UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

PROYECTO DE GRADO

Presentado ante la ilustre UNIVERSIDAD DE LOS ANDES como requisito parcial para obtener el Título de INGENIERO DE SISTEMAS

DISEÑO, CONSTRUCCIÓN Y CONTROL DE UN EXOESQUELETO ORIENTADO A LA REHABILITACIÓN DEL DEDO ÍNDICE DE LA MANO.

Por

Br. Fannia K. Pacheco M.

Tutor: Prof. Mariela Cerrada.

Septiembre 2012

Diseño, construcción y control de un exoesqueleto orientado a la rehabilitación del dedo índice de la mano

Br. Fannia K. Pacheco M.

Proyecto de Grado – Control y Automatización, 173 Páginas

Escuela de Ingeniería de Sistemas, Universidad de Los Andes, 2012.

Resumen: En el presente trabajo se expone el diseño, construcción y control de un exoesqueleto para el dedo índice. Los exoesqueletos mecánicos o exoesqueleto robot es un armazón metálico externo que ayuda a mover a su portador y a realizar cierto tipo de actividad, como lo es el cargar peso o para la rehabilitación de lesiones. Se presenta el diseño del dispositivo y además se desea controlar el ángulo de desplazamiento de falange proximal del dedo índice, usando un motor paso a paso como actuador en el primer eslabón del exoesqueleto. Los movimientos controlados tienen el propósito de emular el movimiento de pinza y realizar ejercicios según la capacidad del paciente para la rehabilitación de lesiones en el dedo índice.

Palabras claves: Exoesqueleto, rehabilitación, robótica, control, ingeniería biomédica.

Índice

Índice de Tablas

Índice de Figuras

Agradecimientos

A DIOS, alabado y misericordioso, quien siempre ilumino mi camino para cumplir mis metas.

A mi mamá, por ser siempre un apoyo incondicional a lo largo de mi vida y durante mi carrera.

A mi papa y mis hermanos George, Yorman y Faviana, porque siempre están allí cuando los necesito.

A la Dra. Mariela Cerrada, por que sin su ayuda y dedicación no podría haberse realizado este proyecto.

A mis amigos, quienes siempre me ayudaron con mis problemas, tanto a nivel académico como personal.

Al Ing.Alfredo Lacruz, por su ayuda y paciencia en el desarrollo de este proyecto.

A la Universidad de los Andes, por ser la casa de estudio que me acogió durante mi carrera.

Al Consejo de Desarrollo Científico, Humanístico y Tecnológico de la Universidad de los Andes (CDCHTA), por el financiamiento otorgado a este trabajo.

Capítulo 1

Introducción

El término exoesqueleto en biología es conocido como el esqueleto externo que recubre toda la superficie de los animales filo artrópodos (arácnidos, insectos, crustáceos y otros grupos relacionados), el mismo cumple una función protectora, de respiración y otra mecánica, proporcionando el sostén necesario para la eficiencia del aparato muscular. Ahora bien, basado en una idea similar, en el campo de la robótica y la ortésica se utiliza este término para denominar estructuras externas rígidas que proporcionan soporte a las funciones motoras de la persona (Chávez et al., 2010).

Se puede definir un exoesqueleto como una estructura que sirve de apoyo y se usa para asistir los movimientos y/o aumentar las capacidades del cuerpo humano. Pueden ser estructuras pasivas o activas, es decir, que contengan, o no, actuadores para el movimiento y por lo tanto necesiten, o no, un sistema de control asociado a los mismos. La mayoría de los exoesqueletos utilizados en el campo de la medicina, se adapta al cuerpo con sistemas inteligentes de procesamiento y sensado para la toma de decisiones, con el fin de realizar una tarea previamente definida (Chávez et al., 2010).

1.1 Antecedentes

En la actualidad existe gran variedad de desarrollos realizados con exoesqueletos robóticos, la mayoría de ellos no están disponibles en el mercado, debido al costo de los mismos, ya que la implementación de estos dispositivos (actuación y almacenamiento de energía) puede estar limitada con la tecnología existente (Chávez et al., 2010).

Los primeros exoesqueletos robóticos construidos fueron desarrollados en el área de la milicia, los científicos se enfocaron principalmente en aumentar las capacidades de los soldados, ya sea en fuerza o velocidad. Con el fin de tener un grupo más poderoso con respecto a sus contrincantes. Con el avance de la tecnología y con el descubrimiento de estos dispositivos se fueron ampliando sus aplicaciones al área de la medicina y la rehabilitación de lesiones principalmente (Chávez et al., 2010).

1.1.1 Aplicaciones de los exoesqueletos robóticos

- Amplificadores de potencia:

Los exoesqueletos como amplificadores de potencia, es el principal enfoque de desarrollo concebido desde la ciencia ficción. La persona suministra señales de control al exoesqueleto, mientras el dispositivo proporciona gran parte de la potencia necesaria para llevar a cabo la tarea. La persona está totalmente integrada con el sistema y percibe la fuerza ejercida por el exoesqueleto en una magnitud pequeña (Kazerooni, 1990).

El dispositivo presentado por Zoss et al., (2006), Bleex Exoskeleton, es un desarrollo de exoesqueletos como amplificadores de potencia, que consiste en un dispositivo capaz de

aumentar las capacidades físicas de los soldados (véase la figura 1.1).

Figura 1. 1: "Bleex Exoskeleton" exoesqueleto robótico para aumentar la carga que puede soportar una persona, destinado a aplicaciones militares (Zoss et al., 2006)

Estos tipos de exoesqueletos también son utilizados para las personas con debilidad muscular o problemas neuromusculares, donde se desempeñan como asistentes para realizar actividades diarias.

- Dispositivo para realimentación háptica[1]:

Producen la sensación de estar tocando realmente un mundo virtual o remoto. El objetivo ideal es que el operador no distinga entre lo real y lo virtual. En la figura 1.2, se observa la estructura de estos dispositivos, donde el operador se encuentra a distancia del objetivo y, por medio de la interacción entre el exoesqueleto, la interfaz gráfica y el robot esclavo, se realiza la actividad deseada (Sabater, 2003).

En el área médica, estos métodos están siendo utilizados con gran éxito. Un ejemplo de ello es el robot Da Vinci. En la actualidad, existen más de 800 robots quirúrgicos Da Vinci en el mundo, con aplicaciones en Urología, Ginecología, Cirugía General, Cirugía Pediátrica, Cirugía Torácica, Cirugía Cardíaca y otorrinolaringología (ORL).

[1] Háptica, estrictamente hablando significa todo aquello referido al contacto, especialmente cuando éste se usa de manera activa.

El robot Da Vinci utiliza el mismo principio de los dispositivos hápticos (Mourad, 2008). En Venezuela, pionera en Suramérica, hay tres instituciones en las que se realizan algunas operaciones ya estandarizadas en cirugía robótica, entre ellas el Hospital Universitario de Caracas.

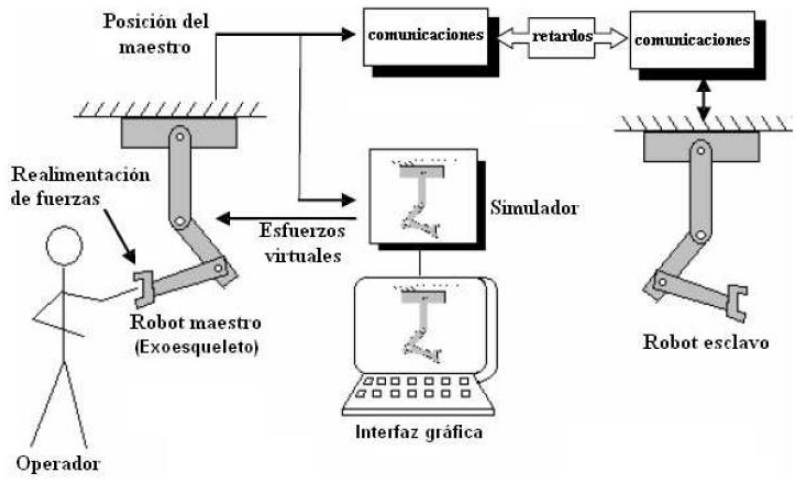


Figura 1. 2: Elementos que intervienen en un sistema teleoperado (Sabater, 2003)

- Rehabilitación:

En pacientes con problemas físicos estos dispositivos permiten asistir las terapias de rehabilitación guiando los movimientos de las trayectorias correctas para ayudar al paciente a reaprender los patrones de motricidad y dar fuerza de soporte para realizar los movimientos, recibiendo en este caso particular el nombre de órtesis activa. En la figura 1.3 se puede observar la evolución que puede tener un paciente en silla de ruedas, al utilizar exoesqueletos para la recuperación del movimiento en los miembros inferiores (Morón, 2007).

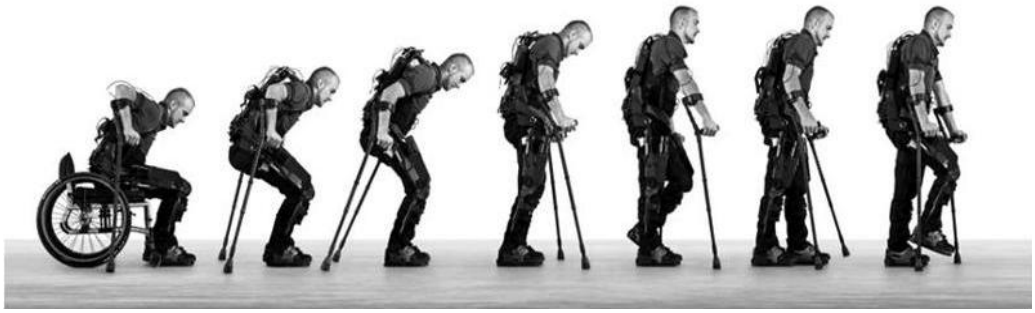


Figura 1. 3: Rehabilitación de la parte inferior del cuerpo mediante el uso de exoesqueletos (véase: http://ehlersdanlos-info-mas-mi-experiencia.blogspot.com/2011/06/esoesqueletos-para-scapacitados.html)

- Estudio e investigación

La biomecánica es un área de conocimiento interdisciplinaria que estudia los modelos, fenómenos y leyes en el movimiento de los seres vivos, por tal definición los exoesqueletos han brindado un gran aporte a esta área, como herramientas de cuantificación de parámetros del miembro superior, como lo son el tono, inercia, impedancia, entre otros (Lucas et al., 2004).

También existen amplias investigaciones relacionadas con el control motor, y en la neurofisiología de pacientes con lesiones cerebrales. A través de señales bioeléctricas o electromiográficas (EMG), captadas por el exoesqueleto y generadas por la persona, se permite el registro de dicha información para el correspondiente estudio de las señales por los expertos (Lucas et al., 2004).

Hoy en día, siguen creciendo las áreas de investigación en las cuales los exoesqueletos son de gran apoyo.

1.1.2 Exoesqueletos robóticos de mano en la literatura

Es de interés en esta investigación conocer los desarrollos que se han enfocado en la mano, específicamente en los dedos humanos, es por ello que a continuación se presenta una breve reseña de los mismos.

Uno de los exoesqueletos de manos más conocido que trabaja con interfaz háptica es el "Cybergrasp Exoskeleton", el cual interactúa mecánicamente con objetos virtuales en un entorno simulado o remoto, este sistema es de gran utilidad para la teleoperación en entornos peligrosos y sirve también como simulador para entrenamiento quirúrgico (Basdogan et al., 2001), ver figura 1.4.

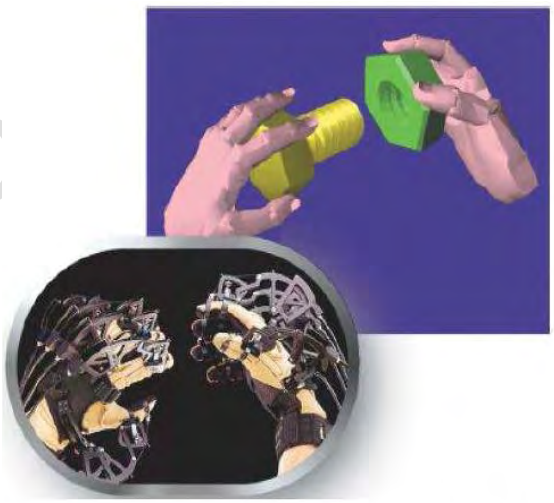


Figura 1. 4: "Cybergrasp Exoskeleton" dispositivo robótico para la mano con interfaz háptica (Basdogan et al., 2001)

Para la rehabilitación de lesiones en manos, es importante tomar en cuenta que el dispositivo debe ser liviano para no interferir en la realización de los movimientos previamente definidos, es por ello que se ha utilizado diferentes dispositivos de instrumentación para cumplir con este requerimiento.

Un ejemplo de ello es el mecanismo presentado por DiCicco et al., (2004). Este dispositivo se implementó con actuadores neumáticos con el objetivo de hacerlo ligero (véase la figura 1.5). En su estudio, el autor compara diferentes estrategias de control, para ser usadas con el dispositivo y presenta los resultados obtenidos de la utilización del exoesqueleto con una persona tetrapléjica.

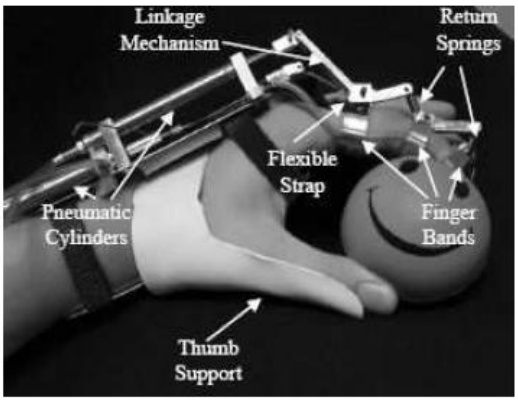


Figura 1. 5: Exoesqueleto para la rehabilitación de mano controlado con señales EMG (DiCicco et al., 2004)

En el estudio presentado por Brown et al., (1993), se describe el diseño e implementación de un dispositivo en forma de guante para asistir los movimientos de personas que han perdido control de los músculos de la mano y dedos. El dispositivo se acciona por motores electromagnéticos DC y mediante una transmisión por cable.

Un exoesqueleto destinado a la rehabilitación de la mano es el presentado por Wege y Hommel, (2005). El dispositivo es capaz de intervenir en cuatro grados de libertad (GdL) de los dedos por medio de actuadores eléctricos y con cables de transmisión (véase la figura 1.6).

Sarakoglou et al., (2004) han investigado el uso de un exoesqueleto para la rehabilitación de los músculos de la mano, que permite al usuario repetir exactamente los movimientos de los dedos, mediante ejercicios en un entorno de realidad virtual (véase la figura 1.7). El dispositivo posee siete GdL activados por motores electromagnéticos de DC y cables para transmisión, para la reflexión de fuerzas sobre los dedos.

Figura 1. 6: Exoesqueleto para la rehabilitación de los dedos de la mano capaz de intervenir cuatro GdL (Wege y Hommel, 2005)

Figura 1. 7: Exoesqueleto de mano destinado a la investigación para la rehabilitación muscular (Sarakoglou et al., 2004)

De igual manera se consiguen diferentes diseños de exoesqueletos para manos como el presentado en la figura 1.8, por Fiorilla et al., (2007).

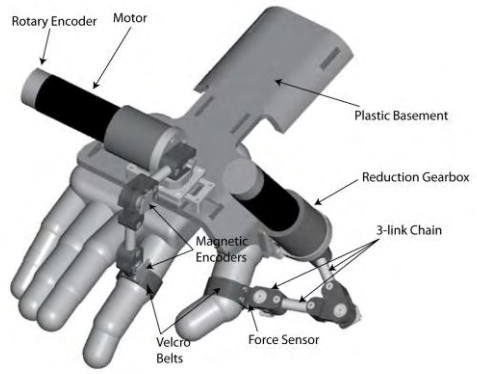


Figura 1. 8: Diseño del Exoesqueleto que muestra las partes del dispositivo (Fiorilla et al., 2007)

Otro diseño interesante es el presentado por Wang et al., (2009), el cual consta de cuatro GdL para la rehabilitación del dedo índice, el dispositivo puede generar un movimiento bi-direccional para todas las articulaciones del dedo a través de cables de transmisión como se muestra en la figura 1.9.

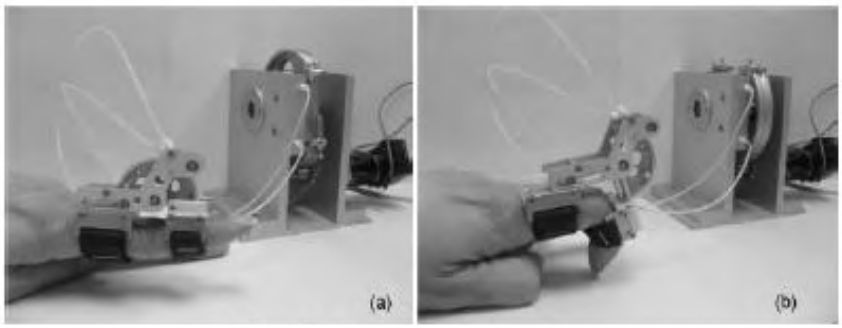


Figura 1. 9: Prototipo del sistema diseñado por Wang et al., (2009)

La mayoría de estudios y desarrollos de exoesqueletos robóticos destinados a rehabilitación y asistencia motora se han enfocado al miembro superior, debido al importante rol que esta parte del cuerpo juega en la realización de actividades diarias.

1.2 Planteamiento del Problema

En Venezuela no se cuenta con amplia investigación y desarrollo en el ámbito de exoesqueletos robóticos para la rehabilitación, es por ello que diferentes grupos de investigación en la Facultad de Ingeniería de la Universidad de los Andes, se han propuesto iniciar esta línea de investigación en diferentes direcciones.

Específicamente en este trabajo se propone el diseño, construcción y control de un exoesqueleto robótico para el dedo índice con dos GdL, con el fin de implementar un patrón de rehabilitación según las especificaciones provenientes de la Sala de Terapia Ocupacional del Instituto Autónomo Hospital Universitario de los Andes (IHULA), como apoyo en la recuperación de lesión de dedos y permitir recobrar movimientos esenciales del paciente.

1.3 Justificación

La tendencia en la rehabilitación hoy en día es la construcción de mecanismos protésicos y ortésicos que aceleren el proceso de recuperación en los pacientes. La mano es una de las extremidades sujeta a rehabilitación, por su importancia al permitir un mayor desarrollo de las actividades mecánicas finas de los humanos.

La mano es capaz de maniobrar casi en cualquier posición que se desee, para ello, se basa en la destreza de sus dedos y la habilidad de la muñeca para acomodarla en dicha posición. El movimiento de pinza de las manos permite la toma objetos, es una característica única de los

seres humanos que no lo tiene ningún otro ser vivo. En este movimiento está involucrado principalmente el dedo pulgar y el dedo índice, los demás dedos sirven de apoyo. El desplazamiento del dedo pulgar no es muy significativo, ya que normalmente es el dedo índice el cual incide en distintos ángulos de curvatura para obtener el movimiento de pinza de manera precisa, es por ello que es importante la recuperación de forma rápida y eficaz de la señalada habilidad (Morón, 2007).

Cuando una persona sufre de una lesión de manos, atraviesa por un proceso de recuperación muy lento que limita su independencia. El exoesqueleto propuesto va a permitir la apertura hacia un campo de investigación en la Universidad de los Andes conjuntamente con la Sala de Terapia Ocupacional del IHULA, para la aplicación de estos dispositivos y los beneficios que los mismos pueden proporcionar. Se espera que la implementación de este prototipo permita ampliar los estudios en este ámbito y tener una referencia para futuros trabajos.

Cabe destacar que la Sala de Terapia Ocupacional del IHULA no tiene dispositivos automatizados para la rehabilitación de lesiones, las terapias se realizan de manera manual con dispositivos mecánicos o en su defecto de madera, es por ello la motivación de trabajar en esta área.

1.4 Objetivos

1.4.1 Objetivo general

Diseñar, construir y controlar un exoesqueleto para el dedo índice, que implemente un patrón de rehabilitación específico.

1.4.2 Objetivos específicos

- Diseñar un sistema mecánico que reproduzca el movimiento de pinza del dedo índice con dos grados de libertad.
- Modelar la dinámica aproximada del sistema.
- Diseñar la ley de control para el exoesqueleto robótico, que implemente un patrón de rehabilitación específica.
- Construir el prototipo del sistema de control

1.5 Metodología

La metodología empleada en el desarrollo del proyecto para cumplir con los objetivos específicos planteados fué la siguiente:

1. Diseñar un sistema mecánico que reproduzca el movimiento de pinza del dedo índice con dos grados de libertad.

   1.1 Investigar la anatomía de la mano y el movimiento de la mano humana. Con base en ello se realizó el diseño del dispositivo y se tomaron otras consideraciones futuras.
   1.2 Decidir cuáles son los movimientos más importantes para la rehabilitación del dedo índice. En esta fase se utilizó la información adquirida de la Sala de Terapia Ocupacional del IAHULA.
   1.3 Diseñar, con la información obtenida, el exoesqueleto del dedo índice, con ayuda del paquete de modelado paramétrico de sólidos en 3D **Autodesk Inventor**.

2. Modelar la dinámica aproximada del sistema dinámico

   2.1 Realizar una analogía con un sistema dinámico, a partir del dispositivo diseñado. En esta fase se estudió la dinámica de diferentes configuraciones de péndulos de un grado de libertad.

   2.2 Estudiar diferentes configuraciones de péndulos de dos grados de libertad.

   2.3 Realizar simulaciones, para observar los comportamientos que se pueden suscitar con distintas entradas (control en lazo abierto), mediante herramientas computacionales como **Matlab** y **Maple**.

3. Diseñar la ley de control para el exoesqueleto robótico, que implemente un patrón de rehabilitación específica.

   3.1 Diseñar una ley de control, según los actuadores disponibles con apoyo de la información obtenida en la primera fase. Se fijaron los requerimientos a cumplir según los patrones de rehabilitación establecidos.

   3.2 Simular el sistema controlado para verificar que funcione correctamente.

4. Construir el prototipo del sistema de control.

   4.1 Estudiar los diferentes componentes que permitirán la implementación del dispositivo y la instrumentación a utilizar que cumple con el diseño de la ley de control para el exoesqueleto del dedo índice, esto es, definir los sensores y actuadores.

4.2 Construir el dispositivo del exoesqueleto para el dedo índice y probarlo con apoyo de las simulaciones.

4.3 Implementar la ley de control usando la herramienta de software **Visual Studio 2010**.

4.4 Validar el prototipo elaborado con el apoyo de la Sala de Terapia Ocupacional.

1.6 Estructura del Documento

El manuscrito está organizado de la siguiente manera:

Capítulo 2, describe las bases teóricas para el correcto desarrollo del trabajo de investigación.

Capítulo 3, se muestran los diferentes componentes del sistema de control, esto es el diseño del exoesqueleto robótico, análisis del actuador y del sensor, modelado dinámico de los mismos y validación de modelos.

Capítulo 4, se presenta la implementación del sistema de rehabilitación asistida. Así como también, el diseño y la programación de la lógica de control y de la interfaz gráfica.

Capítulo 5, se muestran las conclusiones del estudio realizado y las recomendaciones sugeridas.

Capítulo 2

Marco Teórico

Las bases teóricas expuestas ayudarán a entender el comportamiento de la mano, para diseñar el prototipo de acuerdo a ello, así como a definir la instrumentación adecuada a utilizar, esto es actuador y sensores adecuados.

2.1 Generalidades del funcionamiento de la mano humana

De manera de conocer el rango de movimiento del exoesqueleto a diseñar y cómo se va a realizar el control, es importante tomar en cuenta las capacidades de la mano humana. Es por ello, que a continuación se presenta esta información que servirá de guía en el desarrollo del trabajo.

En la tabla 2.1 se muestra un resumen de los movimientos de flexión y extensión, que permite cada articulación de los dedos, como referencia se toman los ejes del dedo índice mostrado en la figura 2.1. Tomando en cuenta que, para cada persona estos valores pueden variar, este estudio fue realizado a 400 manos donde se aportó el resultado mencionado (Palastanga et al., 2000).

Tabla 2. 1: Rangos de movimientos permitidos por las articulaciones del dedo índice

Articulación	DIP	PIP	MCP1	MCP2
Rango de movimiento(°)	0-80	0-100	0-85	0-45

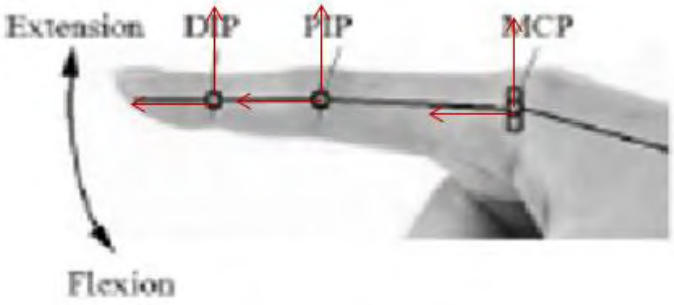


Figura 2. 1: Articulaciones del dedo índice, DIP: articulación interfalágica distal, PIP: articulación interfalángica proximal, MCP: articulación metacarpofalángica (Palastanga et al., 2000)

La articulación metacarpofalángica (MPC) mostrada en la figura 2.1, es la que se desea controlar para que cumpla un patrón de rehabilitación. La misma, está compuesta por dos articulaciones internas, la metacarpofalángica 1 (MCP1) que permite el movimiento de flexión y la metacarpofalángica 2 (MCP2) que realiza el movimiento de extensión.

2.1.1 Lesiones de mano

Para el desarrollo de un exoesqueleto es importante conocer las lesiones de manos más frecuentes, ya que esto permitirá visualizar mejor los problemas que enfrenta cada persona para realizar terapia ocupacional y los requerimientos que debe cumplir el exoesqueleto

robótico para desempeñar un papel importante en la recuperación de estas lesiones. En este sentido, se mencionarán las siguientes (Calennoff, 1972):

i. Lesiones vasculares de la mano. Las lesiones del sistema vascular pueden producirse por elementos cortantes, contusos, quemantes, armas de fuego, explosivos, tracción. La mano está irrigada por las arterias radial y cubital, hasta en un 10% por la arteria mediana remanente.

ii. Lesiones nerviosas de la mano. La lesión nerviosa en la mano puede ser de tipo sensitivo o motor, dependiendo del nervio comprometido y del nivel de la lesión.

iii. Lesiones tendiosas de la mano. Estas lesiones pueden afectar a la mano, tanto traumáticas como por sobrecarga o movimientos repetitivos.

iv. Fracturas y lesiones articulares de la mano. Son las más comunes en los humanos, ya que pueden suscitarse realizando las actividades diarias de la persona, ya sea en el hogar, en el trabajo o en el entorno donde se encuentre. Frecuentemente se dan en la mano derecha y en los hombres en su mayoría.

Es importante el conocimiento de las lesiones mencionadas, ya que el proceso de recuperación incluye una fase de aplicación de terapias, una vez que ha pasado un tiempo estipulado por el médico, dependiendo de la lesión. El paciente debe someterse a diversas terapias para recobrar el movimiento normal de la mano o en su defecto semejante.

2.1.2 Protocolos más comunes en la rehabilitación del dedo índice

En términos generales, la rehabilitación es un proceso por el cual la persona tiene que aprender a vivir con una discapacidad dentro de su entorno. El equipo interdisciplinar que trabaja con personas afectadas de una lesión fisiátrica tiene como objetivo reducir o compensar los efectos que la lesión ha producido en su capacidad física, adquiriendo habilidades que permitan a la persona conseguir el mayor grado de independencia posible (Chavez et al.,2010).

Para realizar la rehabilitación del dedo es necesario que el terapista ocupacional realice un primer examen articular para conocer el nivel de la lesión, este procedimiento se realiza mediante un instrumento llamado goniómetro formado por un semicírculo y un brazo fino (véase la figura 2.2). El goniómetro permite precisar cuántos grados puede mover el paciente una articulación (Chavez et al., 2010).

Figura 2. 2: Goniometría de la flexoextensión de la articulación Metacarpofalángica (Chavez et al., 2010)

2.1.3 Tratamiento de fisioterapia según el nivel de lesión

Comúnmente se distinguen en 6 niveles en la mano, que están asociados en la numeración de la figura 2.3, a continuación se mencionarán las terapias de interés en esta investigación, las cuales están asociadas a las zonas 5 y 6 (Miralles, 2004).

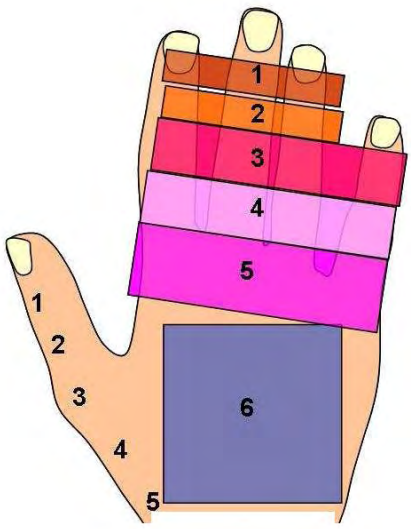


Figura 2. 3: Mano humana dividida en 6 zonas (Miralles, 2004)

Entre el carpo y la articulación metacarpofalángica (MCP) (zonas 5 y 6), durante las tres primeras semanas el paciente realiza 10 repeticiones de flexión activa de algunas de las articulaciones del dedo (DIP y PIP) cada 2 horas seguido de extensión pasiva. A partir de la tercera semana se inicia el movimiento activo suave de la articulación MCP. Con posición inicial de 20º-30º de flexión en la muñeca se recomiendan tres ejercicios:

1. El paciente debe mantener activamente la posición de partida durante varios segundos y seguidamente se le solicita flexión activa MCP hasta 30º que también deberá mantener varios segundos. Posteriormente debe hacer extensión activa de MCP hasta

la posición neutra. Se recomiendan de 10-20 repeticiones cada 1-2 horas (véase la figura 2.4).

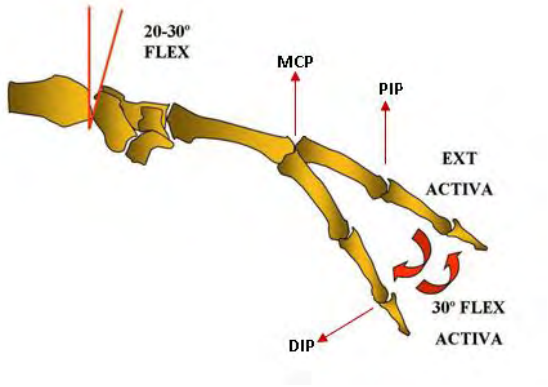


Figura 2. 4: Lesión en zonas 5 y 6 de la mano, ejercicio 1(Miralles, 2004)

2. Extensión pasiva de muñeca de 45°. Se pide al paciente flexión activa MCP hasta 40°-60° manteniendo la PIP y DIP en extensión (véase la figura 2.5).

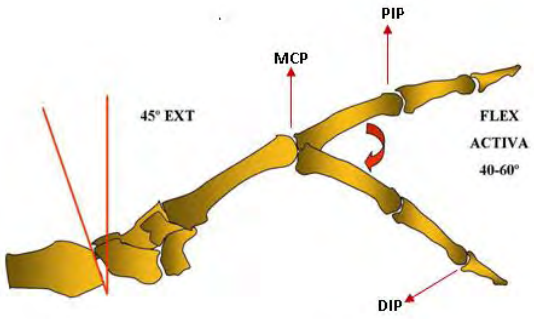


Figura 2. 5: Lesión en zonas 5 y 6 de la mano, ejercicio 2 (Miralles, 2004)

3. Muñeca a 20°-30° de Extensión y MCP a 0°: manteniendo activamente dicha posición, solicitaremos flexo-extensión activa (véase la figura 2.6).

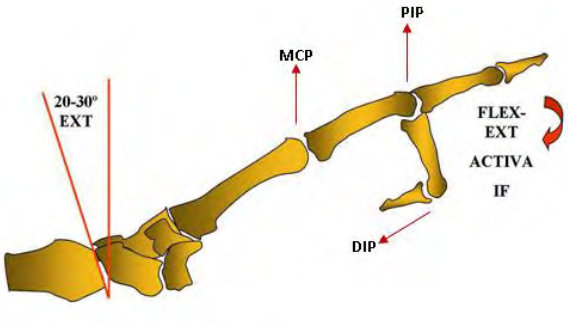


Figura 2. 6: Lesión en zonas 5 y 6 de la mano, ejercicio 3 (Miralles, 2004)

Finalmente en el presente trabajo solo se implementaron dos de los patrones de rehabilitación que implica el movimiento de la MPC (véase la figura 2.4 y 2.5), pero el nivel de diseño del sistema de control se muestra que todas pueden implementarse. Sin embargo, se realizarán simulaciones para abarcar los demás patrones de rehabilitación restantes.

2.2 Modelo del péndulo

El objetivo principal de este trabajo es controlar el ángulo de desplazamiento de la falange proximal. Inicialmente, para simplificar el sistema, se va a considerar que el exoesqueleto robótico con el dedo índice conforma una barra rígida que a simple vista es análogo a un péndulo simple (véase la figura 2.7), en esta fase es despreciado el movimiento del segundo grado de libertad.

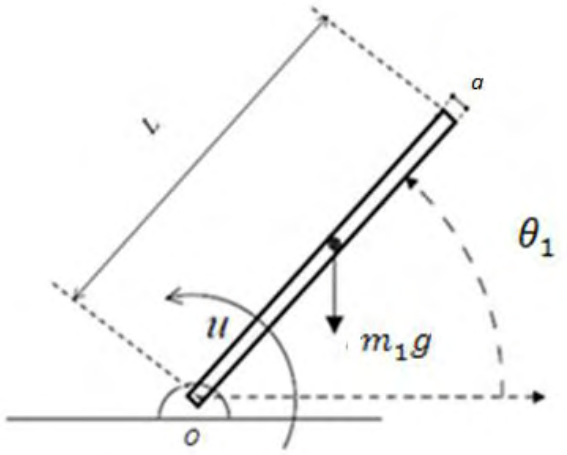


Figura 2. 7: Péndulo simple análogo al primer eslabón del exoesqueleto

El torque *u* aplicado a la barra rígida en el punto de rotación, es considerado la entrada que va a permitir mover la articulación metacarpiana del dedo índice. *L* es la longitud de la barra rígida y *a* es el ancho de la barra.

Considere el péndulo de un eslabón mostrado en la figura 2.7, usando la ecuación de la 2da. Ley de Newton para el movimiento rotacional (Ogata, 1987):

$$\sum_i T_o^i = J_o \alpha = J_o \ddot{\theta} \qquad (2.1)$$

Donde:

T_o^i son los torques respecto al punto de rotación $(N.m)$.

J_o es el momento de inercia del cuerpo con respecto al punto de rotación $(kg.m^2)$.

θ_1 es desplazamiento angular (rad).

w es $\dot{\theta}_1$, la velocidad angular (rad/s).

α es $\ddot{\theta}_1$, la aceleración angular (rad/s^2).

Utilizando la ec. (2.1) para modelar el péndulo de un eslabón, se tiene:

$$J_1\ddot{\theta}_1 + m_1 g l cos\theta_1 + b\dot{\theta}_1 = u \qquad (2.2)$$

Donde:

m_1 es la masa del eslabón (kg).

L es la longitud del eslabón (m).

u es un torque externo $(N.m)$.

$l = \frac{L}{2}$

a es el ancho del eslabón (m).

g es la aceleración de gravedad (m/s^2).

b es el coeficiente de fricción.

