es la línea que se encuentra en contacto directo con el suelo.

P_c - Puntos de contacto – los puntos de la trayectoria que se encuentran sobre la línea de contacto.

P_s - Puntos suspendidos – los puntos de la trayectoria fuera de la línea de contacto.

Dar un paso – pasar de un punto al siguiente.

Existen diversas técnicas que se utilizan para recorrer las trayectorias y con ello lograr un movimiento en el robot. Las que se van a utilizar son: 3, 2 y 1 pata por movimiento.

Técnica	Número mínimo de patas en contacto con el suelo	Número máximo de patas separadas del suelo
3 patas por movimiento	3	3
2 patas por movimiento	4	2
1 pata por movimiento	5	1

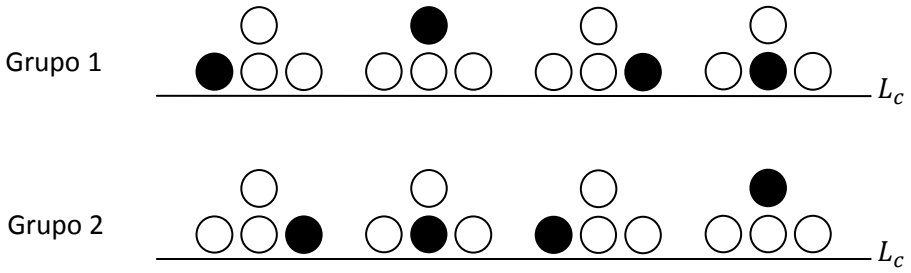
Tabla 8: El mínimo número de patas en contacto con el suelo y el máximo número de patas separadas del suelo para cada técnica de desplazamiento

El número mínimo de puntos de contacto que se necesitan en la trayectoria así como el incremento que se puede realizar en ellos depende de la técnica que se vaya a utilizar. El número de puntos suspendidos depende tanto del número de puntos de contacto como de la técnica elegida.

6.5.1 Tres patas por movimiento

El número mínimo de puntos de contacto es de 3 y el incremento puede ser cualquier número.

En esta técnica se hacen 2 grupos de 3 patas cada uno.



$$P_c = 3 + incremento \qquad P_s = P_c - 2$$

Figura 50: Diagrama de movimientos para la técnica de desplazamiento "tres patas por movimiento"

El grupo 1 está compuesto por la patas 1, 4 y 5 y el grupo 2 por las patas 2, 3 y 6.

En la Figura 51 se muestra un ejemplo de cómo interpretar el diagrama de la Figura 50.

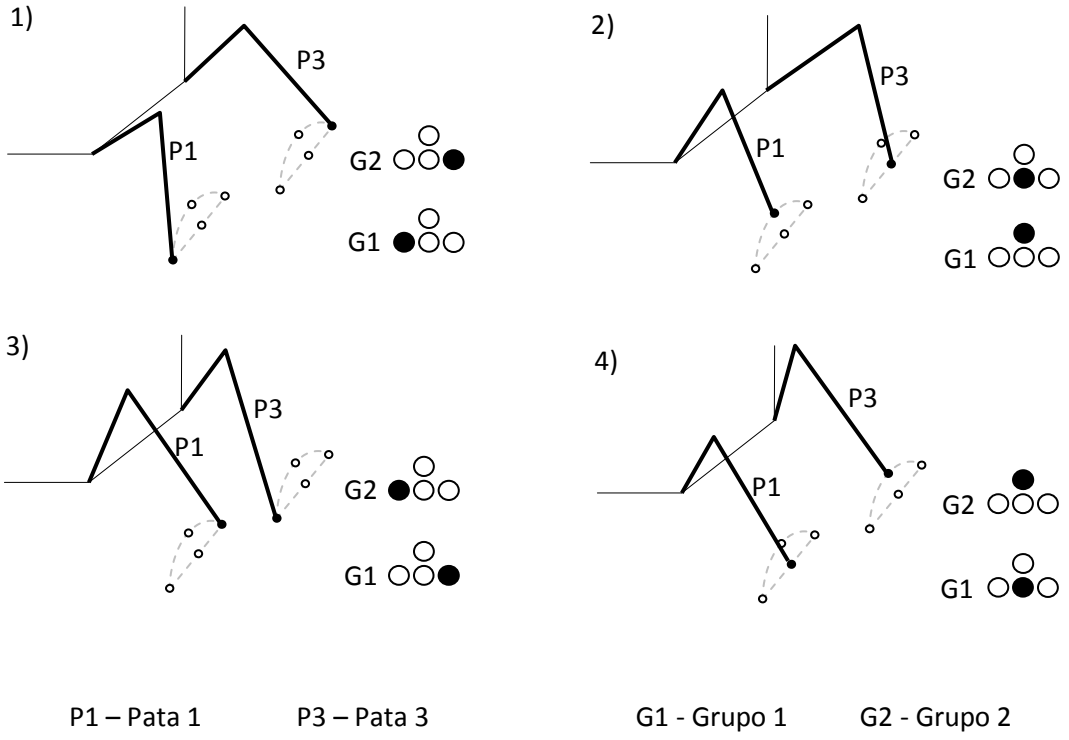
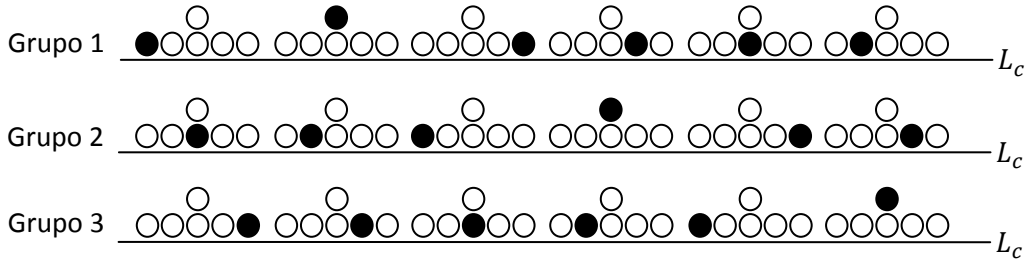


Figura 51: Interpretación de un diagrama de movimientos

6.5.2 Dos patas por movimiento

El número mínimo de puntos de contacto es de 5 y el incremento debe de ser en múltiplos de 2. En esta técnica se hacen 3 grupos de 2 patas cada uno.



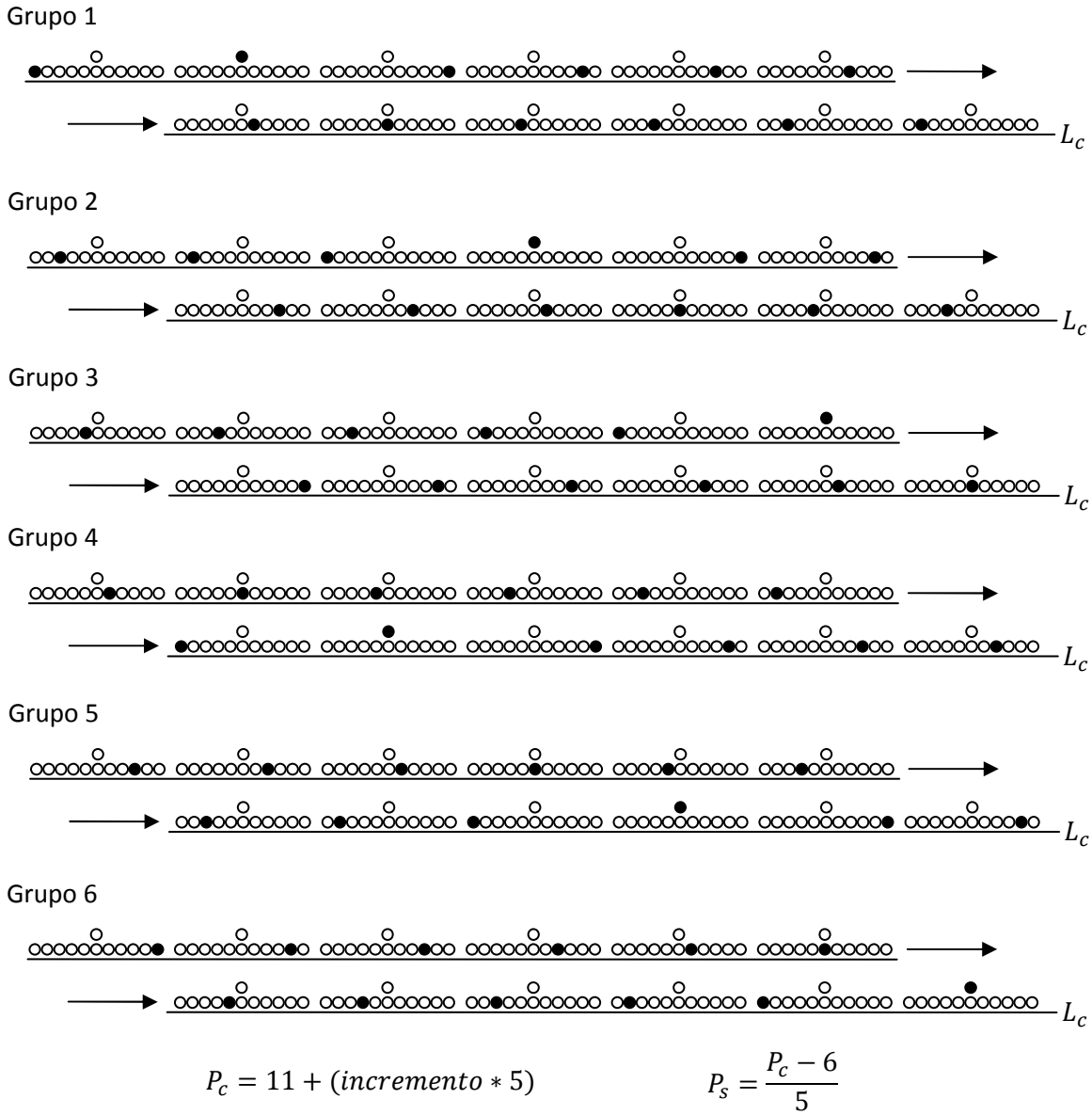
$$P_c = 5 + (incremento * 2) \qquad P_s = \frac{P_c - 3}{2}$$

Figura 52: Diagrama de movimientos para la técnica de desplazamiento "dos patas por movimiento"

El grupo 1 está compuesto por las patas 1 y 6, el grupo 2 por las patas 3 y 4, y el grupo 3 por las patas 2 y 5.

6.5.3 Una pata por movimiento

El número mínimo de puntos de contacto es de 11 y el incremento debe de ser en múltiplos de 5, en esta técnica se hacen 6 grupos de 1 pata cada uno.



$$P_c = 11 + (incremento * 5) \qquad P_s = \frac{P_c - 6}{5}$$

Figura 53: Diagrama de movimientos para la técnica de desplazamiento "una pata por movimiento"

Con esta técnica cada grupo corresponde a una pata.

6.6 Generación de trayectorias

Anteriormente se revisó la manera en que deben de recorrerse las trayectorias para lograr un desplazamiento dependiendo de la técnica de locomoción elegida, ahora se analizará el método utilizado para generar dicha trayectoria.

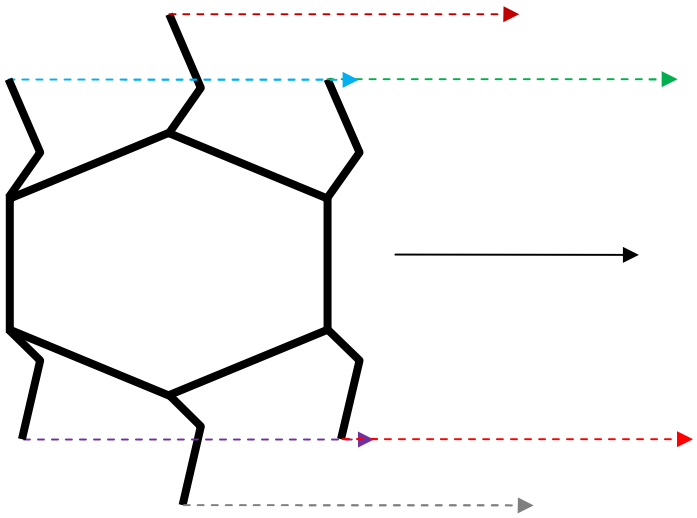


Figura 54: Desplazamiento de traslación

En una traslación pura, todas las trayectorias son paralelas entre si y tienen una misma orientación.

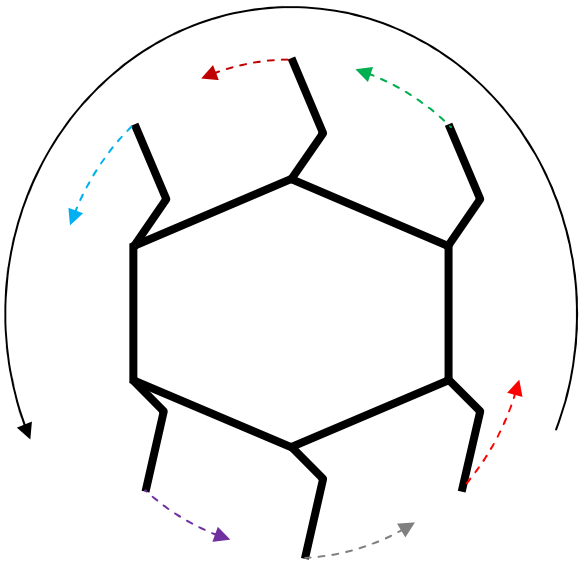


Figura 55: Desplazamiento de rotación

En una rotación pura, todas las trayectorias comparten el mismo centro de rotación.

Para lograr un movimiento de traslación y rotación simultánea, todas las trayectorias deben de cumplir con los mismos parámetros, avance lineal y avance angular por cada paso.

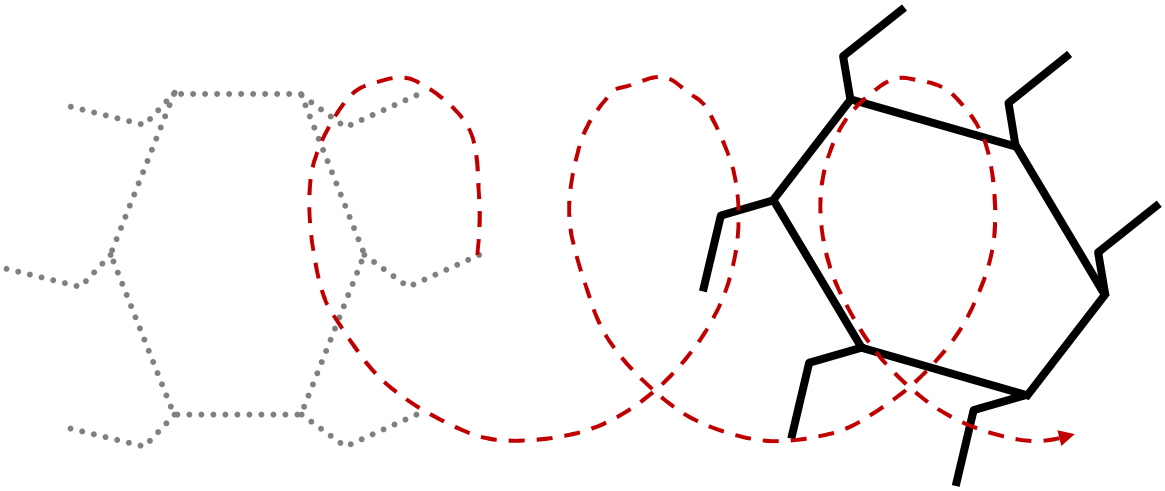
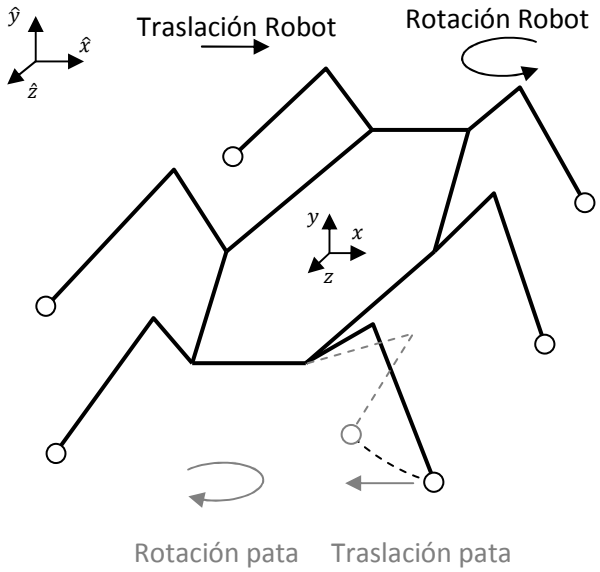


Figura 56: Desplazamiento de traslación y de rotación

Cada que el robot recorre un punto de la trayectoria se calcula la nueva posición de cada pata, esto se muestra en la Figura 57.



A cada pata se le aplica una traslación y una rotación en sentido contrario a la traslación y rotación que se desea en el robot. Esto último se hace siempre y cuando la pata este en contacto con el suelo y además no sea el último punto de contacto.

Figura 57: Cálculo de la nueva posición de cada pata para un desplazamiento de traslación y de rotación simultánea

Una vez que se han completado todos los puntos de contacto es momento de que la pata se eleve y vuelva a su posición original, este movimiento se realiza de forma lineal en el eje X y Z, y en el eje Y lo hace de manera senoidal. Este último movimiento tiene como puntos mínimos los

extremos de la línea de contacto y la altura del punto máximo está dada por la altura del paso previamente definida.

En la Figura 58 se muestra el método utilizado para calcular los puntos que deben de seguir cada una de las patas durante la marcha del robot.

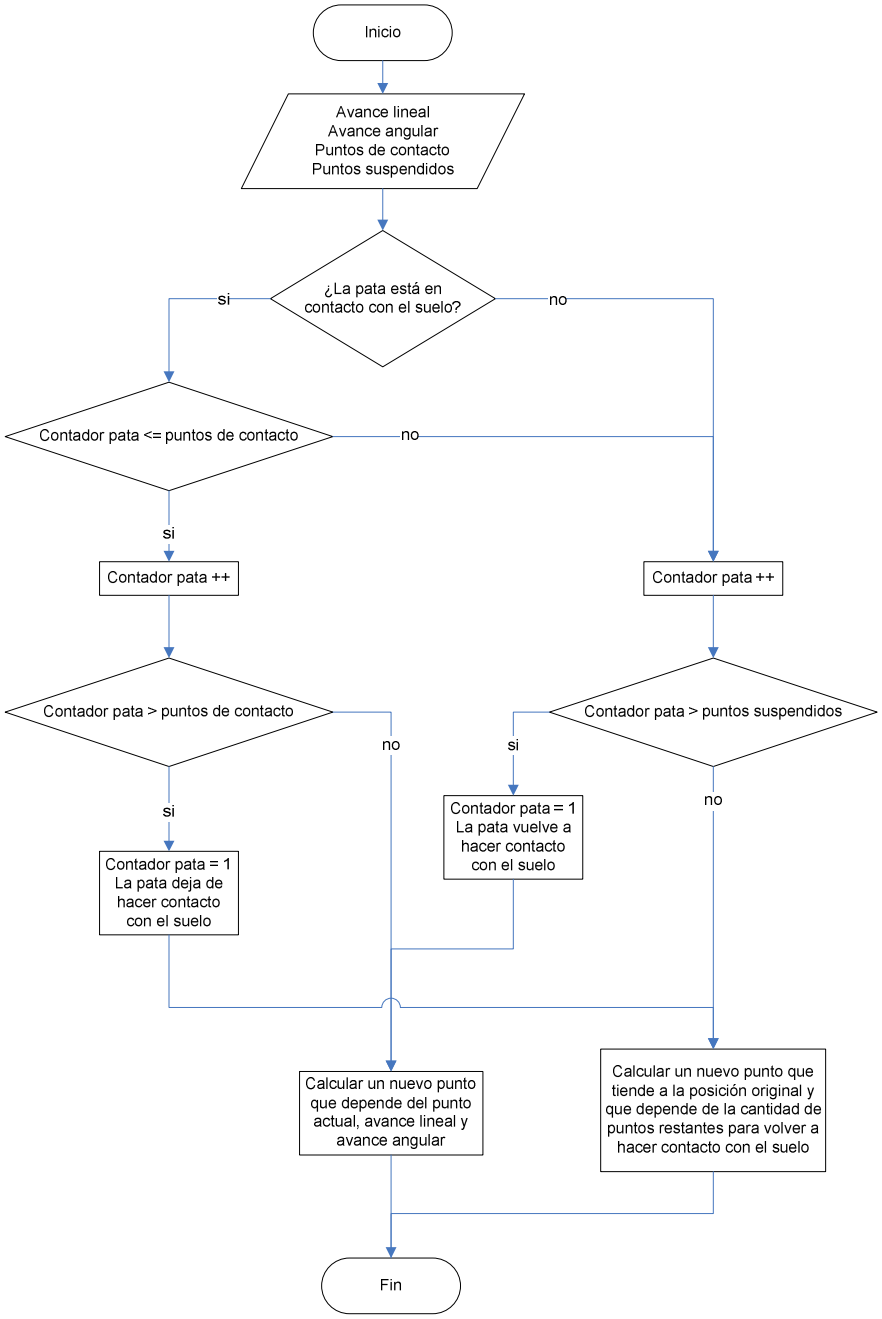


Figura 58 Método utilizado para calcular los puntos de la trayectoria de cada pata

Todo el control del robot está discretizado, lo que quiere decir es que sus movimientos se basan en la posición que tiene cada una de sus patas en un momento determinado. Para lograr un

movimiento aparentemente continuo lo que se hace es unir varios puntos y formar una trayectoria y dependiendo de la forma que tenga esta será el movimiento que el robot realizará. La precisión con que siga una trayectoria de un punto a otro dependerá de la cantidad de puntos que se tengan en dicha trayectoria.