J_1 es calculado por medio del teorema de los eje paralelos $(kg.m^2)$, es decir

$$J_1 = \frac{1}{12} m_1 (a^2 + l^2) + m_1 \left(\frac{l}{2}\right)^2$$

Expresando el modelo en la representación de variables de estados se tiene:

$x_1(t) = \theta_1(t)$

$x_2(t) = \dot{\theta}_1(t)$

$\ddot{\theta}_1(t) = \dot{x}_2$

Entonces, reescribiendo la ec.(2.2), se obtiene:

$$J_1\dot{x}_2(t) + mglcosx_1(t) + bx_2(t) = u(t) \qquad (2.3)$$

y representando el modelo de de estados se tiene:

$$\dot{x}_1(t) = x_2(t) \tag{2.4}$$

$$\dot{x}_2(t) = \frac{\left(-m_1 glcosx_1(t) - bx_2(t) + u(t)\right)}{J_1} \tag{2.5}$$

- Análisis de estabilidad del punto de operación

Con el modelo expuesto en las ec. (2.4) y (2.5), se realiza el análisis de estabilidad del punto de operación parametrizado respecto al desplazamiento angular θ_1. El análisis de estabilidad permitirá determinar el rango de valores de θ_1 para el cual está definido el torque.

El punto de operación *(X_1, X_2, U)* parametrizado respecto de θ_1, respectivamente X_1, es:

$x_2^* = 0$

$x_1^* = X_1$

$u^* = m_1 glcosX_1$

Es decir, el punto de operación es *(X_1, 0, $m_1$glcosX_1)*.

El modelo lineal obtenido por linealización Jacobiana alrededor del punto de operación viene dada por la ec. (2.6).

$$\dot{x}_\delta(t) = Ax_\delta(t) + Bu_\delta(t) \tag{2.6}$$

Donde:

$$A = \frac{\partial f}{\partial x} \qquad \text{y} \qquad B = \frac{\partial f}{\partial u}$$

De esta manera:

$$A = \left[\begin{matrix} 0 & 1 \\ \frac{mglsenx_1(t)}{J} & \frac{-b}{J} \end{matrix}\right]_{PO} \quad , \quad B = \left[\begin{matrix} 0 \\ 1/J \end{matrix}\right]_{PO}$$

Luego, el sistema linealizado obtenido es el siguiente:

$$\dot{x}_\delta(t) = \left[\begin{matrix} 0 & 1 \\ \frac{mglsenx_1{}^*}{J} & \frac{-b}{J} \end{matrix}\right] x_\delta(t) + \left[\begin{matrix} 0 \\ 1/J \end{matrix}\right] u_\delta(t) \qquad (2.7)$$

Analizando el polinomio característico *Pc(s)* dado en la ec. (2.8), y aplicando el criterio de estabilidad es evidente que el sistema es estable para ángulos negativos.

$$\mathrm{Pc(s)} = \det(sI - A) = det\left[\begin{matrix} s & -1 \\ -\frac{mglsenx_1{}^*}{J} & s + \frac{b}{J} \end{matrix}\right] = s^2 + \frac{b}{J}s - \frac{mglsenx_1{}^*}{J} \qquad (2.8)$$

- Control en lazo abierto del péndulo de un eslabón

El análisis de estabilidad alrededor del punto de operación $(X_1,\ 0,\ m_1 g l cos X_1)$ provee de información interesante para el control en lazo abierto:

a) El valor del torque está restringido a la función que define el punto de operación, en el intervalo $-mgl < u < mgl$.
b) Por tanto, se podrá "controlar" el sistema a cualquier desplazamiento angular en el intervalo $-\pi < \theta_1 < 0$.
c) Cualquier otro desplazamiento angular deseado debe lograrse con una ley de control en lazo cerrado.

Con esta información es posible determinar el torque máximo que permite que el sistema se mantenga estable, utilizando el paquete matemático **Matlab** se simuló el modelo con los siguientes valores de los parámetros:

$m_1 = 0.03\ kg$

$g = 9.8\ m/s^2$

$l = 0.06\ m$

$b = 0.005$

$a = 0.01\ m$

$$J_1 = \frac{1}{12} m_1 (a^2 + l^2) + m_1 \left(\frac{l}{2}\right)^2 = 1.5417e^{-5} kg.m^2$$

Evaluando los parámetros obtenidos y con el apoyo de simulaciones el mínimo y máximo torque que puede recibir el sistema para que sea estable es:

$$-0.0176 < u < 0.0176$$

Se realizaron simulaciones con valores dentro del rango especificado para u, observando que el sistema es estable como se muestra en la figura 2.8. Así mismo, se efectuaron pruebas con valores fuera del rango, corroborando que el ángulo no es acotado, véase la figura 2.9.

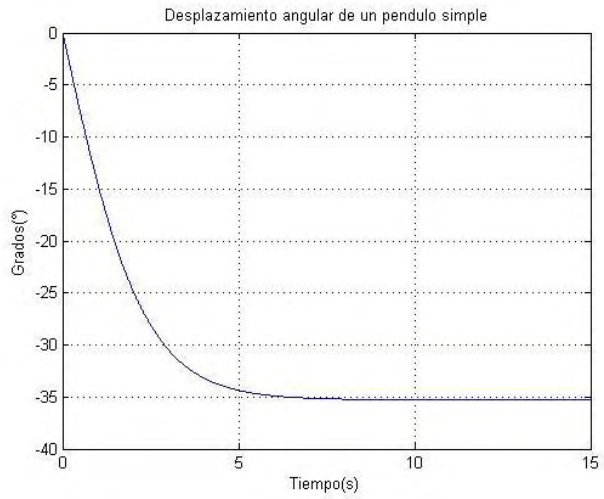


Figura 2. 8: Desplazamiento angular del péndulo simple de un eslabón ante una entrada $u=0.006\ N.m$

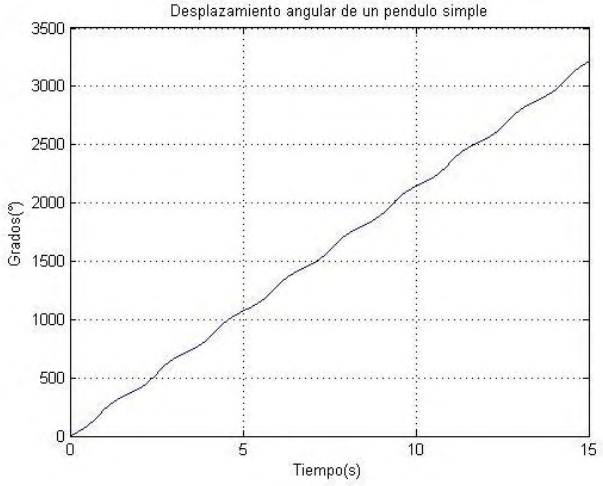


Figura 2. 9: Desplazamiento angular del péndulo simple de un eslabón ante una entrada $u=0.02\ N.m$

A partir de este análisis, se decidió usar un motor paso a paso, a fin de no implementar un control en lazo cerrado usando como actuador un servomotor.

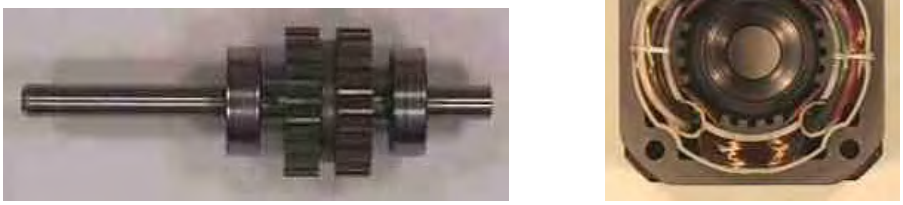
2.3 Modelo de un motor paso a paso

Los motores paso a paso (PAP) son ideales para la construcción de mecanismos que requieren una alta precisión en sus movimientos, como su nombre lo indica, estos dispositivos se desplazan paso a paso, el ángulo por paso puede variar de 0.7º hasta 90º. Estos motores poseen la capacidad de sostener la carga en el ángulo que permite cada paso. Por tal razón para la actuación en el exoesqueleto del dedo índice, se va utilizar un motor de este tipo ya que permite ejercer un control en lazo abierto.

Existen tres tipos de motores PAP, estos son de imán permanente, de reluctancia variable e híbridos. Este último resulta de combinar las características más importantes de los dos primeros. En este trabajo se usa un motor de imán permanente.

2.3.1 Motores de imán permanente

Estos motores están constituidos normalmente por un rotor, sobre el que van aplicados distintos imanes permanentes y por un cierto número de bobinas excitadoras bobinadas en su estator. Las bobinas son parte del estator y el rotor es un imán permanente, toda la conmutación (o excitación de las bobinas) debe ser externamente manejada por un controlador (Kuo, 1996). En la figura 2.10 se puede observa la estructura del estator y del rotor de un motor de imán permanente.



(a) (b)

Figura 2. 10: (a) Rotor de un motor de imán permanente (b) Estator de un motor de imán permanente

Existen dos tipos de motores de imán permanente, bipolares y unipolares. El motor utilizado en la presente investigación es un motor unipolar con dos bobinas, ambas son idénticas y no están conectadas eléctricamente. Cada bobina tiene una toma central, un cable que sale de la bobina que está a medio camino de longitud entre sus dos terminales (véase la figura 2.11).

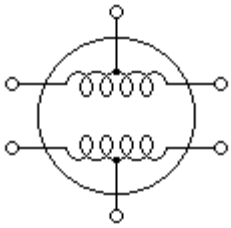


Figura 2. 11: Estructura interna de un motor paso a paso unipolar (Khorrami et.al., 2003)

2.3.2 Control de un motor paso a paso unipolar

La corriente que fluye a través de una bobina produce un campo magnético que atrae un rotor de imán permanente, que está conectado al eje del motor. El principio básico de

control es invertir la dirección de la corriente a través de las 2 bobinas, en secuencia, con el fin de mover el rotor (Khorrami et. al., 2003).

Dado que hay dos bobinas y dos direcciones, es posible obtener una secuencia de cuatro fases, alternando el paso de la corriente a través de los interruptores como se observa en la figura 2.12.

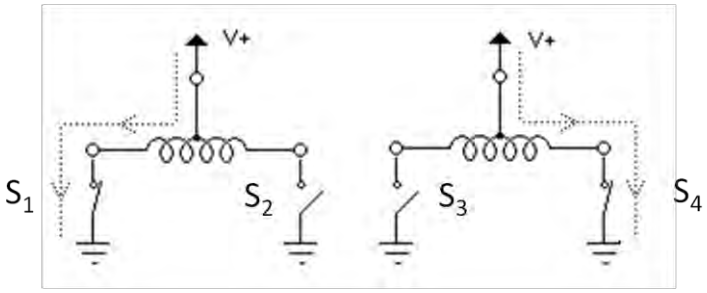


Figura 2. 12: Cambio de sentido de la corriente en una bobina de un motor paso a paso unipolar (Khorrami et.al., 2003)

En la figura 2.13 se muestra el diagrama conceptual del motor PAP unipolar utilizado para la actuación del exoesqueleto, se puede observar las uniones centrales de las bobinas se conectan a la alimentación, los terminales de las bobinas se alimentan en secuencia, para atraer el rotor según el movimiento deseado.

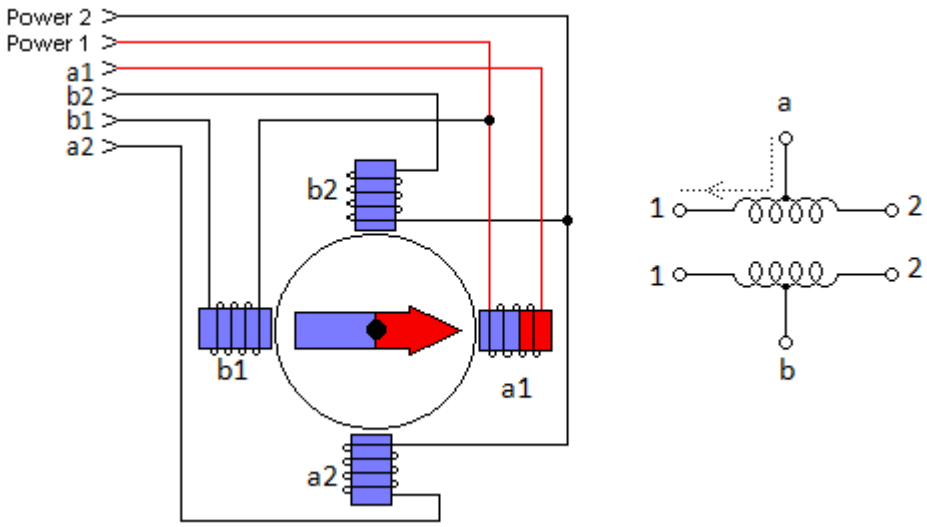


Figura 2. 13: Diagrama conceptual de un motor paso a paso unipolar

En la tabla 2.2 se proporcionan tres secuencias útiles para el movimiento hacia la izquierda de un motor paso a paso unipolar, cambiar la dirección del motor implica invertir la secuencia dada en la tabla. El patrón de secuencia se representa con cuatro bits, un "1" indica un devanado energizado y "0" no energizado.

Tabla 2. 2: Secuencia para el movimiento de motores paso a paso unipolares

Secuencia	Nombre	Descripción
0001 0010 0100 1000	Un paso-Una Fase	Consume menos energía. Solo una fase se activa en un momento. Asegura exactitud posicional.
0011 0110 0110 1100 1001	Torque elevado, dos fases	Esta secuencia activa dos fases adyacentes, ofrece mayor par-velocidad y mayor par de sostenimiento.

Continua en la página siguiente …

		Viene de la página anterior …
0001 0011 0010 0110 0100 1100 1000 1001	Medio paso	Duplica la resolución paso a paso del motor, pero el par no es uniforme para cada paso

Para el control del motor paso a paso utilizado en el presente trabajo, se emplea la secuencia de torque elevado.

2.3.3 Modelo dinámico de un motor paso a paso unipolar

El modelo dinámico de cualquier motor se divide en dos partes, la parte mecánica y la parte eléctrica (véase la figura 2.14). La dinámica mecánica viene dada por las leyes de Newton relacionando el torque con la aceleración. La parte eléctrica es gobernada por las leyes de Kirchoff y se pueden derivar haciendo un circuito equivalente (Kuo, 1996).

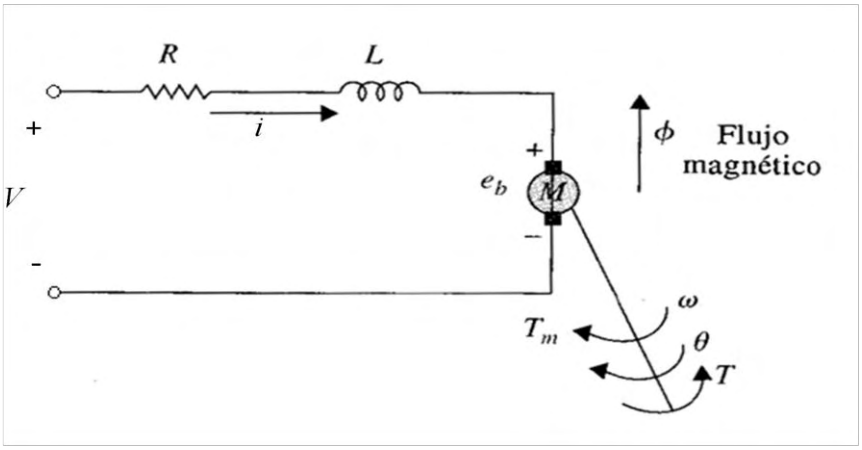


Figura 2. 14: Circuito equivalente de una bobina de un motor paso a paso (Kuo, 1996)

Ambos comportamientos, el eléctrico y el mecánico, se acoplan mediante la fuerza que depende de las corrientes y de la inductancia que depende de la posición. Las ecuaciones dinámicas que rigen el comportamiento mecánico del motor, vienen dadas por las ec.(2.9) y (2.10) (Badinez, 2007).

$$\dot{\theta} = \omega \qquad (2.9)$$

$$\dot{\omega} = \frac{1}{J}\left(T_m - T_c - T_v - T_f\right) \qquad (2.10)$$

Donde θ es la posición angular del rotor dada en *rad*, ω es la velocidad angular, J es el momento de inercia del rotor, T_m torque generado por el motor, T_c torque de carga, T_v torque de fricción estática y T_f es el torque de fricción, este último usualmente es modelado como:

$$T_f = D\omega \qquad (2.11)$$

D es el factor de amortiguamiento viscoso.

Las ecuaciones eléctricas del sistema, dependen de la cantidad de bobinas que tenga el motor paso a paso y su circuito equivalente está compuesto por una resistencia, un inductor y la carga como se muestra en la figura 2.14 , por tal razón aplicando ley de mallas se tiene:

$$V_j(t) = i_j(t)R_j + L_j\frac{di_j(t)}{dt} + e_j \qquad j = 1, \ldots, N \qquad (2.12)$$

N es el número de bobinas del motor.

$i_j(t)$ son las corrientes que pasan por cada bobina (A).

$V_j(t)$ es el voltaje de alimentación de cada bobina (V).

R_j son las resistencias que tiene cada bobina (Ω).

e_j es el voltaje de salida del sistema eléctrico y viene dado por:

$$e_j = K_m\frac{d\theta_m(t)}{dt} = K_m\bar{\omega} \qquad (2.13)$$

Donde $\bar{\omega}$ es un vector de velocidad angular, desarrollando la expresión vectorial de $\bar{\omega}$ se tiene:

$$e_j = K_m \frac{d\theta_m(t)}{dt} = K_m\, \omega sin\left(p\theta - \frac{(j-1)\pi}{N}\right) \qquad (2.14)$$

Al evaluar la ec.(2.14) en la ec.(2.12) se obtiene el comportamiento dinámico de las corrientes de las bobinas, dada en la ec.(2.15).

$$\frac{di_j(t)}{dt} = \frac{1}{L_j}\left(V_j(t) - i_j(t)R_j - K_m\, \omega sin\left(p\theta - \frac{(j-1)\pi}{N}\right)\right) \qquad j = 1, \dots, N \qquad (2.15)$$

El torque generado por el motor se puede obtener a partir de la expresión (2.16), en la cual relaciona las corrientes de la fase, con la posición y las componentes de la matriz de inductancia *L* del motor (Badinez, 2007).

$$T_m(i_1, \dots, i_N, \theta) = \frac{1}{2}\bar{\imath}^T.\frac{\partial[L]}{\partial\theta}.\bar{\imath} \qquad (2.16)$$

Donde $\bar{\imath}$ es un vector de corriente, para el caso particular de un motor de imán permanente la expresión de torque queda expresada en la ec. (2.17).

$$T_m = -K_m \sum_{j=1}^{N} i_j sin\left(p\theta - \frac{(j-1)\pi}{N}\right) - K_d sin(4p\theta) \qquad (2.17)$$

Siendo p el número de polos del motor, K_m es un parámetro del motor y K_d representa el torque de detención debido a la interacción entre el rotor de imán permanente y el material ferromagnético de los polos del estator (Badinez, 2007).

Así pues, utilizando el resultado obtenido de las ec.(2.9), (2.10), (2.15) y (2.17), el modelo genérico del motor paso a paso del motor unipolar viene dado por el siguiente conjunto de ecuaciones:

$$\dot{\theta} = \omega \quad (2.18)$$

$$\dot{\omega} = \frac{1}{J}\left(T_m - T_c - T_v - T_f\right) \quad (2.19)$$

$$\frac{di_a(t)}{dt} = \frac{1}{L}(V_a(t) - i_a(t)R - \omega K_m sin(p\theta)) \quad (2.20)$$

$$\frac{di_b(t)}{dt} = \frac{1}{L}(V_b(t) - i_b(t)R - \omega K_m cos(p\theta)) \quad (2.21)$$

$$T_m = -K_m i_a sin(p\theta) + K_m i_b cos(p\theta) - K_d sin(4p\theta) \quad (2.22)$$

Se realizó una simulación para un motor paso a paso unipolar, con las siguientes características, los parámetros fueron tomados de Badinez (2007) y Morar (2003):

$N = 2$

$R = 20\ \Omega$

$L = 0.006\ H$

$K_m = 0.2\ N.\frac{m}{A}$

$K_d = \frac{k_m}{20}\ N.m$

$D = 0.008 \quad N.m.\frac{s}{rad}$

$J = 5e^{-5}\ N.m.\frac{s^2}{rad}$

$p = 54$

Con esta información es posible calcular el ángulo de paso del motor, donde el número de pasos por revolución del rotor es dado por:

$$S = 2Np \quad (2.23)$$

Y el ángulo de paso es:

$$\Delta\theta = \frac{360}{S} \quad (2.24)$$

$$\Delta\theta = 1.666°$$

V_a y V_b son la alimentación, para efectos de simulación ambas puede tener tres valores 0*volt* indica no hay paso de corriente por la bobina, 12*volt* hay paso de corriente en la bobina en un sentido y -12*volt* hay paso de corriente en la bobina en sentido contrario. Cabe destacar que en la figura 2.12 se muestra que solo hay una alimentación de $V+$ y así es en la realidad, pero en la simulación se utilizan con tres valores para emular el cambio de dirección de corriente al cerrar los interruptores.

El desplazamiento angular del motor unipolar simulado se puede observar en la figura 2.15, allí se evidencia que este motor se mueve por paso 1.6° y se mantiene allí mientras no cambien los pulsos de control del motor, como se había previsto con la ec.2.24. Así mismo, el torque generado por el motor viene dado en la figura 2.16, allí se evidencia el cambio de magnitud para equilibrar la carga del motor en el ángulo de paso. La secuencia de control para la activación en corrientes de las bobinas es dada en la figura 2.17, en la cual se aprecia el cambio de sentido de la corriente para lograr el movimiento adecuado del motor.

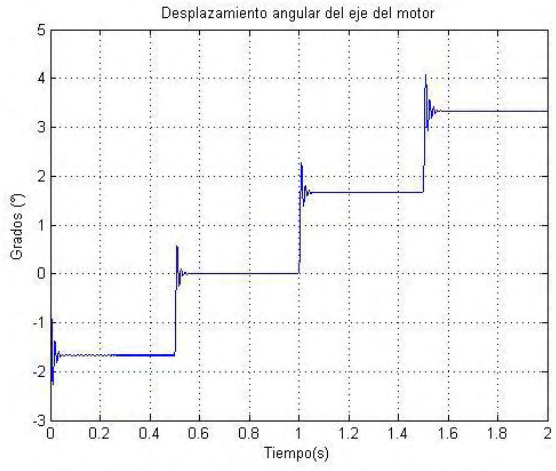
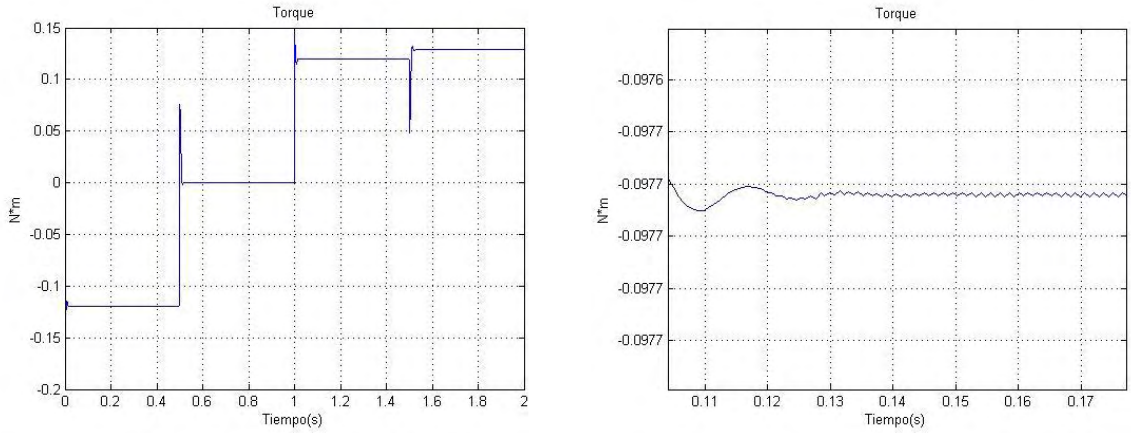
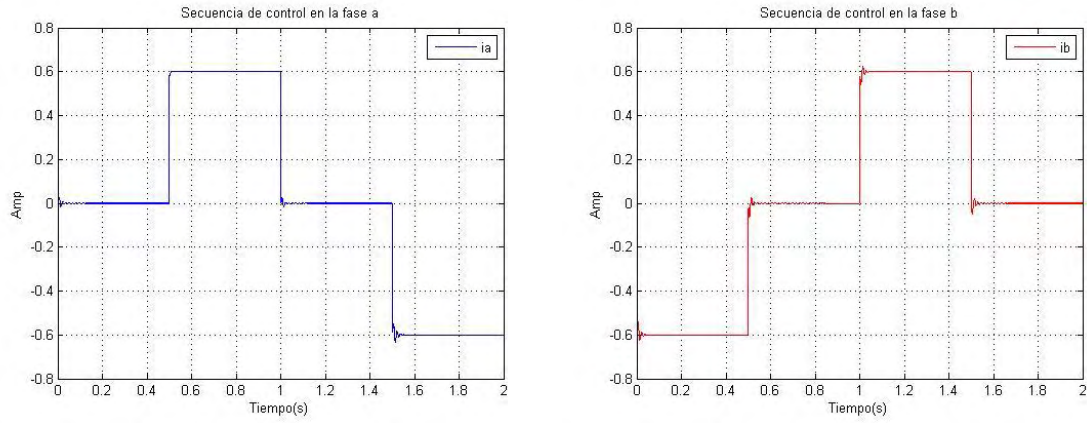


Figura 2. 15: Desplazamiento angular del eje del motor unipolar



(a) (b)

Figura 2. 16: (a) Torque generado por el motor paso a paso unipolar, (b) Acercamiento del torque generado por el motor paso a paso unipolar



(a) (b)

Figura 2. 17: (a) Corriente que circula por la bobina a, (b) Corriente que circula por la bobina b

2.4 Modelo de un potenciómetro

Se desea controlar el ángulo de desplazamiento del eslabón del exoesqueleto asociado a la falange proximal del dedo índice, para ello es necesario un dispositivo que permita la medición de esta variable. Una de las alternativas más adecuadas en este caso es el uso de un potenciómetro de rotación.

Un potenciómetro es un transductor electromecánico que convierte energía mecánica en energía eléctrica. Este elemento se alimenta con un voltaje constante, lo compone una resistencia variable que cambia de acuerdo al desplazamiento rotacional, a través del mismo se deriva una relación lineal, entre el voltaje medido en la resistencia variable y el movimiento mecánico realizado (Kuo, 1996). En la figura 2.18 se puede observar el circuito equivalente de un potenciómetro rotatorio, el cual se considera un sensor adecuado para la medición del desplazamiento angular.

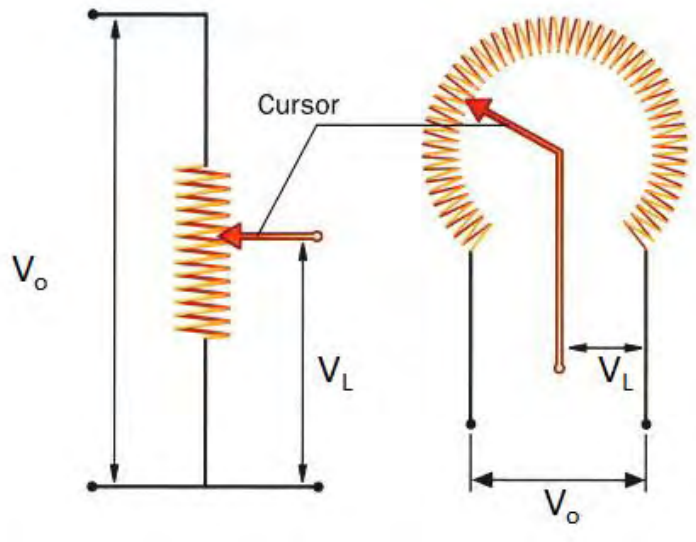


Figura 2. 18: Representación del circuito de un potenciómetro (Kuo 1996)

Una resistencia variable es un elemento lineal sobre el cual desliza un contacto eléctrico, capaz de inyectar corriente en un punto intermedio de su elemento resistivo (véase la figura 2.19).

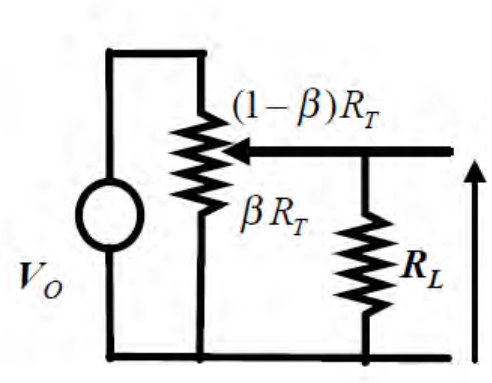


Figura 2. 19: Circuito equivalente de un potenciómetro rotatorio

Donde:

β es la posición relativa del cursor.

R_T es la resistencia total (Ω).

V_o es el voltaje de alimentación $(volt)$.

R_L es la resistencia que presenta el cursor (Ω).

V_o es el voltaje en la terminal variable $(volt)$.

Por medio de un divisor de tensión se puede determinar el voltaje en la terminal variable V_L mostrado en la figura 2.19, como se muestra a continuación:

$$V_L = \frac{\beta R_T // R_L}{(1-\beta)R_T + \beta R_T // R_L} V_o = \frac{\beta R_L}{(\beta - \beta^2)R_T + R_L} V_o \qquad (2.25)$$

Si $R_L \gg R_T$

$V_L \approx \beta V_o$

Capítulo 3

Diseño del Sistema de Control

Para realizar control sobre un proceso existen diferentes elementos que interactúan entre sí para lograr este objetivo, así como también diferentes esquemas del sistema controlado, por ello se debe definir previamente la estructura a utilizar. En la figura 3.1 se define el esquema de control escogido en el presente trabajo, con el objetivo de definir cada uno de los elementos que lo componen.

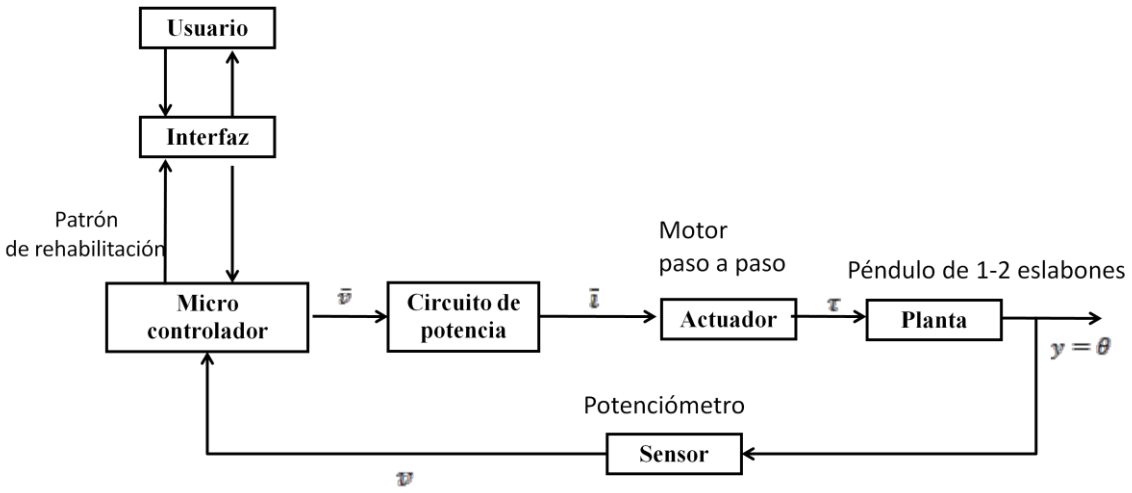


Figura 3. 1: Esquema del Sistema de Control

El usuario introducirá al programa una serie de parámetros que permitirán fijar el patrón de rehabilitación, esa información es procesada por la aplicación que decide que instrucciones enviar al microcontrolador. La salida del microcontrolador es un vector de cuatro voltajes (0*volt* o 1*volt*) que indica la activación o no de los interruptores del circuito de potencia. Las corrientes que circulan por el circuito de potencia son enviadas al actuador para ser traducidas a un torque en el exoesqueleto. El desplazamiento angular es medido por el sensor para ser realimentada al algoritmo de control y así continuar con la terapia de manera correcta (véase la figura 3.1).

A continuación se describen los diferentes componentes del sistema de control.

3.1 Diseño mecánico del exoesqueleto robótico

El exoesqueleto robótico debe permitir la interacción o acoplamiento con el dedo a rehabilitar, dicha interacción no debe afectar a la persona que lo está utilizando, es decir el diseño se debe realizar de manera adecuada, para facilitar su uso.

Es necesario diseñar un prototipo del dispositivo y realizar una aproximación adecuada de un modelo dinámico del sistema acoplado. El modelo obtenido va a permitir realizar estudios para cumplir los objetivos previamente definidos.

Para el diseño del exoesqueleto robótico se toman en cuenta las características mencionadas en la tabla 2.1, hay que considerar que la longitud de la falange proximal del dedo índice varía dependiendo de la persona, al igual que la falange media. Tomando en cuenta las dimensiones promedio, se diseñó una estructura a la cual se le pueda variar la longitud de la falange proximal y así garantizar que el dispositivo lo puedan usar diferentes personas.

Mediante el uso del paquete modelado paramétrico de sólidos en 3D **Autodesk Inventor**, se diseñó pieza por pieza los elementos que conforman el exoesqueleto robótico, en la figura 3.2 se muestran las piezas ensambladas. Para más detalle de los planos de cada pieza ver el Anexo A.

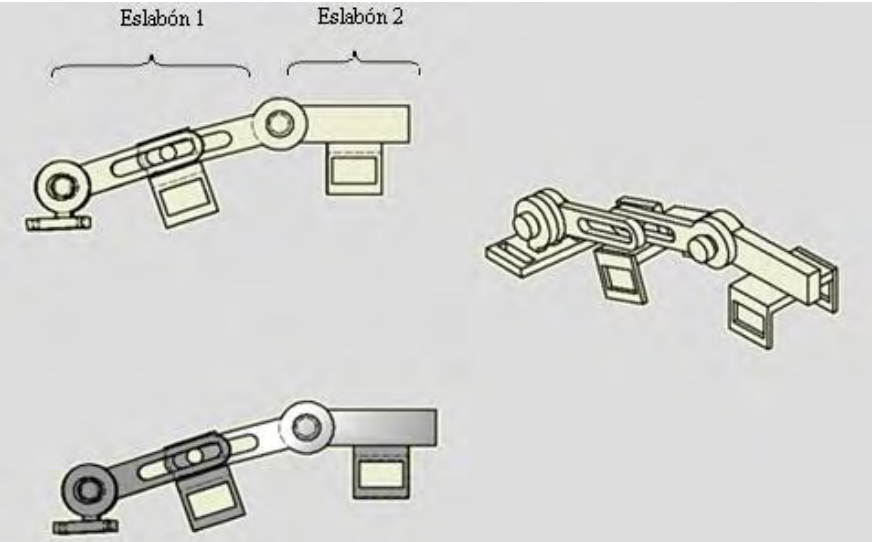


Figura 3. 2: Prototipo diseñado del exoesqueleto robótico

El prototipo se ajusta a la mano como se muestra en la figura 3.3, las articulaciones del dedo índice están asociadas a los grados de libertad de los eslabones.

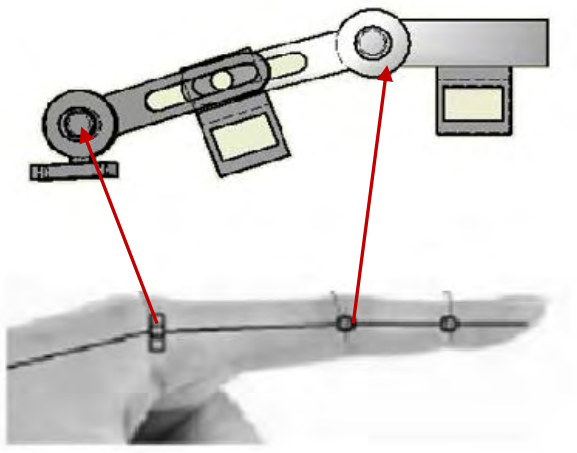


Figura 3. 3: Acople de las articulaciones del dedo índice con el exoesqueleto robótico

El prototipo consta de los siguientes elementos identificados en la figura 3.4:

- La pieza A es la base del exoesqueleto que se fija a una base de madera, dicha pieza tiene dos orificios los cuales permitirán el ajuste con la madera por medio de tornillos, también representa la articulación de la falange proximal ya que se une con el primer eslabón.
- Las piezas B y D van a permitir la reducción o ampliación de la longitud del eslabón 1, para el ajuste de la falange proximal de cada persona.
- La pieza C está compuesta por un tornillo que se ajusta a través de una tuerca, unidos al él se encuentran dos eslabones más pequeños B y D que se corren de acuerdo al tamaño de la falange.
- El eslabón 2 está representado por E, ya que es la pieza terminal del prototipo no se considera la variación de longitud, su tamaño final viene dado por el promedio de longitud de la falange media de las personas.
- Por último el prototipo dispone de dos piezas muy semejantes señaladas en F, la función principal de estas piezas es acoplar el dedo índice con el exoesqueleto diseñado, semejante a un carril, el dedo se introduce y se fija una cinta ajustable.

Con el apoyo del grupo de investigación DIMMA de la Escuela de Ingeniería Mecánica de la Universidad de los Andes, se implementa el primer prototipo del exoesqueleto, que consta de dos eslabones para realizar los estudios pertinentes a esta investigación. El resultado obtenido se observa en la figura 3.5.

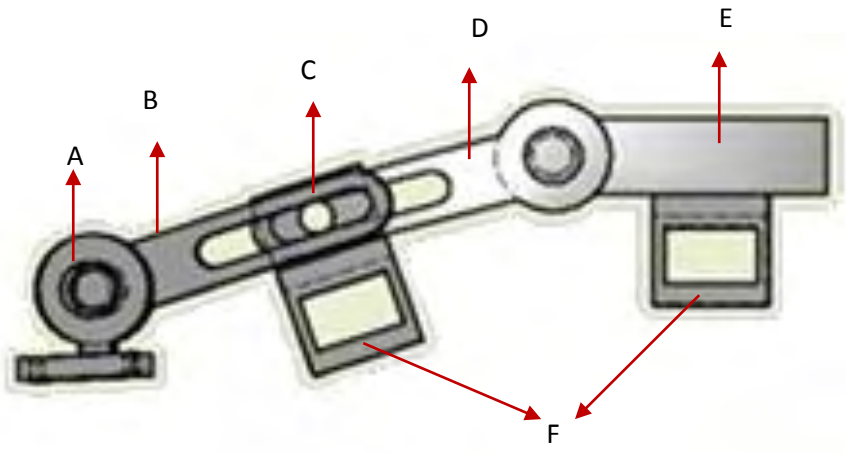


Figura 3. 4: Elementos que conforman el exoesqueleto robótico diseñado

Figura 3. 5: Prototipo del exoesqueleto robótico de dos eslabones

3.2 Diseño del sensor

Es necesario medir el desplazamiento angular de la falange proximal del dedo índice para su posterior control. Un sensor adecuado es un potenciómetro rotacional, el mismo fijado en el eje de rotación del exoesqueleto permitirá establecer una relación lineal entre el ángulo y el voltaje de salida de la resistencia variable, tal y como se explicó en el modelado de la sección 2.4. Dicha relación va a permitir asociar el voltaje de salida al ángulo de la falange proximal, con el fin realizar la acción correspondiente a esa salida.

En la figura 3.6 se muestra el potenciómetro escogido, la selección se hizo considerando sensibilidad al movimiento del eje. En la figura 3.7 se puede observar el sensor acoplado a la planta.

Figura 3. 6: Potenciómetro seleccionado como sensor del sistema

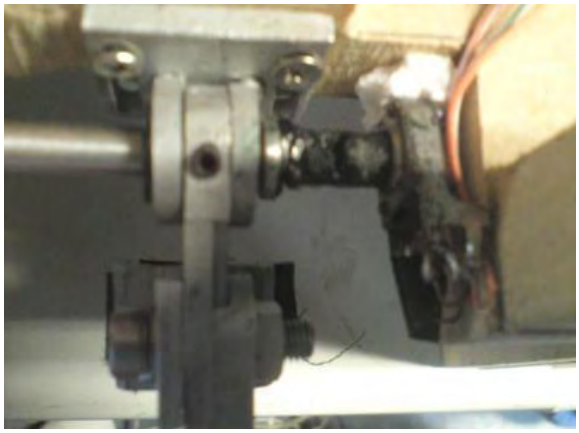


Figura 3. 7: Prototipo del exoesqueleto robótico de un eslabón, con el potenciómetro acoplado al eje de rotación

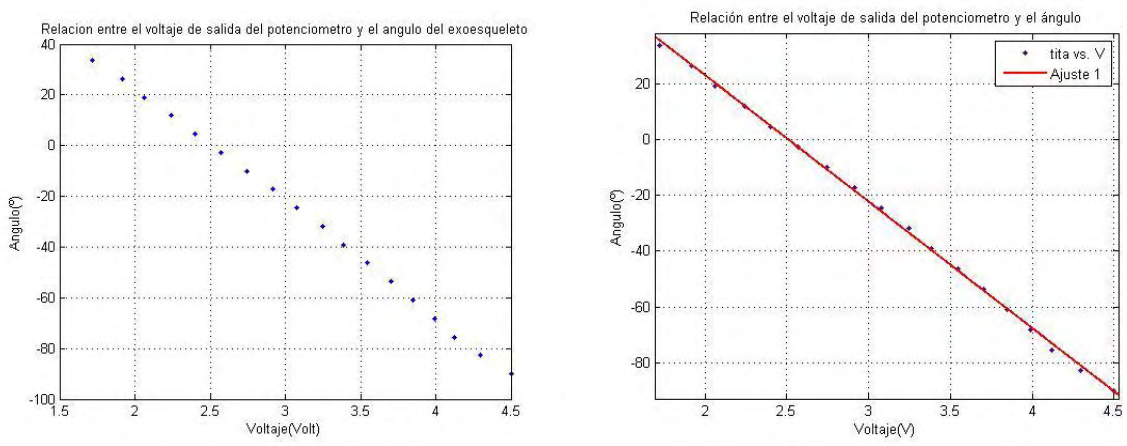
Para conocer la relación lineal se utilizó una tarjeta de adquisición de datos LabJack U3, la cual permite muestrear en tiempo real el voltaje de salida del sensor para cierto ángulo, una vez tomadas las mediciones se almacenaron en la tabla 3.1. El rango de valores de voltajes que se presenta fue escogido así, ya que el potenciómetro se calibró de manera que, para el ángulo mínimo de -90º el voltaje fuera el máximo y para el ángulo máximo 60º el voltaje fuera el mínimo.

Tabla 3. 1: Datos adquiridos del voltaje de salida del potenciómetro y del ángulo medido

Voltaje(Volt)	Ángulo(º)
4.5	-90.0000
4.2960	-82.7273
4.1230	-75.4546
3.9910	-68.1819
3.8480	-60.9092
3.7060	-53.6365
3.5480	-46.3638
3.3850	-39.0911
3.2470	-31.8184
3.0790	-24.5457
2.9160	-17.2730
2.7480	-10.0003
2.5700	-2.7276
2.4020	4.5451
2.2440	11.8178
2.0610	19.0905
1.9190	26.3632
1.7200	33.6359

Con los datos obtenidos en la tabla 3.1, es posible obtener una aproximación lineal para explicar este comportamiento (véase la figura 3.8), para ello se empleó la herramienta

cftool de **Matlab**, la cual utiliza el método de los mínimos cuadrados para realizar aproximaciones de acuerdo a muestras tomadas en varios puntos.



(a) (b)

Figura 3. 8: (a) Relación entre el voltaje de salida del potenciómetro y el ángulo del primer eslabón, (b) Recta de ajuste a la relación entre el voltaje de salida del potenciómetro y el ángulo del primer eslabón.

La recta obtenida por la aproximación lineal se muestra en la ec. (3.1).

$$\theta(V) = -45.23V + 119.6 \qquad (3.1)$$

Con este resultado se puede saber cuál es el ángulo del eslabón para un voltaje específico dado por el sensor.

3.3 Sistema de control para el exoesqueleto de un eslabón

A continuación se presenta el estudio del motor paso a paso, como elemento actuador del exoesqueleto. También las diferentes pruebas que se realizaron para la validación del modelo y las propuestas para el estudio de dos eslabones mediante simulaciones.

3.3.1 Análisis del motor

El motor unipolar seleccionado es usado para el movimiento del cabezal de una impresora, disponible en el laboratorio de control de la Escuela de Sistemas de la Universidad de los Andes, el mismo no tiene especificaciones técnicas debido a su antigüedad. Es por ello que se plantean las siguientes características, que posteriormente serán validadas con la identificación:

$R = 20\ \Omega$

$L = 0.006\ H$

$K_m = 0.1\ N\frac{m}{A}$

$K_d = \frac{km}{20}$

$D = 0.002\ N.m.\frac{s}{rad}$

$J = 5e^{-5}\ N.m.\frac{s^2}{rad}$

$N = 2$

$p = 16$

$S = 2Np = 64$

$\Delta\theta = \frac{360}{S} = 5.625°$

Con el uso de la ecuaciones (2.18)-(2.22), se plantea el modelo dinámico del motor paso a paso.

$$\dot{\theta} = \omega \tag{3.2}$$

$$\dot{\omega} = \frac{1}{J}\left(T_m - T_c - T_v - T_f\right) \tag{3.3}$$

$$\frac{di_a(t)}{dt} = \frac{1}{0.006}\left(V_a(t) - 20i_a(t) - 0.2\omega sin(p\theta)\right) \tag{3.4}$$

$$\frac{di_b(t)}{dt} = \frac{1}{0.006}(V_b(t) - 20i_b(t) - 0.2\omega cos(p\theta)) \quad (3.5)$$

$$T_m = -0.2i_a sin(12\theta) + 0.2i_b cos(12\theta) - 0.01sin(4p\theta) \quad (3.6)$$

Cabe destacar que para realizar estas primeras pruebas se consideran despreciables los torques de fricción estática y de carga del motor paso a paso, además los valores de los parámetros del motor son estimados ya que no se cuenta con las especificaciones técnicas del mismo.

Utilizando el modelo genérico de un motor paso a paso unipolar y realizando simulaciones en **Matlab**, se pueden observar los comportamientos de todas las variables del sistema en las figuras 3.9 a 3.13, evidenciando que con estas características el motor se mueve por paso aproximadamente 5.6°.

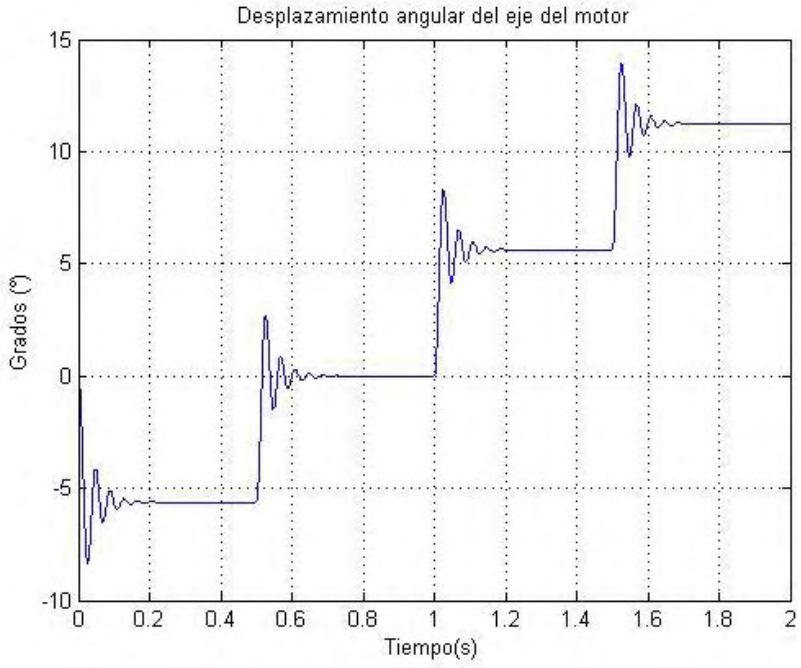


Figura 3. 9: Desplazamiento **a**ngular del eje del motor paso a paso

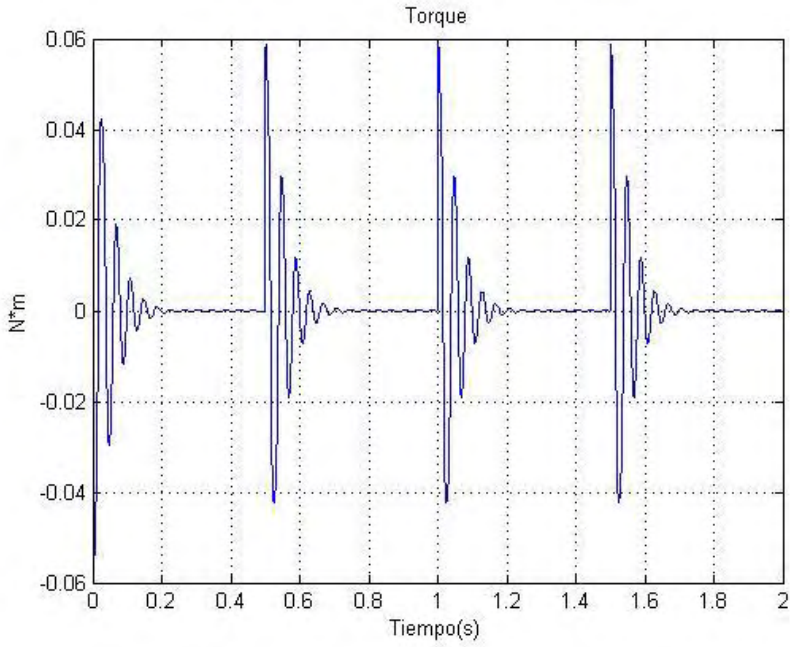


Figura 3. 10: Torque generado por el motor paso a paso

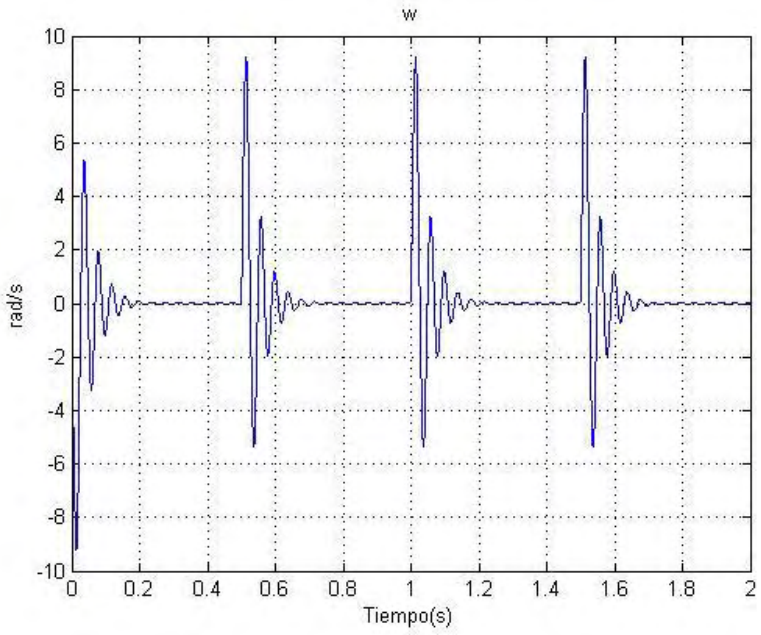


Figura 3. 11: Velocidad angular del eje del motor paso a paso

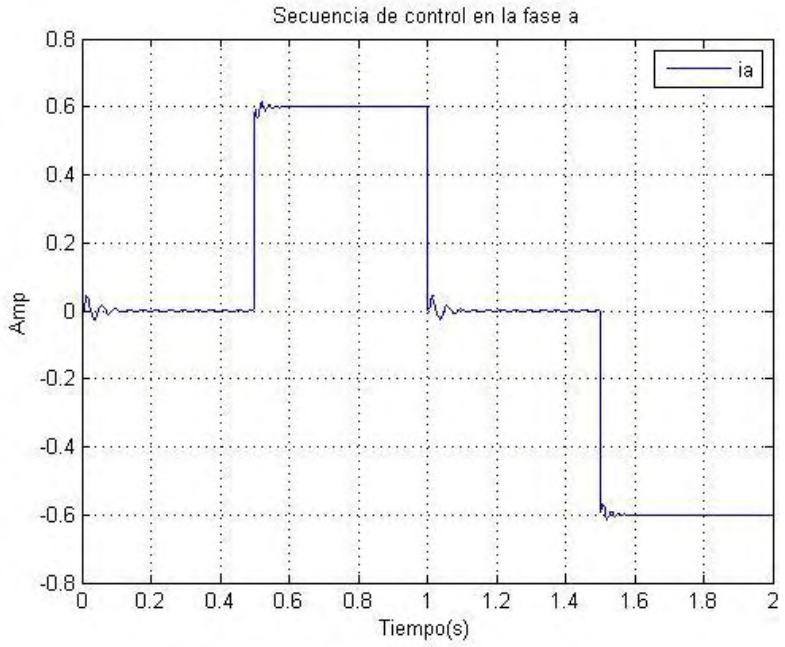


Figura 3. 12: Corriente que circula por la bobina a

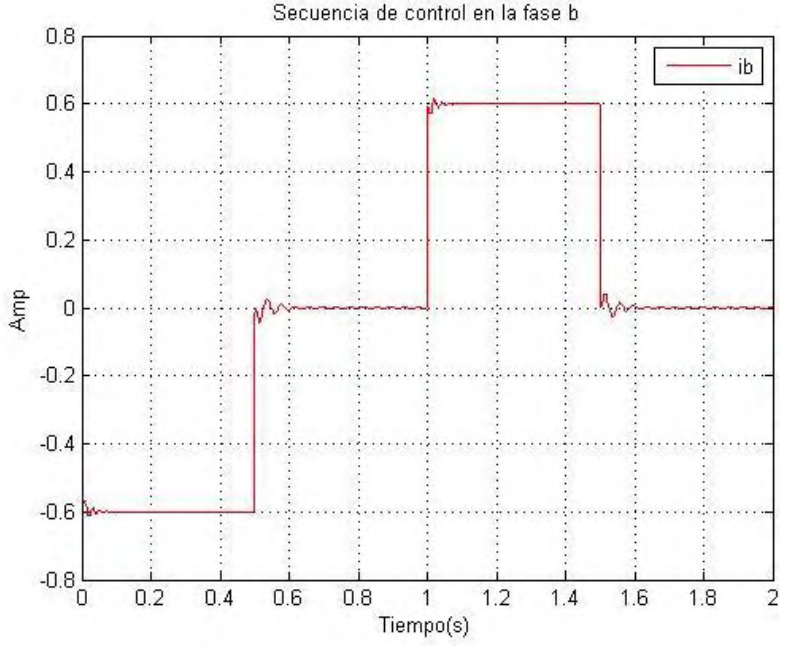


Figura 3. 13: Corriente que circula por la bobina b

El motor paso a paso no se pudo comparar con valores experimentales, debido a que no fue posible acoplar el potenciómetro en el eje del motor, sin embargo, se verá más adelante que los parámetros ajustados son los adecuados para el modelo del motor.

3.3.2 Análisis del motor con el exoesqueleto acoplado

Se acopló el motor paso a paso al eje del exoesqueleto del primer eslabón como se muestra en la figura 3.14, para realizar las pruebas pertinentes.

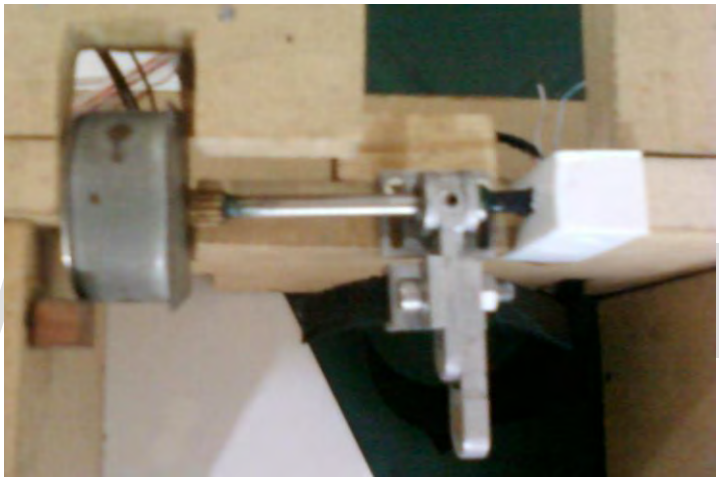


Figura 3. 14: Motor paso a paso acoplado al eje del primer eslabón del exoesqueleto

Una vez acoplado el eslabón con el sensor, es posible adquirir datos con el motor energizado y con una secuencia de control especificada.

Por medio de la tarjeta de adquisición de datos LabJack U3, se obtuvo el ángulo de desplazamiento de una secuencia de movimiento y se realizó la comparación del modelo obtenido con los datos adquiridos, para ello se modificó el modelo del motor agregando el torque de carga y la inercia del péndulo de un eslabón, como se muestran en la ec. (3.8). Se sigue despreciando el torque de fricción estática T_v .

$m_1 = 0.03\ kg$

$lc_1 = \frac{l}{2} = 0.03m$

$b = 0.005$

$J_1 = 1.5417e^{-5} kg.m^2$

$$\dot{\theta} = \omega \tag{3.7}$$

$$\dot{\omega} = \frac{1}{(J + J_1)}\left(T_m - (b + D)w - m_1 g lc_1 cos(\theta)\right) \tag{3.8}$$

$$\frac{di_a(t)}{dt} = \frac{1}{0.006}(V_a(t) - 20i_a(t) - 0.2\omega sin(p\theta)\ ) \tag{3.9}$$

$$\frac{di_b(t)}{dt} = \frac{1}{0.006}(V_b(t) - 20i_b(t) - 0.2\omega cos(p\theta)\ ) \tag{3.10}$$

$$T_m = -0.2i_a sin(12\theta) + 0.2i_b cos(12\theta) - 0.01sin(4p\theta) \tag{3.11}$$

Se energizó el motor paso a paso con la secuencia de activación mostrada en la figura 3.15, la cual permite que el eje del motor gire hacia la derecha y el eslabón se desplace hacia arriba, con un tiempo de ancho de pulso de 500ms. Cabe destacar que se hace uso de la secuencia de torque elevado expuesta en la tabla 2.1.

En este caso, la actuación se realiza mediante un puerto paralelo, del mismo se toman 4 salidas digitales las cuales llegan al circuito activador de las fases, comúnmente llamado circuito de potencia (véase la figura 3.16). La función de este circuito es la activación de las fases de la bobina según el valor proveniente de los pines 2,3 ,4 y 5 (0V o 1V), que permitirá emular el cierre o apertura de los interruptores.

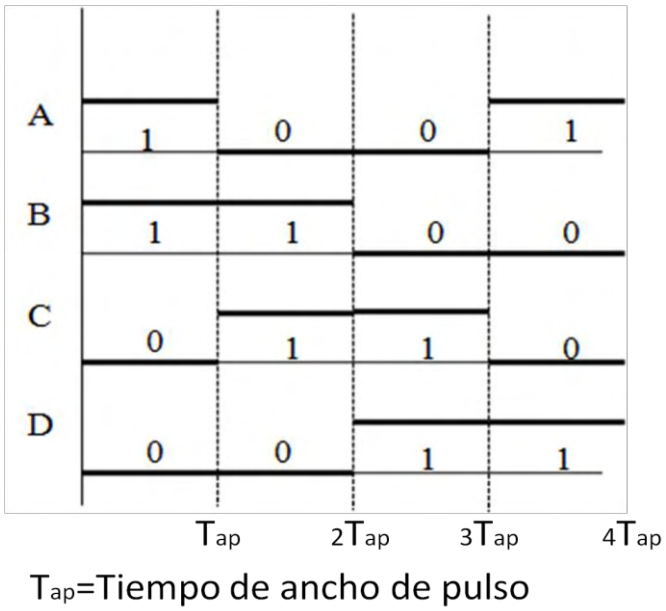


Figura 3. 15: Secuencia de activación de las fases del motor para el movimiento hacia la derecha del eje

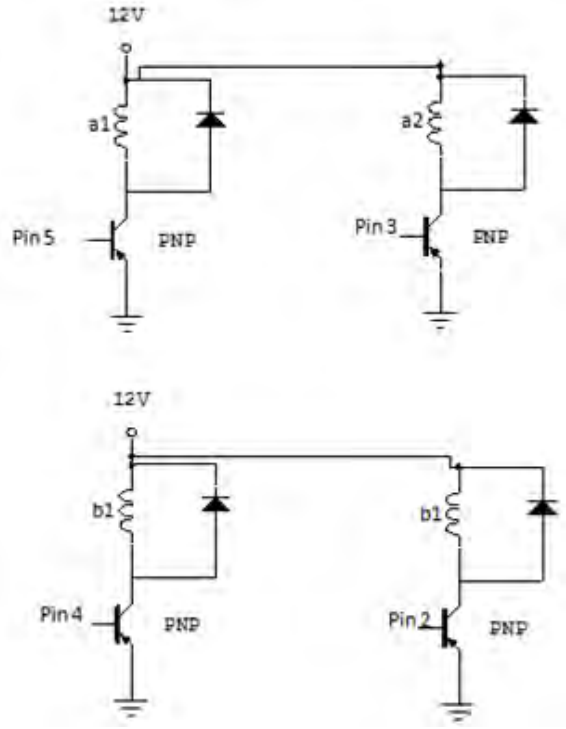


Figura 3. 16: Circuito que permite el control del motor paso a paso

En la figura 3.17 se puede observar el circuito implementado con el motor y el puerto paralelo.

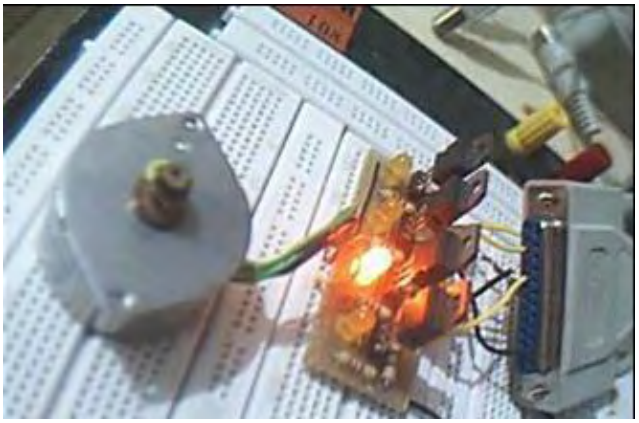


Figura 3. 17: Circuito de potencia implementado para el motor paso a paso

Para controlar el movimiento del motor en la fase de prueba del modelo, se realizó una interfaz de comunicación con el puerto paralelo por medio de la herramienta gráfica **GUI** de **Matlab**, la misma permite especificar el tiempo de activación de las fases y la dirección que se desea. En la figura 3.18 se muestra el esquema de la misma.

Figura 3. 18: Interfaz de comunicación con el puerto paralelo

Se procedió a conectar la salida del sensor con la tarjeta de adquisición de datos, con condición inicial de -90° y un periodo de muestreo de 0.001s. El periodo de muestreo fue escogido analizando la figura 3.9 a partir de su tiempo de respuesta, a fin de satisfacer el teorema de Shannon.

Se comparan los datos adquiridos con la simulación realizada, como se muestra en la figura 3.19, se puede observar que ambas gráficas se ajustan al inicio del movimiento, a medida que va pasando el tiempo se va observando una diferencia. Puede atribuirse esta diferencia al efecto de la gravedad sobre el eslabón y a la potencia requerida por el motor en este rango, sin embargo la aproximación del modelo es adecuada.

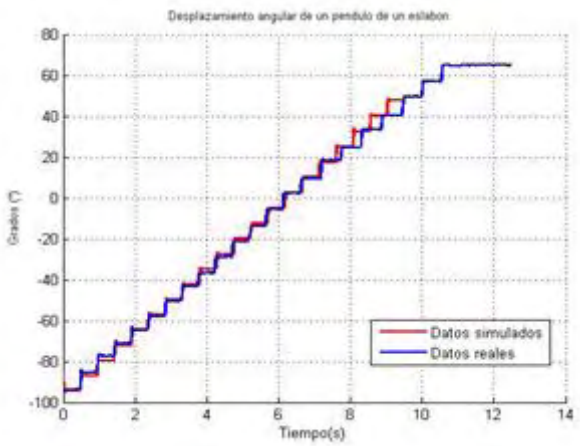


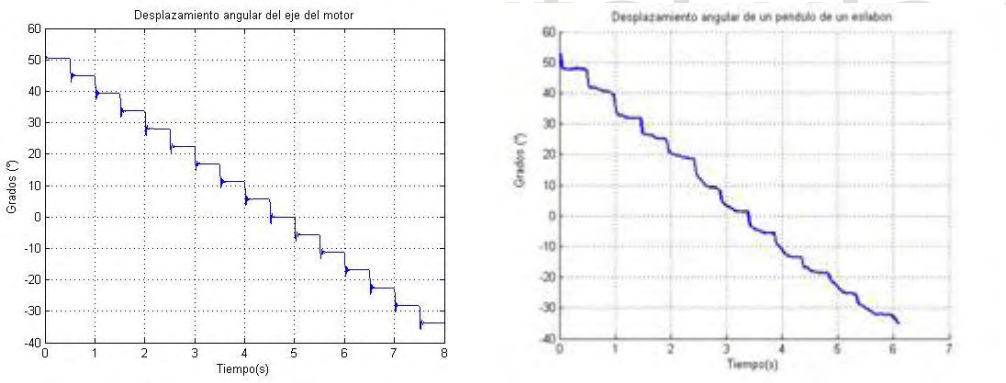
Figura 3. 19: Desplazamiento angular de un solo eslabón valores simulados y valores reales

El resultado obtenido en este apartado permite validar el modelo de la sección 3.3.1, ya que se utilizan los mismos parámetros del motor PAP para las simulaciones. Si bien es cierto que, la dinámica del eslabón afecta el desplazamiento angular del eje del motor, como la masa es relativamente pequeña, se puede concluir que los parámetros supuestos del modelo del motor PAP son adecuados.

3.3.3 Análisis del motor con el exoesqueleto acoplado y el dedo índice

Una vez analizado y validado el modelo del exoesqueleto controlado con el motor, es preciso realizar pruebas con el dedo acoplado al exoesqueleto, para observar el comportamiento del ángulo del eslabón y considerar los cambios que le implicarían al modelo.

Al efectuar la medición del desplazamiento angular del exoesqueleto con el dedo índice, se obtiene el comportamiento dado en la figura 3.20, es evidente que el modelo planteado por las ec. (3.7)-(3.11) para el motor con la planta no se ajusta a este nuevo comportamiento. Es por ello que se plantea modificar los parámetros del péndulo de un eslabón.



(a) (b)

Figura 3. 20: (a) Desplazamiento angular del motor acoplado con la planta valores simulados (b) Desplazamiento angular del motor con la planta y el dedo índice integrado valores identificados

La suposición efectuada es que el dedo le esta añadiendo mas fricción al movimiento del exoesqueleto, se modificó el modelo dinámico cambiando $b = 0.08$, en la figura 3.21 se puede observar el resultado obtenido.

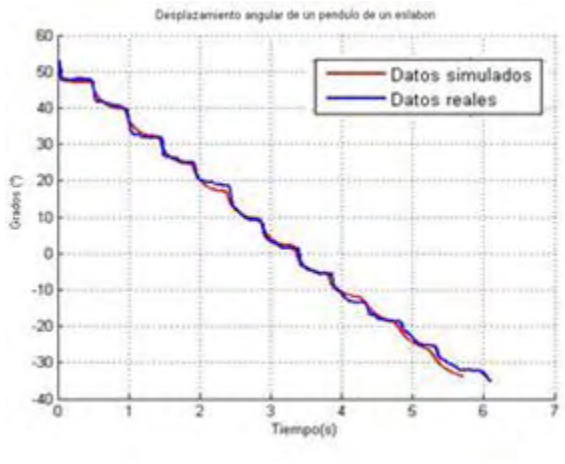


Figura 3. 21: Desplazamiento angular del motor con la planta y el dedo índice integrado datos simulados y reales con *b=0.08*

El valor de *b* fue conseguido realizando distintas pruebas sobre el modelo dinámico y comparando simultáneamente con los datos adquiridos, se puede observar que la aproximación realizada es adecuada ya que se ajusta al comportamiento de los datos reales.

Se realizó la prueba con una persona diferente, en la figura 3.22 se puede observar el resultado obtenido, igual que el caso anterior se modificó el coeficiente de fricción *b* y se fue ajustando de acuerdo al comportamiento adquirido de la tarjeta de adquisición.

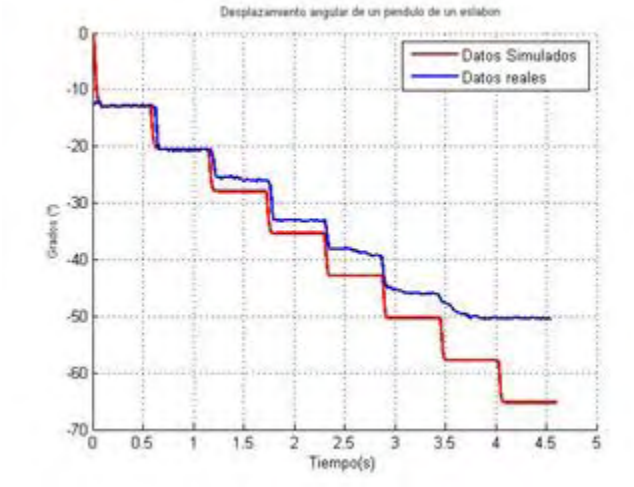


Figura 3. 22: Desplazamiento angular del motor con la planta y el dedo índice integrado datos simulados y reales con *b=0.06*

Es de notar en las figuras 3.21 y 3.22 la diferencia del comportamiento del ángulo de la falange proximal y como se ajusta el modelo para cada dedo, es posible que se pueda ajustar el *b* del modelo dentro de un rango para un grupo de personas o un promedio para lograr una mejor aproximación. Se debe emplear un estudio estadístico, para definir el comportamiento del coeficiente de fricción, con el fin de generalizar el modelo.

3.4 Análisis dinámico con dos eslabones:

Para adicionar el comportamiento del segundo eslabón integrado al exoesqueleto en el modelo dinámico, se realizarán simulaciones donde se van a considerar las siguientes premisas que servirán de apoyo a investigaciones futuras: (i) El segundo eslabón se considera como una masa concentrada. (ii) El segundo eslabón se considera actuado con un motor paso a paso, análogo al utilizado en el primer eslabón.

i. Segundo eslabón como una masa concentrada

En este caso se puede fijar el segundo eslabón a cierto ángulo y considerarse como una masa concentrada que genera un torque adicional al primer eslabón (véase la figura 3.23). Por lo tanto el modelo presentado de un eslabón con el motor integrado en la sección 3.3.2, se modifica agregando el torque generado por el segundo eslabón como se muestra en la ec.(3-13).