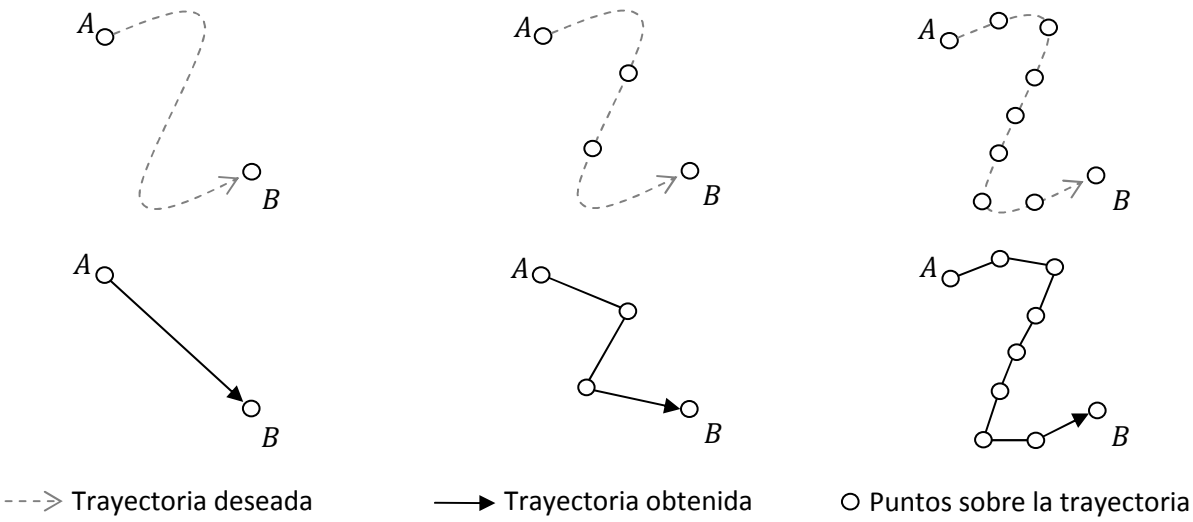


Figura 59: Diferencias ocasionadas por la cantidad de puntos intermedios en la trayectoria deseada sobre la trayectoria obtenida

Como se puede observar en la Figura 59, entre menos puntos intermedios en la trayectoria se tengan menor será la aproximación entre el movimiento real y el deseado.

6.7 Control del robot

Para mover el robot se utiliza un control de Xbox 360 conectado a la computadora. Dentro de las herramientas y librerías con que cuenta XNA se encuentran las encargadas de manejar dicho control por lo que se decidió utilizarlo.

Figura 60: Control de Xbox 360 utilizado para manejar el robot

Se diseñaron 3 modos de controlar el robot.

1. Dar pasos: en este modo el robot se traslada ya sea con una traslación o una rotación o simultáneas. Se controla la técnica de desplazamiento: 1, 2 y 3 patas por movimiento y también se puede cambiar la altura máxima que tiene la pata cuando se encuentra suspendida.

2. Traslaciones y rotaciones absolutas del cuerpo del robot: se controlan las traslaciones en el eje X y Z y las rotaciones en los 3 ejes con variables analógicas directamente del control. La traslación en el eje Y se maneja de forma incremental.

3. Traslaciones y rotaciones incrementales del cuerpo del robot: todas las traslaciones y rotaciones se controlan de forma incremental. La ventaja de esta última forma de controlar el robot sobre la anterior es que puede dejarse a este en una posición determinada para luego cambiar al modo “Dar pasos” manteniendo las traslaciones y rotaciones en el cuerpo.

Tanto el stick derecho como el izquierdo son de carácter analógico y proporcionan un vector de 2 dimensiones por lo que se puede determinar tanto dirección como magnitud.

LT y RT corresponden al gatillo izquierdo y gatillo derecho respectivamente. Son variables analógicas y devuelven un valor entre 0 y 1 con la precisión de un dato flotante.

El D-Pad es digital y sus funciones son: arriba, abajo, izquierda, derecha y combinadas.

Los otros botones son pulsadores.

Para cambiar entre los distintos modos se utiliza el D-Pad a la izquierda y derecha. A la derecha se incrementa el índice del modo y a la derecha se decrementa.

Modo 1: Dar pasos	
Mando	Controla…
Left Stick	Magnitud y dirección del avance lineal
Right Stick	La componente X controla la magnitud y signo del avance angular
RB	Aumenta la altura del paso
LR	Disminuye la altura del paso
RT	Controla la frecuencia con que se dan los pasos
D-Pad	Arriba y abajo cambia el número de patas por movimiento

Tabla 9: Funciones asociadas al control en el modo 1: "Dar pasos"

Modo 2: Traslaciones y rotaciones absolutas del cuerpo del robot	
Mando	Controla…
Left Stick	Componente X controla la magnitud y signo de la traslación X Componente Y controla la magnitud y signo de la traslación Z
Right Stick	Componente X controla la magnitud y signo de la rotación en Z Componente Y controla la magnitud y signo de la rotación en X
LT	La magnitud negativa de la rotación en Y
RT	La magnitud positiva de la rotación en Y
LB	Incrementos negativos de la traslación en Y
RB	Incrementos positivos de la traslación en Y

Tabla 10: Funciones asociadas al control en el modo 2: "Traslaciones y rotaciones absolutas del cuerpo del robot"

Modo 3: Traslaciones y rotaciones incrementales del cuerpo del robot	
Mando	Controla…
Left Stick	Componente X controla los incrementos de la traslación X Componente Y controla los incrementos de la traslación Z
Right Stick	Componente X controla los incrementos de la rotación en Z Componente Y controla los incrementos de la rotación en X
LT	Incrementos negativos de la rotación en Y
RT	Incrementos positivos de la rotación en Y
LB	Incrementos negativos de la traslación en Y
RB	Incrementos positivos de la traslación en Y

Tabla 11: Funciones asociadas al control en el modo 3: "Traslaciones y rotaciones incrementales del cuerpo del robot"

CAPÍTULO 7

Resultados

7.1 Resultados obtenidos en el simulador

Posición en reposo, sin traslaciones ni rotaciones del cuerpo

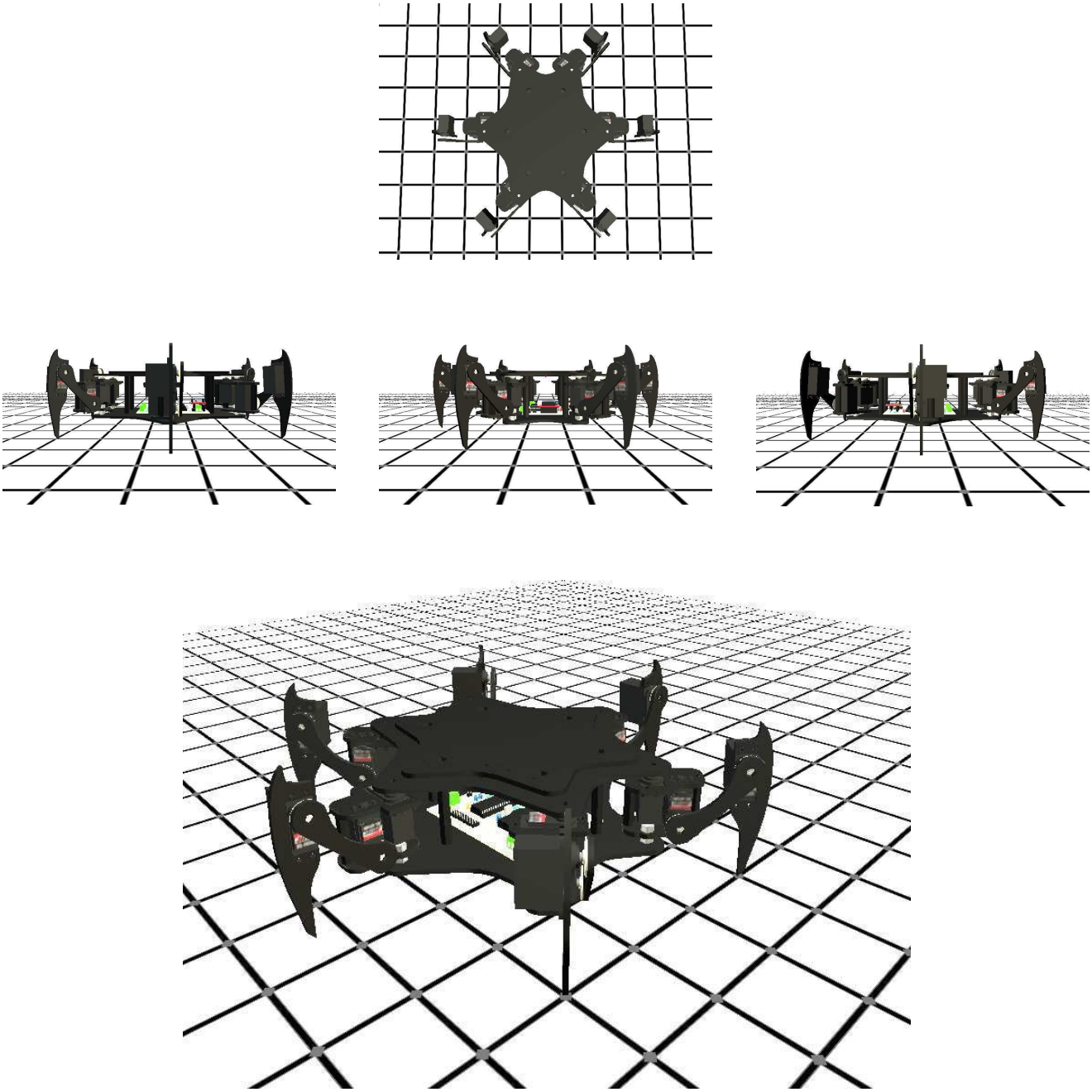


Figura 61: Imágenes tomadas desde diferentes vistas en el simulador y que muestran al robot en su posición de reposo

En la Figura 62 se muestra una imagen del robot con las transformaciones de la Tablas 12.

	Traslación [mm]	Rotación [°]
X	25.2	11.4
Y	19.8	15.3
Z	−8.1	−13.5

Tabla 12: Traslaciones y rotaciones utilizadas para mover el cuerpo del robot

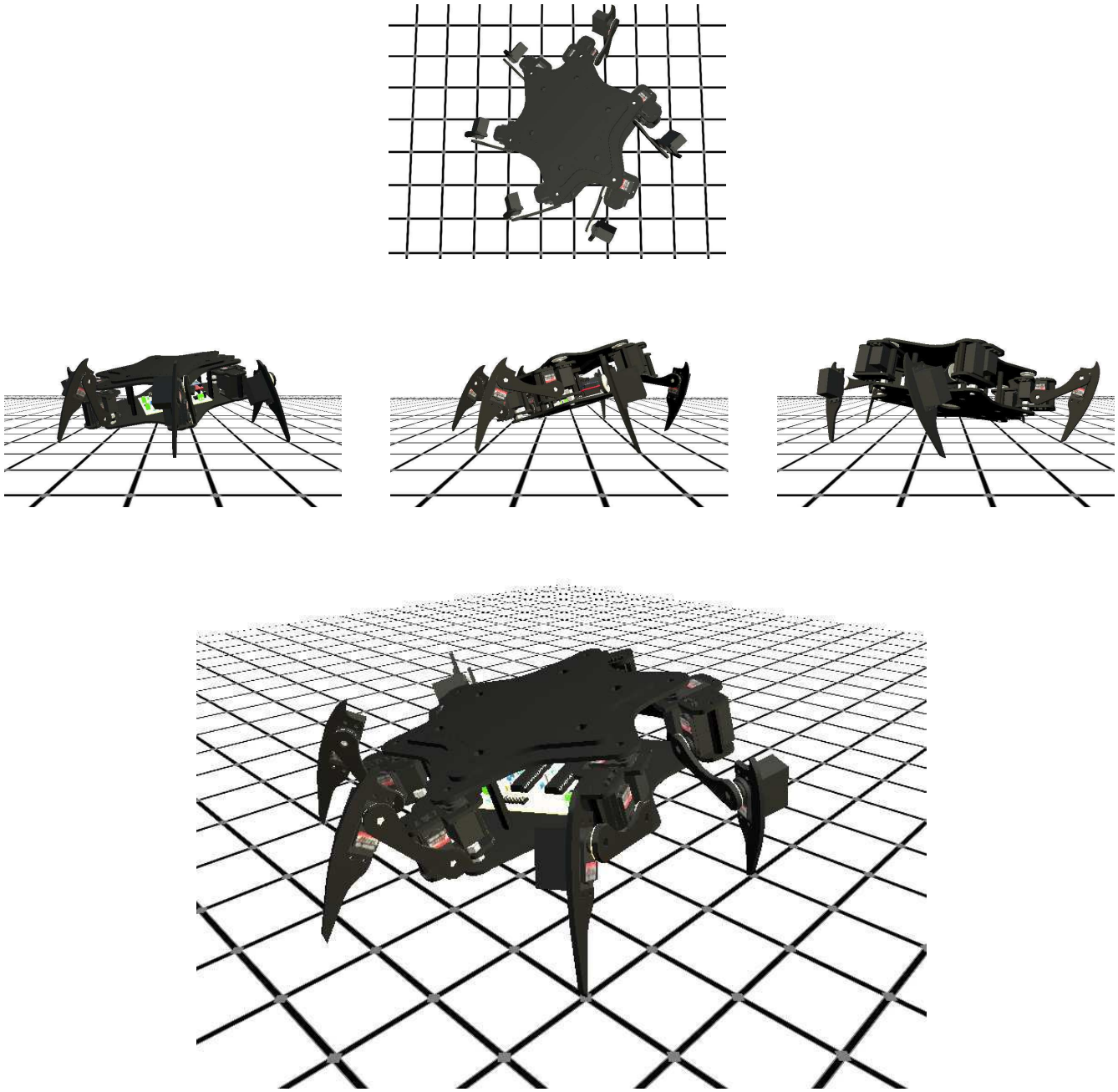


Figura 62: Imágenes tomadas desde diferentes vistas en el simulador que muestran los movimientos del cuerpo

7.2 Trazando un logotipo con movimientos del cuerpo

Para comprobar los movimientos que tiene el cuerpo, sin separar ninguna de sus patas del suelo, se diseñó un experimento que consistió en una serie de traslaciones previamente definidas con el fin de seguir un patrón determinado.

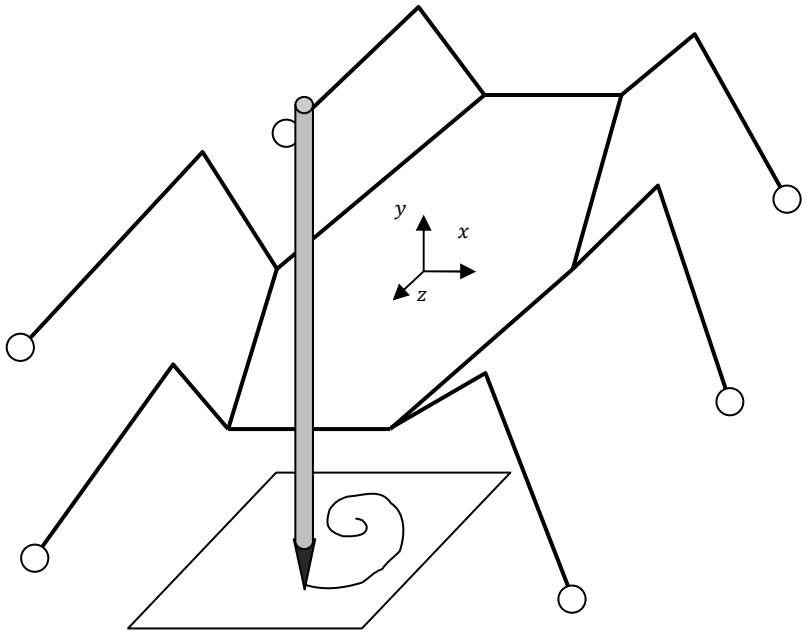


Figura 63: Diagrama que muestra el experimento diseñado para comprobar los movimientos del cuerpo

Para tener un registro de los movimientos realizados se colocó un plumón en uno de los extremos del robot (Figura 63) con el fin de que fuera trazando en una hoja de papel las traslaciones. Puesto que para este experimento sólo se utilizaron traslaciones la ubicación del plumón en el eje X y Z es arbitraria, la posición en el eje Y debe de ser la adecuada para que mantenga contacto con el papel mientras el robot se encuentra en posición de reposo.

Mientras el plumón se encuentra directamente sobre el papel se pueden comprobar las traslaciones tanto en el eje X como en el Z. Por otra parte, el eje Y se utiliza cuando se cambia de posición el plumón pero sin hacer trazo alguno sobre el papel.

El simulador puede ser fácilmente modificado para que realice animaciones y al modelo virtual también se le pueden agregar tantas partes como se quieran, para este experimento se le agregaron funciones para realizar un logotipo, al modelo se le agregó una nueva parte que es un plumón y tambien se hizo un gráfico tridimensional donde se indican los puntos en los que el plumón hace contacto con el suelo.