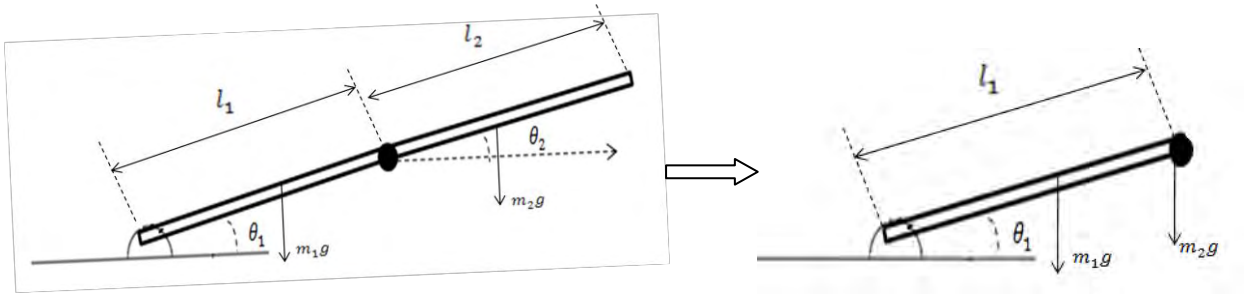


Figura 3. 23 : Péndulo doble, segundo eslabón como una masa concentrada en el extremo del primer eslabón

$$\dot{\theta} = \omega \tag{3.12}$$

$$\dot{\omega} = \frac{1}{(J + J_1 + J_2)}(T_m - (b + D)w - m_1 g l c_1 cos(\theta) - Ts) \tag{3.13}$$

$$\frac{di_a(t)}{dt} = \frac{1}{0.006}(V_a(t) - 20i_a(t) - 0.2\omega sin(p\theta)) \tag{3.14}$$

$$\frac{di_b(t)}{dt} = \frac{1}{0.006}(V_b(t) - 20i_b(t) - 0.2\omega cos(p\theta)) \tag{3.15}$$

$$T_m = -0.2i_a sin(12\theta) + 0.2i_b cos(12\theta) - 0.01sin(4p\theta) \tag{3.16}$$

Donde *Ts* es el torque generado por el peso del segundo eslabón hacia el primer eslabón. El ángulo al cual es fijado el segundo eslabón no es conocido, ya que depende de cómo lo coloca el terapista ocupacional según el paciente. Actualmente no se cuenta con un sensor en esa articulación. Una aproximación válida es considerar que la masa del segundo eslabón está concentrada en el extremo del primer eslabón, de esta manera *Ts* queda expresado en la ec.(3.17).

$$T_s = m_2 g l_1 cos(\theta) \tag{3.17}$$

Se energizó el motor paso a paso con la secuencia de activación mostrada en la figura 3.15, la cual permite que el eje del motor gire hacia la derecha y el eslabón se desplace hacia arriba, con un tiempo de ancho de pulso de 500ms.

En la figura 3.24 se muestra el resultado obtenido simulando el eslabón con el torque integrado y los valores obtenidos de realizar el proceso de identificación del ángulo del primer eslabón. En la figura 3.24 se evidencia que el segundo eslabón, previamente fijado en un ángulo, no afecta el comportamiento de la primera articulación integrada con el motor paso a paso, ya que el valor final es el mismo. Sin embargo, en la figura 3.25 se pueden apreciar variaciones en el estado transitorio por cada paso realizado por el motor, donde influyen varios factores como la gravedad y la dinámica del segundo eslabón que se está simplificando a un torque aplicado al primer eslabón.

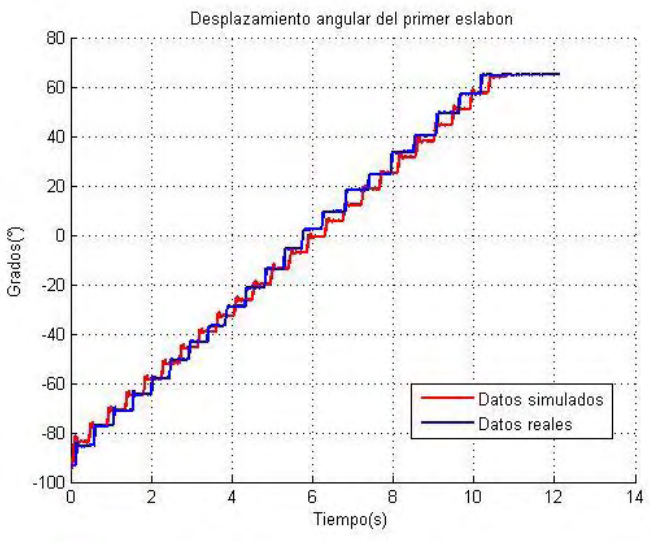


Figura 3. 24: Desplazamiento angular del primer eslabón con el torque *Ts* integrado valores simulados y valores reales

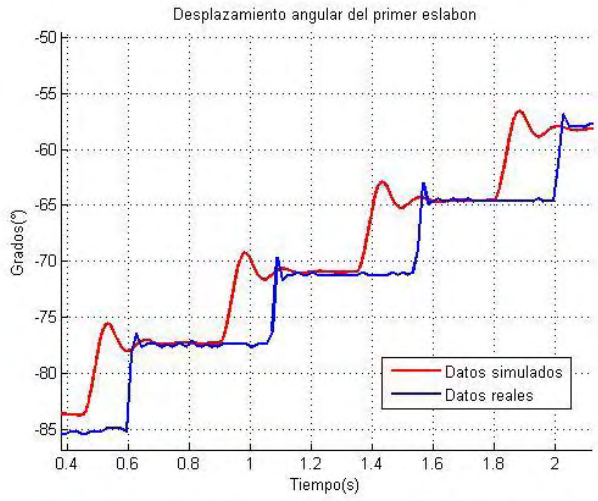


Figura 3. 25: Acercamiento del desplazamiento angular del primer eslabón con el torque *Ts* integrado valores simulados y valores reales

ii. Motor en el segundo eslabón

En este caso se considera que se tienen dos motores paso a paso en el exoesqueleto robótico, uno en la primera articulación y uno en la segunda. Para simular el comportamiento de los

dos eslabones con dos motores paso a paso se realizan las siguientes pruebas: (a) Se considera una dinámica a dos etapas. (b) Ambas articulaciones se mueven simultáneamente.

a. Dinámica a dos etapas

Se piensa en un modelo a dos etapas, la primera etapa mueve el primer eslabón y la siguiente que mueve el segundo eslabón. El movimiento del segundo eslabón está sujeto a la estabilización del primero y viceversa.

Con dicha suposición se podría realizar la siguiente terapia: Se mueve el primer eslabón un ángulo fijado, seguidamente se mueve el segundo eslabón y así sucesivamente hasta conseguir una secuencia de repeticiones deseadas. En la figura 3.26 se ilustra el movimiento de los eslabones.

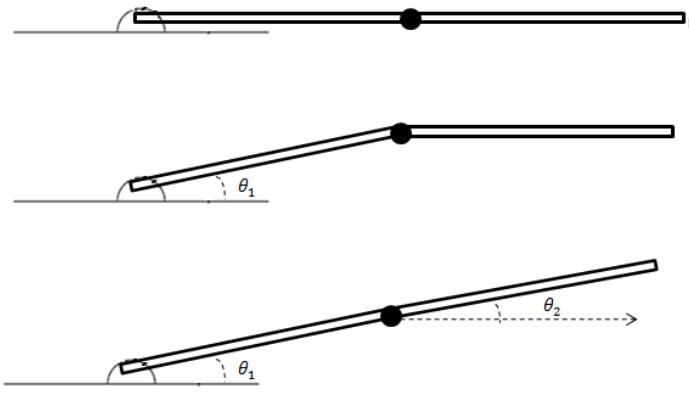


Figura 3. 26: Secuencia de movimientos de dos eslabones para una terapia

Se trabaja con el modelo del péndulo de un eslabón para la primera articulación, para la segunda articulación con el mismo modelo, pero con los parámetros correspondientes al segundo eslabón del exoesqueleto.

Se hace la suposición que ambos modelos no están acoplados. Esta aproximación es admisible ya que para el movimiento del segundo eslabón el primero ya se encuentra en estado estacionario. Sin embargo, en la realidad ambos están relacionados uno con el otro.

Los parámetros de los modelos son los expuestos en la sección 3.3.2, se utilizan dos motores iguales para ambas articulaciones y adicionalmente se definen los parámetros de la segunda articulación.

$m_2 = 0.02kg$

$lc_2 = \frac{l_2}{2} = 0.015m$

$b_2 = 0.002$

$J_2 = 2.5x10^{-6}\ kg.m^2$

Ecuaciones utilizadas para simular el primer eslabón:

$$\dot{\theta}_1 = \omega_1 \tag{3.18}$$

$$\dot{\omega}_1 = \frac{1}{(J + J_1 + J_2)}(T_{m1} - (b_1 + D)w_1 - m_1 g lc_1 cos(\theta_1) - T_s) \tag{3.19}$$

$$\frac{di_{a1}(t)}{dt} = \frac{1}{0.006}(V_{a1}(t) - 20i_{a1}(t) - 0.2\omega_1 sin(p\theta_1)) \tag{3.20}$$

$$\frac{di_{b1}(t)}{dt} = \frac{1}{0.006}(V_{b1}(t) - 20i_{b1}(t) - 0.2\omega_1 cos(p\theta_1)) \tag{3.21}$$

$$T_{m1} = -0.2i_{a1}sin(12\theta_1) + 0.2i_{b1}cos(12\theta_1) - 0.01sin(4p\theta_1) \tag{3.22}$$

Ecuaciones utilizadas para simular el segundo eslabón:

$$\dot{\theta}_2 = \omega_2 \tag{3.23}$$

$$\dot{\omega}_2 = \frac{1}{(J + J_2)}\left(T_{m2} - (b_2 + D)w_2 - m_2 g l c_2 cos(\theta_2)\right) \quad (3.24)$$

$$\frac{di_{a2}(t)}{dt} = \frac{1}{0.006}\left(V_{a2}(t) - 20i_{a2}(t) - 0.2\omega_2 sin(p\theta_2)\right) \quad (3.25)$$

$$\frac{di_{b2}(t)}{dt} = \frac{1}{0.006}\left(V_{b2}(t) - 20i_{b2}(t) - 0.2\omega_2 cos(p\theta_2)\right) \quad (3.26)$$

$$T_{m2} = -0.2i_{a2}sin(12\theta_2) + 0.2i_{b2}cos(12\theta_2) - 0.01sin(4p\theta_2) \quad (3.27)$$

En la figura 3.27 se observa el resultado obtenido y se evidencia el comportamiento deseado por la figura 3.26. Inicialmente ambos eslabones están en la condición inicia con 0º en ambas articulaciones, cuando el primer eslabón está en estado estacionario se realiza el movimiento del segundo eslabón. Al alcanzar el estado estacionario el segundo eslabón se procede al movimiento del primer eslabón y así sucesivamente se puede realizar esta repetición hasta obtener los ángulos deseados para la terapia (Para más detalle acerca de los programas revisar Apéndice C).

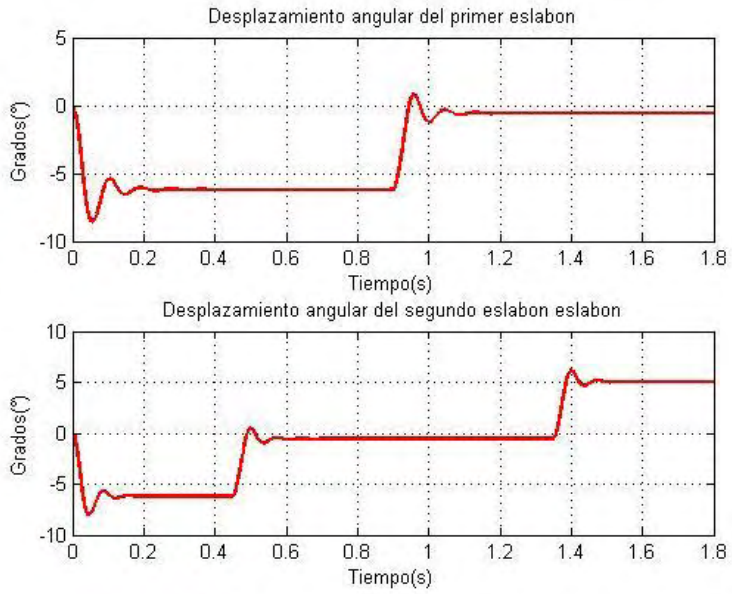


Figura 3. 27: Comportamiento del ángulo de ambas articulaciones con la terapia (a) propuesta

b. Ambas articulaciones moviéndose simultáneamente

Ambas articulaciones se muevan simultáneamente con motores paso a paso en cada articulación.

Esta segunda podría responder a patrones de rehabilitación más complejos a implementar en trabajos futuros.

En este caso se utiliza el modelo de dos eslabones, ya que el diseño mostrado en la figura 3.2 evidencia que básicamente el prototipo consta de dos eslabones, que fácilmente pueden ser representados por dos barras rígidas unidas. La configuración de un péndulo doble (véase la figura 3.28) permite emular el comportamiento del exoesqueleto robótico presentado.

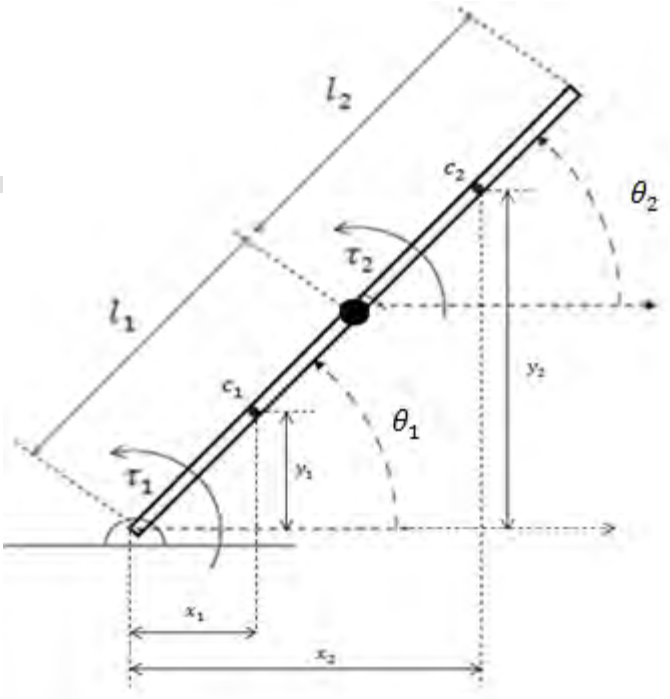


Figura 3. 28: Péndulo doble análogo al exoesqueleto robótico

Donde:

m_1 es la masa del primer eslabón (kg).

c_1 es el centro de masa del primer eslabón (m).

l_1 es la longitud de m_1 (m).

l_{c1} es la longitud del centro de masa del primer eslabón (m).

θ_1 es el desplazamiento angular de m_1 (°).

J_1 es la inercia de m_1 $(kg.m_2)$.

m_2 es la masa del segundo eslabón (kg).

c_2 es el centro de masa del segundo eslabón (m).

l_2 es la longitud de m_2 (m).

l_{c2} es la longitud del centro de masa del segundo eslabón (m).

θ_2 es el desplazamiento angular de m_2 (°).

J_2 es la inercia de m_2 $(kg.m^2)$.

La elaboración del modelo dinámico del péndulo doble se basó en las ecuaciones de movimiento de Lagrange para un sistema conservativo,

$$\frac{d}{dt}\left[\frac{\partial L(\theta,\dot{\theta})}{\partial \dot{\theta}}\right] - \frac{\partial L(\theta,\dot{\theta})}{\partial \theta} = \tau$$

o, de manera equivalente:

$$\frac{d}{dt}\left[\frac{\partial L(\theta,\dot{\theta})}{\partial \dot{\theta}}\right] - \frac{\partial L(\theta,\dot{\theta})}{\partial \theta} = \tau_i \qquad i = 1 \dots n$$

donde $L(\theta,\dot{\theta})$ es el Lagrangiano definido como $L(\theta,\dot{\theta}) = K(\theta,\dot{\theta}) - U(\theta)$, $K(\theta,\dot{\theta})$ es la energía cinetica, $U(\theta)$ es su energía potencial, τ_i son las fuerzas y pares ejercidos externamente en cada articulación y *n* es el número de articulaciones (Kumar, 2010).

El procedimiento a seguir para determinar el modelo por medio de Lagrange es:

a. Obtener de las coordenadas de los centros de masas para cada eslabón (*x1, y1, x2, y2*) analíticamente.

b. A partir de las coordenadas de las posiciones obtener las velocidades de cada eslabón derivando las ecuaciones correspondientes.
c. Calcular la energía cinética con la fórmula:

$$K(\theta,\dot{\theta}) = \frac{1}{2}mv^2 + \frac{1}{2}I\left(\frac{d}{dt}\theta\right)^2$$

m es la masa del eslabón, *v* es la velocidad del eslabón en su centro de masa, *I* es el momento de inercia y θ es el ángulo desde el punto de equilibrio hasta el lugar donde se halla el eslabón.

d. Se obtiene la energía potencial $U(\theta)$, de cada eslabón analíticamente.
e. Una vez obtenidas estas energías se calcula el *Lagrangiano*.

El modelo obtenido es dado por las ec. (3.27) y (3.28). Los detalles se presentan el Apéndice C.

$$(m_1 {l_{c1}}^2 + J_2 + m_1 {l_1}^2)\ddot{\theta}_1 + bm_1\dot{\theta}_1 + (al_{c2}l_1 cos(\theta_1 - \theta_2) + J_2)\ddot{\theta}_2 + 2l_{c2}l_1 sin(\theta_1 - \theta_2)(\dot{\theta}_2)^2 - gsin(\theta_1)(l_{c1}m_1 + m_2 l_1) = \tau_1 \quad (3.27)$$

$$(J_2 + m_1 {l_1}^2)\ddot{\theta}_2 + bm_2\dot{\theta}_2 + (al_{c2}l_1 cos(\theta_1 - \theta_2) + J_2)\ddot{\theta}_1 - 2l_{c2}l_1 sin(\theta_1 - \theta_2)(\dot{\theta}_1)^2 - m_2 g l_{c2} sin(\theta_2) = \tau_2 \quad (3.28)$$

Donde τ_1 y τ_2 provienen de la ec.(3.11), en la figura 3.29 se muestra el resultado de colocar dos motores paso a paso iguales en cada articulación, se puede observar que permite el movimiento simultáneo en cada eslabón y a su vez la estabilización de los mismos.

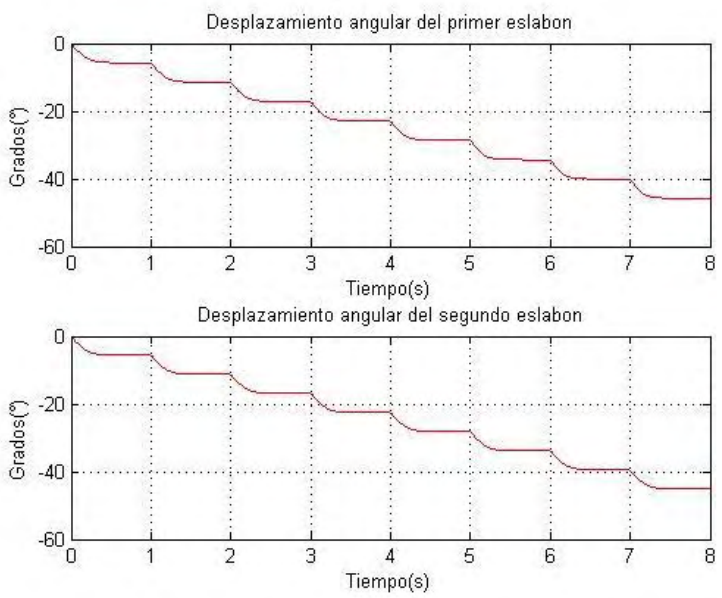


Figura 3. 29: Comportamiento del ángulo de ambas articulaciones con la terapia (b) propuesta

En las respuestas obtenidas del desplazamiento angular de sistema para un eslabón de las secciones 3.3 y 3.4, el comportamiento obtenido tiene una dinámica subamortiguada. Mientras que en este caso en particular (véase la figura 3.29), el desplazamiento angular de las articulaciones tienen una forma sobreamortiguada, esto se debe a que se está en presencia de dos eslabones. El peso de cada eslabón afecta el movimiento del motor paso a paso aportando más fricción en el eje de rotación.

Capítulo 4

Implementación del Sistema de Rehabilitación Asistida

Una vez construido el exoesqueleto robótico, y realizadas las pruebas teóricas necesarias para el control y el uso del mismo, se procede a la programación de la lógica de control. Dentro de una aplicación de software que provee de una interfaz gráfica amigable para el usuario.

Es necesario utilizar un equipo de adquisición de datos y actuación del exoesqueleto que cumpla con todas las exigencias fijadas, desde su versatilidad de operación hasta su vialidad económica.

Uno de los dispositivos que cubre las expectativas es el microcontrolador Arduino Uno[1]. El Arduino Uno puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores.

El microcontrolador en la placa Arduino Uno se programa mediante el lenguaje de programación **Arduino** (basado en el lenguaje de programación **Wiring)** y el entorno de desarrollo **Arduino** (basado en la librería **Processing**). Los proyectos hechos con Arduino

[1] http://www.arduino.cc/es/

Uno pueden ejecutarse sin necesidad de conectarse a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software.

El esquema de implantación utilizado se muestra en la figura 4.1, en el cual la Interfaz Hombre Máquina (HMI) será desarrollado en **Visual Studio 2010**, que es un entorno de desarrollo integrado y soporta varios lenguajes de programación. La comunicación del HMI con el microcontrolador es a través del puerto serial.

La señal enviada al puerto es tomada por el Arduino Uno, que enviará a cuatro pines la secuencia de activación de los interruptores, correspondiente al circuito de potencia para el movimiento del motor. Seguidamente la planta es actuada y el sensor mide la señal para enviarla a una entrada analógica del microcontolador, finalmente es emitida al puerto serial para que el HMI decida la nueva instrucción a realizar.

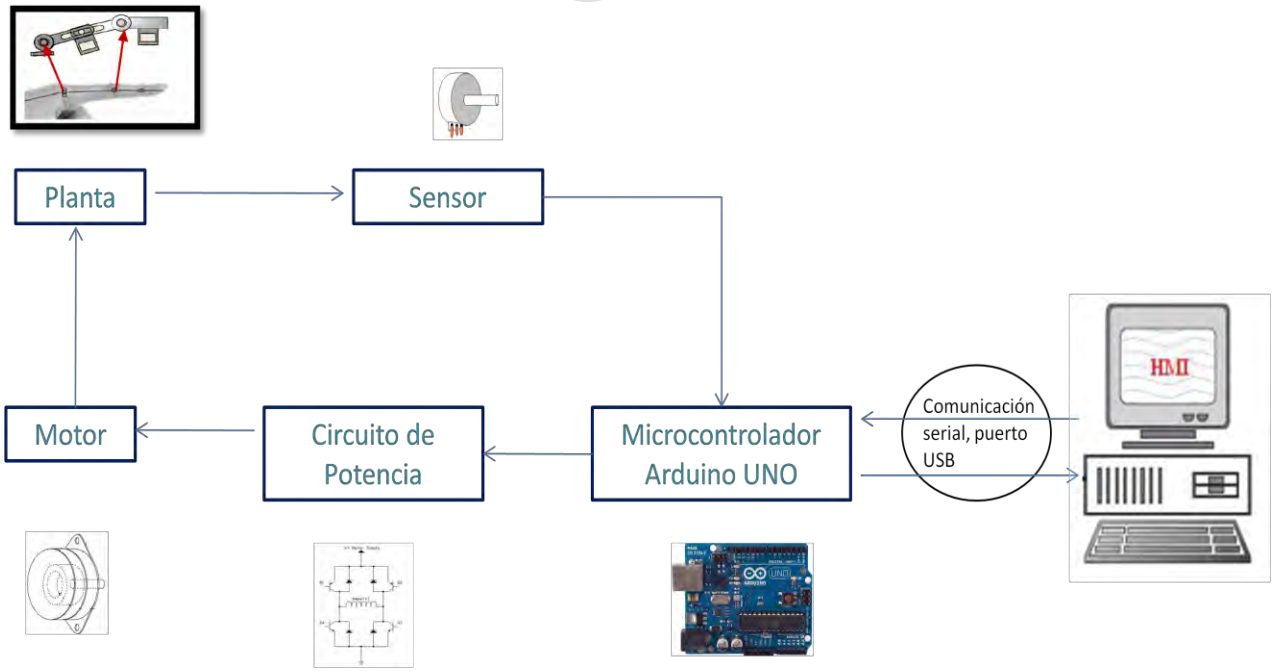


Figura 4. 1: Esquema de implantación del sistema de rehabilitación asistida

Debido a que el dispositivo robótico está destinado a pacientes de rehabilitación del dedo índice de la mano y principalmente el uso lo define el terapista ocupacional a cargo, es preciso pautar requerimientos adecuados para que se utilice el exoesqueleto.

Por los motivos previamente mencionados y con ayuda de la Sala de Terapia Ocupacional del IHULA, los requerimientos fijados fueron los siguientes:

- La aplicación debe permitir fijar el ángulo superior al cual se desea que llevar el dedo índice y así mismo el ángulo inferior, ya que ambos pueden variar según el nivel de lesión del paciente.
- El terapista ocupacional puede cambiar el tiempo de sostenimiento o tiempo de ancho de pulso, el cual es el tiempo que se tarda en moverse de un paso a otro el eje del motor.
- Establecer un tiempo de duración de la terapia.
- Observar cómo transcurre el tiempo, una vez iniciada la terapia.
- Observar en tiempo real el comportamiento del ángulo de la falange proximal, ya sea gráficamente o en un mímico que represente el dispositivo.
- El sistema debe tener una fase de información, que permita a un nuevo usuario aprender su funcionamiento y uso.

4.1 Programación de la lógica de control

El exoesqueleto está actuado por un motor paso a paso, que funciona mediante la activación de fases como se describió en la sección 3.3.2. Dicha secuencia de activación de las fases del motor se asocia a un estado del conjunto de interruptores, del circuito de potencia del motor

(abierto o cerrado), los cuales se pueden describir como un Sistema a Eventos Discretos (SED). Cada estado del SED se corresponde a una situación particular de los interruptores (abierto, cerrado), que determinan el flujo de corriente a través de las bobinas del motor.

Así pues, la lógica de control se concibe como un sistema de control supervisorio, que indica la secuencia de activación o desactivación de los interruptores.

Para conceptualizar la lógica de control que determina la secuencia de activación de fases, se propone el análisis del SED por medio de autómatas. Se parte de la descripción de cada componente por separado y después se hace la composición paralela de todos los elementos, el resultado de esta composición permite definir las especificaciones que debe cumplir el supervisor, porque en él se pueden observar la secuencia de eventos permitidos y las no permitidas, esta última es la que el supervisor deshabilita para que el sistema funcione correctamente.

A continuación se muestra la descripción de cada uno de los autómatas que conforman el motor paso a paso.

- **Interruptor 1:**

Este autómata modela la activación de la fase A del motor paso a paso (véase la figura 4.2), sus estados son interruptor abierto (S1A) e interruptor cerrado (S1C). Los eventos que lo conforman para pasar de un estado al otro son: cerrar (c1) y abrir (a1). La condición inicial es S1A, es decir no está energizada la bobina ya que no hay paso de corriente.

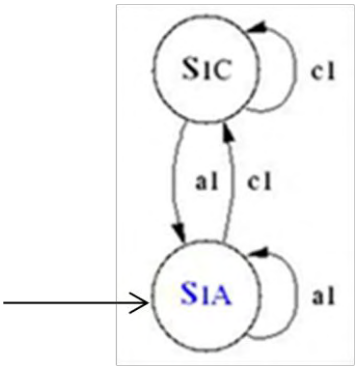


Figura 4. 2: Autómata G1, que representa el comportamiento del interruptor 1

El resto de los interruptores se comportan de forma análoga como se indica a continuación.

- **Interruptor 2:**

Su comportamiento es igual al caso a anterior como se muestra en la figura 4.3, sus estados son interruptor abierto (S2A) e interruptor cerrado (S2C). Los eventos que lo conforman para pasar de un estado al otro son: cerrar (c2) y abrir (a2). La condición inicial es S2A.

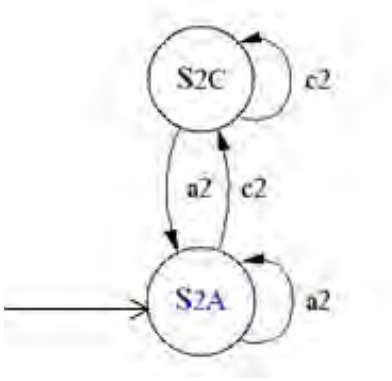


Figura 4. 3: Autómata G2, que representa el comportamiento del interruptor 2

- **Interruptor 3:**

Sus estados son interruptor abierto (S3A) e interruptor cerrado (S3C). Los eventos que lo conforman para pasar de un estado al otro son: cerrar (c3) y abrir (a3). La condición inicial es S3A (véase la figura 4.4).

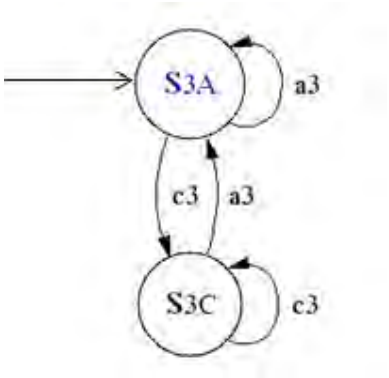


Figura 4. 4: Autómata G3, que representa el comportamiento del interruptor 3

- **Interruptor 4:**

Sus estados son interruptor abierto (S4A) e interruptor cerrado (S4C). Los eventos que lo conforman para pasar de un estado al otro son: cerrar (c4) y abrir (a4). La condición inicial es S4A (véase la figura 4.5).

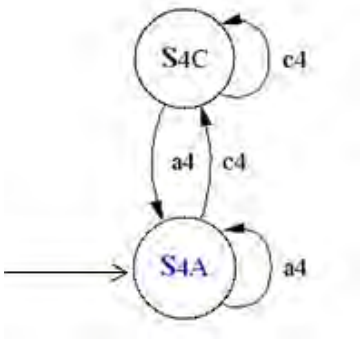


Figura 4. 5: Autómata G4, que representa el comportamiento del interruptor 4

El comportamiento del sistema de interruptores viene dado por su secuencia de activación que permite la energización las fases del motor, es por ello que se realiza la composición paralela de G1, G2, G3 y G4. En la figura 4.6 se observa el resultado obtenido representado por el autómata G5.

Se considera que todos los eventos son observables y controlables, para el diseño del supervisor primero se definen las especificaciones que debe cumplir el mismo. En el modelado con autómatas de especificaciones de control se realiza el diseño con el uso de dos técnicas, la primera eliminando los estados ilegales y la segunda por medio de la división estados (Ramadge y Wonham, 1987), veamos cómo se aplica a nuestro caso en estudio.

- **Estados ilegales:**

En la figura 4.7 se engloban con un eclipse rojo los estados ilegales de G5. En este caso, por ejemplo S1C, S2C, S3C, S4C es un estado no permitido, el motor funciona con la activación de dos interruptores, el mismo caso ocurre con los estados que tienen 3 interruptores activados, como por ejemplo S1A, S2C, S3C, S4C y todas las demás combinaciones. Por último se eliminan los estados que nunca llegan a ocurrir con la secuencia de control indicada en la tabla 2.2.

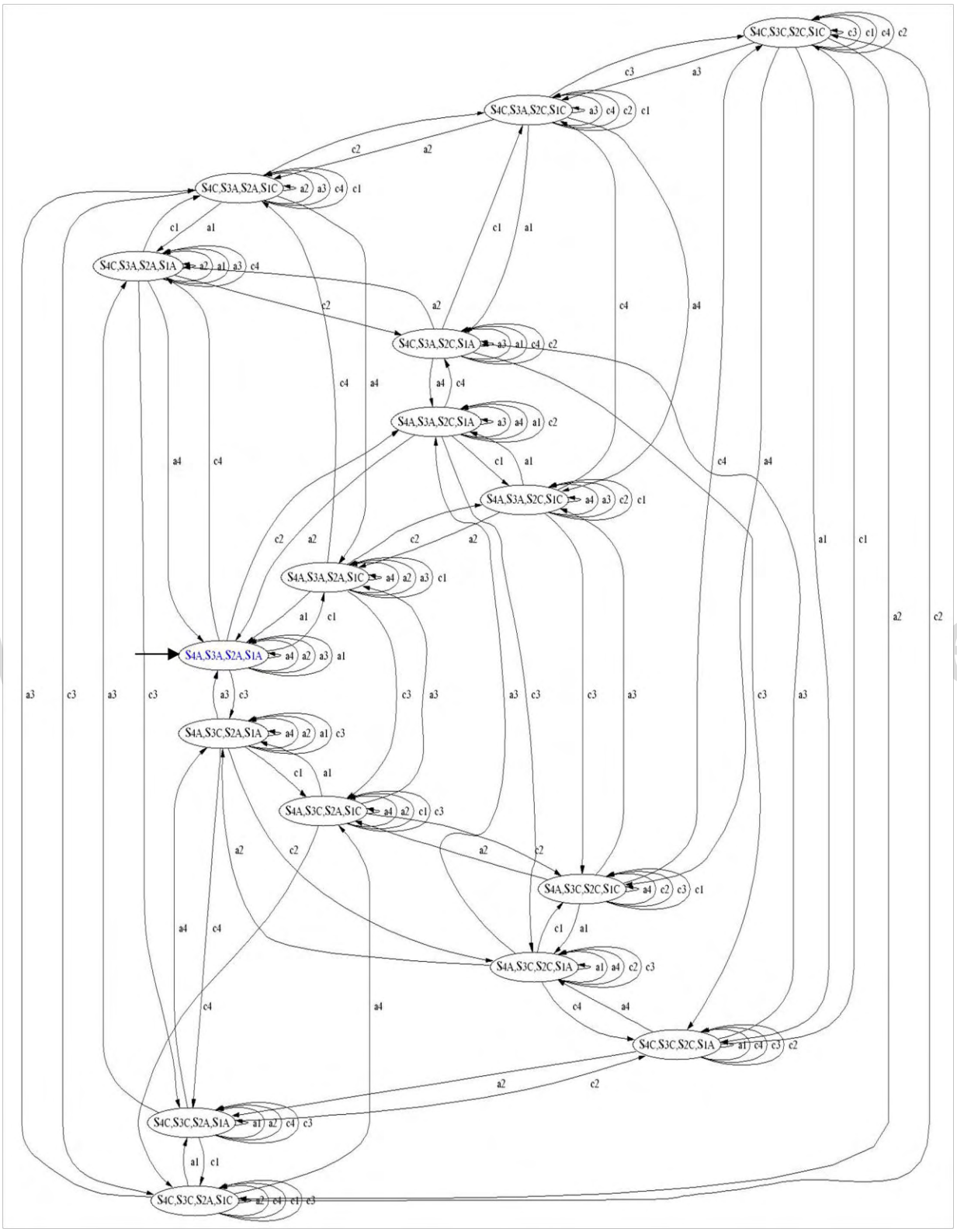


Figura 4. 6: Autómata G5, composición paralela de G1, G2, G3 y G4

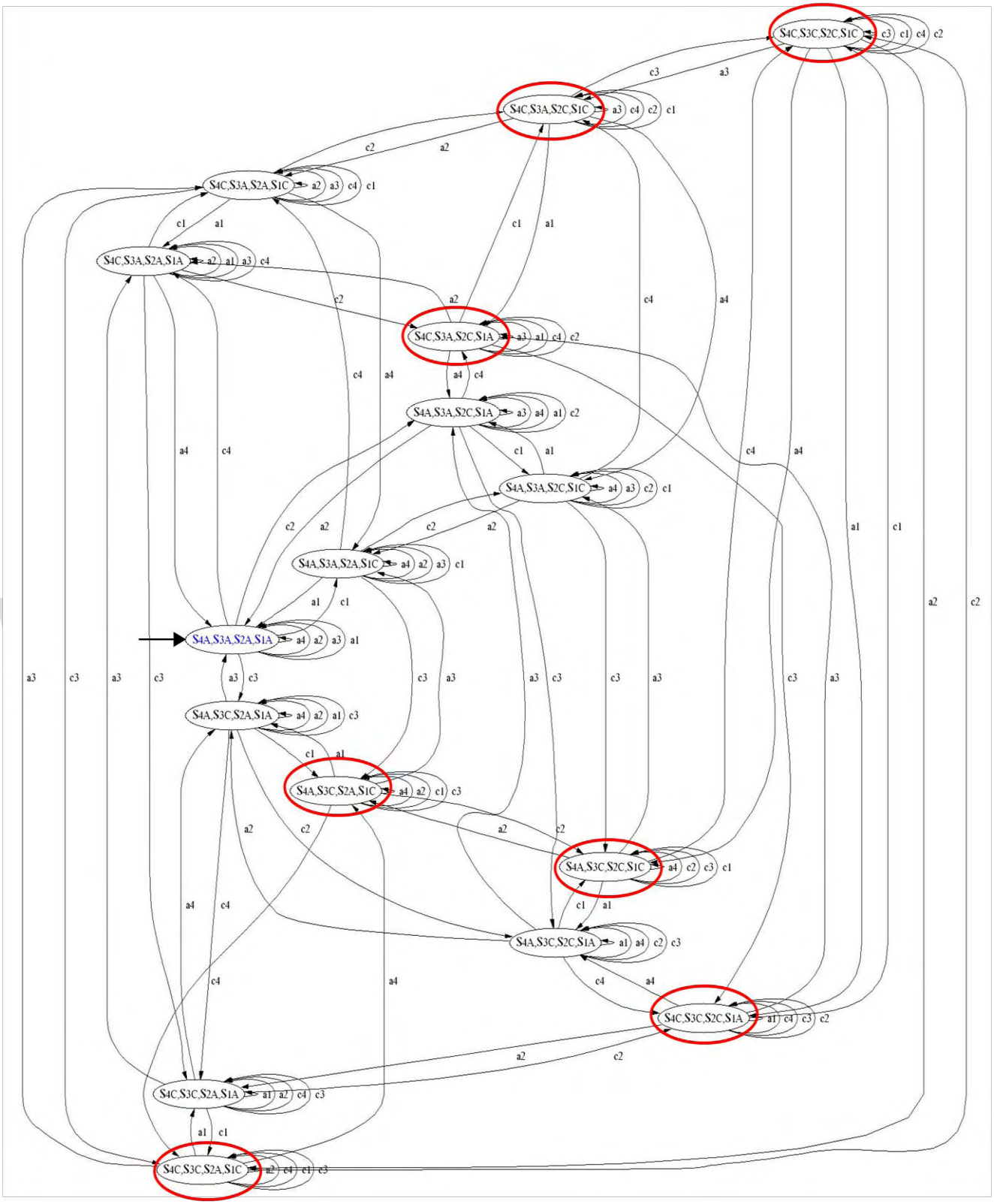


Figura 4. 7: Autómata G5, estados ilegales de la composición paralela entre G1, G2, G3 y G4

El resultado obtenido al eliminar estados ilegales se muestra en la figura 4.8 denotado por el autómata G6. Si bien es cierto que este autómata genera todos los eventos que permiten el funcionamiento de manera correcta del motor paso a paso, no hay un orden en la secuencia de activación de los interruptores y no se puede especificar la dirección de giro del eje del motor paso a paso.

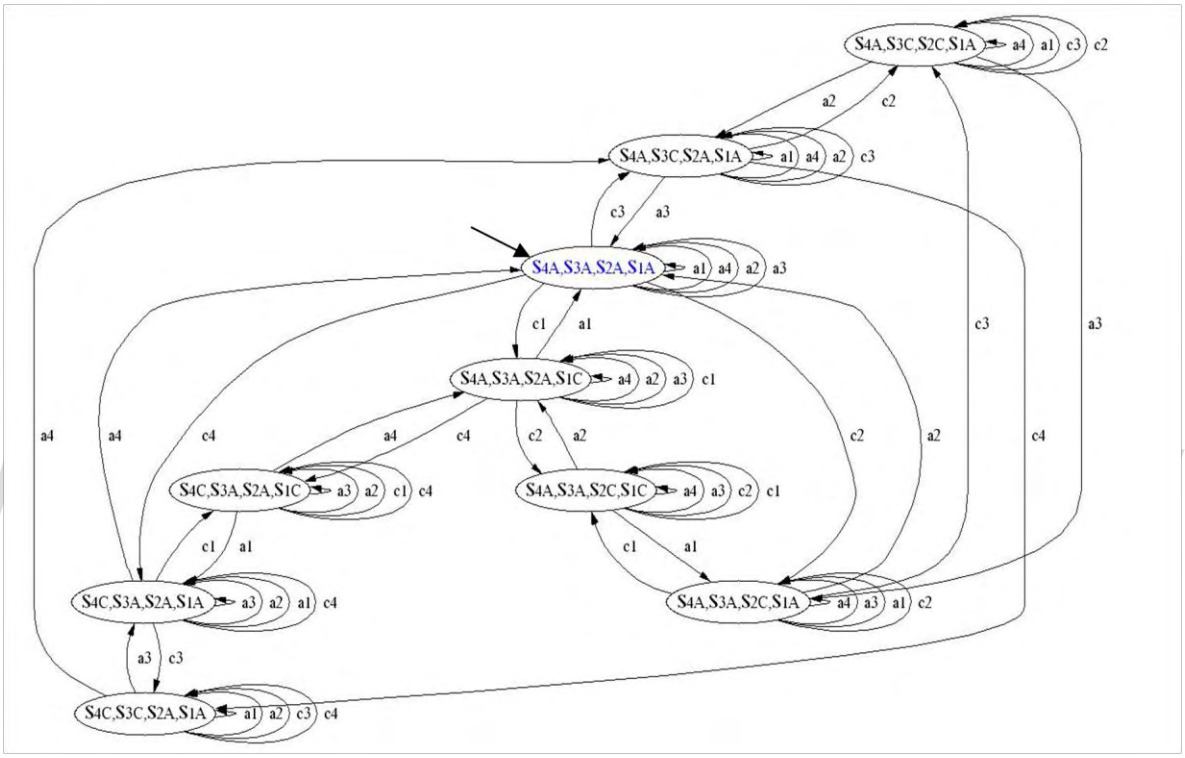


Figura 4. 8: Autómata G6, resultado de eliminar estados ilegales de la composición paralela entre G1, G2, G3 y G4

Por tal motivo se plantea el autómata, G7 que adiciona la condición de dirección de giro y cuando lo debe hacer.

El sistema está en reposo (estado Stop), cuando se inicia la terapia (evento i) se comienza girando el eje del motor hacia la derecha denotado como MD. Debido a la secuencia de activación indicada, pueden ocurrir dos situaciones, que se detuvo la terapia (evento p) o que

se alcanzó al ángulo deseado superior (evento Ad). Si ocurrió esto último se cambia la dirección del eje del motor hacia la izquierda (estado MI) y se verifica si se llegó al ángulo deseado inferior (evento Aizq) o hay una parada (evento p). Si no se llega al desplazamiento angular superior o inferior deseado, el motor se mueve en el mismo sentido continuando con la secuencia especificada .El autómata obtenido se muestra en la figura 4.9. Se asume por diseño, que siempre se inicia con movimiento a la derecha.

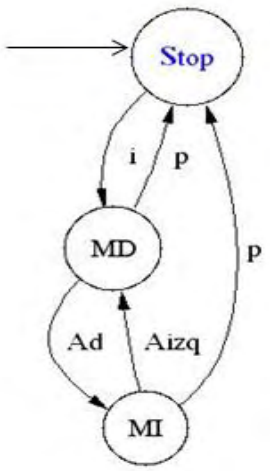


Figura 4. 9: Autómata G7, que representa la planta controlada

Es importante señalar, que los eventos *Ad*, *Aizq*,*i* y *p*, son eventos observables pero no controlables, y son provenientes de sensores o del exterior.

Realizando la composición paralela entre G6 y G7 el resultado se muestra en la figura 4.10, se eliminan los estados no deseados. El estado no permitido es MI, S1A, S2A, S3A, S4A ya que este caso no ocurre en el sistema controlado (véase la figura 4.11), debido a que si se está moviendo el motor a la izquierda (MI) debe estar activada al menos una fase, se admite solo en parada ya que es el estado inicial y en movimiento a la derecha ya que permitirá cumplir con la secuencia deseada para el movimiento del motor, según la tabla 2.2.

El resultado de eliminar el estado no deseado es dado por el autómata G9, como se muestra en la figura 4.12.

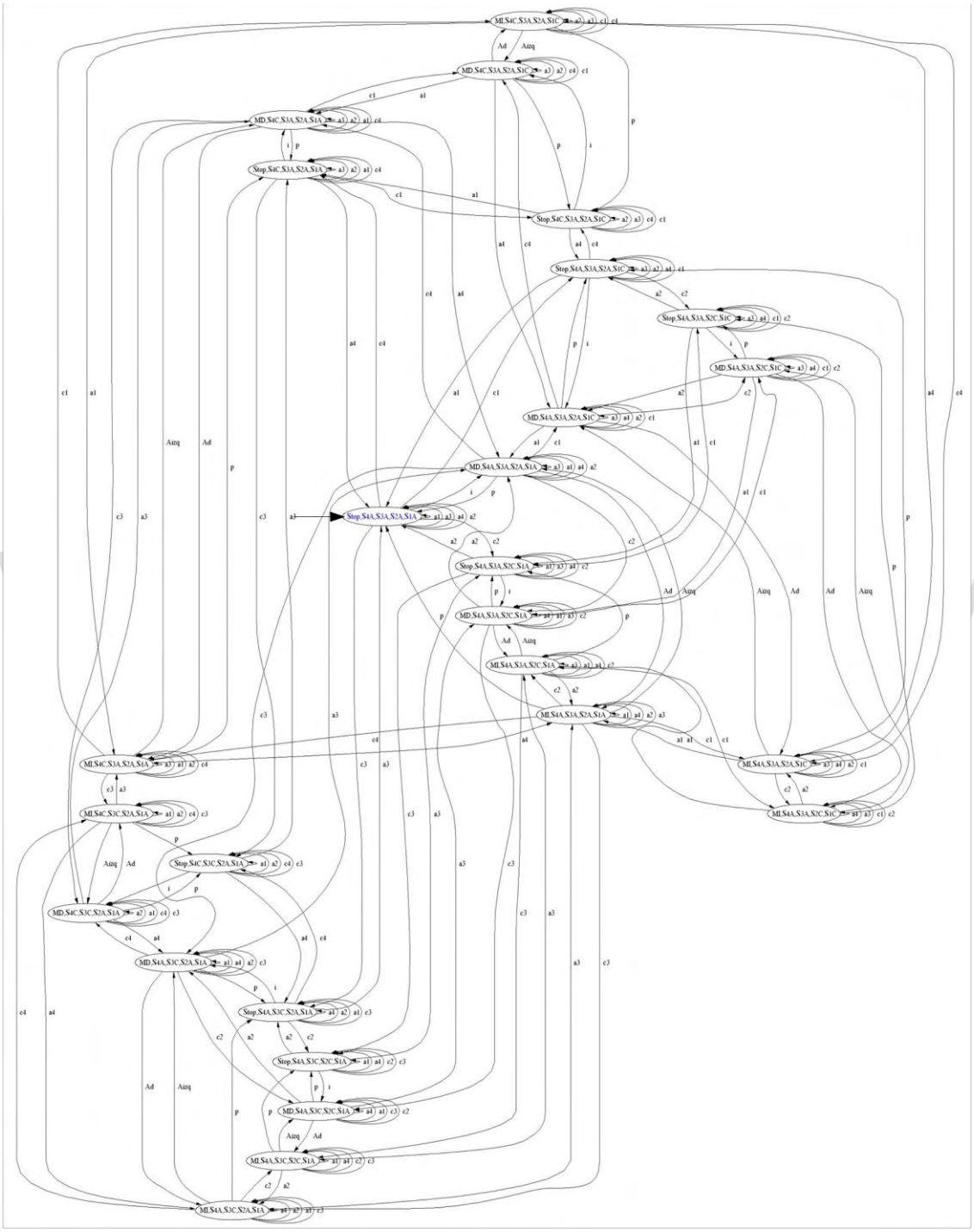


Figura 4. 10: Autómata G8, que representa la dinámica del sistema a eventos

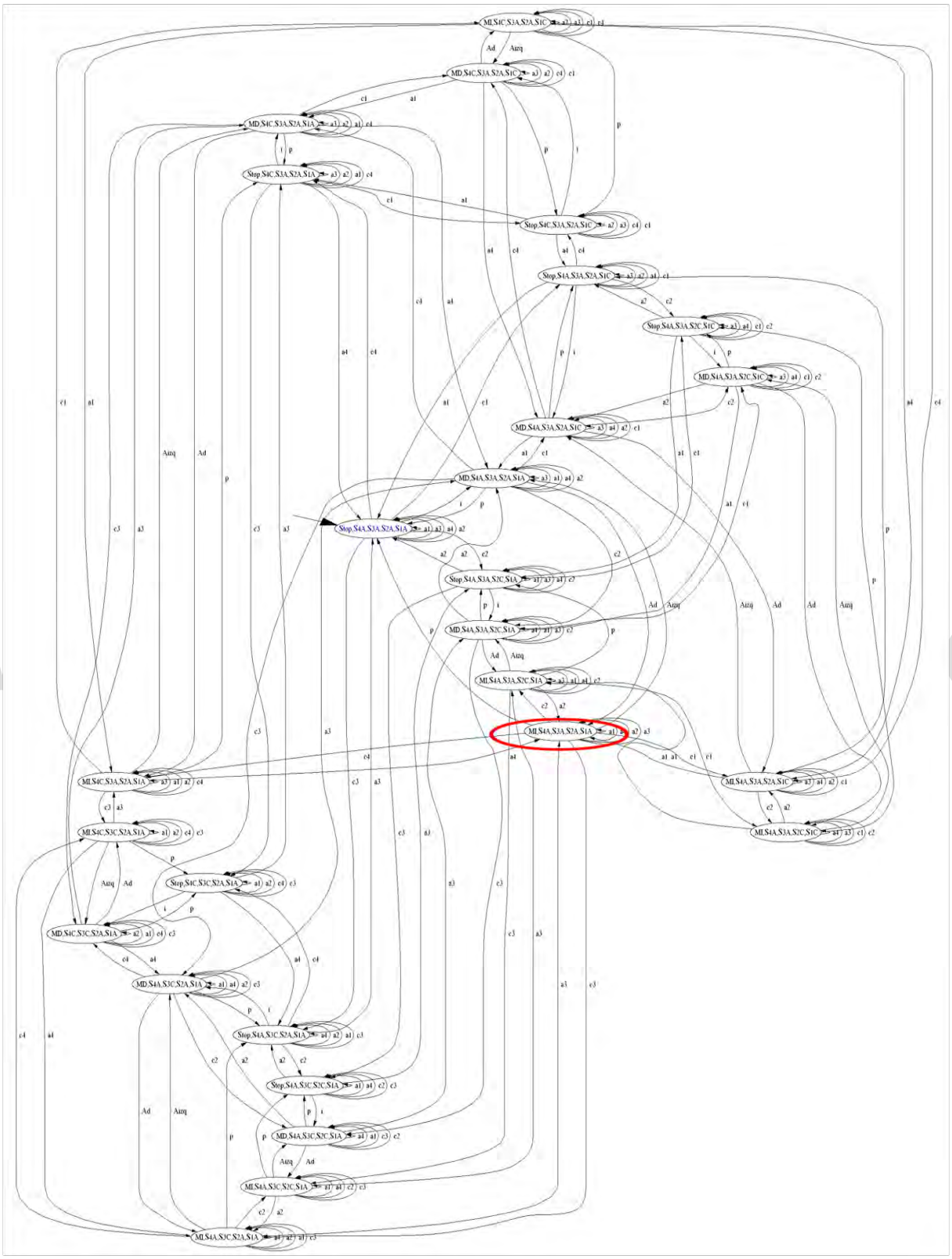


Figura 4. 11: Autómata G8, que representa la dinámica del sistema a eventos

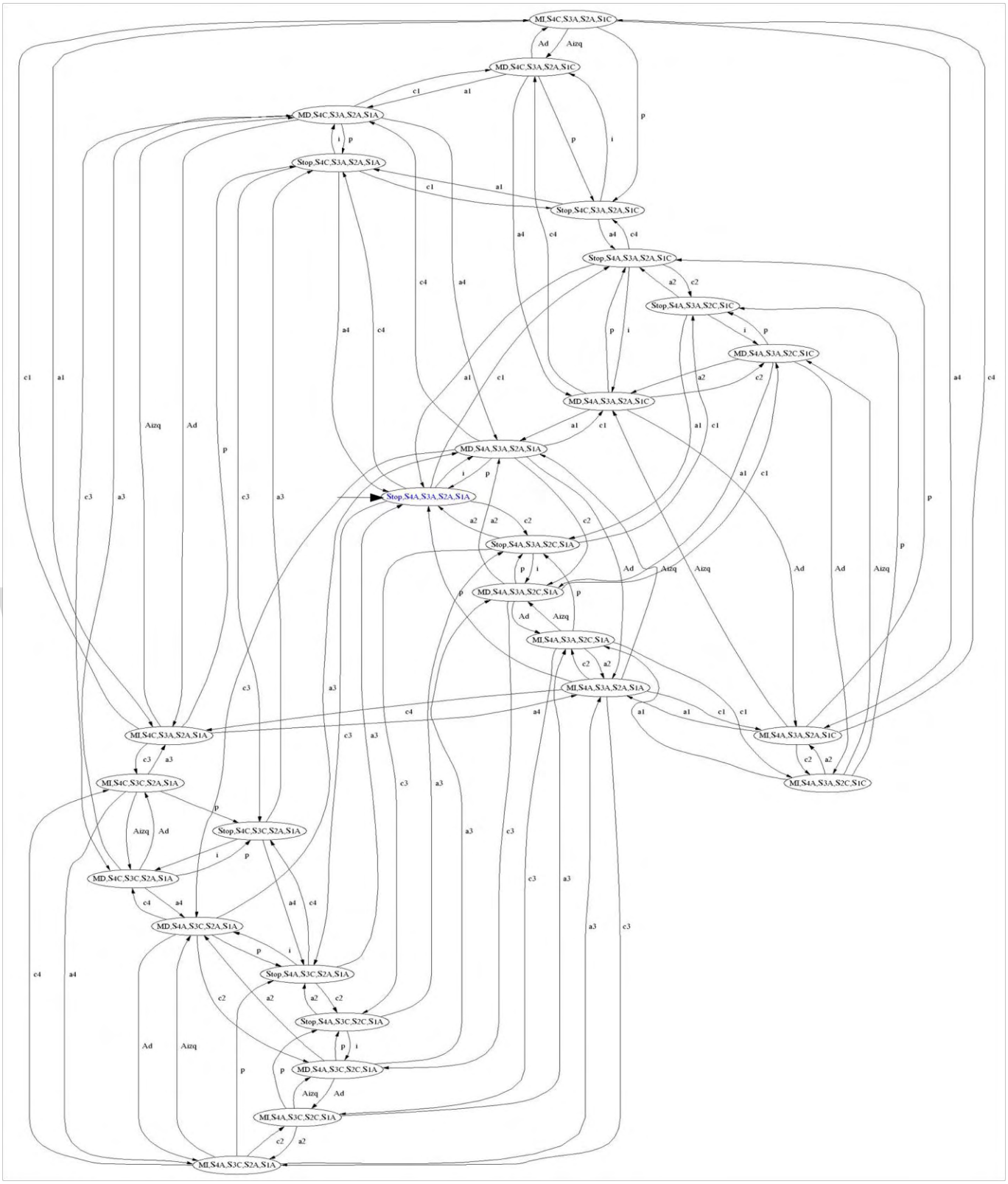


Figura 4. 12: Autómata G9, que representa la dinámica del sistema a eventos eliminado estados ilegales

El autómata de las especificaciones de control se logra bajo las siguientes consideraciones, tomadas de la secuencia de activación de la tabla 2.2:

- Para que el motor realice un paso se activan los dos interruptores instantáneamente, para que el autómata de las especificaciones cumpla con esto se considera que primero se activa un interruptor y después el otro. Con el autómata obtenido se asume que esto ocurre muy rápido para que represente el comportamiento real.
- Para que la activación de dos interruptores se realice en orden, pueden ocurrir dos secuencias diferentes, una de las cuales será eliminada. Por ejemplo, para llegar al estado MD, S1C, S2C, S3A, S4A por ejemplo, pueden ocurrir dos secuencias {*c1c2*} ó {*c2c1*}. Este problema se reduce a llegar a este estado solo con {*c1c2*}, si bien la otra secuencia es posible, pero como la activación de las fases es tan rápida cualquiera de las dos funciona. Esta condición aplica para los demás estados donde se activan dos fases.
- Se verifica que se ha llegado al ángulo deseado, ya sea el evento *Ad* o el evento *Aizq*, si y solo si están activados dos interruptores, esto quiere decir que el motor ya realizó el paso que correspondía.
- Al igual que el caso anterior, las paradas solo ocurren cuando están activados dos interruptores, ya que como se mencionó, cerrar dos interruptores es muy rápido en la realidad.
- Se escogió una sola secuencia de parada de cualquier estado válido. Por ejemplo, si se quiere realizar una parada estando en el estado Stop, S1C, S2C, S3A, S4A una secuencia permitida es {a1a2} y otra es {a2a1}, ambas conllevan al mismo resultado (Stop, S1A, S2A, S3A, S4A), pero para la lógica de control se dejan solo las secuencias que abren los interruptores en orden, es decir {a1a2}, {a2a3}, {a3a4} y {a1a4}.

- **División de estados:**

Finalmente se divide el estado MI, S4A, S3A, S2C, S1A, con la finalidad de cumplir con la secuencia de activación dada en la tabla 2.2 y se verifica que el modelo está funcionando correctamente como se muestra en la figura 4.13.

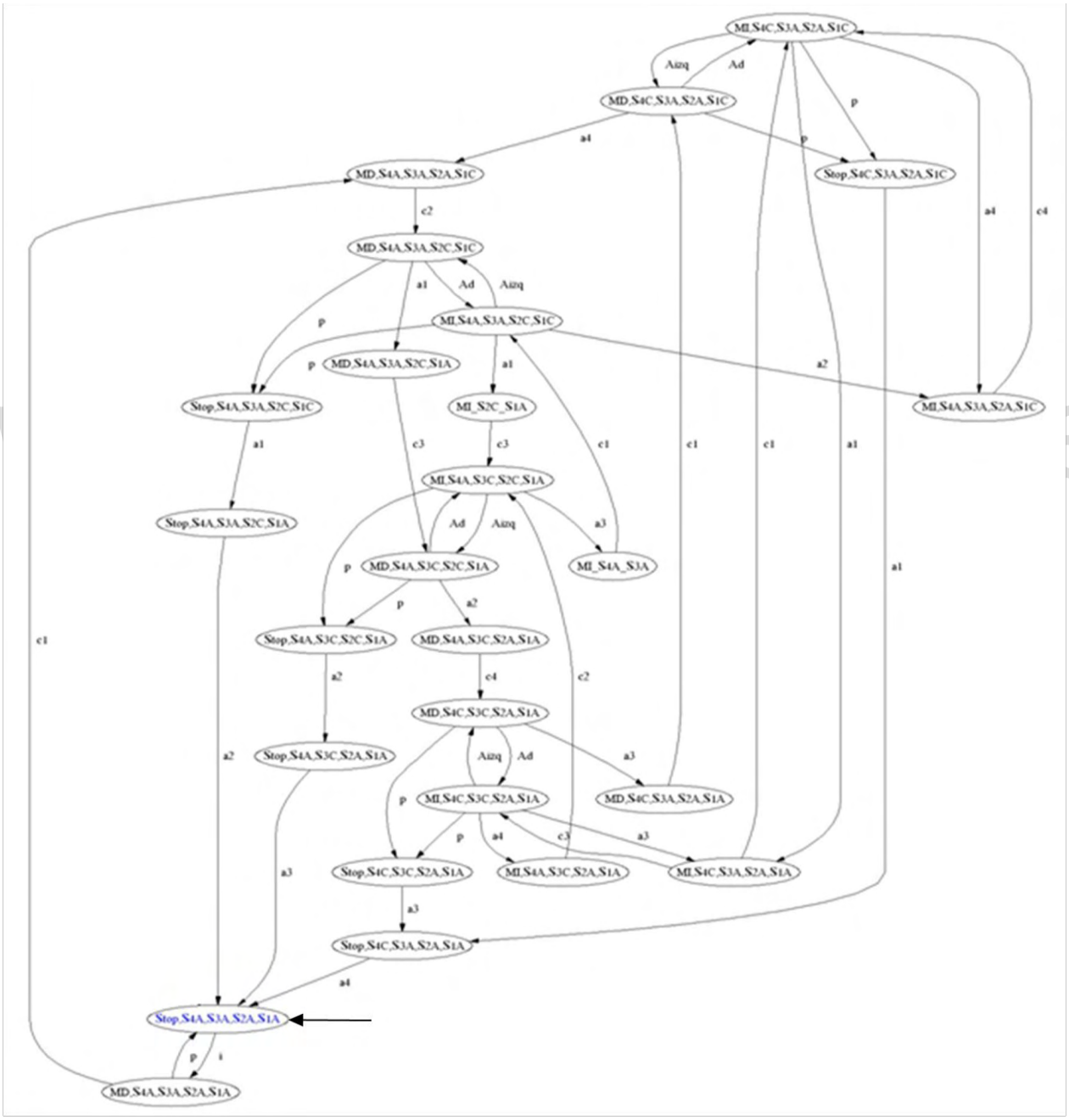


Figura 4. 13: Autómata Hesp, que representa las especificaciones de control

El autómata supervisor obtenido a partir del autómata Hesp, se muestra en la figura 4.14 y la tabla de control se muestra en la tabla 4.1.

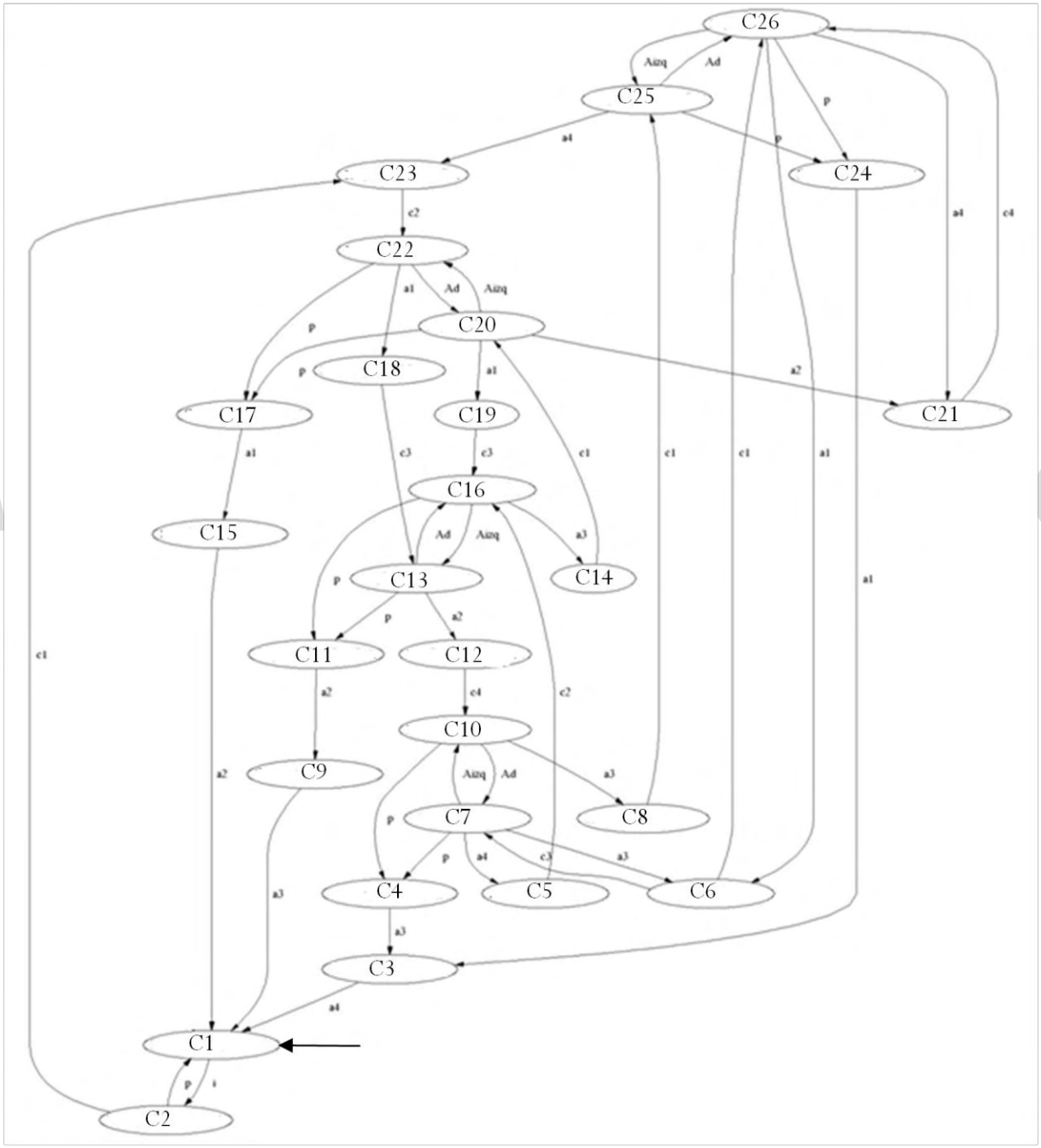


Figura 4. 14: Autómata C, supervisor del sistema

A continuación se presenta la tabla de control binario, que permite la implementación del supervisor en la herramienta computacional utilizada (véase la tabla 4.1). Se puede observar que los eventos *p*, *i*, *Ad* y *Aizq* están habilitados, pero no son controlables.

Tabla 4. 1: Patrones de control binario del supervisor C

Φ	i	P	Ad	Aizq	a1	a2	a3	a4	c1	c2	c3	c4
C1	1	0	0	0	0	0	0	0	0	0	0	0
C2	0	1	0	0	0	0	0	0	1	0	0	0
C3	0	0	0	0	0	0	0	1	0	0	0	0
C4	0	0	0	0	0	0	1	0	0	0	0	0
C5	0	0	0	0	0	0	0	0	0	1	0	0
C6	0	0	0	0	0	0	0	0	1	0	1	0
C7	0	1	0	1	0	0	1	1	0	0	0	0
C8	0	1	0	0	0	0	0	0	0	0	0	0
C9	0	0	0	0	0	0	0	0	1	0	0	0
C10	0	1	1	0	0	0	1	0	0	0	0	0
C11	0	0	0	0	0	1	0	0	0	0	0	0
C12	0	0	0	0	0	0	0	0	0	0	0	1
C13	0	1	1	0	0	1	0	0	0	0	0	0
C14	0	0	0	0	0	0	0	0	1	0	0	0
C15	0	0	0	0	0	1	0	0	0	0	0	0
C16	0	1	0	1	0	0	1	0	0	0	0	0
C17	0	0	0	0	1	0	0	0	0	0	0	0
C18	0	0	0	0	0	0	0	0	0	0	1	0
C19	0	0	0	0	0	0	0	0	0	0	1	0
C20	0	1	0	0	1	1	0	0	0	0	0	0
C21	0	0	0	0	0	0	0	0	0	0	0	1
C22	0	1	1	0	1	0	0	0	0	0	0	0
C23	0	0	0	0	0	0	0	0	0	1	0	0
C24	0	0	0	0	1	0	0	0	0	0	0	0
C25	0	1	1	0	0	0	0	1	0	0	0	0
C26	0	1	0	1	0	0	0	1	0	0	0	0

Para demostrar que efectivamente el autómata Hesp cumple con el comportamiento deseado, se realizaron las pruebas pertinentes para verificar que todas las secuencias de movimiento ocurran. Por ejemplo, si se desea mover el motor hacia la derecha dos pasos desde el estado inicial la secuencia es la siguiente:

$$f_1(Stop\_S4A\_S3A\_S2A\_S1A, ic1c2a1c3) = MD\_S4A\_S3C\_S2C\_S1A \quad (4.1)$$

Donde el evento *i* indica inicio de la terapia, *c1* cierra el interruptor 1 y *c2* cierra el interruptor 2, en este instante el motor se mueve un paso. Seguidamente se abre el interruptor 1 con *a1* y se cierra el interruptor con *c3* para mover dos pasos. Si ocurre una parada p la secuencia de activación es dada en f_2.

$$f_2(Stop\_S4A\_S3A\_S2A\_S1A, ic1c2a1c3pa2a3) = Stop\_S4A\_S3A\_S2A\_S1A \quad (4.2)$$

Ahora bien, si el motor se mueve un paso a la derecha y ya alcanzó el desplazamiento angular deseado, cambia la dirección a la izquierda. Veamos el ejemplo de f_3 con dos pasos a la izquierda después de llegar al ángulo superior deseado.

$$f_3(Stop\_S4A\_S3A\_S2A\_S1A, ic1c2Ada1c3) = MI\_S4A\_S3C\_S2C\_S1A \quad (4.3)$$

El sensor cumple una función vital para que el sistema supervisado funcione, debido a que con la información proveniente del mismo se verifica si se logró el desplazamiento angular deseado sea superior o inferior, y se produce el cambio de dirección del motor.

En la figura 4.15 se observa el comportamiento del supervisor C sobre el autómata que representa la dinámica del sistema a eventos G8, este resultado evidencia los eventos *e* que ocurren en la planta son vistos por el supervisor ϕ que habilita los eventos Υ controlables activos a la planta, garantizando así el funcionamiento deseado. Es importante señalar que todos los eventos son observables por el supervisor y se asume que los interruptores están libres de fallas.

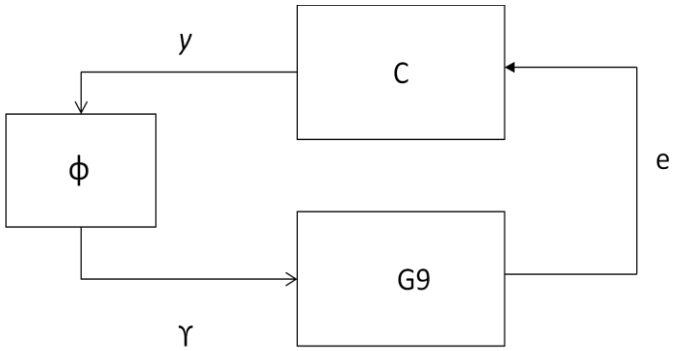


Figura 4. 15: Comportamiento del SED controlado

Con la descripción del sistema como un SED y el diseño del supervisor, se proporciona gran información del comportamiento del sistema y sirven para comprender el control propuesto, que más adelante será implementado.

4.2 Diseño de la interfaz gráfica

Recordando los requerimientos pautados, se necesita que el usuario introduzca una serie de datos para el inicio de la terapia. Así como también, mostrar la evolución del desplazamiento angular en tiempo real y por último información acerca del dispositivo.

Con la información recopilada, se plantea un área de trabajo para el terapista ocupacional, la misma cuenta con un menú en la parte superior de la ventana y cada menú con su submenú. En la figura 4.16 se muestra la pantalla principal de la aplicación.

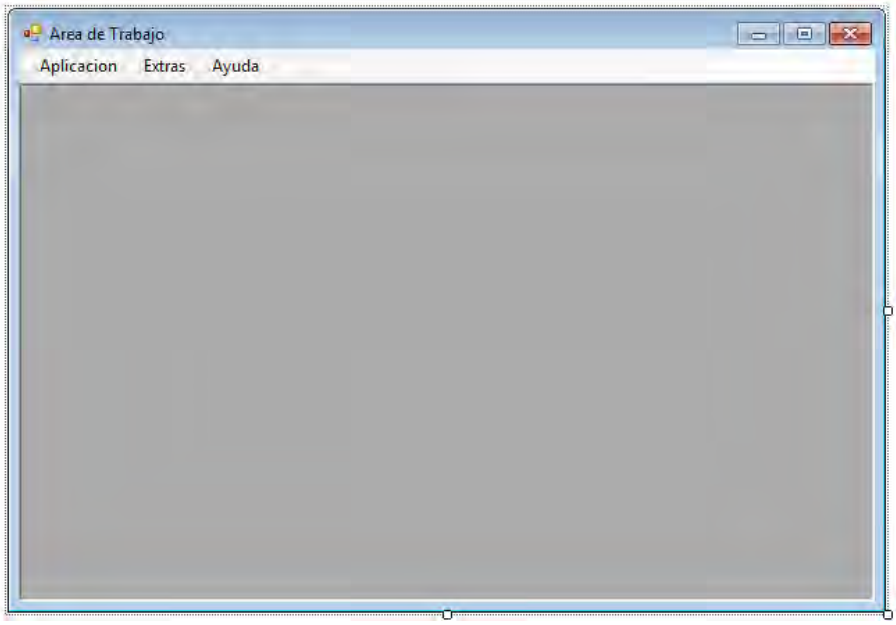


Figura 4. 16: Ventana de área de trabajo de la interfaz gráfica

Dicha área de trabajo está diseñada para que se abran varias ventanas a la vez. A continuación se muestran las ventanas que conforman el área de trabajo, posteriormente se explicará cómo fueron codificadas en **Visual Studio 2010.**

i. Aplicación

En esta opción se almacena la función más importante del programa, como lo es el control del exoesqueleto robótico y está compuesta por los siguientes ítems:

a. Inicio: Contiene el programa que permite fijar los parámetros y comenzar la terapia. En la figura 4.17 se muestra que se cumplen todos los requerimientos pedidos por el terapista ocupacional, las casillas enmarcadas en rojo son los campos obligatorios que se deben completar.