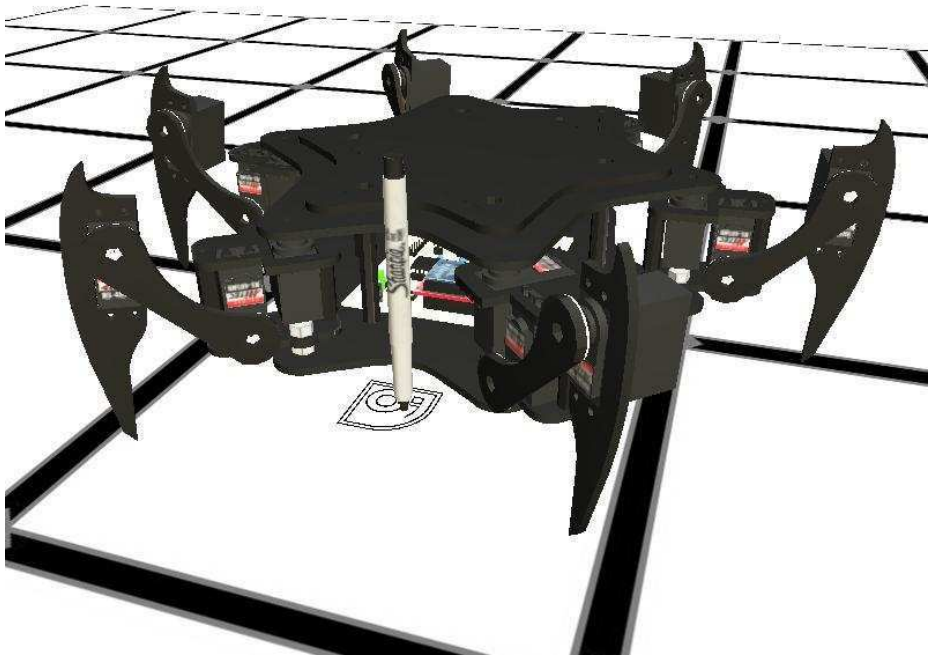


Figura 64: Imagen tomada del simulador que muestra al robot trazando el logotipo

Gracias a que el plumón se puede colocar de manera arbitraria en el cuerpo del robot, su instalación es muy simple. Se colocó un contrapeso sobre el plumón para que el contacto entre éste y el papel sea el adecuado y que sirve como un tope para que cuando el plumón se despegue del papel éste no se deslice (Figura 65).

Figura 65: Fotografía tomada del robot luego de trazar el logotipo

En la Figura 66 se muestra del lado izquierdo el patrón que se buscó reproducir y del lado derecho el resultado obtenido.

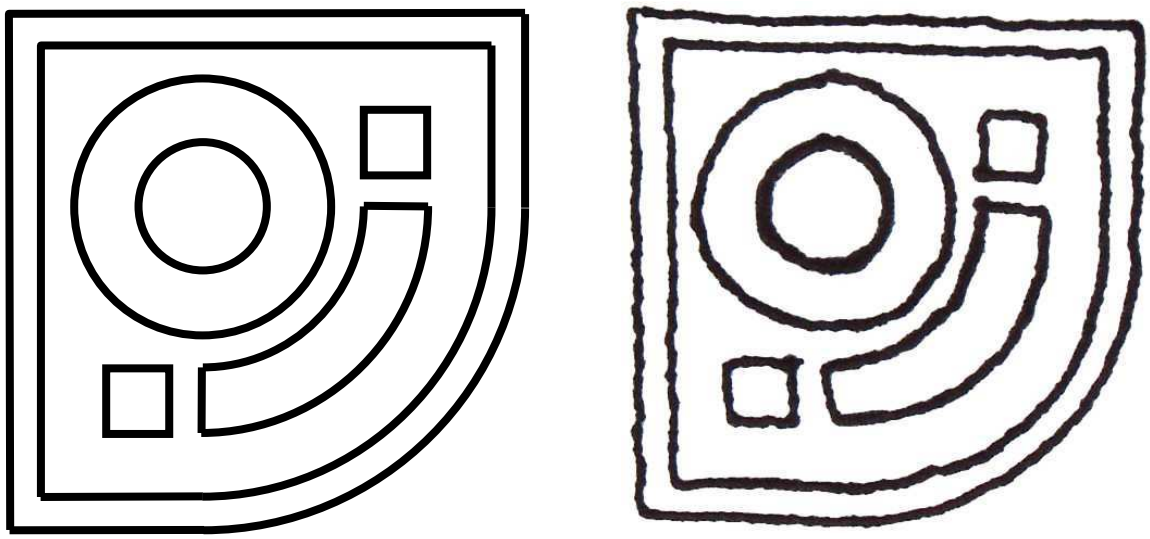


Figura 66: El logotipo que se quería obtener y el que se obtuvo

El resultado difiere del original y esto puede deberse a distintas razones:

- Fuerzas inerciales existentes en el robot las cuales no están consideradas en el simulador lo que provoca que los cambios de dirección no sean instantáneos como se espera.
- La precisión de las piezas y del ensamblado en general puede diferir del modelo simulado.
- La precisión y torque que tienen los servomotores, la vibración que estos generan puede apreciarse en la calidad de los trazos obtenidos.

En el apéndice A pueden observarse las 18 gráficas que corresponden a las orientaciones en los 18 servomotores durante el trazado del logotipo.

7.3 Movimientos omnidireccionales

En el experimento anterior se comprobaron los movimientos de traslación que tiene el cuerpo del robot sin despegar las patas del suelo. Sin embargo, el objetivo principal de este trabajo era demostrar el movimiento omnidireccional que se puede lograr con los tres grados de libertad en cada pata. Este movimiento no se limita únicamente a los movimientos del cuerpo sino que debe de ser logrado en los desplazamientos que el robot realiza.

Este movimiento omnidireccional se logró por completo tanto en el simulador como en el modelo físico el cuál se comprobó al controlar remotamente el robot.

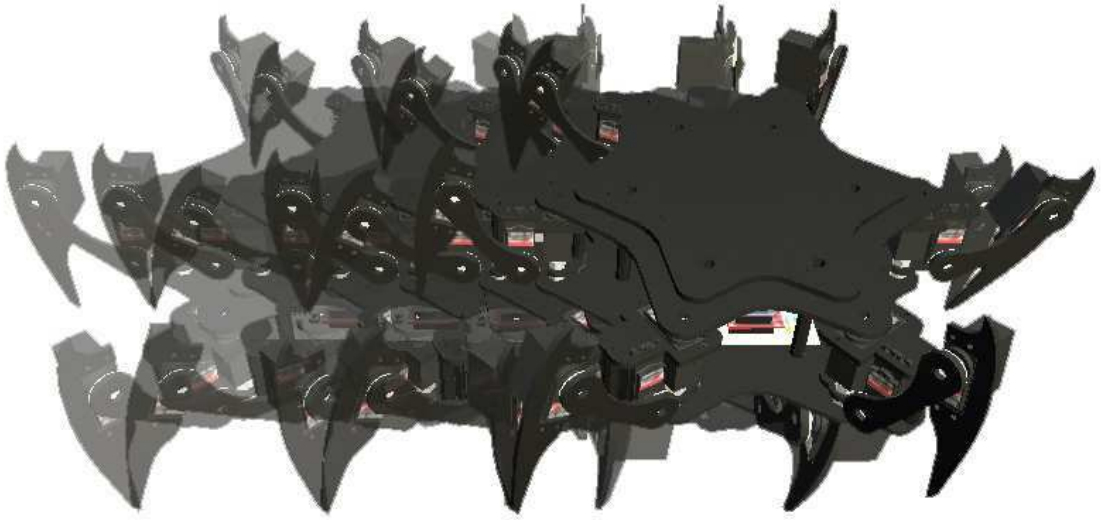


Figura 67 Movimiento de traslación en el eje X del robot

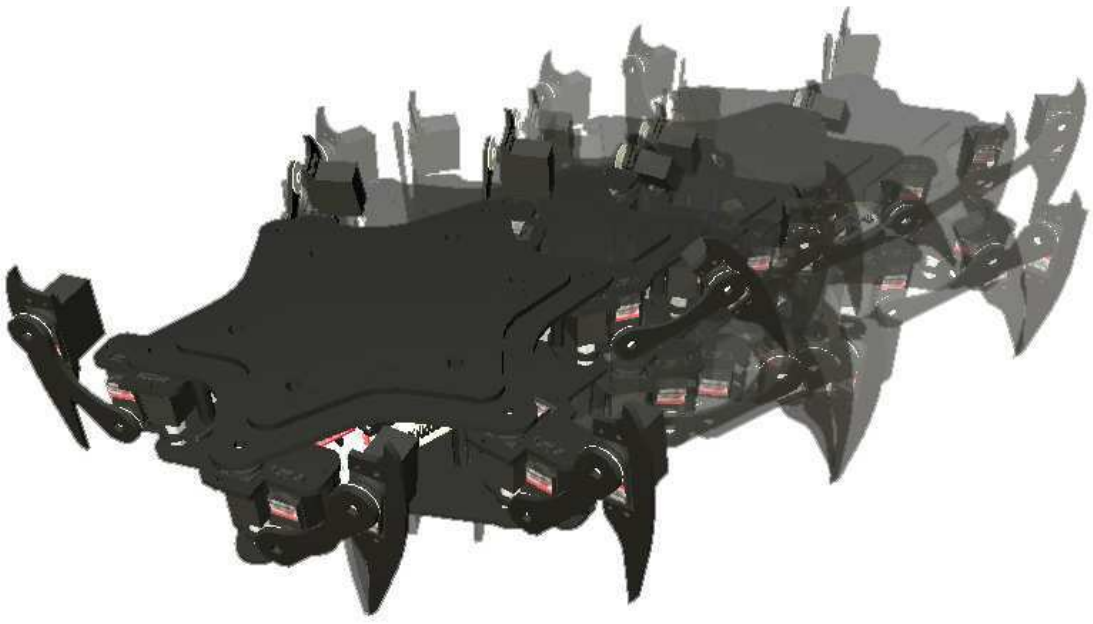


Figura 68 Movimiento de traslación en el eje Z del robot

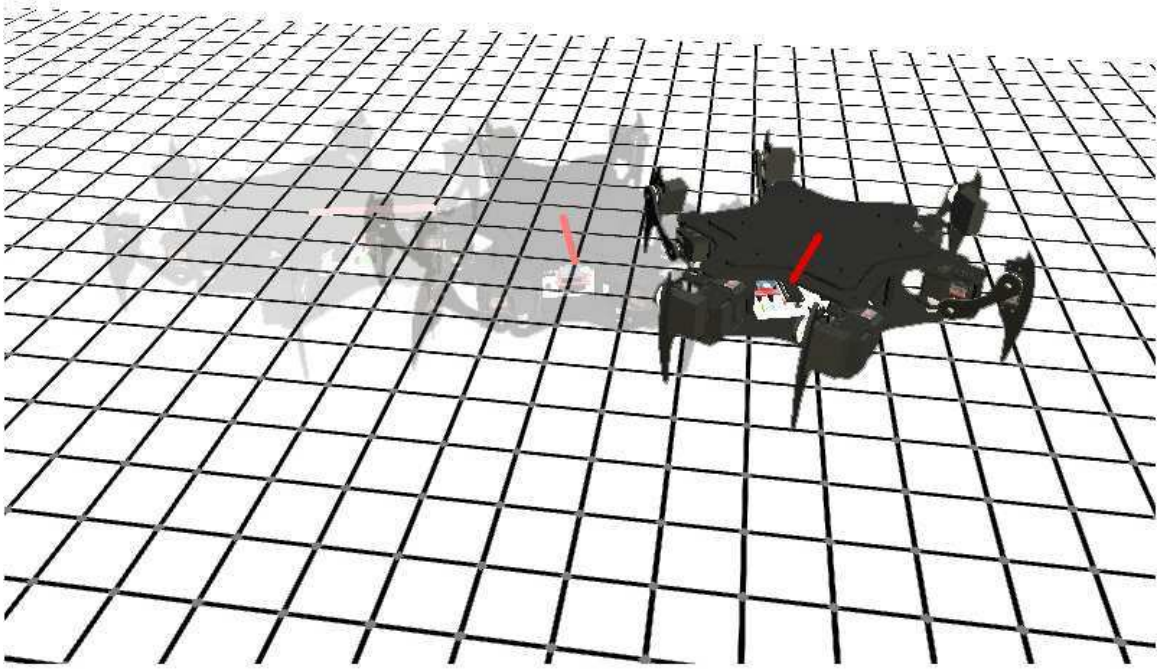


Figura 69 Movimiento de traslación y rotación simultánea

CAPÍTULO 8

Conclusiones

Se logró simular y demostrar físicamente de manera satisfactoria que es posible tener un movimiento omnidireccional en un robot hexápodo con tres grados de libertad en cada pata.

Dentro de la robótica móvil, los robots caminantes como el presentado en este trabajo tienen grandes ventajas sobre los robots impulsados por ruedas u orugas, sin embargo conllevan grandes dificultades. La mayor de todas es la complejidad que trae consigo su control y construcción. Es importante que todas sus piezas estén fabricadas y ensambladas de manera precisa con el fin de que trabajen en conjunto para lograr un mismo fin ya que de lo contrario, con las mínimas imperfecciones existirán fuerzas que se opongan entre sí provocando un mal funcionamiento del robot. El controlador, en este caso el simulador, puede compensar muchas de las imperfecciones para que el resultado final sea el deseado. Sin embargo al no tener una realimentación por parte del robot es necesario conocer a fondo estas imperfecciones para realizar el ajuste manualmente. Se ha demostrado que la principal ventaja que este tipo de robot tiene frente a sus contrapartes ya sean con ruedas u orugas es la facilidad para desplazarse por terrenos no estructurados, por lo que son una buena opción para labores de búsqueda y rescate, exploración, etc.

El robot presentado en este trabajo cuenta con 18 servomotores, sin embargo el torque que se tiene en ellos es insuficiente para lograr movimientos con facilidad y por esta razón fue que la batería se tuvo que colocar fuera del cuerpo del robot. Los principales problemas que ocasiona la falta de torque en los motores además de la ya mencionada es que se sobre exigen y causan mayores vibraciones de las deseadas y por ello es probable que disminuyan su vida útil. Se probaron 3 tipos de desplazamiento: una, dos y tres patas por movimiento. En la primera sólo una pata se levanta del suelo en cada etapa, en la segunda dos y en la tercera tres. La técnica que mejor se llevó a cabo fue cuando se tienen en todo momento cuando menos cinco puntos de contacto en el suelo, que corresponde a “una pata por movimiento” lo cual era de esperarse debido a la falta de torque en los servomotores. Todos estos problemas son fácilmente solucionados o cuando menos reducidos con la elección de servomotores de mayor torque con el mismo tamaño que los utilizados o en su defecto hacer un rediseño que pueda albergar a los nuevos motores; sin embargo esta solución no es tan trivial como parece debido a la inversión que esto conlleva, es importante hacer una evaluación en cuanto al costo y a las especificaciones de los servomotores para decidir cuál es la opción más conveniente y sensata.

En cuanto a la construcción, todas las piezas se realizaron en acrílico de 4 *mm* y se cortaron con láser. Se decidió utilizar acrílico debido a la facilidad con que este puede ser trabajado. Otra opción que se tuvo para realizar las piezas fue en aluminio ya que tiene características deseables para este tipo de aplicaciones pero debido a que existen muy pocos lugares en donde cortan aluminio con láser en México y que el corte por chorro de agua es muy costoso se optó por el acrílico. Al utilizar máquinas de control numérico para realizar los cortes se garantiza que todas las piezas tengan las mismas dimensiones reduciendo con ello las imperfecciones al momento de ensamblar todas las partes. El corte láser de acrílico es una técnica bastante rápida, limpia y precisa lo que es ideal para realizar modelos y prototipos a bajo costo.

CONCLUSIONES

El modelo virtual que se diseñó para luego ser utilizado en el simulador se realizó en un programa que fue desarrollado principalmente para hacer animaciones y videojuegos, incluso existen películas que se han hecho en este programa. El simulador, donde se controla el modelo virtual se realizó en C# con XNA, este último es un conjunto de herramientas y librerías para C# y que fue desarrollado para realizar videojuegos. La unión de un modelo virtual con un programa como C# nos da una valiosa herramienta con la cual es posible evaluar tantos parámetros como nosotros podamos programar antes de siquiera construir una pieza. En este caso toda la simulación que se hizo fue cinemática, sin embargo al ser un programa con lenguaje de alto nivel las posibilidades de simulación son infinitas. Tal vez una de las mayores cualidades de estos dos programas utilizados para completar el simulador es que son libres, lo que permite que cualquier persona de forma legal y gratuita pueda diseñar y simular un sistema tan complejo como pueda. Existen programas muy especializados que muy pocas personas pueden pagar sus costosas licencias lo que limita, de forma legal, el uso para el público en general.

Trabajo futuro

- Realimentación tanto de la posición de los servomotores como del contacto con el suelo para saber si se está alcanzando el punto deseado. También se pude equipar con diferentes tipos de sensores con los cuales se determine el estado actual del robot, inclinación, aceleración, etc. Sensores ultrasónicos o infrarrojos para detectar obstáculos en su entorno y con ello hacer la toma de decisiones.
- Se utilizó sólo un microcontrolador para manejar los 18 servomotores con lo que se deja casi por completo el otro microcontrolador para la recolección y análisis de información.
- Equipar al robot con motores de mayor torque que le permitan realizar sus movimientos con mayor facilidad y también para que pueda cargar una batería y sea autosuficiente.
- Cargar toda la cinemática inversa necesaria para su funcionamiento en un microcontrolador para que con ello se pueda prescindir del uso de la computadora.
- En caso del simulador, aunque están restringidos los movimientos de los servomotores a su rango de operación, el simulador no detecta las colisiones que puedan existir entre las mismas partes del robot.
- Realizar un programa de control que no solamente contemple el modelo cinemático como se hizo en este trabajo sino que incorpore al mismo un modelo dinámico que mejore el control del robot.
- Y por último, diseñar un programa que sea completamente autónomo en cuanto a las decisiones y la ejecución de las mismas, claro que para que esto se pueda llevar a cabo es necesario que el robot tenga una misión asignada.

Gráficas de la orientación en los servomotores para el trazado del logotipo

Tanto el inicio como el final de la rutina están representados en la siguiente imagen por una “X”.