Para comenzar con la terapia se presiona el botón “Inicio”, instantáneamente este botón cambia a “Stop” para que se pueda detener cuando se desee.

Debido a que hay un límite de movimiento del dedo índice de ángulo superior e inferior. En el programa se verifica esta restricción, colocando una barra de desplazamiento que permite variar el ángulo superior de 0º a 60º e inferior de 0 a -80º, estas barras permitirán al usuario colocar solo los ángulos admitidos. Se aplica igual con el tiempo de sostenimiento del motor paso a paso.

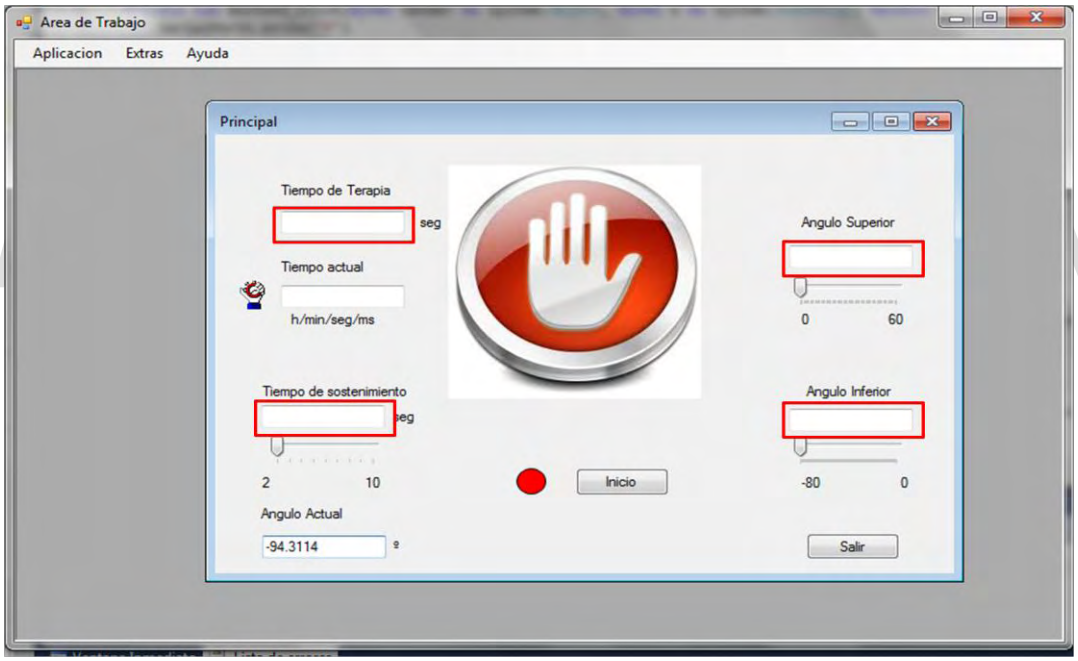


Figura 4. 17: Ventana principal de la aplicación de la interfaz gráfica

El *TextBox* nombrado como “Ángulo Actual” se actualiza una vez que se abre la aplicación con el valor del ángulo de la falange proximal. Mientras que el “Tiempo Actual” se comporta como un cronómetro que inicia cuando comienza la terapia y culmina cuando se llega al tiempo de terapia fijado.

El botón “Salir” cierra la ventana, el área de trabajo sigue activo.

b. Salir: Cierra el área de trabajo y así mismo todas las ventanas abiertas contenidas en él.

ii. Extras

Esta opción es integrada para complementar la aplicación principal y observar mejor el comportamiento del exoesqueleto. Cabe destacar que lo adjunto en este menú está inactivo hasta que se comienzan las terapias. A continuación se muestra el contenido de este menú:

a. Desplazamiento angular: Se muestra la gráfica en tiempo real del comportamiento del ángulo de la falange proximal. En la figura 4.18, se muestra el diseño de la ventana.

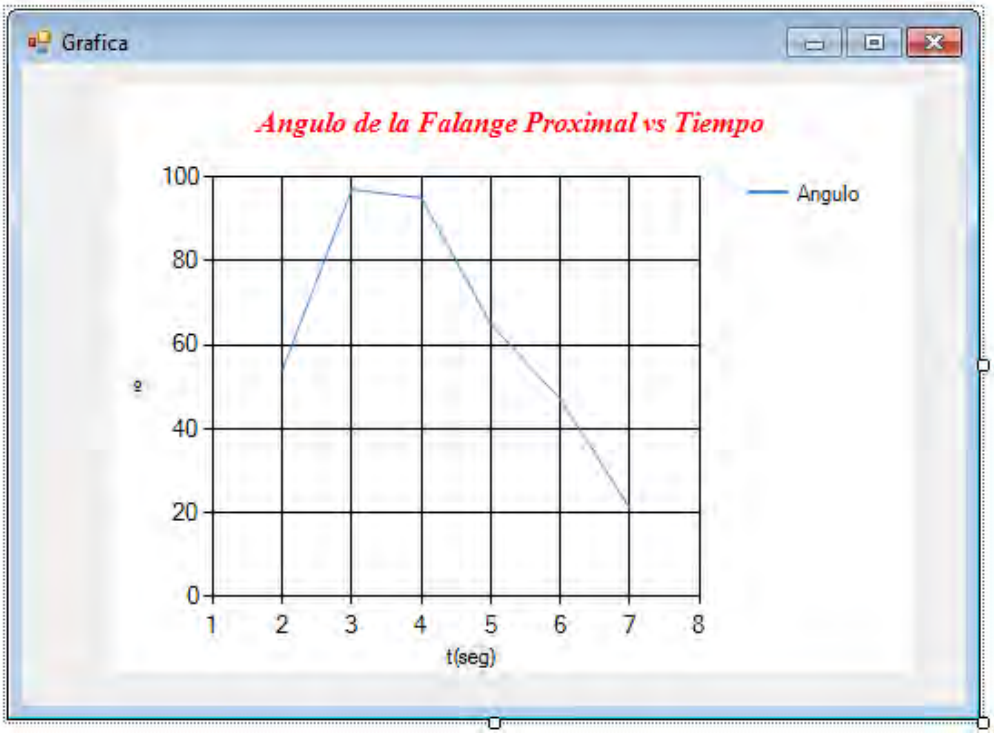


Figura 4. 18: Ventana que muestra el desplazamiento angular

b. Mímico: En esta ventana se presenta un mímico que emula el comportamiento en tiempo real del exoesqueleto robótico, con la finalidad de hacer amigable la interacción con el usuario como se muestra en la figura 4.19 Es una recta que se mueve como un péndulo según el ángulo proporcionado por el sensor.

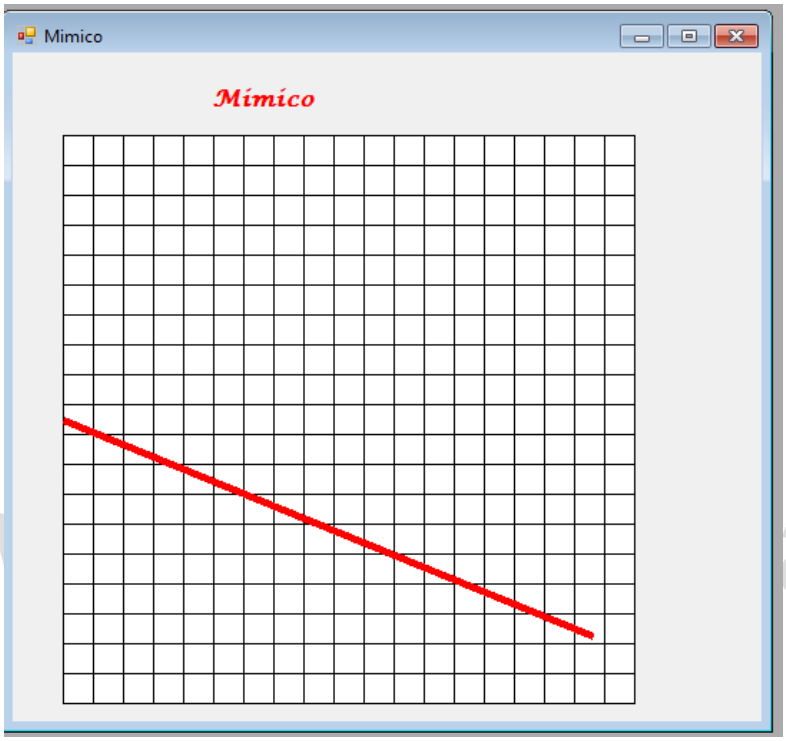


Figura 4. 19: Ventana que muestra el mímico del exoesqueleto.

iii. Ayuda:

Por último se dispone de un menú de ayuda al usuario, que proporcionará información a preguntas que pueda tener el usuario.

a. Información: Esta opción contiene toda la información que necesite saber el usuario para el uso del dispositivo.

b. Conexiones: Se muestra como se deben conectar los elementos del dispositivo, tales como la alimentación del exoesqueleto y la conexión a la computadora. En la figura 4.20 se muestra gráficamente como se deben realizar estas conexiones para que sea más sencillo de entender para el usuario.

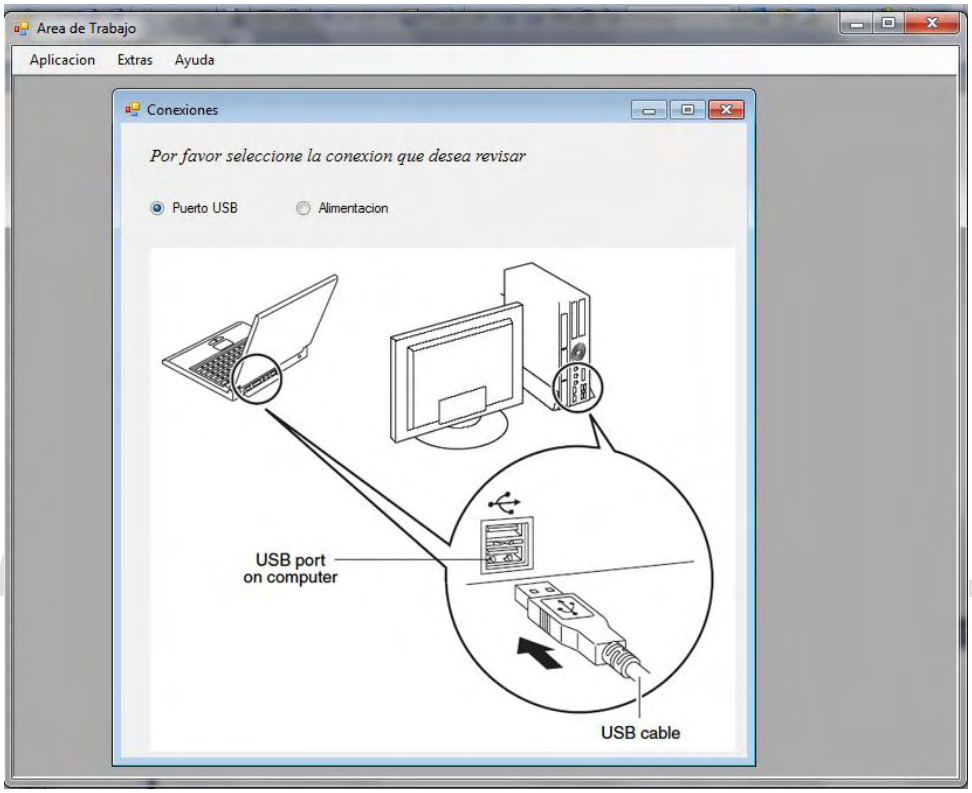


Figura 4. 20: Ventana que muestra como se deben realizar las conexiones en el dispositivo

Finalmente el usuario puede ver paralelamente lo todas las ventanas mostradas anteriormente como se muestra en la figura 4.21.

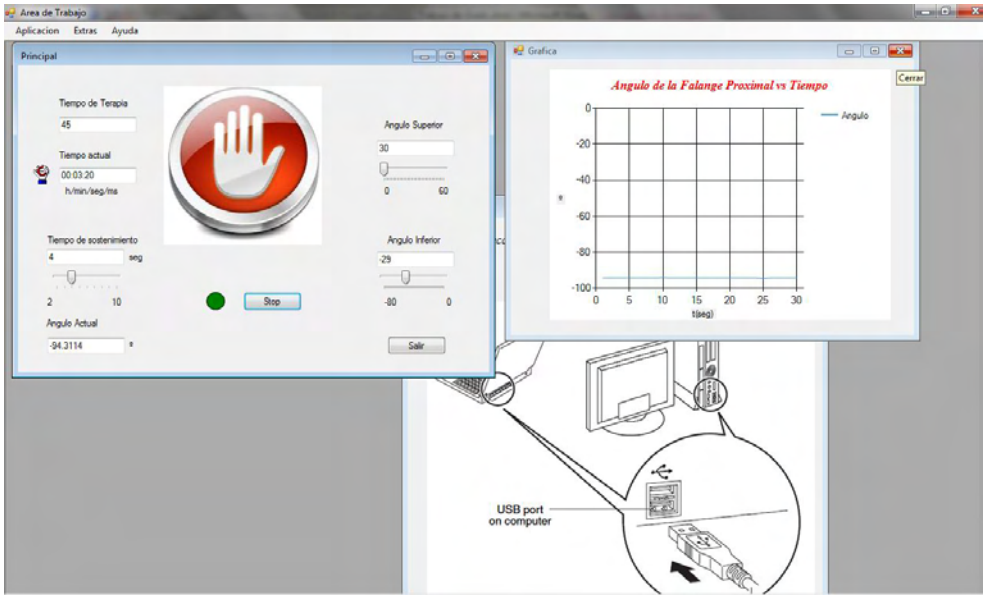


Figura 4. 21: Ventana de área de trabajo con varia ventanas abiertas

4.3 Implementación

4.3.1 Lógica de control

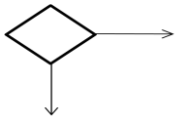
Para comprender la codificación de la lógica de control, se plantea un diagrama de flujo sencillo, el cual es el algoritmo que se siguió para la implementación.

En la tabla 4.2 se muestran los símbolos empleados con mayor frecuencia en el diseño de diagramas de flujo.

Tabla 4. 2: Algunos símbolos empleados en el diseño de diagramas de flujo

Símbolo	Descripción
	Terminal, representa el comienzo y el final de un programa.

Continua en la próxima página …

Viene de la página anterior …	
	Entrada/Salida, representa cualquier entrada salida de datos.
	Decisión, muestra una operación lógica o de comparación de datos.
	Llamada de procedimiento, llama una rutina independiente al programa principal

Los datos que se necesitan para implementar el control supervisorio diseñado son: tiempo_sos, tiempo, angI, angS y ang_actual. Con estos datos se ejecuta un subprograma que será explicado posteriormente y se va verificando el tiempo de terapia introducido por el usuario, cuando se haya cumplido este tiempo pautado el programa finaliza (véase la figura 4.22).

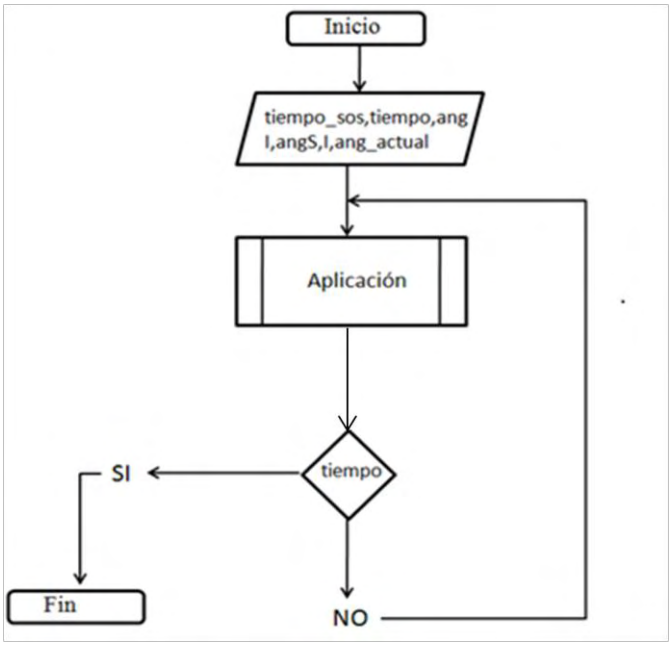


Figura 4. 22: Diagrama de flujo de la aplicación principal

La función del subprograma consiste en implementar el control supervisorio diseñado en la sección 4.1, se inicia llevando el exoesqueleto al ángulo superior deseado (*AngS*), una vez esté allí se mueve el motor en la dirección contraria hasta llegar al ángulo inferior deseado (*AngI*) y así sucesivamente hasta haber culminado con el tiempo de terapia definido (véase la figura 4.23).

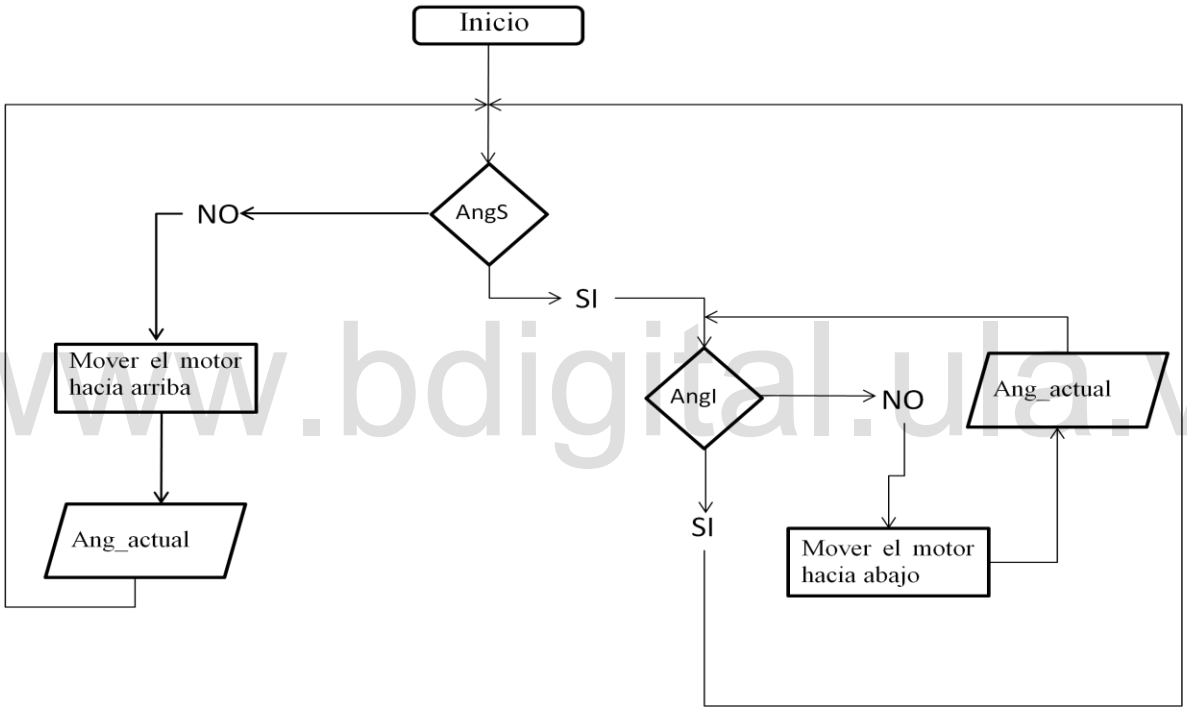


Figura 4. 23: Diagrama de flujo del subprograma

El algoritmo propuesto en la figura 4.23 implementa la dinámica de la planta controlada diseñada con autómatas. Se puede realizar una analogía como se puede observar el la figura 4.24, entendiendo mover el motor hacia arriba y hacia abajo como el comportamiento que se le adiciona al autómata con los interruptores de cada fase del motor.

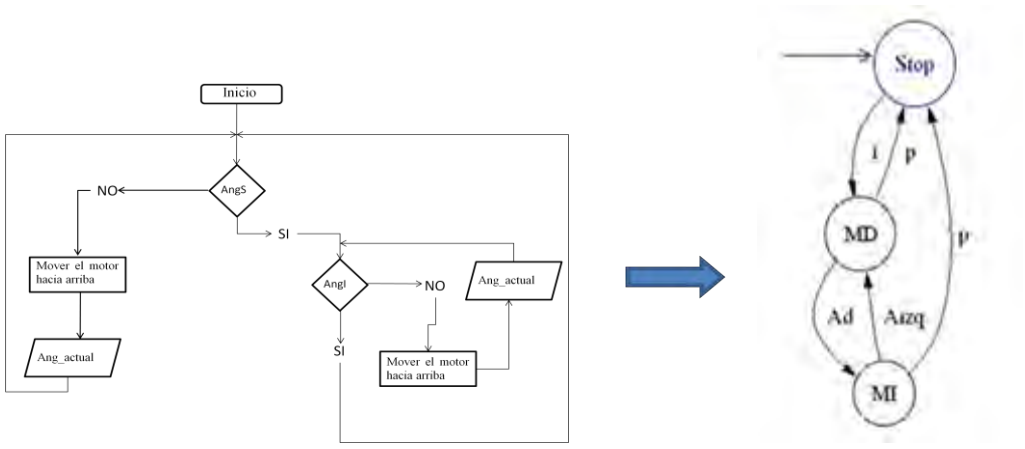


Figura 4. 24: Analogía entre el diagrama de flujo de la implementación del control y el autómata propuesto

La lógica de control es implementada una parte en **Arduino**, la cual es la activación de los interruptores y la adquisición de datos del sensor. Mientras que en **Visual Studio** se decide la dirección del motor, dado el desplazamiento angular proveniente del puerto serial.

- **Programación del Arduino Uno**

Para que la ventana de aplicación funcione correctamente y se vea reflejado en el dispositivo las acciones que se deben realizar, el microcontrolador Arduino Uno permite la codificación de un programa que se estará ejecutando continuamente en el chip al conectar con la computadora.

Este programa se graba en el PIC que incluye la placa, para el trabajo en estudio lo que se desea es enviar la dirección de giro del motor paso a paso y la adquirir el ángulo que proviene del sensor. Para ello se utiliza la comunicación serial la cual consiste en el envío de un bit de información de manera secuencial, esto es, un bit a la vez y a un ritmo acordado entre el emisor y el receptor.

En la placa Arduino Uno el emisor es el pin "TX" y el receptor es el pin "RX", se realiza un protocolo de comunicación entre **Arduino** con **Visual Studio 2010** por medio del uso de una librería.

En la tabla 4.3 se observa el lazo principal del programa el cual se almacena en el PIC, se comienza con la lectura del puerto serial "RX", es la información que proviene del programa de Visual Studio, si hay bits disponible se asigna el valor a la variable *dato* , sino se le asigna cualquier otra letra.

La variable *count* se utiliza para saber cual fase se moverá, recordando que se trabaja con un motor paso a paso unipolar con la configuración de cuatro fases y la alimentación de dichas fases debe realizarse de manera ordenada, el contador garantiza que esto ocurra.

Por medio de una estructura de decisión se conoce cuál es el movimiento a realizar, si dato tiene el carácter "d" el eje del motor se mueve hacia la derecha, sino si es "i" se mueve hacia la izquierda, si es "f" es el final de la terapia se colocan todos los pines en cero y si no es ninguno de los anteriores quiere decir que se está esperando cumplir el tiempo de sostenimiento, por lo tanto no hace nada hasta que la nueva instrucción sea proporcionada.

El último bloque del código es la emisión del valor adquirido con el sensor, se realiza la lectura análoga del valor en voltaje, se aplica la relación lineal obtenida en la sección 3.2 ec.(3.1), para llevarlo a grados y seguidamente enviarlos al puerto serial. Se introduce un *delay* o retardo, que permite fijar el periodo de muestreo en el cual se toma el valor del ángulo de la falange proximal, dicho periodo de muestreo es tomado de la sección 3.3.2.

Tabla 4. 3 : Código que se implementa para la placa Arduino Uno, lazo principal

Código
void loop() {
Continua en la próxima página …

Viene de la página anterior …

```
if(Serial.available()>0){
    dato = Serial.read();
  }else{
    dato = 'n';
    }
  if(count>3){
     count = 0;
     }

   if(dato=='d'){
     moveForward(count);
     count++;
       }else if (dato=='i'){
           moveBackward(count);
          count++;
          }else if (dato=='f'){
             digitalWrite(motorPins[1], LOW);
             digitalWrite(motorPins[0], LOW);
             digitalWrite(motorPins[2], LOW);
             digitalWrite(motorPins[3], LOW);
             }else {}

   tita =-45.23*(analogRead(A0))+119.6;
   Serial.println(tita,4);
   delay(100);
}
```

Para mover hacia la derecha o izquierda el motor, se utiliza la secuencia de doble torque, introduciendo voltaje alto (1volt) o bajo (0volt) para cada interruptor. En la tabla 4.4, se observa que se utilizan cuatro salidas digitales, cada una correspondiente a la alimentación en el circuito de potencia. Para mover a la izquierda se sigue de manera ordenada la secuencia expuesta en la tabla 3.15 y el contador guarda memoria del interruptor en el que quedo para realizar el siguiente paso.

Tabla 4. 4: Código que se implementa para la placa Arduino Uno para mover el motor hacia la izquierda

Código

```
void moveBackward(int count) {


 switch (count) {


  case 0:
   digitalWrite(motorPins[2], LOW);
   digitalWrite(motorPins[1], LOW);
   digitalWrite(motorPins[3], HIGH);
   digitalWrite(motorPins[0], HIGH);
    break;
  case 1:
   digitalWrite(motorPins[1], LOW);
   digitalWrite(motorPins[0], LOW);
   digitalWrite(motorPins[2], HIGH);
   digitalWrite(motorPins[3], HIGH);
         break;
  case 2:
   digitalWrite(motorPins[0], LOW);
```

Continua en la próxima página …

Viene de la página anterior …

```
 digitalWrite(motorPins[3], LOW);
digitalWrite(motorPins[1], HIGH);
digitalWrite(motorPins[2], HIGH);
  break;

case 3:
 digitalWrite(motorPins[3], LOW);
 digitalWrite(motorPins[2], LOW);
 digitalWrite(motorPins[0], HIGH);
 digitalWrite(motorPins[1], HIGH);
    break;

default:
 count =0 ;
}
                                                                    }
```


4.3.2 Programación de la interfaz gráfica

A continuación se muestra como se codificaron en **Visual Studio 2010** las ventanas de la interfaz y parte de la lógica de control (para más detalle revisar el Apéndice D).

i. Ventana de aplicación

Para la codificación de esta etapa se utiliza el diagrama de flujo descrito en la sección 4.2.3, con el uso de cada uno de los elementos dispuestos en la ventana.

Para comenzar se programó el inicio de la terapia con una estructura de decisión, básicamente esta fase está controlada por el evento de presionar el botón **Inicio**. En el momento que este evento ocurre se verifican que los valores obligatorios pedidos estén correctos (véase la tabla 4.5), sino es el caso se abre una ventana de aviso para que verifique la información pedida. Así mismo se inicializa un *timer* o reloj para que comience inmediatamente la terapia.

Tabla 4. 5: Código para la implementación del botón de inicio de la ventana principal

Código

```
'Botón de inicio y parada de la aplicación

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click

        ' Comprueba que los valores pedidos sean correctos

        If CDbl(TextBox2.Text) * 1000 > 550 And CDbl(TextBox6.Text) > 0 And
CDbl(TextBox3.Text) <= 60 And CDbl(TextBox4.Text) <= 80 Then

            ' Asigna el valor del ángulo superior deseado

            angS = CDbl(TextBox3.Text)

            ' Asigna el valor del ángulo inferior deseado

            angI = CDbl(TextBox4.Text)


            If cambio12 = True Then

                'Habilita el reloj

                Timer1.Enabled = True

                Button3.Text = "Stop"

                'Cambia el color del óvalo a verde'

            OvalShape1.BackColor = Color.Green

            I = 0 'Inicializar el contador del tiempo.
```

Continua en la próxima página …

Viene de la página anterior …

```
            TextBox5.Text = " "
            Timer1.Interval = 1    'Iniciar el cronómetro en un segundo

        Else
            'Deshabilita el reloj
            Timer1.Enabled = False
            SerialPort1.Write("f")
            Button3.Text = "Inicio"
            'Cambia el color del óvalo a rojo
            OvalShape1.BackColor = Color.Red
            'Cambia la imagen
            PictureBox1.Image = My.Resources._stop
        End If

        cambio12 = Not (cambio12)

    Else
        MsgBox("Por favor complete la información pedida")
    End If

End Sub
```


Seguidamente se programó la ventana de visualización del desplazamiento angular que tiene el exoesqueleto en tiempo real. Se usan hilos o subprocesos, es una característica que

permite a una aplicación realizar varias tareas a la vez. Esta herramienta es usada, debido a que mientras se controla el dispositivo es necesario observar el ángulo actual del mismo.

En la tabla 4.6 se observa la implementación de la función *ThreadProcSafe,* siendo un hilo codificado de manera segura, el cual genera un bucle infinito, que adquiere la información enviada del puerto serial del Arduino Uno. Además del hilo anterior es necesaria la utilización de la función *SetText*, que permite asignar de manera segura el valor del ángulo actual en un *TextBox*, logrando la visualización de este valor en la pantalla.

El segundo bloque de decisión de la función *ThreadProcSafe*, verifica si el tiempo de terapia se cumplió, de ser el caso inhabilita el reloj e inicializa los valores.

Tabla 4. 6: Código que implementa la adquisición del ángulo en la ventana principal

Código

```
Private Sub ThreadProcSafe()

    Do While True

        ang_actual = CDbl(SerialPort1.ReadLine)

        SetText(SerialPort1.ReadLine)


        If I >= TiempoTerapia Then

            'Deshabilita el reloj

            Timer1.Enabled = False

            SerialPort1.Write("f")

            cambio12 = Not (cambio12)

            SetText1("Inicio")

            SetText2("n")

        End If
```

Continua en la próxima página …

Viene de la página anterior …

```
        Loop
    End Sub


    Private Sub SetText(ByVal [text] As String)


        If Me.TextBox1.InvokeRequired Then
            Dim d As New SetTextCallback(AddressOf SetText)
            Me.Invoke(d, New Object() {[text]})
        Else
            Me.TextBox1.Text = [text]
        End If
    End Sub
```


Finalmente se codifica la parte más importante del programa, como lo es el control supervisorio, esta fase está asociada al *Tick* del reloj declarado (véase la tabla 4.7), entendiendo por *Tick* el intervalo de tiempo que transcurre para que se vuelva a ejecutar todo lo contenido en la función *Timer1_Tick*, en nuestro caso cada milisegundo. Cabe destacar que este *Tick*, tiene que ver con el tiempo de ejecución del programa no con el de muestreo.

Tabla 4. 7: Código que implementa el control supervisorio en la ventana principal

Código

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Timer1.Tick

  'Cuenta los segundos transcurridos una vez habilitado el reloj

   I = I + 1
```

Continua en la próxima página …

Viene de la página anterior …

```
' Asigna a la variable tiempo el valor del tiempo en el formato
horas/minutos/segundos

        Tiempo = Format(Int(I / 36000) Mod 24, "00") & ":" & _

                 Format(Int(I / 600) Mod 60, "00") & ":" & _

                 Format(Int(I / 10) Mod 60, "00")

        'Actualiza el tiempo actual de la terapia

        TextBox5.Text = Tiempo

        ' Ya cada paso se realiza segun el tiempo de sostenimiento deseado

        ' Se mueve el motor una vez cumplido dicho tiempo de sostenimiento

        ' el count_t sirve para actializar el ciclo en el cual debe ser movido

        If I = count_t * tiempo_sos Then


            If cambio13 And ang_actual <= angS Then

                SerialPort1.Write("i")

            Else

                cambio13 = False

            End If


            If Not (cambio13) And ang_actual >= angI Then

                SerialPort1.Write("d")

            Else

                cambio13 = True

            End If

            count_t = count_t + 1


        End If


    End Sub
```

En cada *Tick* se incrementa un contador que almacena los milisegundos transcurridos en total, se actualiza la casilla de tiempo actual para mostrarle en la pantalla al usuario el tiempo transcurrido y seguidamente se ejecuta el control.

Se comienza verificando el alcance al ángulo superior deseado, *cambio13* es un valor booleano que se comporta como un evento, es decir, cuando se llegó al ángulo superior el evento se coloca en falso para cambiar la dirección, esta variable permitirá que el control se realice de manera repetitiva y adecuada.

Una vez alcanzado el ángulo superior deseado, se cambia la dirección del movimiento del eje del motor y se continúa la secuencia hasta que se llega al ángulo inferior deseado, para seguidamente repetir el proceso.

Cabe destacar que para el cambio de dirección del motor, se envía al puerto serial la instrucción correspondiente para el movimiento hacia la derecha (SerialPort1.Write("d")) y el movimiento hacia la izquierda (SerialPort1.Write("i")).

ii. Ventana de Desplazamiento Angular

Se declaran como variables globales en la ventana de aplicación los valores del ángulo actual del exoesqueleto (*ang_actual*) y el tiempo que está transcurriendo una vez iniciada la terapia, este último es dado por el contador de milisegundos *I*.

En el programa se utiliza un control de **Visual Studio 2010** llamado *Chart*, el cual permite la creación de una gráfica. En la tabla 4.8 se observa que se utiliza un hilo para graficar en tiempo real en el *Chart1*, la asignación de *x* y *y* se hace de manera segura en los ejes de la gráfica por medio de la función *SetData*.

Tabla 4. 8: Código que se implementa para la ventana Desplazamiento Angular

Código

```
Private Sub ThreadTask()
        Do While True
            SetData(I / 10, ang_actual )
            Thread.Sleep(1000)
        Loop
 End Sub

 Private Sub SetData(ByVal x As Double, ByVal y As Double)
        If Me.Chart1.InvokeRequired Then
            Dim d As New SetTextCallback(AddressOf SetData)
            Me.Invoke(d, New Object() {x, y})
        Else
            Me.Chart1.Series("Ángulo").Points.AddXY(x, y)
        End If
 End Sub
```


iii. Ventana de Mímico

Al igual que la ventana de Desplazamiento Angular, se utiliza la variable global *ang_actual*. Se dispone de una recta que representa un eslabón del exoesqueleto, la posición de la misma se puede cambiar asignando los valores de (x_0, y_0) y (x_1, y_1), observe la figura 4.25. Los puntos (x_0, y_0) están fijos en el origen del eje de coordenadas, mientras que (x_1, y_1) son los que varían para mostrar el desplazamiento angular que está ocurriendo en tiempo real.

Dado el ángulo θ proveniente del programa aplicación, los valores de (x_1, y_1) se calculan de la siguiente manera:

$$x_1 = \cos\theta \qquad (4.4)$$

$$y_1 = \sin\theta \qquad (4.5)$$

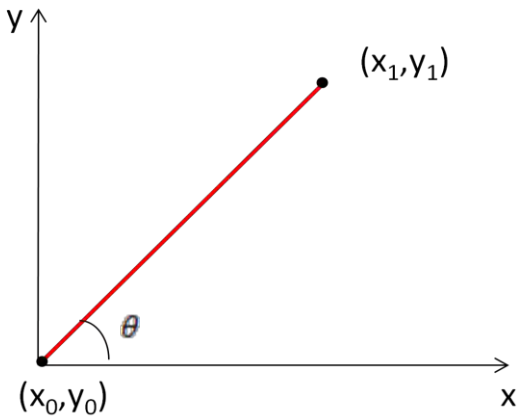


Figura 4. 25: Recta que representa un péndulo en el plano XY

Las ecuaciones (4.4) y (4.5) se adaptan al código mostrado en la tabla 4.9 para implementar la ventana. Se declara un *PictureBox* en él se dibuja un rectángulo, que va a contener una cuadricula de rectas formadas por una estructura de repetición.

Luego se dibuja una recta de color rojo con el ángulo proveniente del sensor. Este código también es implementado con hilos, ya que es un subproceso que siempre debe estar corriendo una vez abierta la ventana.

Tabla 4. 9: Código que implementa la ventana Mímico

Código

`Private Sub ThreadTask()`
 	
`Do While True`

Continua en la próxima página …

Viene de la página anterior …

```
  Private Sub ThreadTask()

        Do While True

            pMyPen = New Pen(Brushes.Black, 1)

            gMyGraphics = PictureBox1.CreateGraphics()

            gMyGraphics.FillRectangle(bMyBrush, 0, 0, PictureBox1.Width,
PictureBox1.Height)

            Dim I As Integer

            For I = 0 To 400

                gMyGraphics.DrawLine(pMyPen, I, 0, I, PictureBox1.Height)

                gMyGraphics.DrawLine(pMyPen, PictureBox1.Width, I, 0, I)

                I = I + 20

            Next I

            pMyPen2 = New Pen(Brushes.Red, 5)

            ang = ((ang_actual / 10000) * 3.1416) / 180

             If ang > 0 Then

                x = Abs(Math.Cos(ang)) * 400

                y = 200 - Abs(Math.Sin(ang) * 400)

            Else

                x = Abs(Math.Cos(ang)) * 400

                y = 200 + Abs(Math.Sin(ang)) * 400

            End If

            aux = PictureBox1.Height / 2

            gMyGraphics.DrawLine(pMyPen2, 0, aux, x, y)

            Thread.Sleep(500)

        Loop

    End Sub
```

Para culminar se probó el funcionamiento de la aplicación con el microcontrolador y la planta, observando el comportamiento adecuado de la interacción de todos los elementos. A continuación en las Figuras 2.26, 2.27, 2.28 y 2.29 se muestra una secuencia de imágenes del sistema funcionando:

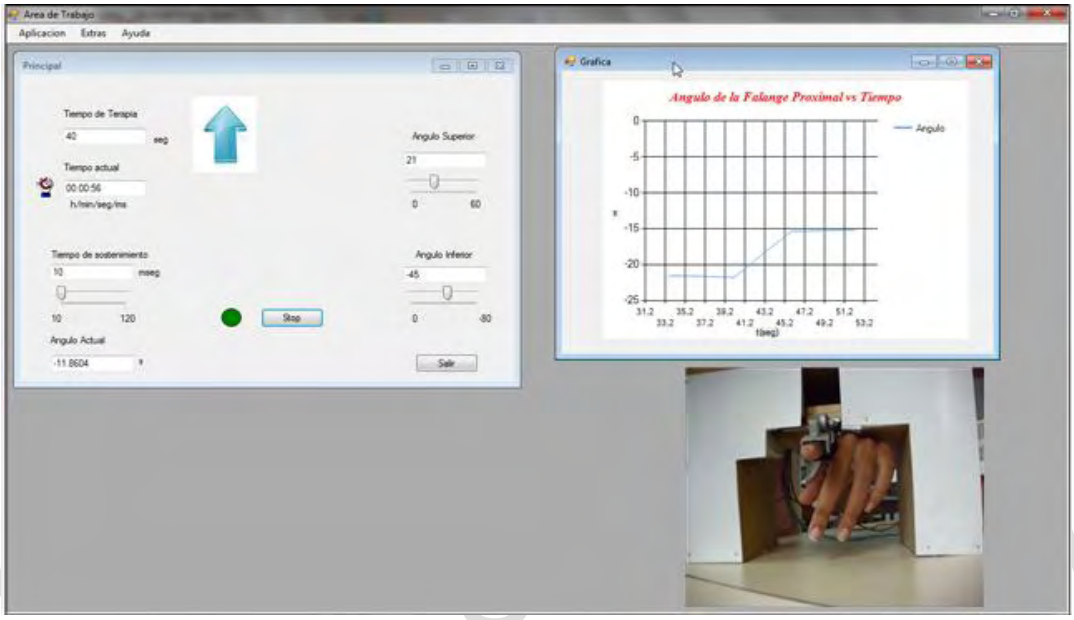


Figura 4. 26: Secuencia de funcionamiento del sistema 1.

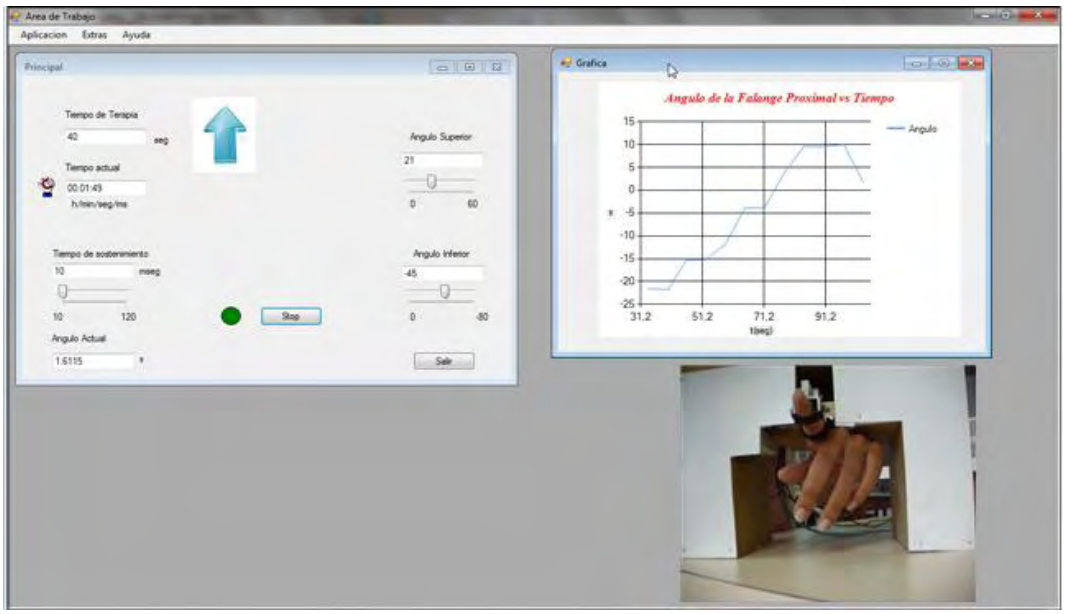


Figura 4. 27: Secuencia de funcionamiento del sistema 2.

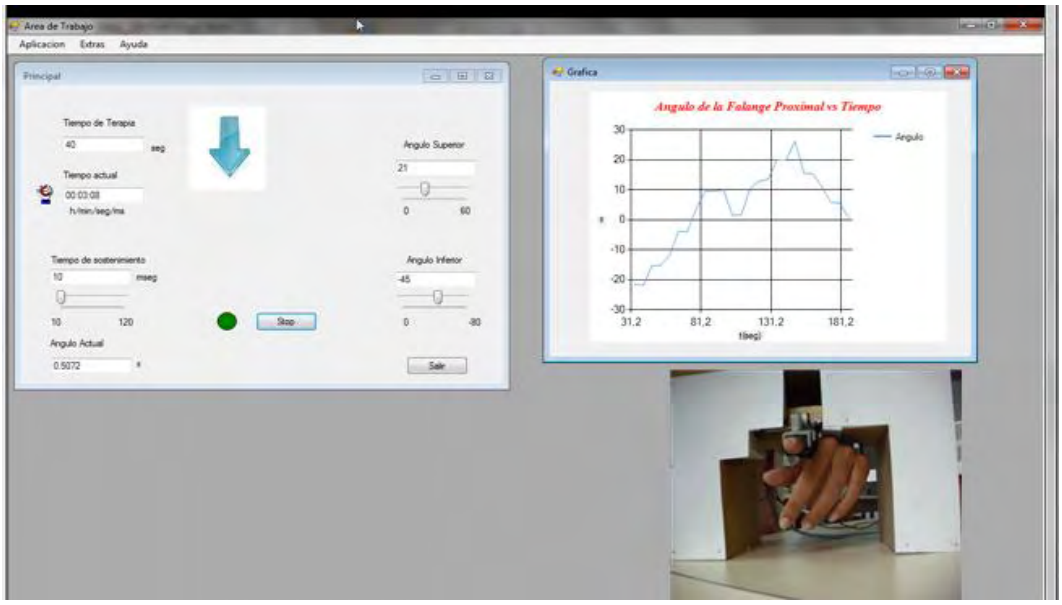


Figura 4. 28: Secuencia de funcionamiento del sistema 3.

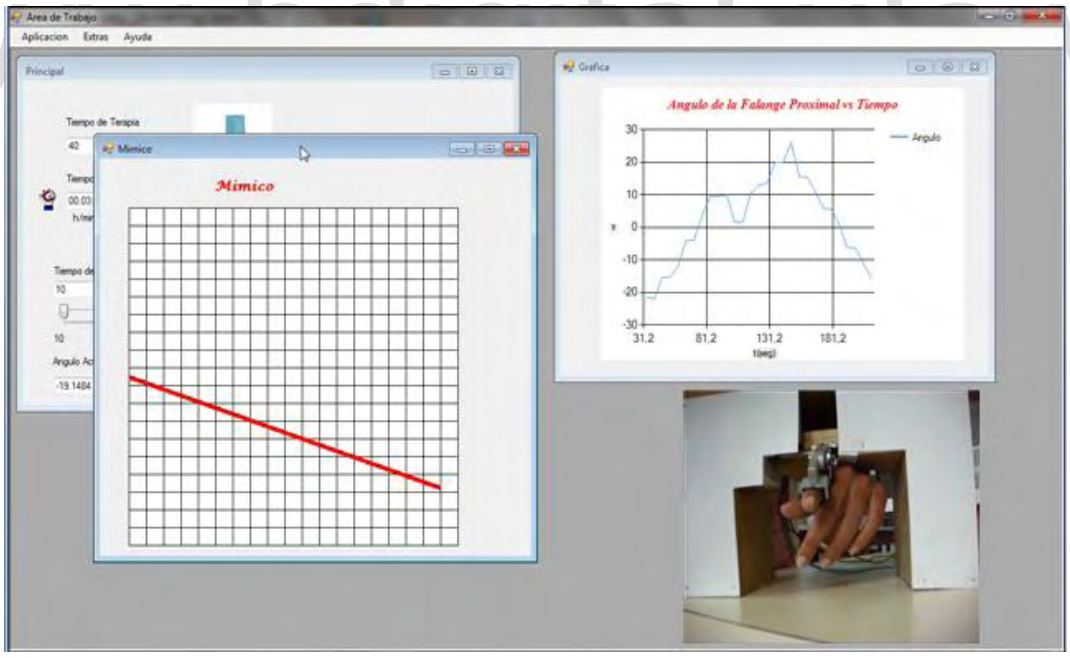


Figura 4. 29: Secuencia de funcionamiento del sistema 4.

Capitulo 5

Conclusiones

El resultado de este proyecto de grado es el diseño, construcción y control de un exoesqueleto orientado a la rehabilitación del dedo índice de la mano.

Se propuso un prototipo que fue implementado y adaptado de acuerdo a las necesidades de la mano humana, las cuales fueron verificadas con la información recopilada de la anatomía de la mano.

El sensor escogido para medir el desplazamiento angular de la falange proximal del dedo índice para su posterior control fue un potenciómetro rotatorio, con una relación lineal permitió obtener la variable deseada. El actuador que se dispuso al prototipo fue un motor paso a paso unipolar que genera el torque necesario para el control en lazo abierto del desplazamiento angular del primer eslabón del exoesqueleto.

El exoesqueleto implementado se adapta a un modelo dinámico del péndulo de un eslabón, en el cual fue incluida la dinámica del motor paso a paso. Los resultados obtenidos al realizar comparaciones fueron satisfactorios.

Al integrar el dedo índice al exoesqueleto, los resultados obtenidos al realizar comparaciones entre los datos reales y simulados fueron satisfactorios. A pesar, de que fue necesario modificar el modelo, debido a que el dedo se opone al movimiento del motor, es admisible

pensar que se está ejerciendo una fricción más elevada. Es importante señalar, que este estudio se probó con varias personas, donde el valor del parámetro *b* cambia, debido a la incertidumbre del movimiento del dedo índice de cada mano.

Para abarcar el comportamiento del segundo eslabón integrado al exoesqueleto en el modelo dinámico, se plantean diferentes alternativas que ayudarán a trabajos futuros.

La descripción del sistema como un SED fue adecuada, debido a que el funcionamiento como tal del motor lo permitía, el supervisor obtenido proporciona gran información del comportamiento del sistema y sirve para comprender el control propuesto, el cual, en conjunto corresponde al control de un sistema hibrido (continuo y a eventos).

El estudio teórico permitió diseñar la lógica de control y la interfaz gráfica, implementada con las herramientas de computación necesarias. A partir de ella, el exoesqueleto de un eslabón permite realizar terapias de manera automática definidas por el usuario.

Este proyecto permitirá abrir nuevos campos de investigación en biomédica, también a promover este tipo de trabajos, de manera que a futuro se cuenten con estos dispositivos en cualquier sala de terapia ocupacional.

5.1 Recomendaciones

El exoesqueleto robótico puede mejorarse cambiando el material que se utilizó para su construcción, también implementando las piezas de una manera más fina. Cambiar la base de madera por un material más ligero y ergonómico para su uso.

Colocar un motor y un sensor en la segunda articulación, para realizar un estudio completo con dos eslabones. Se estudiarían otro tipo de motores, de manera que se adapte de manera adecuada al prototipo. Con esto se pueden ampliar los patrones de rehabilitación para el dedo índice y mejorar la interfaz gráfica.

Diseñar supervisores adicionando el comportamiento con dos eslabones, es decir con las dos terapias propuestas. La primera es que se fije el primer eslabón y se mueva el segundo, para después fijar el segundo y mover el primero, la otra terapia es realizando el movimiento de ambos eslabones simultáneamente.

Diseño de leyes de control más complejas, para cumplir terapias con más detalles. Como por ejemplo la variación de la fuerza aplicada sobre el dedo índice.

Bibliografía

Arduino- Home Page. Extraído el 3 de Agosto de 2012 del sitio web: http://www.arduino.cc/es/.

Badinez R., (2007). Diseño y Simulación de un Instrumento para la Estimación de Torque de un Motor Paso a Paso. Proyecto de Grado. Universidad de Chile, Santiago de Chile. Disponible en: http://www.cybertesis.cl/tesis/uchile/2007/badinez_rl/html/index.html.

Basdogan C., Ho C. and Srinivasan M. (2001). Virtual environments for medical training: Graphical and haptic simulation of laparoscopic common bile duct exploration. IEEE Transactions on Mechatronics. Vol.6(3), pp. 267-285.

Brown P., Jones D. and Singh S. (1993). The exoskeleton glove for control of paralyzed hands. Proceedings of the IEEE International Conference on Robotics and Automation.Vol.1, pp. 642-647.

Calennoff L. (1972). Angiography of the hand. Guidelines for interpretation. Diagnostic Radiology. Vol.102, pp.331-335.

Chavéz M., Spitia F. y Lopéz A. (2010). Exoesqueletos para potenciar las capacidades humanas y apoyar la rehabilitación. Revista Ingeniería Biomédica. Vol. 4(7), pp. 63-73.

DiCicco M., Lucas L. and Matsuoka Y. (2004). Comparison of control strategies for an emg controlled orthotic exoskeleton for the hand. In Proceedings of the 2004 IEEE International Conference on Robotics and Automation. Vol. 2, pp. 1622-1672.

Fiorillaa A., Tsagarakis N., Nori F. and Sandini G.(2007). Design of a 2-Finger Hand Exoskeleton for finger stiffness measurements. Journal of Applied Bionics and Biomechanics. Vol. 6(2), pp. 1-12.

Kazerooni H. (1990). Human-robot interaction via the transfer of power and information signals. IEEE Transaction On Systems, Man and Cybernetics. Vol. 20, pp. 450-463.

Khorrami F., Krishnamurthy P. and Melkote H. (2003). Modeling and Adaptive Nonlinear Control of Electric Motors. Editorial Springer, Heidelberg.

Kumar S. (2010). Introducción a la Robótica. Editorial McGraw-Hill Interamerericana.

Kuo B. (1996). Sistemas de Control Automático. Séptima Edición. Prentice Hall, Hispanoamericana. S.A.

Lucas L., DiCicco M. and Matsuoka Y. (2004). An EMG-Controlled Hand Exoskeleton for Natural Pinching. Journal of Robotics and Mechatronics. Vol. 16(5), pp. 482-488.

Miralles R. (2004). Tratamiento Fisioterapéutico de los tendones extensores. Centre de Cooperació al Desenvolupament, URV Solidaria, Universitat Rovira i Virgili Tarragona. Disponible en: http://www.ucam.edu/revistafisio/numeros/volumen-4/numero-2-diciembre-2005/REVISTA%20FISIO%20VOL4%20NUM2%20-Dic%202005.pdf.

Morar Alexandru. (2003). Stepper Motor Model for Dynamics Simulation. Achta Electronica. Vol 44(2), pp. 620-625.

Morón S. (2007). Terapia Ocupacional en la rehabilitación de la mano de pacientes Tetrapléjicos. Revista Gallega de Terapia Ocupacional TOG. Disponible en: http://www.revistatog.com/num6/pdfs/revision1.pdf.

Mourad W. (2008). Da Vinci opera en Caracas. Revista Estampas Temática. Cirugía General Hospital de Clínicas Caracas. Disponible en: http://www.eluniversal.com/estampastematica/archivo/salud010308/vivirsalud4.shtml.

Ogata K. (1987). Dinámica de Sistemas. Editorial Prentice-Hall Hispanoamericana,S.A.

Palastanga N., Demer F., Field D. y Soames, R. (2000). Anatomía y movimiento humano: estructura y funcionamiento. Editorial Paidotribo.

Ramadge P. and Wonham W. (1987). Supervisory Control of a Class of Discrete Event Processes. SIAM J. Control and Optimazation. Vol. 25(1), pp 206-230.

Sabater J. (2003). Desarrollo de una Interfaz Kinestésica Paralela y Experimentación en Control de Sistemas Hápticos y Teleoperados. PhD thesis, Universidad Miguel Hernández. Disponible en: http://www.isa.umh.es/personal/j.sabater/index.html.

Sarakoglou I., Tsagarakis N. and Caldwell D. (2004). Occupational and physical therapy using a hand exoskeleton based exerciser. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems. Vol. 3, pp. 2973-2978.

Wang J., Jiting L., Zhang Y. and Wang S. (2009). Design of an Exoskeleton for Index Finger Rehabilitation. 31st Annual International Conference of the IEEE EMBS Minneapolis, Minnesota, USA. Conference Publications, pp. 5957-5960.

Wege A. y Hommel G. (2005). Development and control of a hand exoskeleton for rehabilitation of hand injuries. In Proceedings of the IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS). Conference Publications, pp. 3046-3051.

Zoss A. and Kaserooni H. (2006). Desing of an electrically lower extremity exoskeleton. Departament of Mechanical Engineering. Advanced Robotics. Vol. 20(9), pp. 967-988.

Apéndices

Apéndice A

Diseño mecánico del exoesqueleto robótico

A continuación se muestran las piezas diseñadas con **Autodesk Inventor.**

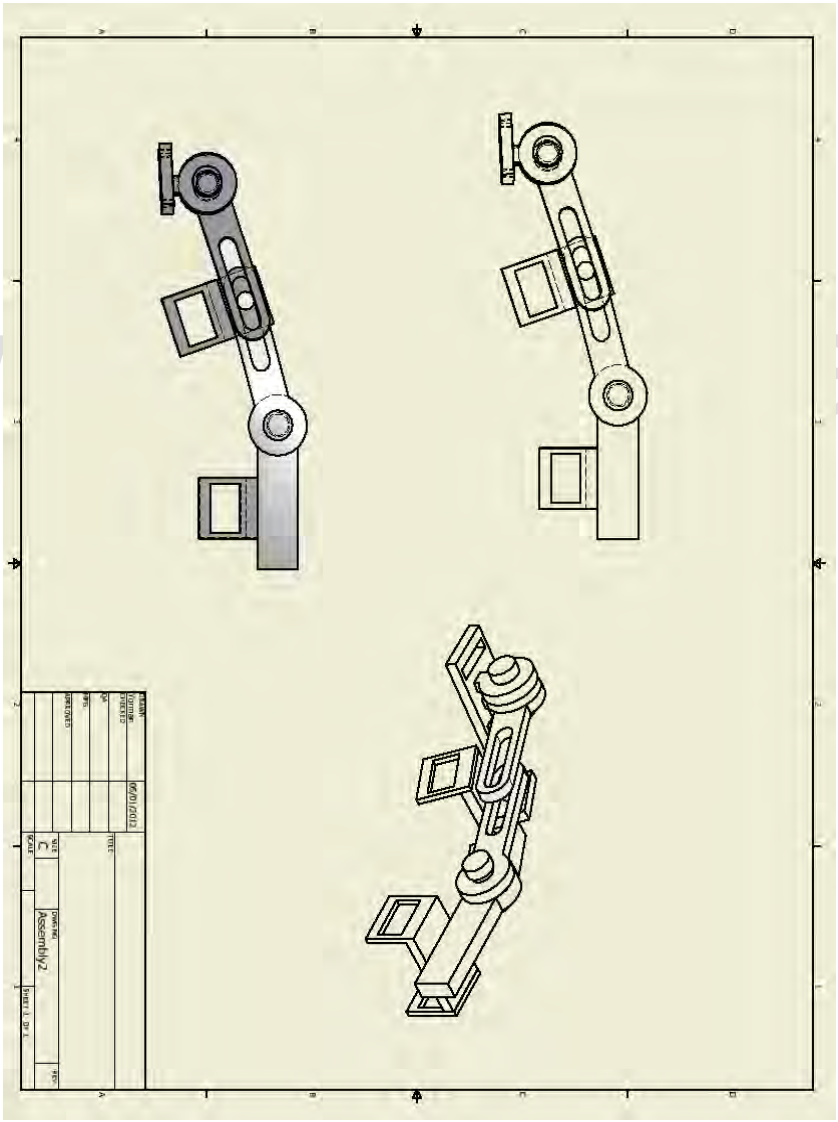


Figura A. 1: Prototipo diseñado y ensamblado del exoesqueleto robótico

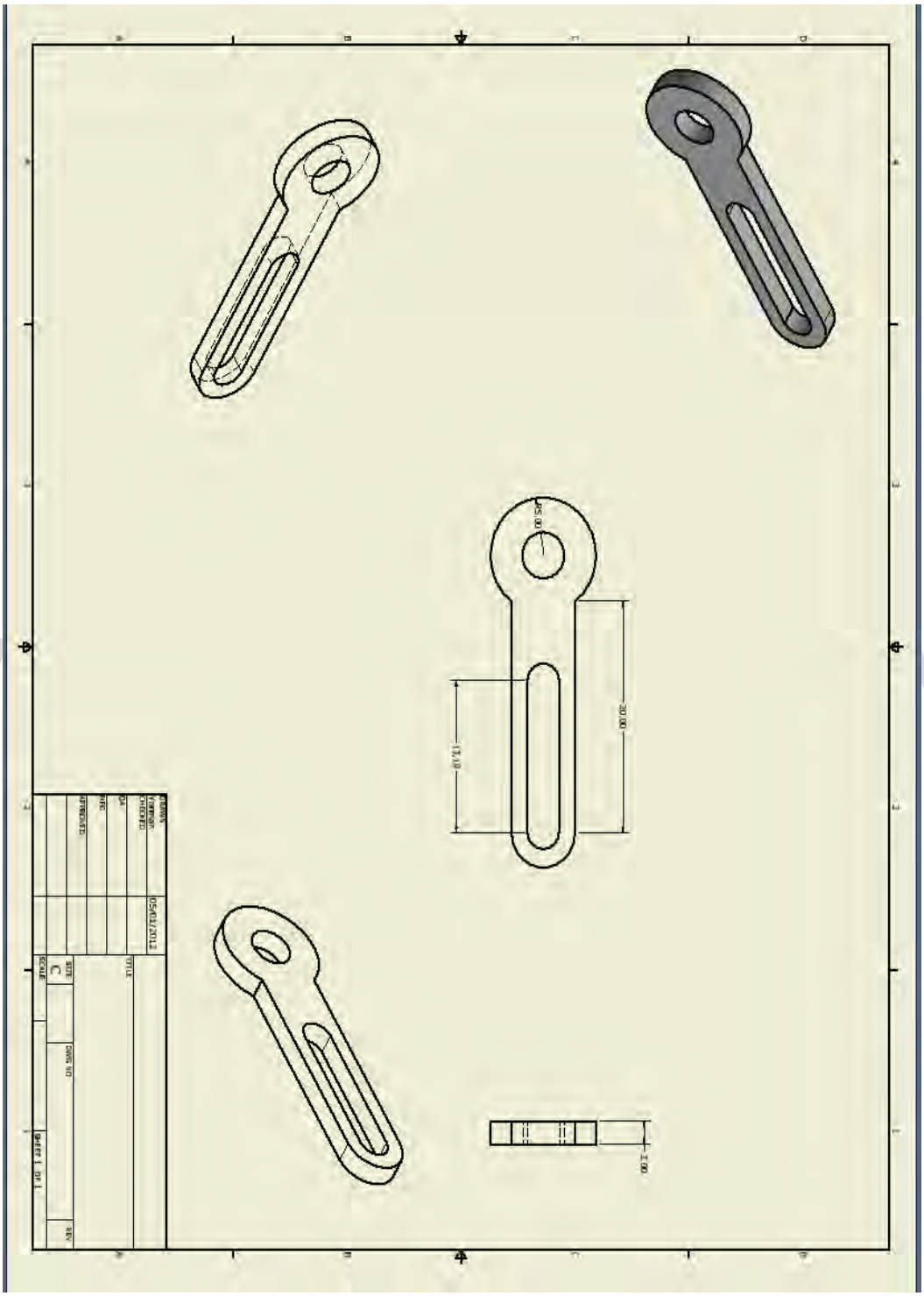


Figura A. 2: Pieza de reducción y ampliación de la longitud del eslabón 1

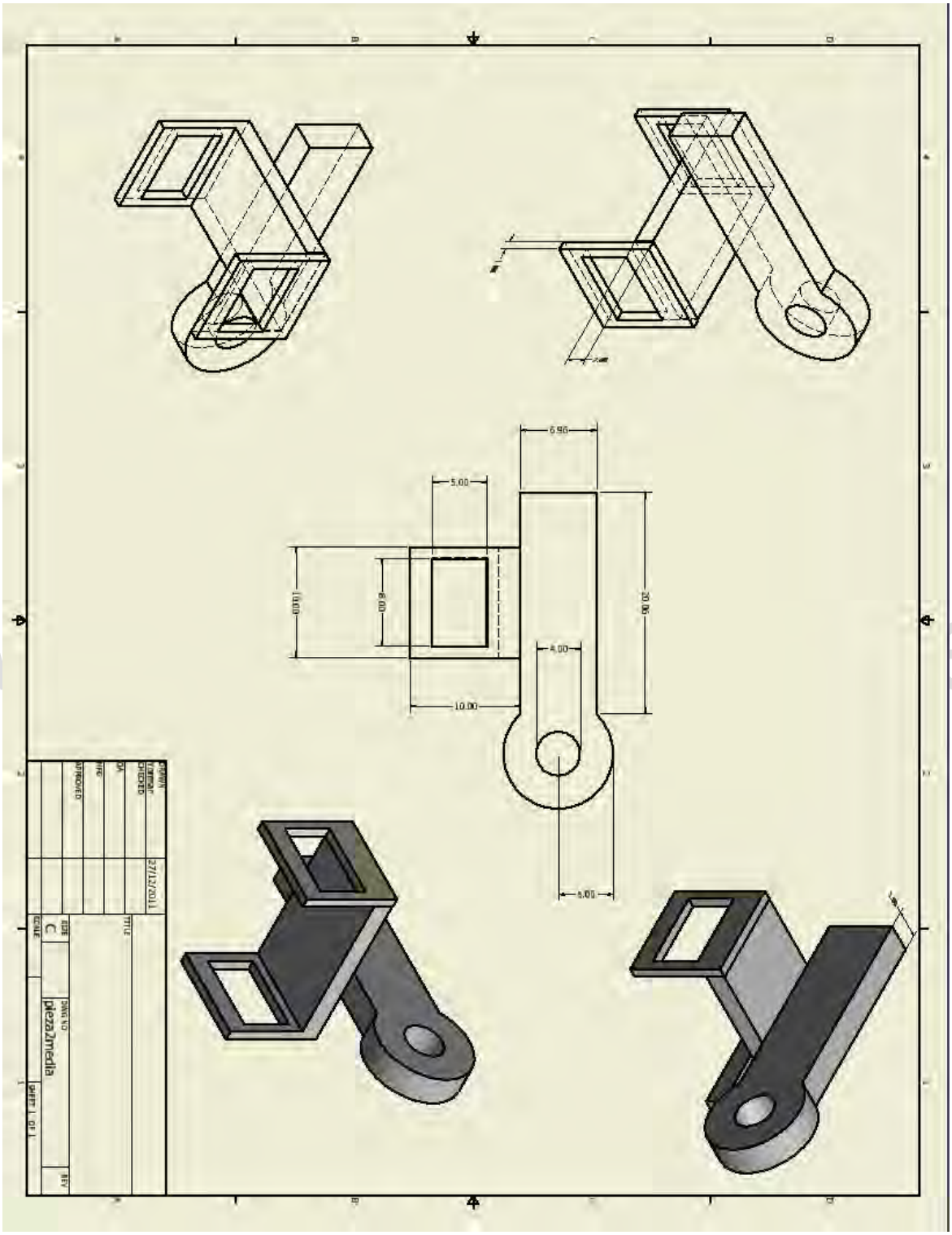


Figura A. 3: Pieza que permite acoplar el dedo en el exoesqueleto robótico

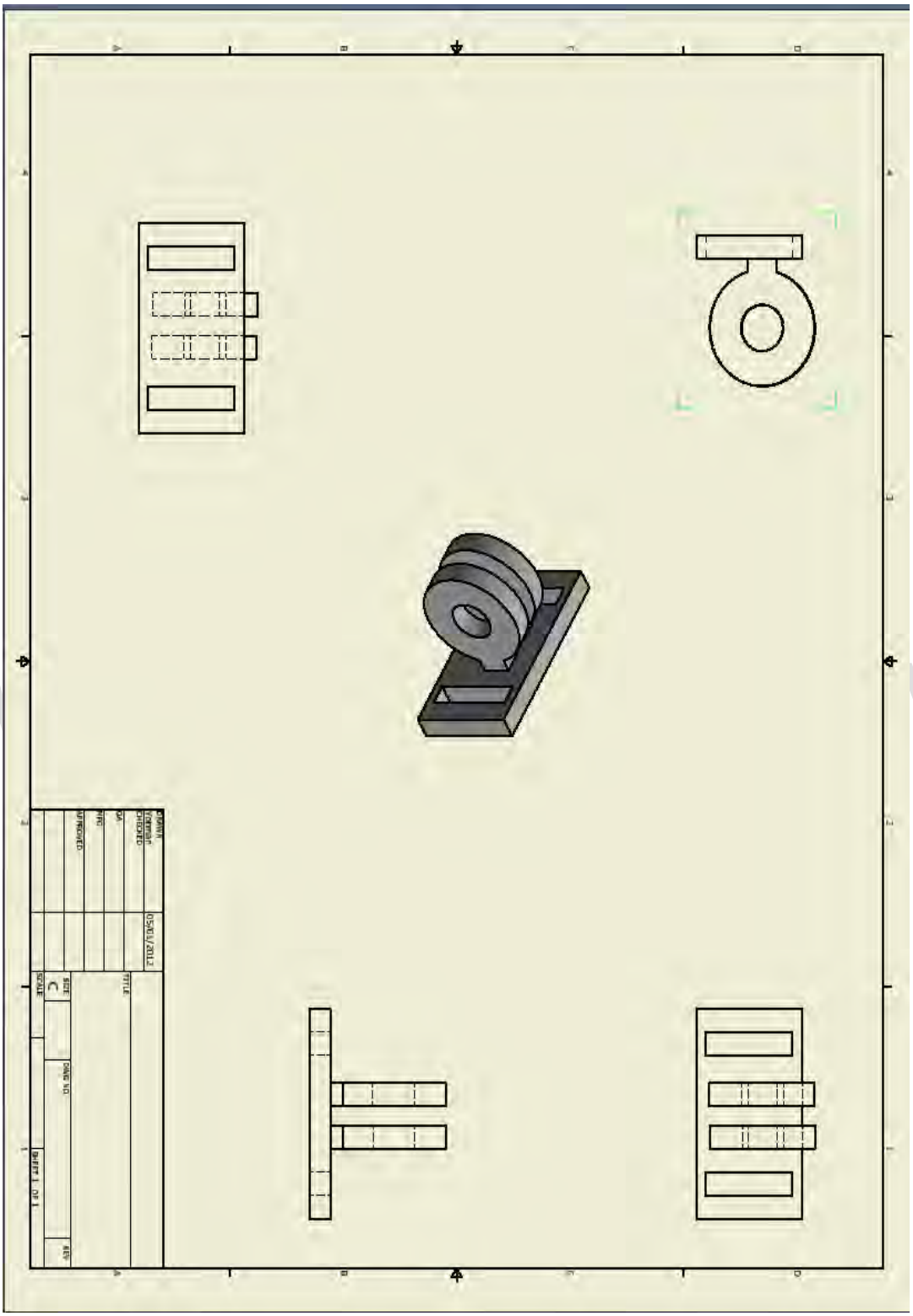


Figura A. 4: Base del exoesqueleto que se fija a una base de madera

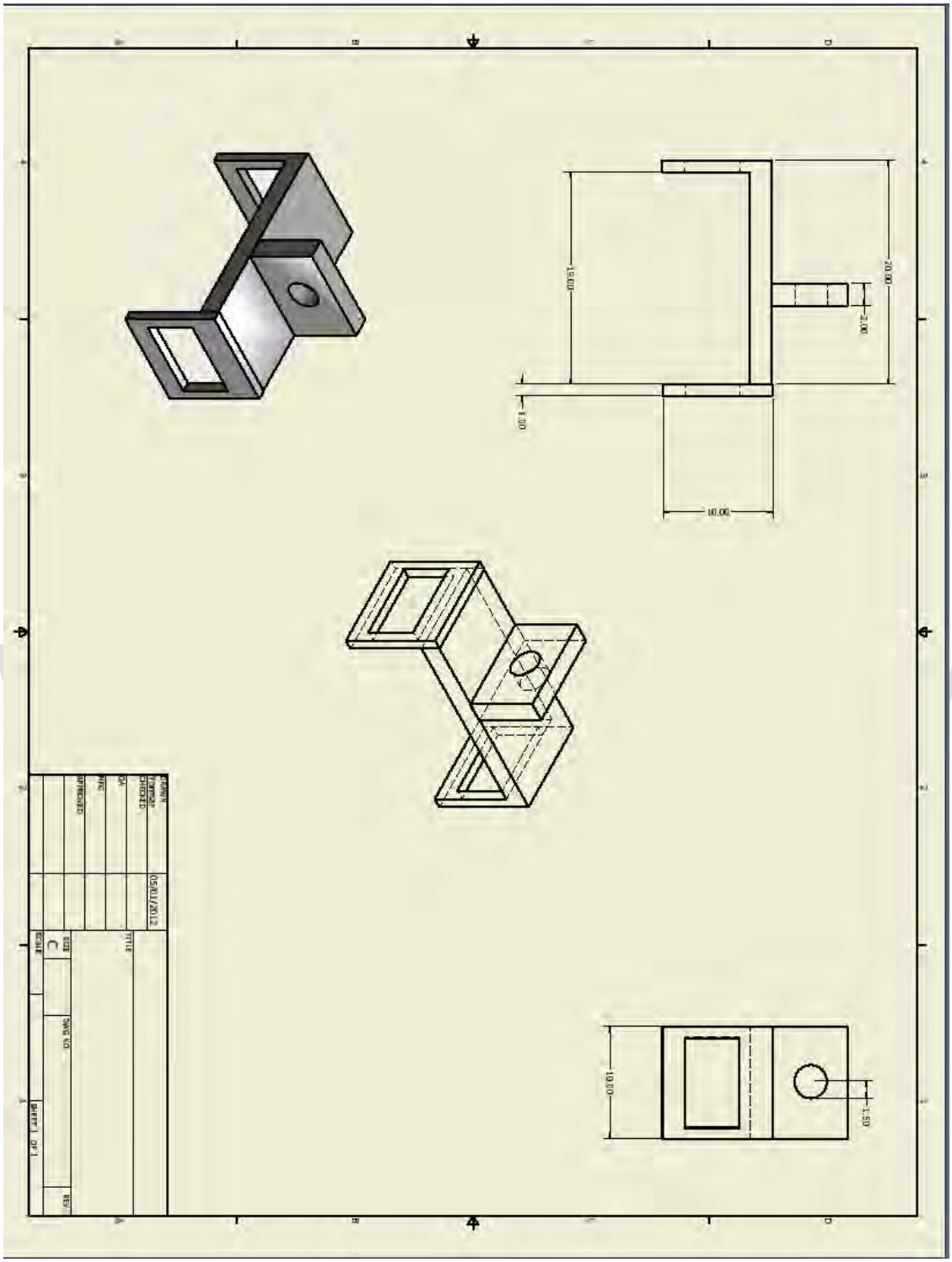


Figura A. 5: Pieza que permite la unión de los dos segmentos que conforman el primer eslabón

Apéndice B

Análisis del motor con la planta

- Identificación con la tarjeta LabJack

Este programa permite identificar el voltaje dado por el sensor, por medio de la tarjeta de adquisición de datos.