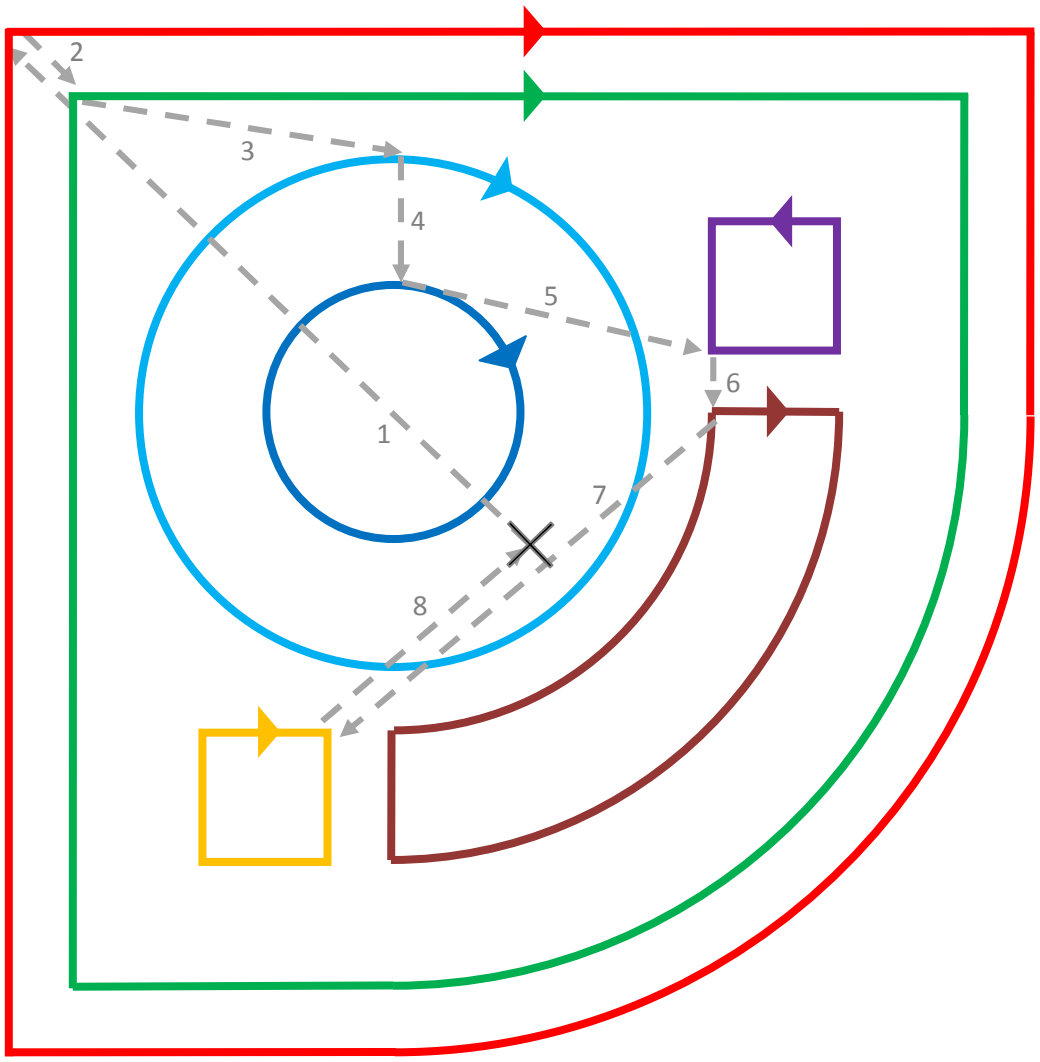
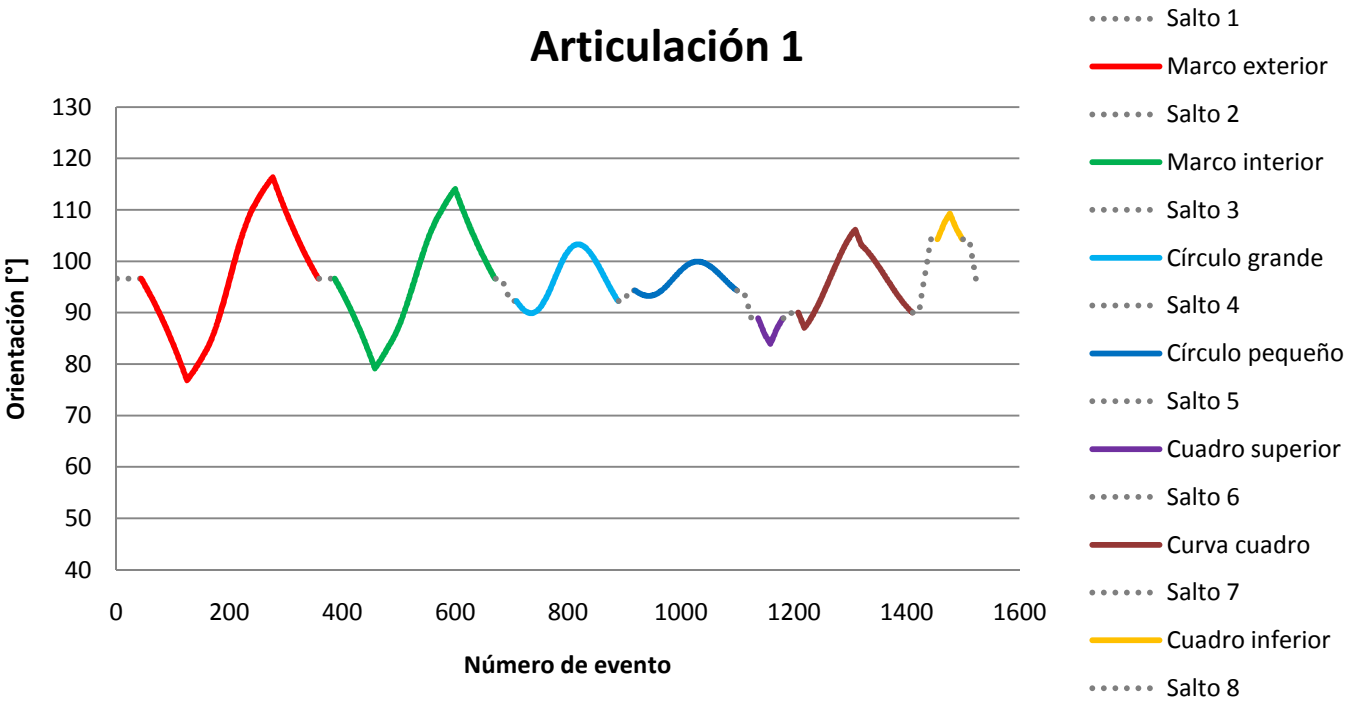


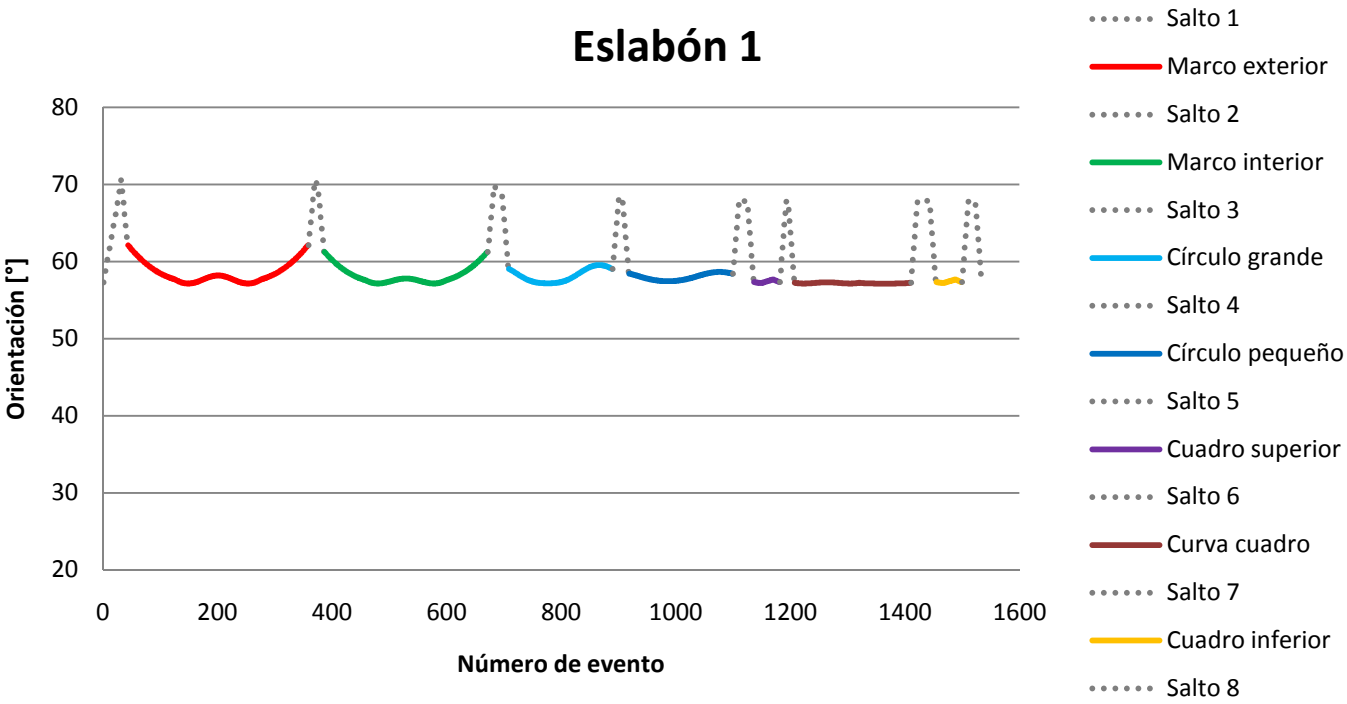
Figura 70: Diferentes trayectorias dentro del logotipo

Las líneas punteadas (- - →) representan los “saltos” que ejecuta el robot, mientras los cuales el plumón se separa del papel.

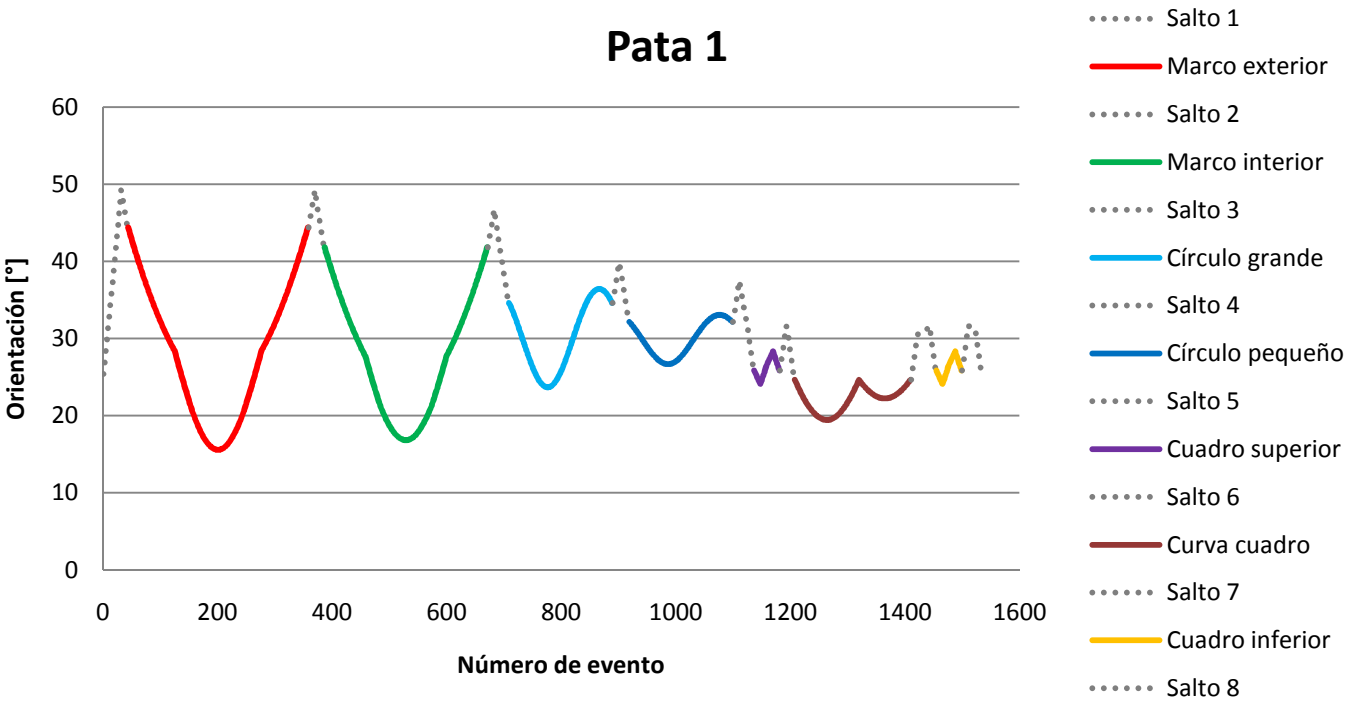
De la Gráfica 1 a la Gráfica 18 se muestran las orientaciones en los servomotores a lo largo de toda la rutina. En el eje X se muestra el número del evento. Al estar toda la rutina discretizada, cada evento representa un arreglo de orientaciones en los 18 servomotores logrando en conjunto una posición determinada en el robot. La duración completa de la rutina depende directamente de la frecuencia con que se cambie de evento.



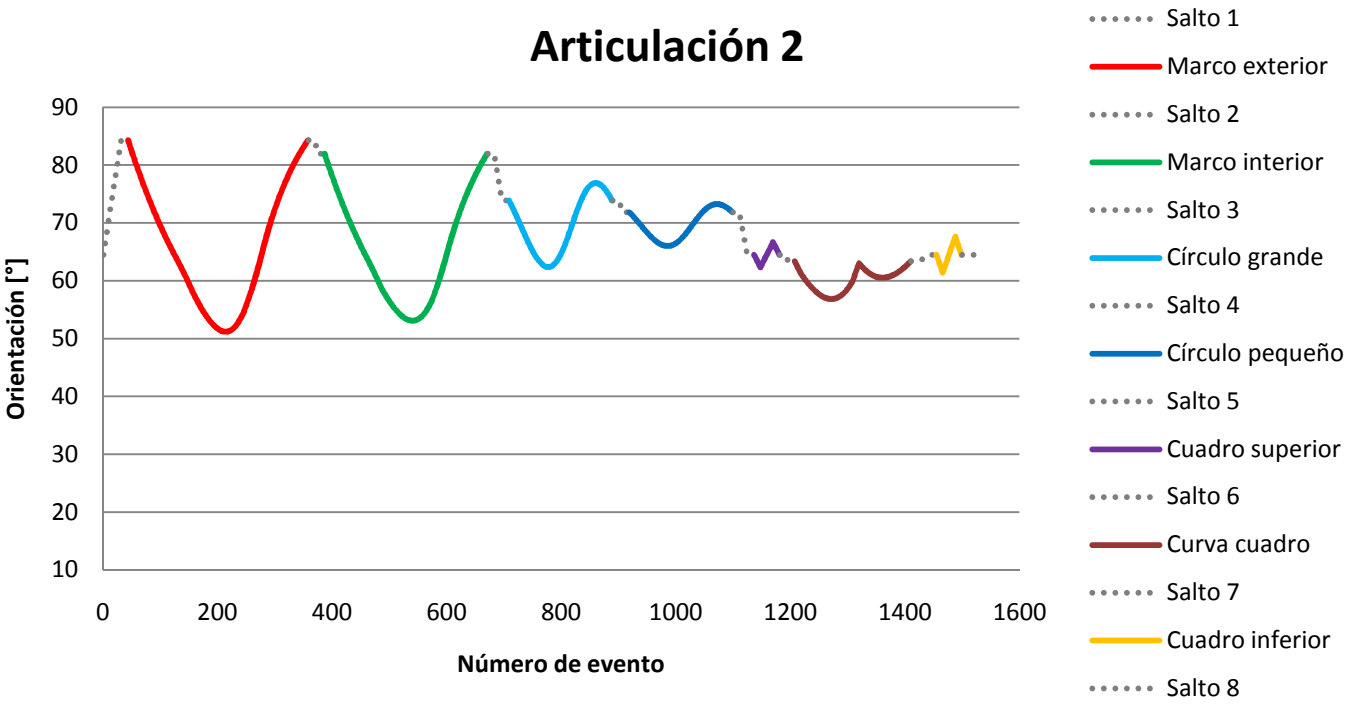
Gráfica 1: Orientación del servomotor 1 (Articulación 1) durante todo el trazado del logotipo



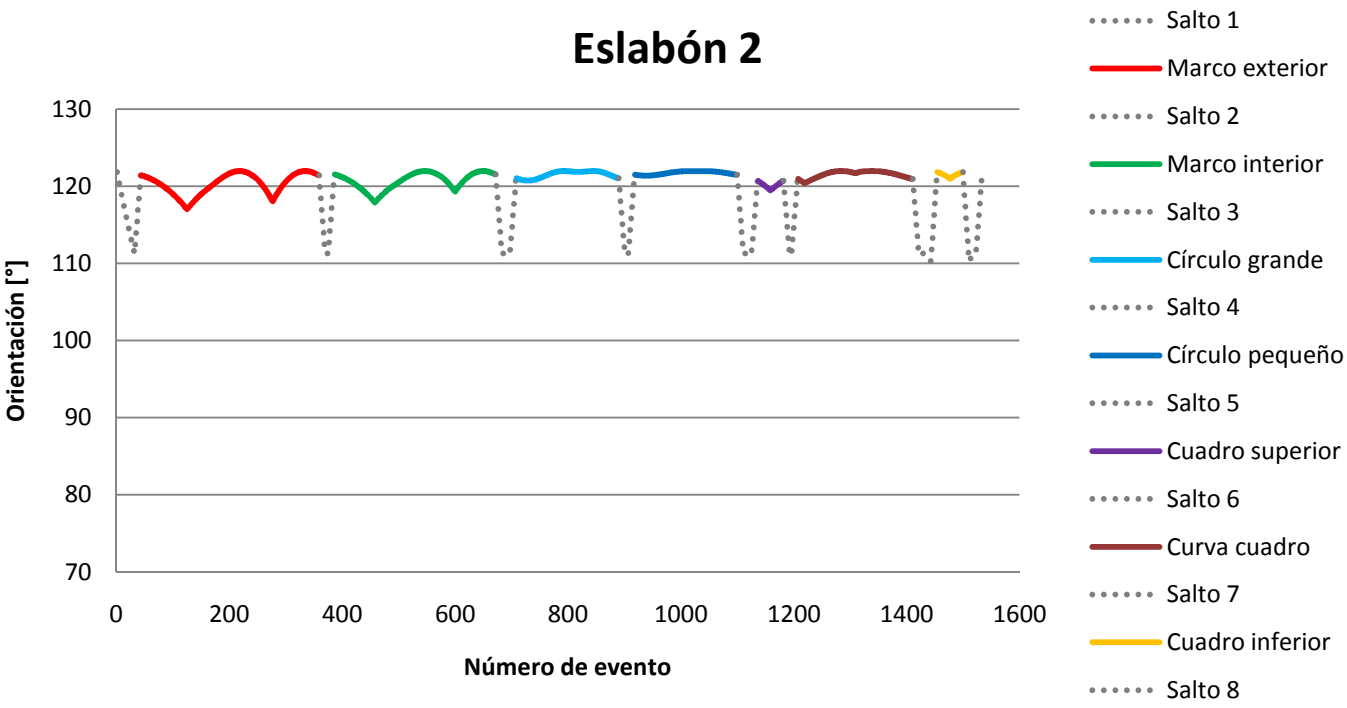
Gráfica 2: Orientación del servomotor 2 (Eslabón 1) durante todo el trazado del logotipo



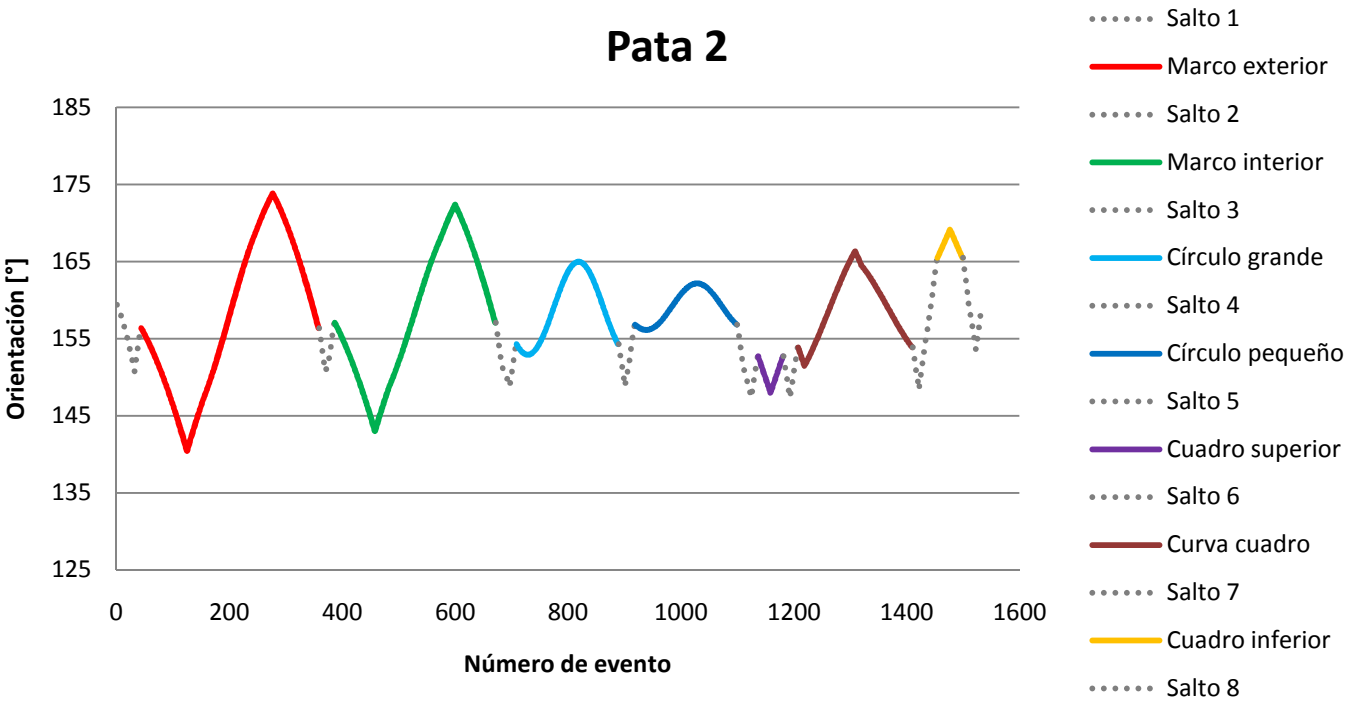
Gráfica 3: Orientación del servomotor 3 (Pata 1) durante todo el trazado del logotipo



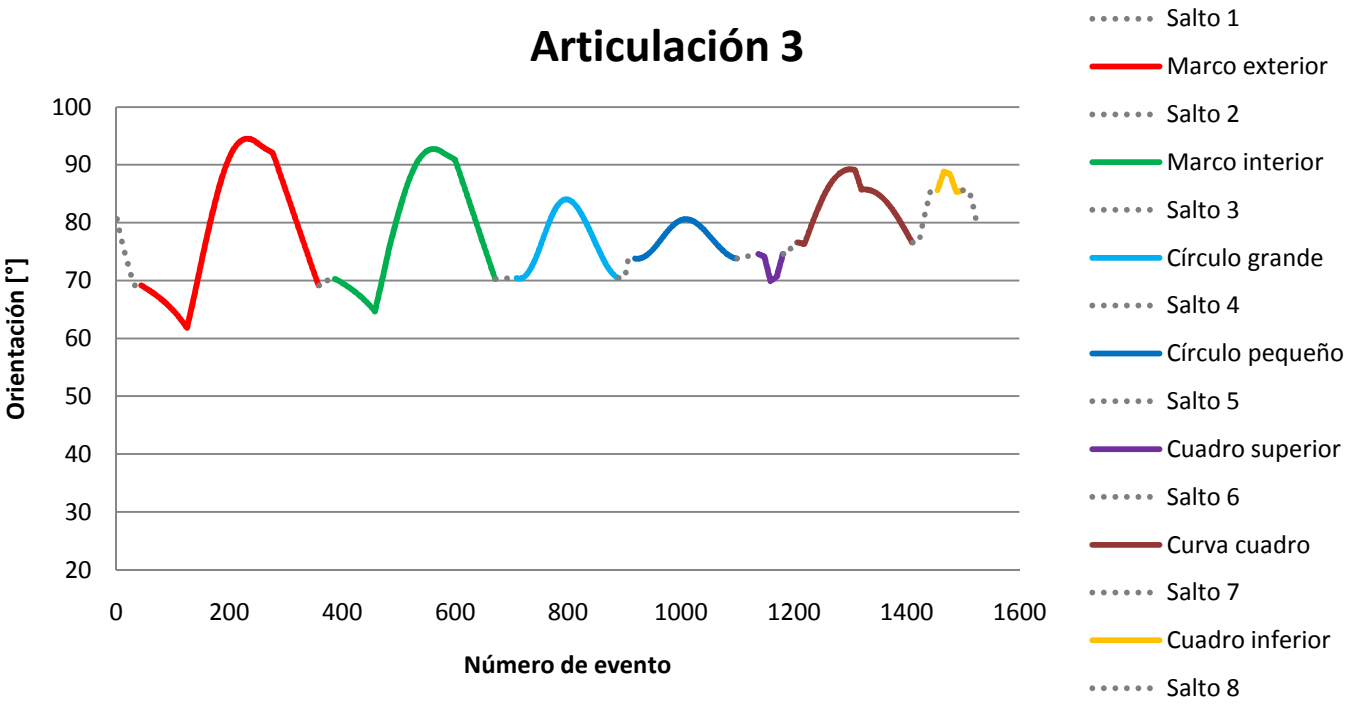
Gráfica 4: Orientación del servomotor 4 (Articulación 2) durante todo el trazado del logotipo



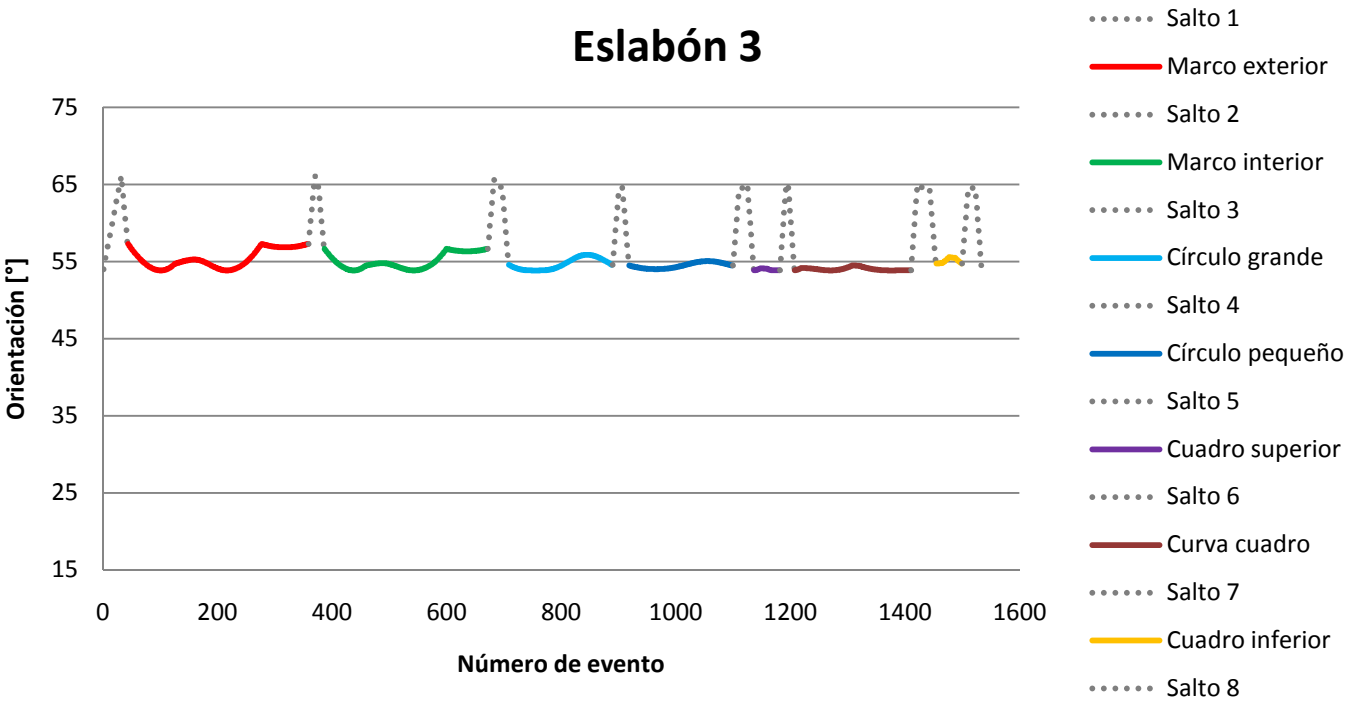
Gráfica 5: Orientación del servomotor 5 (Eslabón 2) durante todo el trazado del logotipo



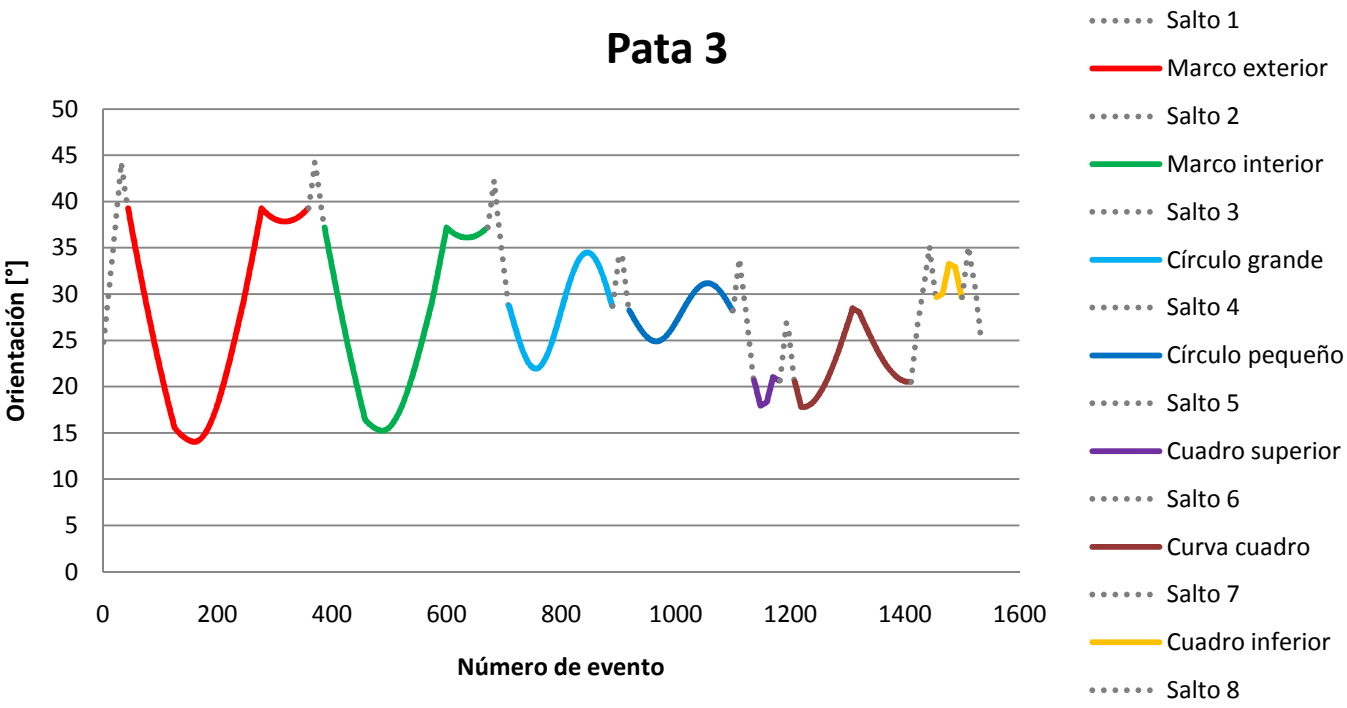
Gráfica 6: Orientación del servomotor 6 (Pata 2) durante todo el trazado del logotipo



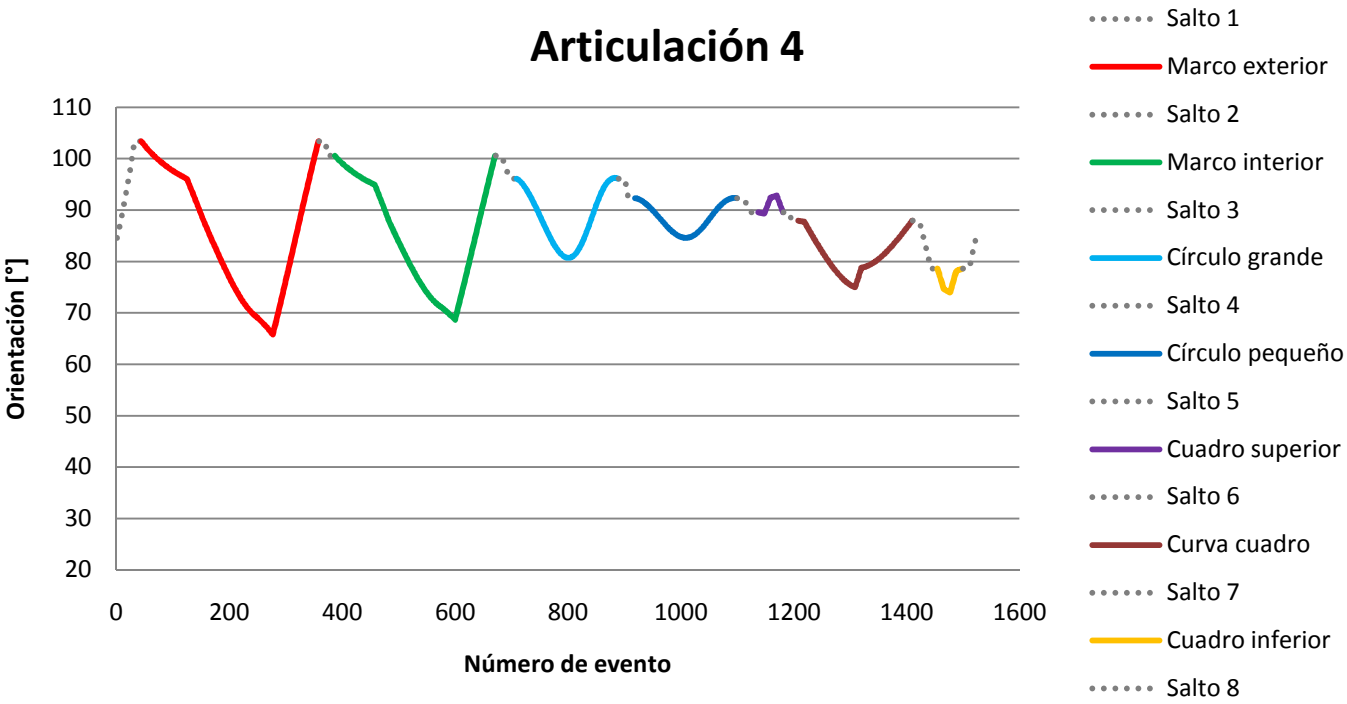
Gráfica 7: Orientación del servomotor 7 (Articulación 3) durante todo el trazado del logotipo



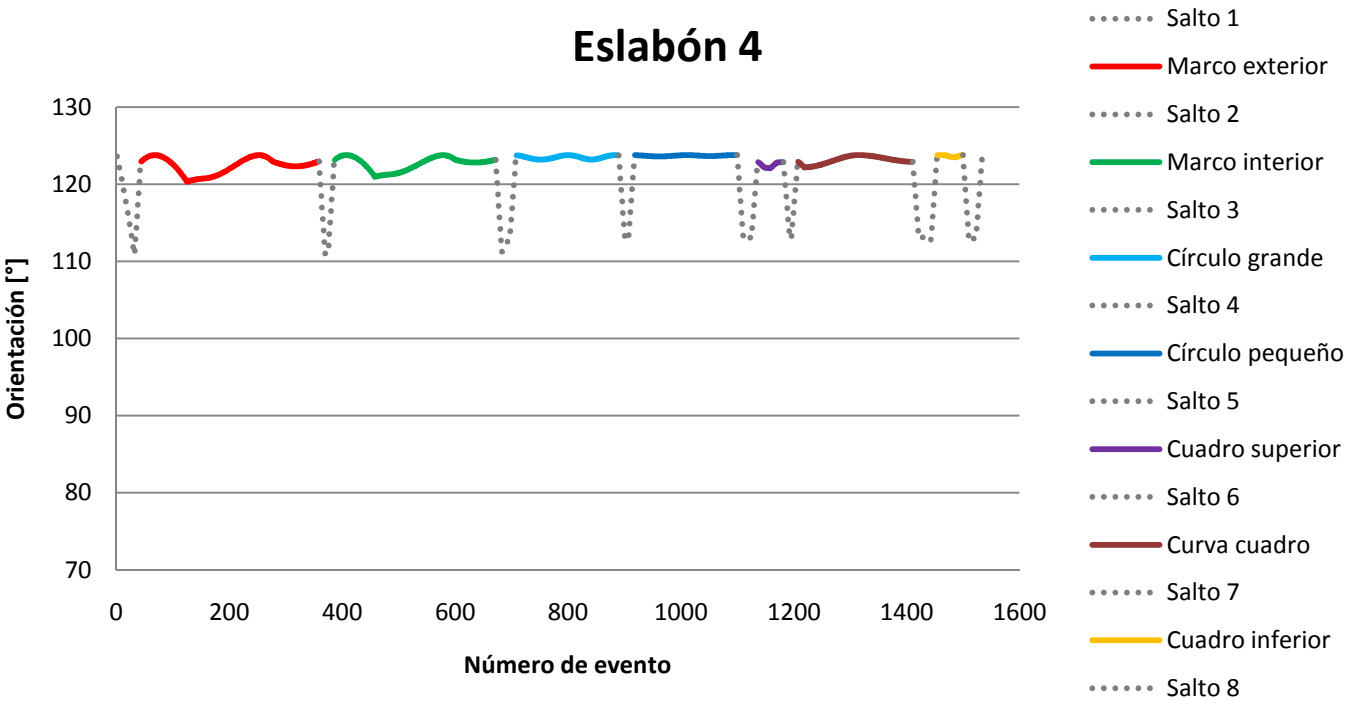
Gráfica 8: Orientación del servomotor 8 (Eslabón 3) durante todo el trazado del logotipo



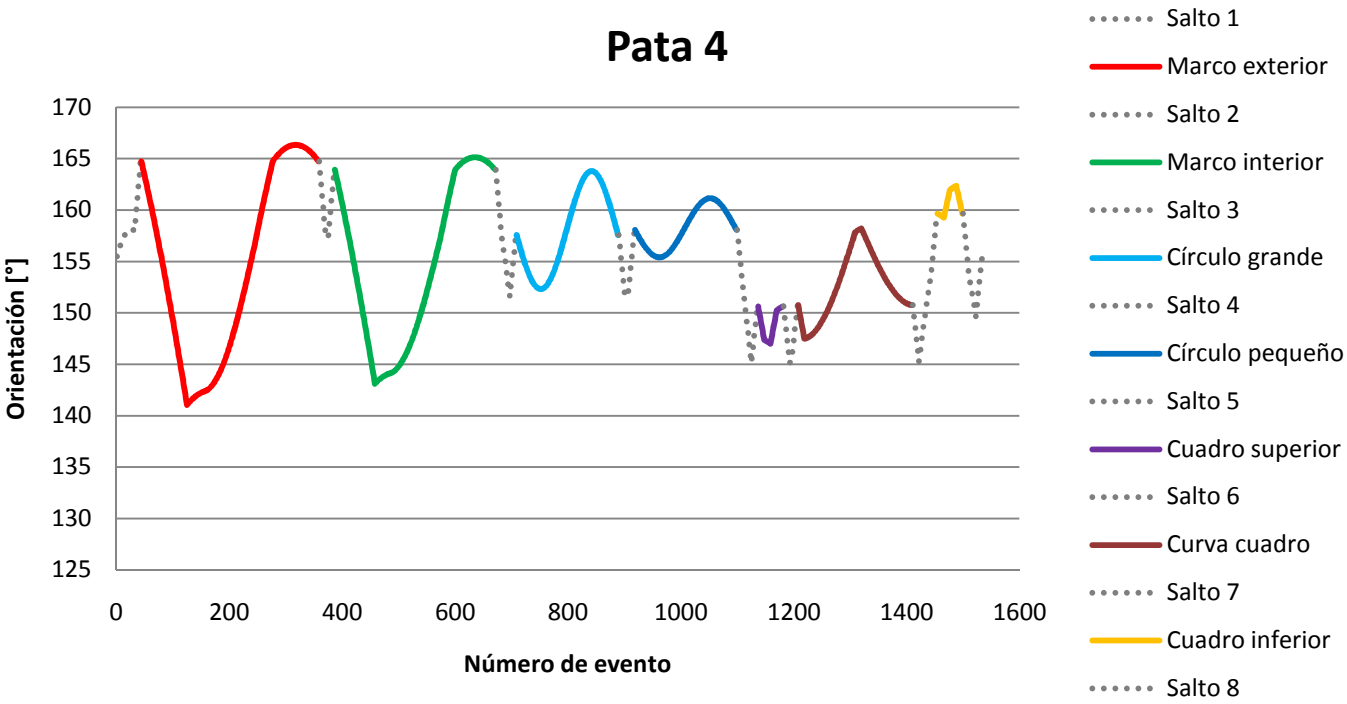
Gráfica 9: Orientación del servomotor 9 (Pata 3) durante todo el trazado del logotipo



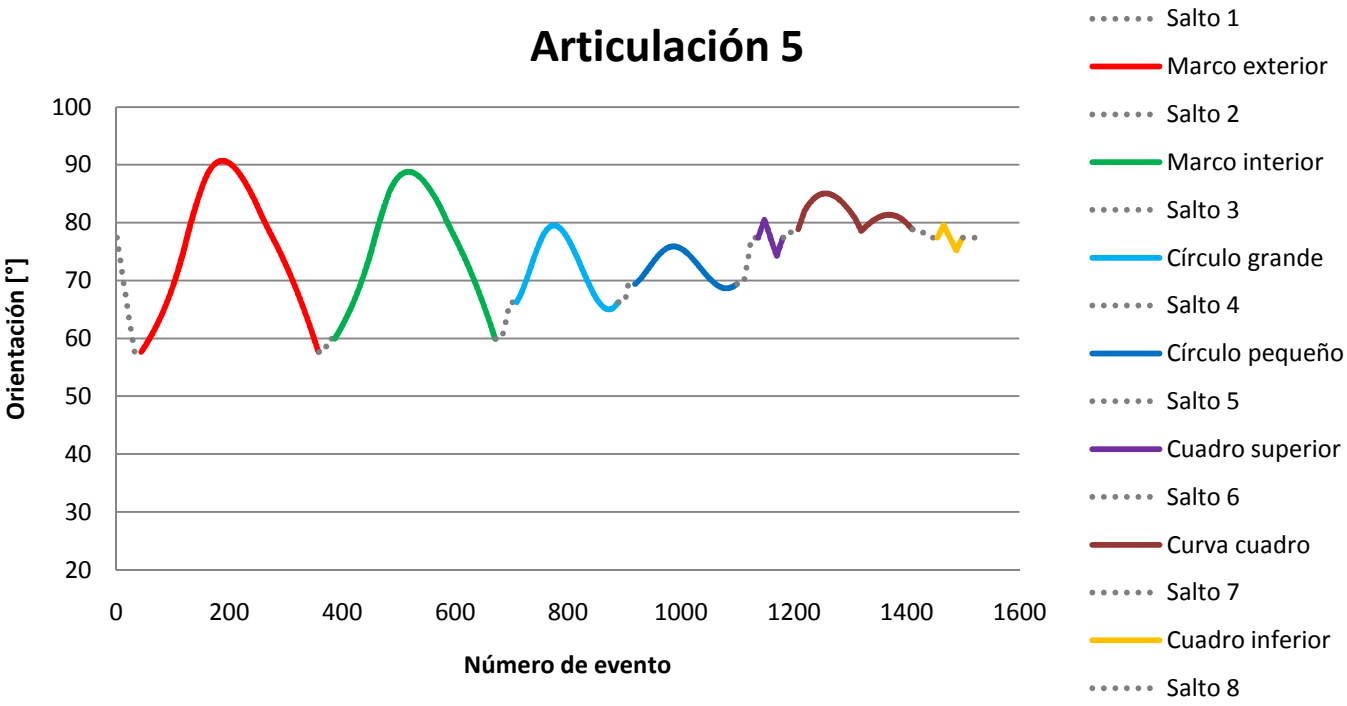
Gráfica 10: Orientación del servomotor 10 (Articulación 4) durante todo el trazado del logotipo



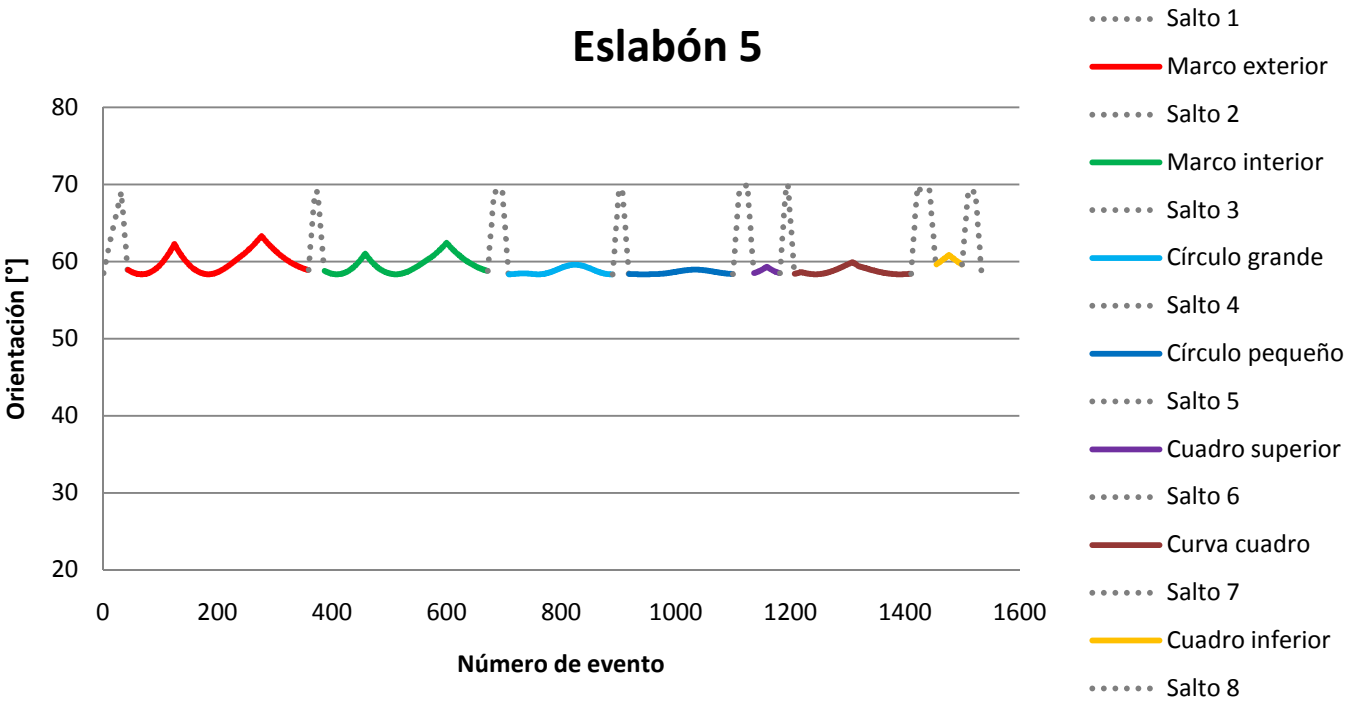
Gráfica 11: Orientación del servomotor 11 (Eslabón 4) durante todo el trazado del logotipo



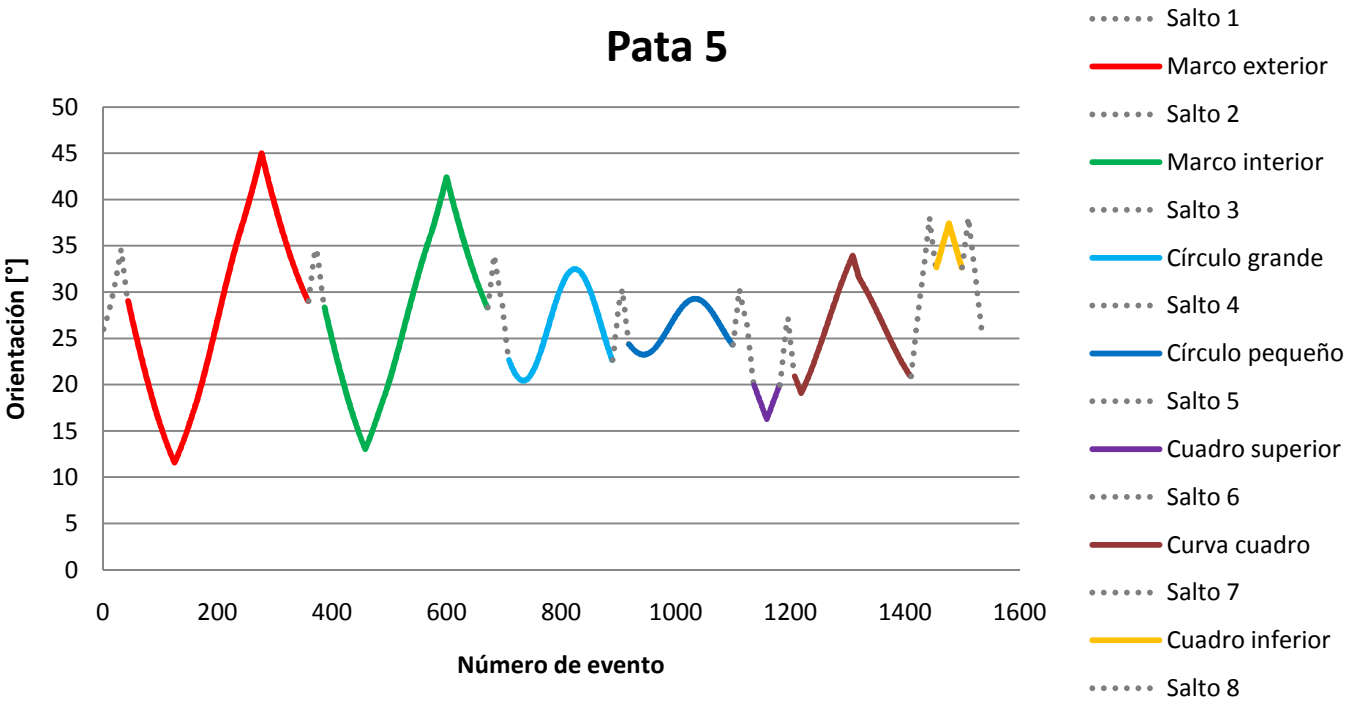
Gráfica 12: Orientación del servomotor 12 (Pata 4) durante todo el trazado del logotipo



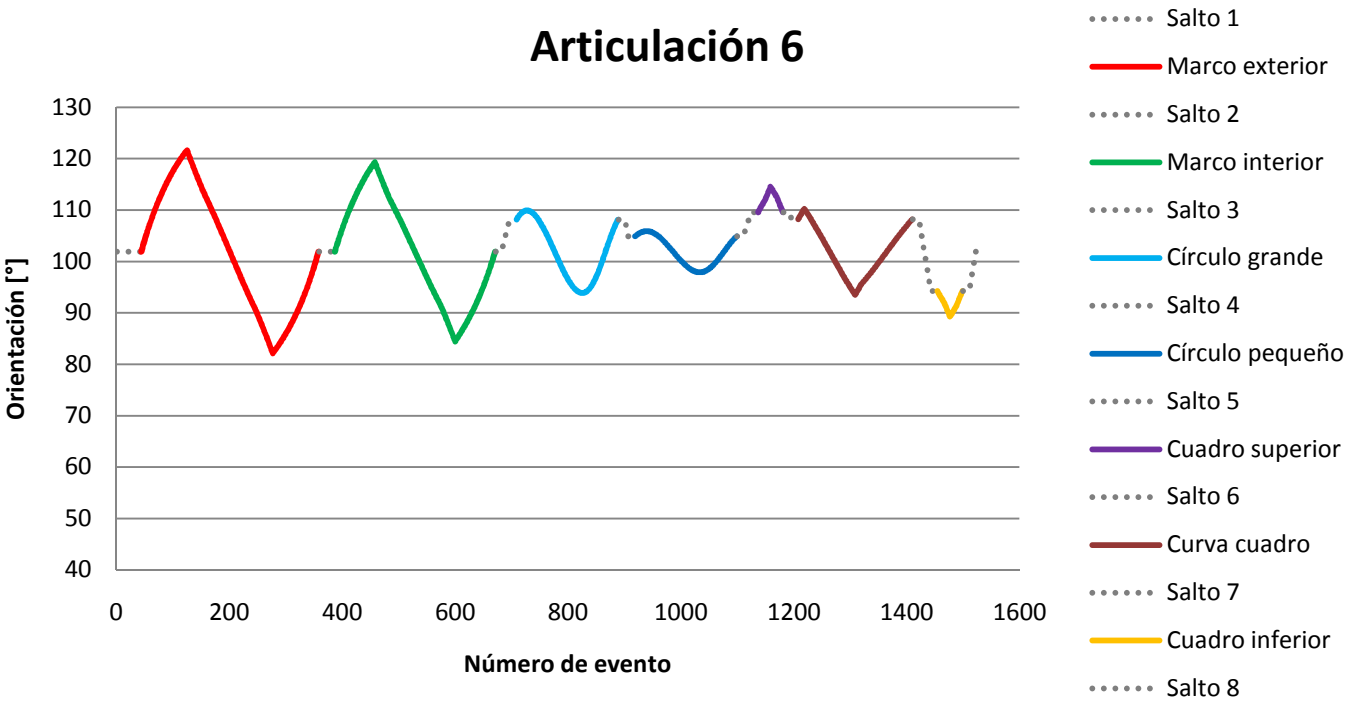
Gráfica 13: Orientación del servomotor 13 (Articulación 5) durante todo el trazado del logotipo



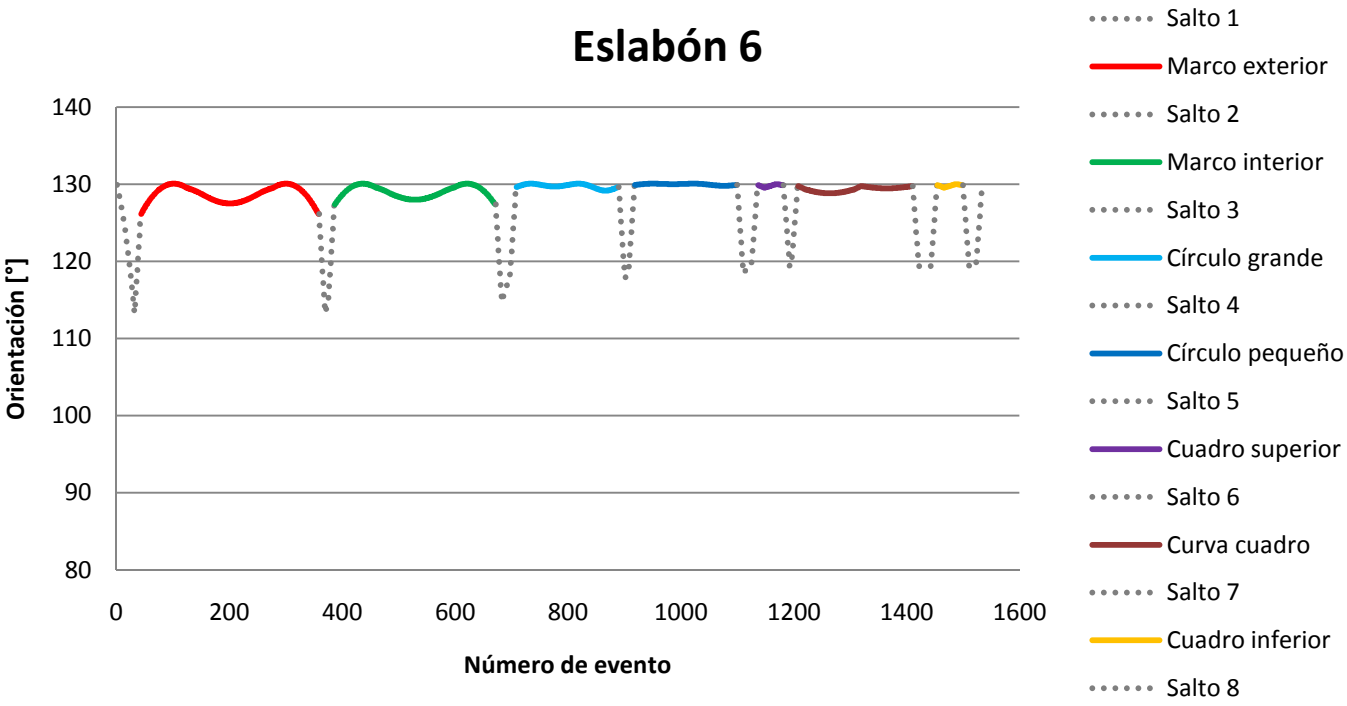
Gráfica 14: Orientación del servomotor 14 (Eslabón 5) durante todo el trazado del logotipo



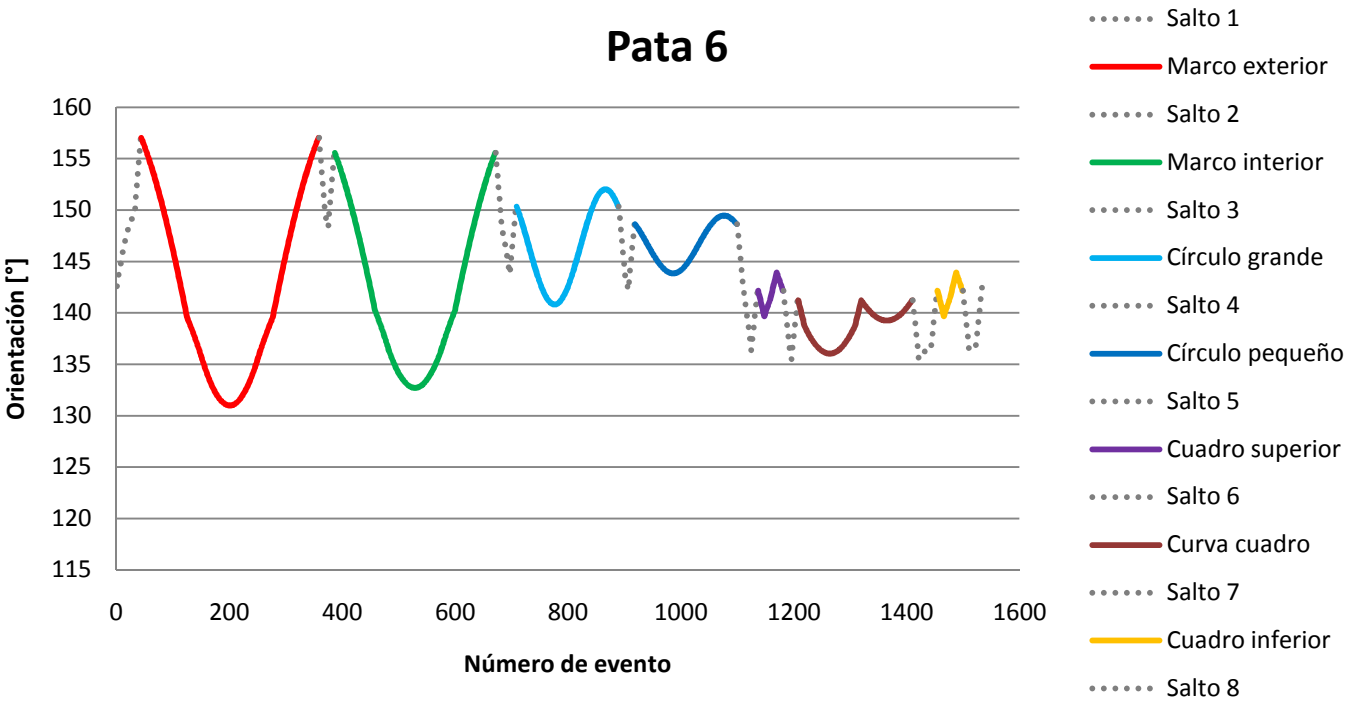
Gráfica 15: Orientación del servomotor 15 (Pata 5) durante todo el trazado del logotipo



Gráfica 16: Orientación del servomotor 16 (Articulación 6) durante todo el trazado del logotipo



Gráfica 17: Orientación del servomotor 17 (Eslabón 6) durante todo el trazado del logotipo



Gráfica 18: Orientación del servomotor 18 (Pata 6) durante todo el trazado del logotipo

Programas del microcontrolador maestro y del esclavo

Programa utilizado por el microcontrolador trabajando como Maestro y que sirve como intermediario entre el módulo Xbee y el microcontrolador Esclavo.

```
#include <18F452.h>
//cristal de 10MHz lo multiplica x4 ->
40MHz
#fuses H4,NOWDT,NOPROTECT
#use delay(clock=40000000)

//Configuración comunicación RS232 a
115200 baudios
#use
rs232(baud=115200,xmit=PIN_C6,rcv=PIN_C7,b
its=8,parity=N)

//configuración de la comunicación i2c a
100 kHz
#use
i2c(master,sda=pin_c4,scl=pin_c3,slow)

#use standard_io(c)
#use standard_io(b)
#use standard_io(d)

//variables para el Servo1
int valorServo1h=70;
int valorServo1l=80;

//variables para el Servo2
int valorServo2h=70;
int valorServo2l=80;

//variables para el Servo3
int valorServo3h=70;
int valorServo3l=80;

//variables para el Servo4
int valorServo4h=70;
int valorServo4l=80;

//variables para el Servo5
int valorServo5h=70;
int valorServo5l=80;

//variables para el Servo6
int valorServo6h=70;
int valorServo6l=80;

//variables para el Servo7
int valorServo7h=70;
int valorServo7l=80;

//variables para el Servo8
int valorServo8h=70;
int valorServo8l=80;

//variables para el Servo9
int valorServo9h=70;
int valorServo9l=80;

//variables para el Servo10
int valorServo10h=70;
int valorServo10l=80;

//variables para el Servo11
int valorServo11h=70;
int valorServo11l=80;

//variables para el Servo12
int valorServo12h=70;
int valorServo12l=80;

//variables para el Servo13
int valorServo13h=70;
int valorServo13l=80;

//variables para el Servo14
int valorServo14h=70;
int valorServo14l=80;

//variables para el Servo15
int valorServo15h=70;
int valorServo15l=80;

//variables para el Servo16
int valorServo16h=70;
int valorServo16l=80;

//variables para el Servo17
int valorServo17h=70;
int valorServo17l=80;

//variables para el Servo18
int valorServo18h=70;
int valorServo18l=80;
int dato=0;

int bandera=0;
int indice=0;
int direccionCorrecta=0;

//le envia la información de los 18 servos
al esclavo por i2c
void enviarDatosI2C()
{
   i2c_start();
   i2c_write(2); //manda la dirección del
esclavo
   //datos servo1
   i2c_write(valorServo1h); //parte alta
   i2c_write(valorServo1l); //parte baja
   //datos servo2
   i2c_write(valorServo2h); //parte alta
   i2c_write(valorServo2l); //parte baja
   //datos servo3
   i2c_write(valorServo3h); //parte alta
   i2c_write(valorServo3l); //parte baja
   //datos servo4
   i2c_write(valorServo4h); //parte alta
   i2c_write(valorServo4l); //parte baja
   //datos servo5
   i2c_write(valorServo5h); //parte alta
   i2c_write(valorServo5l); //parte baja
   //datos servo6
   i2c_write(valorServo6h); //parte alta
   i2c_write(valorServo6l); //parte baja
   //datos servo7
   i2c_write(valorServo7h); //parte alta
   i2c_write(valorServo7l); //parte baja
   //datos servo8
   i2c_write(valorServo8h); //parte alta
   i2c_write(valorServo8l); //parte baja
   //datos servo9
   i2c_write(valorServo9h); //parte alta
   i2c_write(valorServo9l); //parte baja
   //datos servo10
   i2c_write(valorServo10h); //parte alta
   i2c_write(valorServo10l); //parte baja
```

```
   //datos servo11
   i2c_write(valorServo11h); //parte alta
   i2c_write(valorServo11l); //parte baja
   //datos servo12
   i2c_write(valorServo12h); //parte alta
   i2c_write(valorServo12l); //parte baja
   //datos servo13
   i2c_write(valorServo13h); //parte alta
   i2c_write(valorServo13l); //parte baja
   //datos servo14
   i2c_write(valorServo14h); //parte alta
   i2c_write(valorServo14l); //parte baja
   //datos servo15
   i2c_write(valorServo15h); //parte alta
   i2c_write(valorServo15l); //parte baja
   //datos servo16
   i2c_write(valorServo16h); //parte alta
   i2c_write(valorServo16l); //parte baja
   //datos servo17
   i2c_write(valorServo17h); //parte alta
   i2c_write(valorServo17l); //parte baja
   //datos servo18
   i2c_write(valorServo18h); //parte alta
   i2c_write(valorServo18l); //parte baja

   i2c_stop(); //señal de fin de
transmisión
}



#int_rda //interrupción por recepción de
datos
void interrupcionRS232(void)
{
   dato=getchar(); //para que reponga la
bandera de interrupción

   if(bandera==1)
   {
      //Se verifica que la posición 11 del
mensaje corresponda al último byte del
       número de serie del que envía el
mensaje
      if(indice==11)
      {
         if(dato==0x55)
         {
            direccionCorrecta=1;
         }
         else
         {
            direccionCorrecta=0;
         }
      }

      //Si se confirma la dirección, se
busca en la cadena recibida el mensaje
      if(direccionCorrecta==1)
      {
         if(indice==15)
         {
            valorServo1h=dato;
            indice++;
         }
         else if(indice==16)
         {
            valorServo1l=dato;
            indice++;
         }
         else if(indice==17)
         {
            valorServo2h=dato;
            indice++;
         }
         else if(indice==18)
         {
            valorServo2l=dato;
            indice++;
         }
         else if(indice==19)
         {
            valorServo3h=dato;
            indice++;
         }
         else if(indice==20)
         {
            valorServo3l=dato;
            indice++;
         }
         else if(indice==21)
         {
            valorServo4h=dato;
            indice++;
         }
         else if(indice==22)
         {
            valorServo4l=dato;
            indice++;
         }
         else if(indice==23)
         {
            valorServo5h=dato;
            indice++;
         }
         else if(indice==24)
         {
            valorServo5l=dato;
            indice++;
         }
         else if(indice==25)
         {
            valorServo6h=dato;
            indice++;
         }
         else if(indice==26)
         {
            valorServo6l=dato;
            indice++;
         }
         else if(indice==27)
         {
            valorServo7h=dato;
            indice++;
         }
         else if(indice==28)
         {
            valorServo7l=dato;
            indice++;
         }
         else if(indice==29)
         {
            valorServo8h=dato;
            indice++;
         }
         else if(indice==30)
         {
            valorServo8l=dato;
            indice++;
         }
         else if(indice==31)
         {
```

```
            valorServo9h=dato;
            indice++;
         }
         else if(indice==32)
         {
            valorServo9l=dato;
            indice++;
         }
         else if(indice==33)
         {
            valorServo10h=dato;
            indice++;
         }
         else if(indice==34)
         {
            valorServo10l=dato;
            indice++;
         }
         else if(indice==35)
         {
            valorServo11h=dato;
            indice++;
         }
         else if(indice==36)
         {
            valorServo11l=dato;
            indice++;
         }
         else if(indice==37)
         {
            valorServo12h=dato;
            indice++;
         }
         else if(indice==38)
         {
            valorServo12l=dato;
            indice++;
         }
         else if(indice==39)
         {
            valorServo13h=dato;
            indice++;
         }
         else if(indice==40)
         {
            valorServo13l=dato;
            indice++;
         }
         else if(indice==41)
         {
            valorServo14h=dato;
            indice++;
         }
         else if(indice==42)
         {
            valorServo14l=dato;
            indice++;
         }
         else if(indice==43)
         {
            valorServo15h=dato;
            indice++;
         }
         else if(indice==44)
         {
            valorServo15l=dato;
            indice++;
         }
         else if(indice==45)
         {
            valorServo16h=dato;
            indice++;
         }
         else if(indice==46)
         {
            valorServo16l=dato;
            indice++;
         }
         else if(indice==47)
         {
            valorServo17h=dato;
            indice++;
         }
         else if(indice==48)
         {
            valorServo17l=dato;
            indice++;
         }
         else if(indice==49)
         {
            valorServo18h=dato;
            indice++;
         }
         else if(indice==50)
         {
            valorServo18l=dato;
            indice++;
         }
         else if(indice>50)
         {
            //el último byte de la
transmisión CheckSum
            bandera=0;
            indice=0;
            direccionCorrecta=0;

            //Mandar todos los datos
            enviarDatosI2C();
            output_toggle(pin_b5);
         }
         else
         {
            indice++;
         }
      }
      else
      {
         if(indice>50)
         {
            //el último byte de la
transmisión CheckSum
            bandera=0;
            indice=0;
         }
         else
         {
            indice++;
         }
      }
   }
   else
   {
      if(dato==0x7e)
      {
         //quiere decir que es el inicio
de la transmisión
         bandera=1;
         indice++;
      }
   }
}
```

```
void main()
{
   setup_adc_ports(NO_ANALOGS); //no hay
entradas analógicas

   //Se habilita la interrupción por
recepción de datos RS232
   enable_interrupts(int_rda);
   enable_interrupts(global);

   indice=0;
   bandera=0;
   direccionCorrecta=0;

   while(1)
   {
      delay_ms(500);
      output_toggle(pin_b7);
   }
}
```

Programa utilizado por el microcontrolador esclavo que controla la posición de los 18 servomotores con los datos recibidos por protocolo i^2c del microcontrolador maestro

```
#include <18F4550.h>
//configure a 20MHz crystal to operate at
48MHz
#fuses
HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV
,PLL5,CPUDIV1,VREGEN
#use delay(clock=48000000)

//configuración de la comunicación i2c a
100 kHz
#use
i2c(slave,sda=pin_b0,scl=pin_b1,slow,addre
ss=0x02)

#use fast_io(A)
#use standard_io(B)
#use standard_io(C)
#use fast_io(D)
#use fast_io(E)

//Se definen nombres para los pines
utilizados para controlar los servos
#define pinServo1 pin_b2
#define pinServo2 pin_b3
#define pinServo3 pin_b4
#define pinServo4 pin_d2
#define pinServo5 pin_d3
#define pinServo6 pin_d4
#define pinServo7 pin_b5
#define pinServo8 pin_b6
#define pinServo9 pin_b7

#define pinServo10 pin_d5
#define pinServo11 pin_d6
#define pinServo12 pin_d7
#define pinServo13 pin_a0
#define pinServo14 pin_a1
#define pinServo15 pin_a2
#define pinServo16 pin_d1
#define pinServo17 pin_d0
#define pinServo18 pin_a3

//variables para el Servo1
int valorServo1h=74;
int valorServo1l=220;

//variables para el Servo2
int valorServo2h=55;
int valorServo2l=140;

//variables para el Servo3
int valorServo3h=37;
int valorServo3l=164;

//variables para el Servo4
int valorServo4h=63;
int valorServo4l=72;

//variables para el Servo5
int valorServo5h=85;
int valorServo5l=20;

//variables para el Servo6
int valorServo6h=102;
int valorServo6l=72;

//variables para el Servo7
int valorServo7h=67;
int valorServo7l=8;

//variables para el Servo8
int valorServo8h=52;
int valorServo8l=152;

//variables para el Servo9
int valorServo9h=39;
int valorServo9l=120;

//variables para el Servo10
int valorServo10h=68;
int valorServo10l=16;

//variables para el Servo11
int valorServo11h=84;
int valorServo11l=204;

//variables para el Servo12
int valorServo12h=102;
int valorServo12l=144;

//variables para el Servo13
int valorServo13h=68;
int valorServo13l=196;

//variables para el Servo14
int valorServo14h=55;
int valorServo14l=32;

//variables para el Servo15
int valorServo15h=39;
int valorServo15l=192;

//variables para el Servo16
int valorServo16h=66;
int valorServo16l=96;

//variables para el Servo17
int valorServo17h=88;
int valorServo17l=188;

//variables para el Servo18
int valorServo18h=100;
```

```
int valorServo18l=8;

int numeroServo1=1;
int numeroServo2=10;

int nivel1=0;
int nivel2=0;

int direccionCorrecta=0;
int dato=0;
int posicion=0;

//Controla del servo1 al servo9
#int_timer1
void interrupcionTimer1(void)
{
   if(nivel1==0)
   {
      switch(numeroServo1)
      {
         case 1:
            output_high(pinServo1);
            set_timer1(65536-
(valorServo1h*256+valorServo1l));
            break;
         case 2:
            output_high(pinServo2);
            set_timer1(65536-
(valorServo2h*256+valorServo2l));
            break;
         case 3:
            output_high(pinServo3);
            set_timer1(65536-
(valorServo3h*256+valorServo3l));
            break;
         case 4:
            output_high(pinServo4);
            set_timer1(65536-
(valorServo4h*256+valorServo4l));
            break;
         case 5:
            output_high(pinServo5);
            set_timer1(65536-
(valorServo5h*256+valorServo5l));
            break;
         case 6:
            output_high(pinServo6);
            set_timer1(65536-
(valorServo6h*256+valorServo6l));
            break;
         case 7:
            output_high(pinServo7);
            set_timer1(65536-
(valorServo7h*256+valorServo7l));
            break;
         case 8:
            output_high(pinServo8);
            set_timer1(65536-
(valorServo8h*256+valorServo8l));
            break;
         case 9:
            output_high(pinServo9);
            set_timer1(65536-
(valorServo9h*256+valorServo9l));
            break;
      }
      nivel1=1;
   }
   else
   {
      switch(numeroServo1)
      {
         case 1:
            output_low(pinServo1);
            numeroServo1++;

set_timer1(35536+(valorServo1h*256+valorSe
rvo1l));
            break;
         case 2:
            output_low(pinServo2);
            numeroServo1++;

set_timer1(35536+(valorServo2h*256+valorSe
rvo2l));
            break;
         case 3:
            output_low(pinServo3);
            numeroServo1++;

set_timer1(35536+(valorServo3h*256+valorSe
rvo3l));
            break;
         case 4:
            output_low(pinServo4);
            numeroServo1++;

set_timer1(35536+(valorServo4h*256+valorSe
rvo4l));
            break;
         case 5:
            output_low(pinServo5);
            numeroServo1++;

set_timer1(35536+(valorServo5h*256+valorSe
rvo5l));
           break;
         case 6:
            output_low(pinServo6);
            numeroServo1++;

set_timer1(35536+(valorServo6h*256+valorSe
rvo6l));
            break;
         case 7:
            output_low(pinServo7);
            numeroServo1++;

set_timer1(35536+(valorServo7h*256+valorSe
rvo7l));
            break;
         case 8:
            output_low(pinServo8);
            numeroServo1++;

set_timer1(35536+(valorServo8h*256+valorSe
rvo8l));
            break;
         case 9:
            output_low(pinServo9);
            numeroServo1=1;

set_timer1(35536+(valorServo9h*256+valorSe
rvo9l));
            break;
      }
      nivel1=0;
   }
}
//Controla del servo10 al servo18
#int_timer3
void interrupcionTimer3(void)
```

```
{
   if(nivel2==0)
   {
      switch(numeroServo2)
      {
         case 10:
            output_high(pinServo10);
            set_timer3(65536-
(valorServo10h*256+valorServo10l));
            break;
         case 11:
            output_high(pinServo11);
            set_timer3(65536-
(valorServo11h*256+valorServo11l));
            break;
         case 12:
            output_high(pinServo12);
            set_timer3(65536-
(valorServo12h*256+valorServo12l));
            break;
         case 13:
            output_high(pinServo13);
            set_timer3(65536-
(valorServo13h*256+valorServo13l));
            break;
         case 14:
            output_high(pinServo14);
            set_timer3(65536-
(valorServo14h*256+valorServo14l));
            break;
         case 15:
            output_high(pinServo15);
            set_timer3(65536-
(valorServo15h*256+valorServo15l));
            break;
         case 16:
            output_high(pinServo16);
            set_timer3(65536-
(valorServo16h*256+valorServo16l));
            break;
         case 17:
            output_high(pinServo17);
            set_timer3(65536-
(valorServo17h*256+valorServo17l));
            break;
         case 18:
            output_high(pinServo18);
            set_timer3(65536-
(valorServo18h*256+valorServo18l));
            break;
      }
      nivel2=1;
   }
   else
   {
      switch(numeroServo2)
      {
         case 10:
            output_low(pinServo10);
            numeroServo2++;

set_timer3(35536+(valorServo10h*256+valorS
ervo10l));
         break;
         case 11:
            output_low(pinServo11);
            numeroServo2++;

set_timer3(35536+(valorServo11h*256+valorS
ervo11l));
         break;
         case 12:
            output_low(pinServo12);
            numeroServo2++;

set_timer3(35536+(valorServo12h*256+valorS
ervo12l));
         break;
         case 13:
            output_low(pinServo13);
            numeroServo2++;

set_timer3(35536+(valorServo13h*256+valorS
ervo13l));
         break;
         case 14:
            output_low(pinServo14);
            numeroServo2++;

set_timer3(35536+(valorServo14h*256+valorS
ervo14l));
         break;
         case 15:
            output_low(pinServo15);
            numeroServo2++;

set_timer3(35536+(valorServo15h*256+valorS
ervo15l));
         break;
         case 16:
            output_low(pinServo16);
            numeroServo2++;

set_timer3(35536+(valorServo16h*256+valorS
ervo16l));
         break;
         case 17:
            output_low(pinServo17);
            numeroServo2++;

set_timer3(35536+(valorServo17h*256+valorS
ervo17l));
         break;
         case 18:
            output_low(pinServo18);
            numeroServo2=10;

set_timer3(35536+(valorServo18h*256+valorS
ervo18l));
         break;
      }
      nivel2=0;
   }
}
//Recibe los datos del maestro
#int_ssp
void interrupcionI2C(void)
{
   dato=i2c_read();
   if(direccionCorrecta==1)
   {
      switch(posicion)
      {
         case 0:
            valorServo1h=dato;
            posicion++;
            break;
         case 1:
            valorServo1l=dato;
            posicion++;
            break;
         case 2:
```

```
            valorServo2h=dato;
            posicion++;
            break;
         case 3:
            valorServo2l=dato;
            posicion++;
            break;
         case 4:
            valorServo3h=dato;
            posicion++;
            break;
         case 5:
            valorServo3l=dato;
            posicion++;
            break;
         case 6:
            valorServo4h=dato;
            posicion++;
            break;
         case 7:
            valorServo4l=dato;
            posicion++;
            break;
         case 8:
            valorServo5h=dato;
            posicion++;
            break;
         case 9:
            valorServo5l=dato;
            posicion++;
            break;
         case 10:
            valorServo6h=dato;
            posicion++;
            break;
         case 11:
            valorServo6l=dato;
            posicion++;
            break;
         case 12:
            valorServo7h=dato;
            posicion++;
            break;
         case 13:
            valorServo7l=dato;
            posicion++;
            break;
         case 14:
            valorServo8h=dato;
            posicion++;
            break;
         case 15:
            valorServo8l=dato;
            posicion++;
            break;
         case 16:
            valorServo9h=dato;
            posicion++;
            break;
         case 17:
            valorServo9l=dato;
            posicion++;
            break;
         case 18:
            valorServo10h=dato;
            posicion++;
            break;
         case 19:
            valorServo10l=dato;
            posicion++;
            break;
         case 20:
            valorServo11h=dato;
            posicion++;
            break;
         case 21:
            valorServo11l=dato;
            posicion++;
            break;
         case 22:
            valorServo12h=dato;
            posicion++;
            break;
         case 23:
            valorServo12l=dato;
            posicion++;
            break;
         case 24:
            valorServo13h=dato;
            posicion++;
            break;
         case 25:
            valorServo13l=dato;
            posicion++;
            break;
         case 26:
            valorServo14h=dato;
            posicion++;
            break;
         case 27:
            valorServo14l=dato;
            posicion++;
            break;
         case 28:
            valorServo15h=dato;
            posicion++;
            break;
         case 29:
            valorServo15l=dato;
            posicion++;
            break;
         case 30:
            valorServo16h=dato;
            posicion++;
            break;
         case 31:
            valorServo16l=dato;
            posicion++;
            break;
         case 32:
            valorServo17h=dato;
            posicion++;
            break;
         case 33:
            valorServo17l=dato;
            posicion++;
            break;
         case 34:
            valorServo18h=dato;
            posicion++;
            break;
         case 35:
            valorServo18l=dato;
            posicion=0;
            direccionCorrecta=0;
            break;
      }
   }
   else
   {
      if(dato==0x02)
      {
```

```
         output_toggle(pin_c1);
         direccionCorrecta=1;
         posicion=0;
      }
   }
}
void main()
{
   setup_adc_ports(NO_ANALOGS); //no hay
entradas analógicas

   //Se configura el timer1 y se habilita
la interrupción por desbordamiento
   setup_timer_1(t1_internal|t1_div_by_1);
   enable_interrupts(int_timer1);

   //Se configura el timer3 y se habilita
la interrupción por desbordamiento
   setup_timer_3(t3_internal|t3_div_by_1);
   enable_interrupts(int_timer3);

   enable_interrupts(int_ssp);

   set_tris_A(0); //el puerto A es salida
   set_tris_D(0); //el puerto D es salida
   set_tris_E(0); //el puerto E es salida

   enable_interrupts(global);

   while(1)
   {
      delay_ms(500);
      output_toggle(pin_c2);
   }
}
```

Circuito eléctrico y tarjeta fenólica

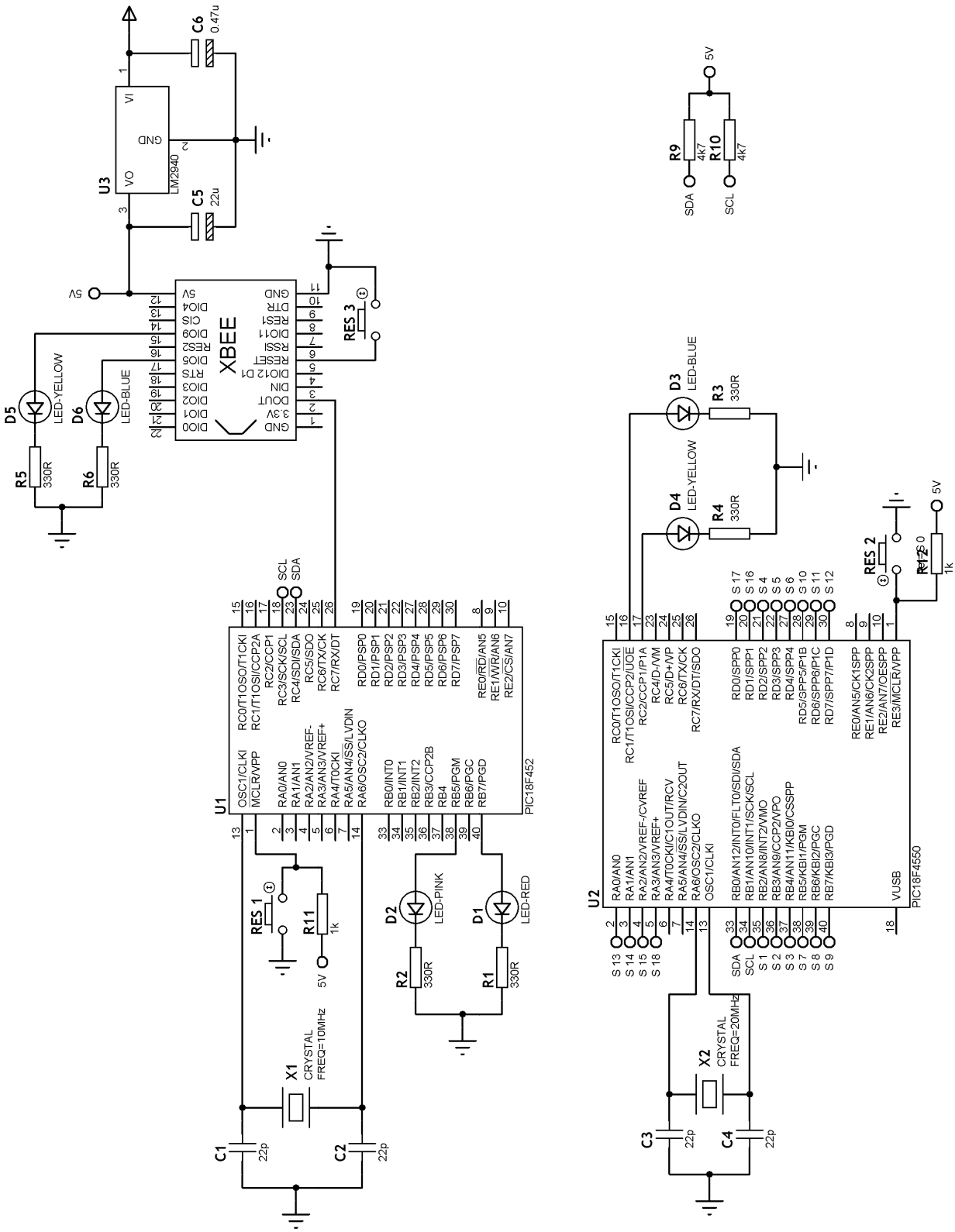


Figura 71: Diagrama eléctrico

Circuito impreso para realizar la tarjeta fenólica

La tarjeta es de 1 sola cara, sin embargo hace falta hacer un puente soldando un cable. En la Figura 72 se muestra la ubicación de dicho puente.

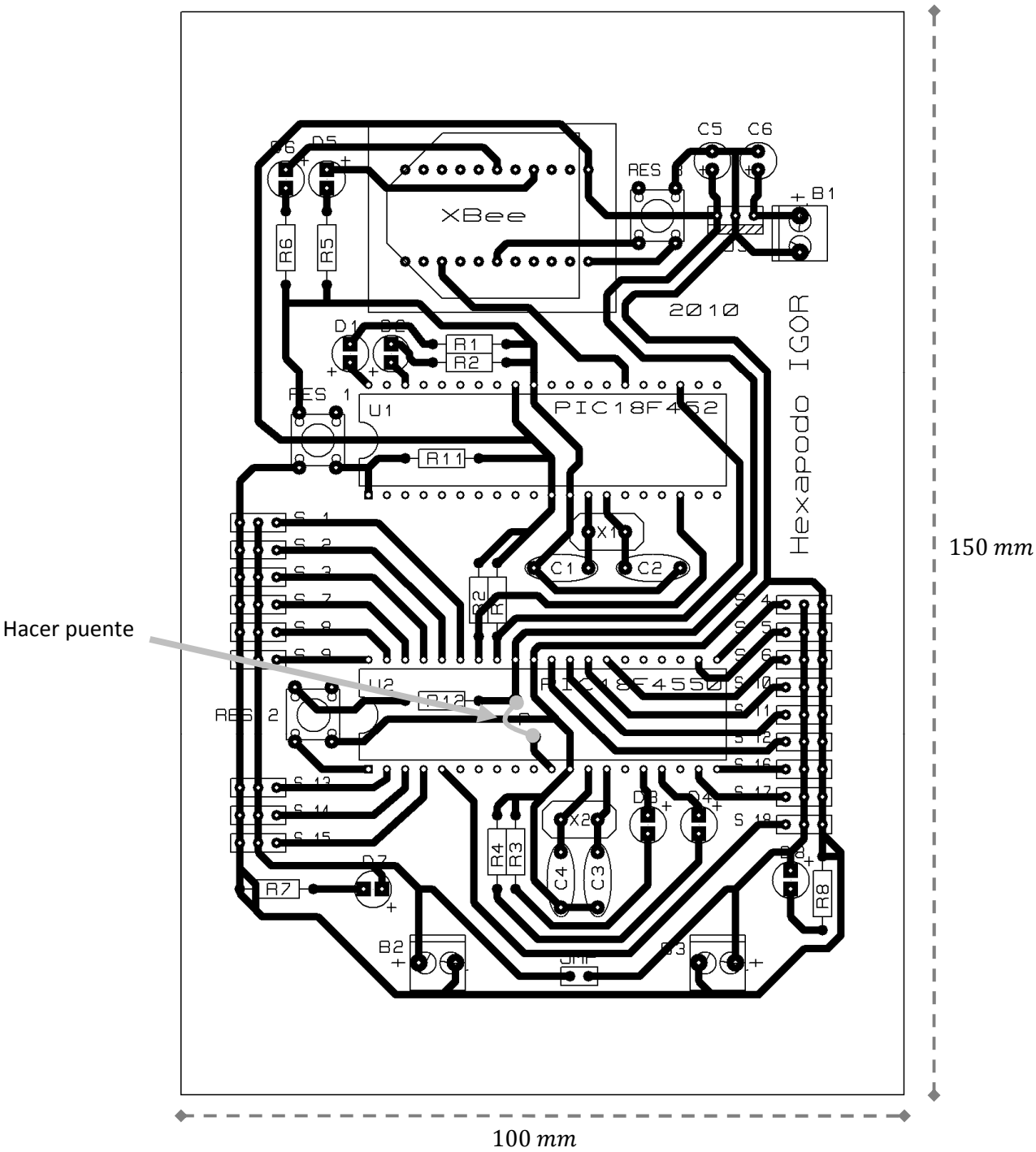


Figura 72: Diagrama completo de la tarjeta fenólica

APÉNDICE C

En la Figura 73 se muestra del lado izquierdo la ubicación de los componentes (en espejo) y es la imagen que debe ser impresa en la placa en la parte superior (la que no tiene cobre), del lado derecho se muestran las pistas que deben de ser impresas en la parte inferior de la placa (lado que tiene cobre), las imágenes están a escala real.

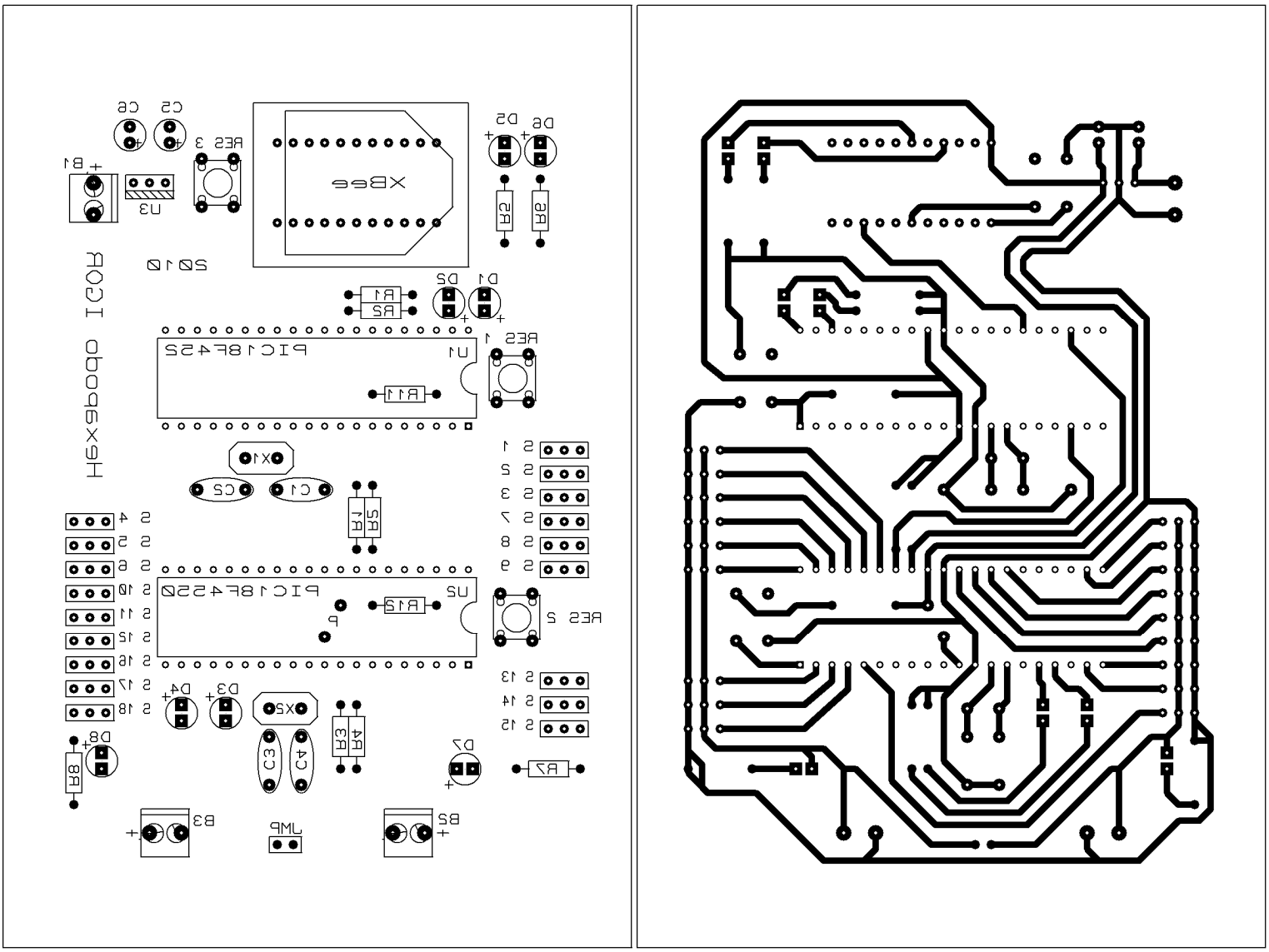


Figura 73: Plantillas para realizar la tarjeta fenólica

Lista de materiales

Designador	Descripción	Valor
C1, C2, C3, C4	Capacitor cerámico	22 pF
C5	Capacitor electrolítico	22 μF
C6	Capacitor electrolítico	0.47 μF
R1, R2, R3, R4 **R5, R6, R7, R8**	Resistencia de ¼ W	330 Ω

R11, R12	Resistencia de ¼ W	1 $k\Omega$
D1	LED rojo	LED 3 mm
D2	LED rosa pastel	LED 3 mm
D3, D6	LED azul pastel	LED 3 mm
D4, D5	LED amarillo	LED 3 mm
D7, D8	LED verde	LED 3 mm
RES 1, RES 2, RES 3	Micro switch de push	4 terminales
U1, U2	Base para CI en DIP	DIP40
U1*	Microcontrolador PIC	PIC18F452
U2*	Microcontrolador PIC	PIC18F4550
U3	Regulador de voltaje 1A 5V	LM2940CT-5.0
XBee	Xbee Explorer regulated	
JMP	Header Jumper	Header 1 x 2 Jumper 1 x 2
X1	Cristal de cuarzo	10 MHz
X2	Cristal de cuarzo	20 MHz
B1, B2, B3	Terminal chica con 2 tornillos para circuito impreso	Terminal 1 x 2
S 1, S 2, S 3, S 4, S 5 S 6, S 7, S 8, S 9, S 10 S 11, S 12, S 13, S 14 S 15, S 16, S 17, S 18	Header vertical	Header 1 x 3
P	Hacer puente	

Tabla 13: Lista de materiales para realizar la tarjeta fenólica

U1* y U2* se deben montar sobre las bases luego de soldar estas últimas a la tarjeta.

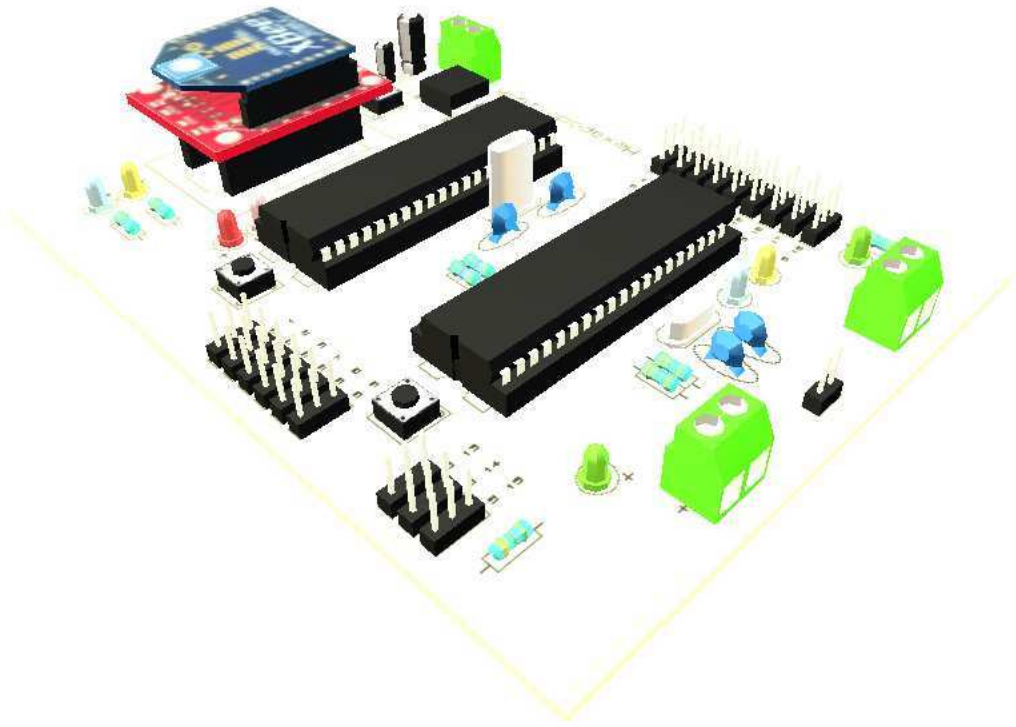


Figura 74: Imagen tomada del simulador que muestra la tarjeta fenólica completamente ensamblada

Referencias

[1] Reed, Aaron, “Learning XNA 3.0”, Editorial O’Reilly, Estados Unidos, 2009.

[2] Grootjans, Riemer, “XNA 3.0 Game Programming Recipes: A Problem-Solution Approach, Editorial Apress, Estados Unidos, 2009.

[3] Blender 3D: Noob to Pro – Beginner Tutorial

[4] Ollero, Aníbal, “Robótica: manipuladores y robots móviles” Editorial Alfaomega, España, 2007.

[5] Craig, John, “Introduction to Robotics: Mechanics and control”, 2ª edición, Editorial Addison Wesley Longman, Estados Unidos, 1989.

[6] Barrientos, Antonio, “Fundamentos de robótica”, 2a edición, Editorial McGraw-Hill, España 2007.

[7] Siegwart, Roland, Nourbakhsh, Illah R., “Introduction to Autonomous Mobile Robots”, A Bradford Book The MIT Press Cambridge, Massachusetts, Londres, Inglaterra, 2004.

[8] Jones, Joseph L., Flynn, Anita M., Peters, A. K., “Mobile Robots: Inspiration to implementation”, Wellesley, Massachusetts, 1993.

[9] Gray, J. O., Caldwell, D. G., “Advanced Robotics & Intelligent machines”, IEE the Institution of Electrical Engineers, 1996.

[10] Williams, Karl, “Insectronics: Build your own walking robot”, TAB Robotics, McGraw Hill, 2003.

[11] P. González de Santos, E. García, J. Estremera, *Quadrupedal Locomotion: An Introduction to the Control of Four-legged Robots,* Springer-Verlag, Alemania, 2006

[12] P. Muñoz-Martínez, M. D. R. Moreno, J. Gómez Elvira, J. J. Romeral Planelló, S. Navarro López, *An Autonomous system for the locomotion of a Hexapod Exploration Robot,* Third IEEE International Conference on Space Mission Challenges for Information Technology, Madrid (Spain), 2009

[13] P. Birkmeyer, R. S. Fearing, *DASH: A Resilient High-Speed 15g Hexapedal Robot,* The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 11-15, 2009 St. Louis, USA

[14] A. M. Hoover, E. Steltz, R. S. Fearing, *RoACH: An autonomous 2.4g crawling hexapod robot,* 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Acropolis Convention Center, Nice, France, September 22-26, 2008

[15] Bessonov, A. P., Umnov, N. W. *Choice of geometric parameters of walking machines*, Mech. Mach. Theory 18, 1976.

[16] http://es.wikipedia.org/wiki/Blender 28-01-11

[17] http://es.wikipedia.org/wiki/Microsoft_XNA 28-01-11