```
clc % Clears the command window
clear global % Clears all global variables

ljud_LoadDriver; % Loads LabJack UD Function Library
ljud_Constants; % Loads LabJack UD constant file
[Error ljHandle] = ljud_OpenLabJack(LJ_dtU3,LJ_ctUSB,'1',1); % Returns
ljHandle for open LabJack
Error_Message(Error) % Check for and display any Errros

%Start by using the pin_configuration_reset IOType so that all
%pin assignments are in the factory default condition.
[Error] = ljud_ePut(ljHandle, LJ_ioPIN_CONFIGURATION_RESET, 0, 0, 0);
Error_Message(Error)

%First some configuration commands.  These will be done with the ePut
%function which combines the add/go/get into a single call.

[Error] = ljud_ePut(ljHandle, LJ_ioPUT_ANALOG_ENABLE_PORT, 0, 12, 16);
Error_Message(Error)


%------------------------------------------------------------------------
acum=0; To=0.0001;
TSIM = 0.3;
tt=[0];
yy=[0];
uu=[0];
i=0;

while acum <= TSIM,

    [Error AIN0] = ljud_eGet(ljHandle,LJ_ioGET_AIN,0,0,0);
    Error_Message(Error)

    y = AIN0;

     tic
    while toc < To,
```

```
    end

    yy=[yy,y];
    tt=[tt;To*i];
    plot(tt,yy);
    drawnow
    i=i+1;
    acum= acum+To;
end
```

- Programa para el uso del motor con Matlab

```
function varargout = motorGUI2(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @motorGUI2_OpeningFcn, ...
                   'gui_OutputFcn',  @motorGUI2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before motorGUI2 is made visible.
function motorGUI2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to motorGUI2 (see VARARGIN)
 diego=digitalio('parallel','LPT1');
 dato=addline(diego,0:3,'out');
 putvalue(dato,0);

% Choose default command line output for motorGUI2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes motorGUI2 wait for user response (see UIRESUME)
```

```
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = motorGUI2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


%lectura de la velocidad del motor
function speed_Callback(hObject, eventdata, handles)
% hObject    handle to speed (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of speed as text
%        str2double(get(hObject,'String')) returns contents of speed as a
double
handles.delay=str2double(get(hObject,'String'))*1e-3;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function speed_CreateFcn(hObject, eventdata, handles)
% hObject    handle to speed (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


%al presionar el boton se incrementa la velocidad
function up_Callback(hObject, eventdata, handles)
global a
%return
a=str2double(get(handles.speed,'String'));
a=a+10;
set(handles.speed,'String',a);
handles.delay=a*1e-3;
guidata(hObject,handles);


%al presionar el boton se decrementa la velocidad
function down_Callback(hObject, eventdata, handles)
```

```
global a
%return
a=str2double(get(handles.speed,'String'));
a=a-10;
set(handles.speed,'String',a);
handles.delay=a*1e-3;
guidata(hObject,handles);



%al presionar el boton se cambia la direccion del motor
% --- Executes on button press in direction.
function direction_Callback(hObject, eventdata, handles)
% hObject    handle to direction (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
f=get(handles.direction,'Value');
if f==1
    set(handles.direction,'String','DIRECTION ''L''');
else
    set(handles.direction,'String','DIRECTION ''R''');
end

% este cambia el estado del motor ya sea encendido o apagado
% --- Executes on button press in state.
function state_Callback(hObject, eventdata, handles)
d=get(hObject,'Value');
if d==1
    set(handles.state,'String','ON');
    diego=digitalio('parallel','LPT1');
    dato=addline(diego,0:3,'out');
    g=1;
    while g
        e=get(handles.direction,'Value');
        if e==0
            mov=[3 6 12 9];
        else
            mov=[9 12 6 3];
        end
        delay=str2double(get(handles.speed,'String'))*1e-3;
        if delay<0 ||isnan(delay)
            errordlg('Time out of range','ERROR');
            delay=500;
            set(handles.speed,'String',500);
            set(handles.state,'String','OFF');
            set(handles.state,'Value',0);
            break;
        end
        if get(hObject,'Value')==0
            break
        end
        putvalue(dato,mov(1));
        pause(delay);
        if get(hObject,'Value')==0
            break
```

```
        end
        putvalue(dato,mov(2));
        pause(delay);
        if get(hObject,'Value')==0
            break
        end
        putvalue(dato,mov(3));
        pause(delay);
        if get(hObject,'Value')==0
            break
        end
        putvalue(dato,mov(4));
        pause(delay);
    end

else
    set(handles.state,'String','OFF');
end
```

- Programa para simular el motor paso a paso con un eslabón

a. Programa principal main.m


```
global R L km kd J D b Vnom Ts Tc t0 dt m1 lc1 I1  g ;

%INICIALIZACION
% peso de un eslabon de 30 gr
% peso en kg
m1= 0.03;

% longitudes en metros m
l1=6/100; %45cm
lc1=l1/2
g=9.8;
b=0.005;
TM=[0];
% Inercia del eslabon
a=1/100;
I1=(1/12)*m1*(a*a+l1*l1)+(1/4)*m1*(l1*l1)

R=20;
L=0.006;
km=0.2;
kd=km/20;
D=0.008;
J=5e-5;

Vnom=9;
```

```
f=1/4;


Ts=0;
%cond inicial [teta w ia ib]
xo=[-(85)*pi/180 0 0 0];
% Parametros de simulacion
t0=0;
tf=1.9;
m=100;
dt=tf;
tts=[0];
xx=xo;
tspan=[t0 tf];
pap=1;

for i=1:pap
options = odeset('RelTol',1e-6,'AbsTol',1e-6);
[t,x] = ode23('motor',tspan, xo,options);
    xo=[x(length(x),1) x(length(x),2) x(length(x),3) x(length(x),4)];
    t0=t(length(t));
    tf=t0+dt;
    tspan=[t0 tf];
    tts=[tts;t];
    xx=[xx;x];
    i=i+1;
end


figure(1)
 hold on

plot(tts,xx(:,1)*180/pi,'r','LineWidth',1.5);
xlabel('Tiempo(s)'), ylabel('°');
title('Angulo de un pendulo de un eslabon');
grid on;

  plot(tt(1:758)*165,(-yy(743:1500)*46.29+113.5),'LineWidth',1.5)

  hold off



TM=-km*(xx(:,3).*sin(16*xx(:,1))+xx(:,4).*sin(16*xx(:,1)-pi/2))-
kd*sin(4*16*xx(:,1));
figure(2);
plot(tts,TM);
xlabel('Tiempo(s)'), ylabel('°');
title('torque');
grid on;
```

```
figure(3)
figure(3);
plot(tts,xx(:,3));
xlabel('Tiempo(s)'), ylabel('°');
title('ia');
grid on;

figure(4);
plot(tts,xx(:,4));
xlabel('Tiempo(s)'), ylabel('°');
title('ib');
grid on;
```

b. Programa dinamica.m

```
function dx = motor(t,x);

global R L km kd J D b m1 lc1 I1  g Ts;

Vg=genera_voltajes(t);
va=Vg(1);
vb=Vg(2);

dx(1)= x(2);

%Tm
Tm=-km*(x(3)*sin(12*x(1)-pi/5)+x(4)*sin(12*x(1)-(pi/2+pi/5)))-
kd*sin(4*12*x(1));

% omega punto del motor mas el eslabon
dx(2)=(1/(J*I1))*(Tm-(b+D)*x(2)-(m1*g*lc1*cos(x(1)))-Ts);

%ia punto
dx(3)= 1/L*(va-x(3)*R+x(2)*km*sin(16*x(1)));
%ib punto
dx(4)=1/L*(vb-x(4)*R+x(2)*km*sin(16*x(1)-pi/2));


dx = dx';
```

Apéndice C

Análisis dinámico con dos eslabones

- Segundo eslabón como masa concentrada

A continuación se muestra el programa utilizado para simular la dinámica a dos etapas.

a. Programa principal main.m

```
global m1 lc1 I1  g b i1 i2  km kd D m2 lc2 I2;


%INICIALIZACION
% peso de un eslabon de 30 gr
% peso del segundo eslabo 15 gr
m1= 0.03; %kg
m2=0.015;
l2=3/100;
lc2=l2/2;
% longitudes en metros m
l1=6/100; %45cm
lc1=l1/2;
g=9.8;
b=0.001;
a=1/100;

% Inercia de los eslabones
I1=(1/12)*m1*(a*a+l1*l1)+(1/4)*m1*(l1*l1);
I2=(1/12)*m1*(a*a+l2*l2)+(1/4)*m2*(l1+l2)^2;

Ts=0;

%tao1=0.00882; torque necesario para mantenerlo en cero

% Corrientes de las fases del motor

i1=0;
i2=0;

%--------------------------------
%Parametros del motor
km=0.1;
kd=0.005;
D=0.002;

tts = [0];
t0 =0;
```

```
xo=[(-95)*pi/180 0];
xx=xo;
Deltat=0.45;
tf=0.45;
tspan=[t0 tf];
pap=6;
k=0;

%-------Simulacion-------
% El siguiente bucle permite que coloque el orden de alimentacion de las
% fases, cambiando el valor en estado estacionario de las corrientes del
% motor
for i=1:pap

    i1=0;
    i2=-0.5;

    options = odeset('RelTol',1e-6,'AbsTol',1e-6);
    [t,x] = ode23('Dinamica2',tspan, xo,options);
    %Actualizacion de las cond. iniciales , el tiempo inicial, el tiempo
    %final para utilizarlos en las siguientes soluciones de ode
    xo=[x(length(x),1) x(length(x),2)];
    t0=t(length(t));
    tf=t0+Deltat;
    tspan=[t0 tf];
    % Vectores que almacenan los resultados
    tts=[tts;t];
    xx=[xx;x];


    i1=0.5;
    i2=0;
    options = odeset('RelTol',1e-6,'AbsTol',1e-6);
    [t,x] = ode23('Dinamica2',tspan, xo,options);
    xo=[x(length(x),1) x(length(x),2)];
    t0=t(length(t));
    tf=t0+Deltat;
    tspan=[t0 tf];
    tts=[tts;t];
    xx=[xx;x];

    i1=0;
    i2=0.5;
    options = odeset('RelTol',1e-6,'AbsTol',1e-6);
    [t,x] = ode23('Dinamica2',tspan, xo,options);
    xo=[x(length(x),1) x(length(x),2)];
    t0=t(length(t));
    tf=t0+Deltat;
    tspan=[t0 tf];
    tts=[tts;t];
    xx=[xx;x];


    i1=-0.5;
```

```
    i2=0;
    options = odeset('RelTol',1e-6,'AbsTol',1e-6);
    [t,x] = ode23('Dinamica2',tspan, xo,options);
    xo=[x(length(x),1) x(length(x),2)];
    t0=t(length(t));
    tf=t0+Deltat;
    tspan=[t0 tf];
    tts=[tts;t];
    xx=[xx;x];

end

figure(1)

plot(tts,xx(:,1)*180/pi,'r','LineWidth',1.5);
xlabel('Tiempo(s)'), ylabel('°');
title('Angulo del primer eslabon');
grid on;



TM=-km*(i1*sin(16*xx(:,1))+i2*sin(16*xx(:,1)-pi/2))- kd*sin(4*16*xx(:,1));
figure(2);
plot(tts,TM);
xlabel('Tiempo(s)'), ylabel('°');
title('torque');
grid on;
```


b. Programa Dinamica2.m

```
function dx = Dinamica2(t,x);

global m1 lc1 I1 g b i1 i2  km kd D m2 lc2 I2;

%Para m1
% x1= q1
% x2= dq1=x1p
%   d^2q1= x2p

%Definicion de la entrada proveniente del motor y con los valores en estado
%estacionarios de la corriente del motor
u=-km*(i1*sin(14*x(1))+i2*sin(14*x(1)-pi/2))- kd*sin(4*14*x(1))-D*x(2);


%---------------------------
% dinamica del pendulo de un eslabon
dx(1)= x(2);
dx(2)=(u-b*(x(2))-m1*g*lc1*cos(x(1))-m2*g*lc2*cos(x(1)))/(I1+I2);


dx = dx';
```

- Motor en el segundo eslabón (i)

Los programas que se presentarán permiten simular ambas articulaciones moviéndose simultáneamente.

a. Programa principal main.m

```
%  clear all
%  Close all

global m1 lc1 I1  g b1 b2 i1 i2  km kd D m2 lc2 I2;


%INICIALIZACION
% peso de un eslabon de 30 gr
% peso en kg
m1= 0.03;
m2=0.015;
l2=3/100;
lc2=l2/2;
% longitudes en metros m
l1=6/100; %45cm
lc1=l1/2;
g=9.8;
b1=0.001;
b2=0.001;
% Inercia del eslabon
a=1/100;
I1=(1/12)*m1*(a*a+l1*l1)+(1/4)*m2*(l1*l1);
I2=(1/12)*m1*(a*a+l2*l2)+(1/4)*m2*(l1+l2)^2;

Ts=0;

%tao1=0.00882; torque necesario para mantenerlo en cero

% Corrientes de las fases del motor

i1=0;
i2=0;

%--------------------------------
%Parametros del motor
km=0.1;
kd=0.005;
D=0.002;

tts = [0];
t0 =0;
tts1 = [0];
```

```
t01 =0;

xo=[(0)*pi/180 0];
xo1=[(0)*pi/180 0];

xx=xo;
Deltat=0.45;
tf=0.45;
tspan=[t0 tf];

xx1=xo1;
tf1=0.45;
tspan1=[t0 tf];


pap=1;
k=0;

%-------Simulacion-------
% El siguiente bucle permite que coloque el orden de alimentacion de las
% fases, cambiando el valor en estado estacionario de las corrientes del
% motor
for i=1:pap

    i1=0;
    i2=-0.5;

    %--------------------ESLABON 1----------------------
    options = odeset('RelTol',1e-6,'AbsTol',1e-6);
    [t,x] = ode23('Dinamica1',tspan, xo,options);
    %Actualizacion de las cond. iniciales , el tiempo inicial, el tiempo
    %final para utilizarlos en las siguientes soluciones de ode
    xo=[x(length(x),1) x(length(x),2)];
    % Vectores que almacenan los resultados
    tts=[tts;t];
    xx=[xx;x];

    %--------------------ESLABON 2-------------------------
    options = odeset('RelTol',1e-6,'AbsTol',1e-6);
    [t1,x1] = ode23('Dinamica2',tspan, xo1,options);
    %Actualizacion de las cond. iniciales , el tiempo inicial, el tiempo
    %final para utilizarlos en las siguientes soluciones de ode
    xo1=[x1(length(x1),1) x1(length(x1),2)];
    % Vectores que almacenan los resultados
    tts1=[tts1;t1];
    xx1=[xx1;x1];

    % Actualizacion del tiempo
    t0=t(length(t));
    tf=t0+Deltat;
    tspan=[t0 tf];
```

```
%--------------------ESLABON 1----------------------
options = odeset('RelTol',1e-6,'AbsTol',1e-6);
[t,x] = ode23('Dinamica1',tspan, xo,options);
%Actualizacion de las cond. iniciales , el tiempo inicial, el tiempo
%final para utilizarlos en las siguientes soluciones de ode
xo=[x(length(x),1) x(length(x),2)];
% Vectores que almacenan los resultados
tts=[tts;t];
xx=[xx;x];

i1=0.5;
i2=0;
%--------------------ESLABON 2-------------------------
options = odeset('RelTol',1e-6,'AbsTol',1e-6);
[t1,x1] = ode23('Dinamica2',tspan, xo1,options);
%Actualizacion de las cond. iniciales , el tiempo inicial, el tiempo
%final para utilizarlos en las siguientes soluciones de ode
xo1=[x1(length(x1),1) x1(length(x1),2)];
% Vectores que almacenan los resultados
tts1=[tts1;t1];
xx1=[xx1;x1];

% Actualizacion del tiempo
t0=t(length(t));
tf=t0+Deltat;
tspan=[t0 tf];


  %--------------------ESLABON 1----------------------
options = odeset('RelTol',1e-6,'AbsTol',1e-6);
[t,x] = ode23('Dinamica1',tspan, xo,options);
%Actualizacion de las cond. iniciales , el tiempo inicial, el tiempo
%final para utilizarlos en las siguientes soluciones de ode
xo=[x(length(x),1) x(length(x),2)];
% Vectores que almacenan los resultados
tts=[tts;t];
xx=[xx;x];

%--------------------ESLABON 2-------------------------
options = odeset('RelTol',1e-6,'AbsTol',1e-6);
[t1,x1] = ode23('Dinamica2',tspan, xo1,options);
%Actualizacion de las cond. iniciales , el tiempo inicial, el tiempo
%final para utilizarlos en las siguientes soluciones de ode
xo1=[x1(length(x1),1) x1(length(x1),2)];
% Vectores que almacenan los resultados
tts1=[tts1;t1];
xx1=[xx1;x1];


% Actualizacion del tiempo
t0=t(length(t));
tf=t0+Deltat;
```

```
    tspan=[t0 tf];

    %--------------------ESLABON 1----------------------
    options = odeset('RelTol',1e-6,'AbsTol',1e-6);
    [t,x] = ode23('Dinamica1',tspan, xo,options);
    %Actualizacion de las cond. iniciales , el tiempo inicial, el tiempo
    %final para utilizarlos en las siguientes soluciones de ode
    xo=[x(length(x),1) x(length(x),2)];
    % Vectores que almacenan los resultados
    tts=[tts;t];
    xx=[xx;x];
    i1=0;
    i2=0.5;
    %--------------------ESLABON 2-------------------------
    options = odeset('RelTol',1e-6,'AbsTol',1e-6);
    [t1,x1] = ode23('Dinamica2',tspan, xo1,options);
    %Actualizacion de las cond. iniciales , el tiempo inicial, el tiempo
    %final para utilizarlos en las siguientes soluciones de ode
    xo1=[x1(length(x1),1) x1(length(x1),2)];
    % Vectores que almacenan los resultados
    tts1=[tts1;t1];
    xx1=[xx1;x1];


end
figure(1)
subplot(2,1,1)
plot(tts,xx(:,1)*180/pi,'r','LineWidth',1.5);
xlabel('Tiempo(s)'), ylabel('°');
title('Angulo del primer eslabon');
grid on;

subplot(2,1,2)
plot(tts1,xx1(:,1)*180/pi,'r','LineWidth',1.5);
xlabel('Tiempo(s)'), ylabel('°');
title('Angulo del segundo eslabon eslabon');
grid on;


TM=-km*(i1*sin(16*xx(:,1))+i2*sin(16*xx(:,1)-pi/2))- kd*sin(4*16*xx(:,1));
figure(3);
plot(tts,TM);
xlabel('Tiempo(s)'), ylabel('°');
title('torque');
grid on;
```

b. Programa Dinamica1.m

```
function dx = Dinamica1(t,x);

global m1 lc1 I1 g b1 b2 i1 i2  km kd D m2 lc2;
```

```
%Para m1
% x1= q1
% x2= dq1=x1p
%   d^2q1= x2p

%Definicion de la entrada proveniente del motor y con los valores en estado
%estacionarios de la corriente del motor
u=-km*(i1*sin(14*x(1))+i2*sin(14*x(1)-pi/2))- kd*sin(4*14*x(1))-D*x(2);


%---------------------------
% dinamica del pendulo de un eslabon
dx(1)= x(2);
dx(2)=(u-b1*(x(2))-m1*g*lc1*cos(x(1))-m2*g*lc2*cos(x(1)))/I1;


dx = dx';
```

c. Programa Dinamica2.m

```
function dx = Dinamica2(t,x);

global m1 lc1 I1 g b1 b2 i1 i2  km kd D m2 lc2;

%Para m1
% x1= q1
% x2= dq1=x1p
%   d^2q1= x2p

%Definicion de la entrada proveniente del motor y con los valores en estado
%estacionarios de la corriente del motor
u=-km*(i1*sin(14*x(1))+i2*sin(14*x(1)-pi/2))- kd*sin(4*14*x(1))-D*x(2);


%---------------------------
% dinamica del pendulo de un eslabon
dx(1)= x(2);
dx(2)=(u-b2*(x(2))-m1*g*lc1*cos(x(1))-m2*g*lc2*cos(x(1)))/I2;


dx = dx';
```

- Motor en el segundo eslabón (ii)

a. Calculo de las ecuaciones dinámicas

Coordenadas de los centros de masa

C1

> $a1 := x1(t) = lc1 \cdot \sin(q1(t)); b1 := y1(t) = lc1 \cdot \cos(q1(t));$

$$a1 := x1(t) = lc1 \sin(q1(t))$$

$$b1 := y1(t) = lc1 \cos(q1(t))$$

C2

> $a2 := x2(t) = lc2 \cdot \sin(q2(t)) + l1 \cdot \sin(q1(t)); b2 := y2(t) = lc2 \cdot \cos(q2(t)) + l1 \cdot \cos(q1(t));$

$$a2 := x2(t) = lc2 \sin(q2(t)) + l1 \sin(q1(t))$$

$$b2 := y2(t) = lc2 \cos(q2(t)) + l1 \cos(q1(t))$$

Velocidades al cuadrado de cada centro de masa

V1

> $x1p := diff(a1, t); y1p := diff(b1, t);$

$$x1p := \frac{d}{dt} x1(t) = lc1 \cos(q1(t)) \left(\frac{d}{dt} q1(t) \right)$$

$$y1p := \frac{d}{dt} y1(t) = -lc1 \sin(q1(t)) \left(\frac{d}{dt} q1(t) \right)$$

> $\left(lc1 \cos(q1(t)) \left(\frac{d}{dt} q1(t) \right) \right)^2 + \left(-lc1 \sin(q1(t)) \left(\frac{d}{dt} q1(t) \right) \right)^2;$
> $collect(\%, [lc1^2, diff(q1(t), t)^2]); v1 := subs(\cos(q1(t))^2 + \sin(q1(t))^2 = 1, \%);$

$$lc1^2 \cos(q1(t))^2 \left(\frac{d}{dt} q1(t) \right)^2 + lc1^2 \sin(q1(t))^2 \left(\frac{d}{dt} q1(t) \right)^2$$

$$\left(\cos(q1(t))^2 + \sin(q1(t))^2 \right) \left(\frac{d}{dt} q1(t) \right)^2 lc1^2$$

$$v1 := \left(\frac{d}{dt} q1(t) \right)^2 lc1^2$$

V2

> $x2p := diff(a2, t); y2p := diff(b2, t);$

$$x2p := \frac{d}{dt} x2(t) = lc2 \cos(q2(t)) \left(\frac{d}{dt} q2(t) \right) + l1 \cos(q1(t)) \left(\frac{d}{dt} q1(t) \right)$$

$$y2p := \frac{d}{dt} y2(t) = -lc2 \sin(q2(t)) \left(\frac{d}{dt} q2(t) \right) - l1 \sin(q1(t)) \left(\frac{d}{dt} q1(t) \right)$$

```
> (lc2*cos(q2(t))*(d/dt q2(t)) + l1*cos(q1(t))*(d/dt q1(t)))^2 + (
  -lc2*sin(q2(t))*(d/dt q2(t)) - l1*sin(q1(t))*(d/dt q1(t)))^2; v2
  := expand(%, size); collect(%, [l1^2, lc2^2, diff(q1(t), t)^2,
  diff(q2(t), t)^2, diff(q1(t), t), diff(q2(t), t)]); v
  := subs([cos(q1(t))^2 + sin(q1(t))^2 = 1, cos(q2(t))^2
  + sin(q2(t))^2 = 1], %)
```

$$\left(lc2\cos(q2(t))\left(\frac{d}{dt}q2(t)\right)+l1\cos(q1(t))\left(\frac{d}{dt}q1(t)\right)\right)^2+\left(-lc2\sin(q2(t))\left(\frac{d}{dt}q2(t)\right)-l1\sin(q1(t))\left(\frac{d}{dt}q1(t)\right)\right)^2$$

$$v2 := lc2^2\cos(q2(t))^2\left(\frac{d}{dt}q2(t)\right)^2 + 2\,lc2\cos(q2(t))\left(\frac{d}{dt}q2(t)\right)l1\cos(q1(t))\left(\frac{d}{dt}q1(t)\right) + l1^2\cos(q1(t))^2\left(\frac{d}{dt}q1(t)\right)^2 + lc2^2\sin(q2(t))^2\left(\frac{d}{dt}q2(t)\right)^2 + 2\,lc2\sin(q2(t))\left(\frac{d}{dt}q2(t)\right)l1\sin(q1(t))\left(\frac{d}{dt}q1(t)\right) + l1^2\sin(q1(t))^2\left(\frac{d}{dt}q1(t)\right)^2$$

$$\left(\cos(q1(t))^2+\sin(q1(t))^2\right)\left(\frac{d}{dt}q1(t)\right)^2 l1^2 + (2\cos(q2(t))\cos(q1(t)) + 2\sin(q2(t))\sin(q1(t)))\left(\frac{d}{dt}q2(t)\right)\left(\frac{d}{dt}q1(t)\right)lc2\,l1 + \left(\cos(q2(t))^2+\sin(q2(t))^2\right)\left(\frac{d}{dt}q2(t)\right)^2 lc2^2$$

$$v := \left(\frac{d}{dt}q1(t)\right)^2 l1^2 + (2\cos(q2(t))\cos(q1(t)) + 2\sin(q2(t))\sin(q1(t)))\left(\frac{d}{dt}q2(t)\right)\left(\frac{d}{dt}q1(t)\right)lc2\,l1 + \left(\frac{d}{dt}q2(t)\right)^2 lc2^2$$

```
> v2 := subs([2*cos(q2(t))*cos(q1(t)) + 2*sin(q2(t))*sin(q1(t)) = 2
  *cos(q1(t) - q2(t))], %);
```

$$v2 := \left(\frac{d}{dt} q1(t)\right)^2 l1^2 + 2\cos(q1(t) - q2(t))\left(\frac{d}{dt} q2(t)\right)\left(\frac{d}{dt} q1(t)\right) lc2\, l1 + \left(\frac{d}{dt} q2(t)\right)^2 lc2^2$$

Energia Cinetica : $T = \frac{1}{2}mv^2 + \frac{1}{2}I\cdot\left(\frac{dq}{dt}\right)^2$

> $EC := T = \frac{1}{2}\cdot m1\cdot v1 + \frac{1}{2}\cdot m2\cdot v2 + \frac{1}{2}I1\cdot\left(\frac{dq1}{dt}\right)^2 + \frac{1}{2}I2\cdot\left(\frac{dq1}{dt} + \frac{dq2}{dt}\right)^2;$

$$EC := T = \frac{1}{2} m1 \left(\frac{d}{dt} q1(t)\right)^2 lc1^2 + \frac{1}{2} m2 \left(\left(\frac{d}{dt} q1(t)\right)^2 l1^2 + 2\cos(q1(t) - q2(t))\left(\frac{d}{dt} q2(t)\right)\left(\frac{d}{dt} q1(t)\right) lc2\, l1 + \left(\frac{d}{dt} q2(t)\right)^2 lc2^2\right) + \frac{1}{2}\frac{I1\, dq1^2}{dt^2} + \frac{1}{2} I2 \left(\frac{dq1}{dt} + \frac{dq2}{dt}\right)^2$$

>

Energia Potencial: $U=mgh$

> $EP := U = m1\cdot g\cdot(lc1\cdot\cos(q1(t))) + m2\cdot g\cdot(l1\cdot\cos(q1(t)) + lc2\cdot\cos(q2(t)));$

$$EP := U = m1\, g\, lc1\cos(q1(t)) + m2\, g\,(lc2\cos(q2(t)) + l1\cos(q1(t)))$$

>

Lagrangiano

> $L = T - U; subs(EC, \%); Lg := subs(EP, \%);$

$$L = T - U$$

$$L = \frac{1}{2} m1 \left(\frac{d}{dt} q1(t)\right)^2 lc1^2 + \frac{1}{2} m2 \left(\left(\frac{d}{dt} q1(t)\right)^2 l1^2 + 2\cos(q1(t) - q2(t))\left(\frac{d}{dt} q2(t)\right)\left(\frac{d}{dt} q1(t)\right) lc2\, l1 + \left(\frac{d}{dt} q2(t)\right)^2 lc2^2\right) + \frac{1}{2}\frac{I1\, dq1^2}{dt^2} + \frac{1}{2} I2 \left(\frac{dq1}{dt} + \frac{dq2}{dt}\right)^2 - U$$

$$Lg := L = \frac{1}{2} m1 \left(\frac{d}{dt} q1(t) \right)^2 lc1^2 + \frac{1}{2} m2 \left(\left(\frac{d}{dt} q1(t) \right)^2 l1^2 + 2 \cos(q1(t) - q2(t)) \left(\frac{d}{dt} q2(t) \right) \left(\frac{d}{dt} q1(t) \right) lc2\, l1 + \left(\frac{d}{dt} q2(t) \right)^2 lc2^2 \right) + \frac{1}{2} \frac{I1\, dq1^2}{dt^2} + \frac{1}{2} I2 \left(\frac{dq1}{dt} + \frac{dq2}{dt} \right)^2 - m1\, g\, lc1 \cos(q1(t)) - m2\, g\, (lc2 \cos(q2(t)) + l1 \cos(q1(t)))$$

>

Finalmente L=T-U es:

> *Lg*

$$L = \frac{1}{2} m1 \left(\frac{d}{dt} q1(t) \right)^2 lc1^2 + \frac{1}{2} m2 \left(\left(\frac{d}{dt} q1(t) \right)^2 l1^2 + 2 \cos(q1(t) - q2(t)) \left(\frac{d}{dt} q2(t) \right) \left(\frac{d}{dt} q1(t) \right) lc2\, l1 + \left(\frac{d}{dt} q2(t) \right)^2 lc2^2 \right) + \frac{1}{2} \frac{I1\, dq1^2}{dt^2} + \frac{1}{2} I2 \left(\frac{dq1}{dt} + \frac{dq2}{dt} \right)^2 - m1\, g\, lc1 \cos(q1(t)) - m2\, g\, (lc2 \cos(q2(t)) + l1 \cos(q1(t)))$$

Lagrangiano 1 : *PLq1p* = *Derivada parcial de L con respecto a q1 punto.*
PLq1 = *Derivada parcial de L con respecto a q1*

> $q1(t); a := diff(\%, t);\ q2(t); b := diff(\%, t)$

$$q1(t)$$

$$a := \frac{d}{dt} q1(t)$$

$$q2(t)$$

$$b := \frac{d}{dt} q2(t)$$

> $PLq1p := m1 \cdot a \cdot lc1^2 + m2 \cdot l1^2 \cdot a + m2 \cdot \cos(q1(t) - q2(t)) \cdot b \cdot lc2 \cdot l1 + I1 \cdot a + I2 \cdot (a + b);$

$$PLq1p := m1 \left(\frac{d}{dt} q1(t) \right) lc1^2 + m2\, l1^2 \left(\frac{d}{dt} q1(t) \right) + m2 \cos(q1(t) - q2(t)) \left(\frac{d}{dt} q2(t) \right) lc2\, l1 + I1 \left(\frac{d}{dt} q1(t) \right) + I2 \left(\frac{d}{dt} q1(t) + \frac{d}{dt} q2(t) \right)$$

> $PLq1 := -m2 \cdot \sin(q1(t) - q2(t)) \cdot a \cdot b \cdot lc1 \cdot l1 + m1 \cdot g \cdot lc1 \cdot \sin(q1(t)) + m2 \cdot g \cdot l1 \cdot \sin(q1(t));$

$$PLq1 := -m2\sin(q1(t) - q2(t))\left(\frac{d}{dt}q1(t)\right)\left(\frac{d}{dt}q2(t)\right)lc1\,l1 + m1\,g\,lc1\sin(q1(t)) + m2\,g\,l1\sin(q1(t))$$

> a

$$\frac{d}{dt}q1(t)$$

> $diff(PLq1p, t) - PLq1 + bm1\cdot a = tao1;$

$$m1\left(\frac{d^2}{dt^2}q1(t)\right)lc1^2 + m2\,l1^2\left(\frac{d^2}{dt^2}q1(t)\right) - m2\sin(q1(t) - q2(t))\left(\frac{d}{dt}q1(t) - \left(\frac{d}{dt}q2(t)\right)\right)\left(\frac{d}{dt}q2(t)\right)lc2\,l1 + m2\cos(q1(t) - q2(t))\left(\frac{d^2}{dt^2}q2(t)\right)lc2\,l1 + I1\left(\frac{d^2}{dt^2}q1(t)\right) + I2\left(\frac{d^2}{dt^2}q1(t) + \frac{d^2}{dt^2}q2(t)\right) + m2\sin(q1(t) - q2(t))\left(\frac{d}{dt}q1(t)\right)\left(\frac{d}{dt}q2(t)\right)lc1\,l1 - m1\,g\,lc1\sin(q1(t)) - m2\,g\,l1\sin(q1(t)) + bm1\left(\frac{d}{dt}q1(t)\right) = tao1$$

> $ec1 := simplify(\%, size);$

$$ec1 := m2\left((-lc2 + lc1)\left(\frac{d}{dt}q1(t)\right) + lc2\left(\frac{d}{dt}q2(t)\right)\right)\left(\frac{d}{dt}q2(t)\right)l1\sin(q1(t) - q2(t)) + m2\cos(q1(t) - q2(t))\left(\frac{d^2}{dt^2}q2(t)\right)lc2\,l1 + \left(m1\,lc1^2 + m2\,l1^2 + I2 + I1\right)\left(\frac{d^2}{dt^2}q1(t)\right) + I2\left(\frac{d^2}{dt^2}q2(t)\right) + bm1\left(\frac{d}{dt}q1(t)\right) - g\sin(q1(t))\,(lc1\,m1 + m2\,l1) = tao1$$

> Lg

$$L = \frac{1}{2}m1\left(\frac{d}{dt}q1(t)\right)^2 lc1^2 + \frac{1}{2}m2\left(\left(\frac{d}{dt}q1(t)\right)^2 l1^2 + 2\cos(q1(t) - q2(t))\left(\frac{d}{dt}q2(t)\right)\left(\frac{d}{dt}q1(t)\right)lc2\,l1 + \left(\frac{d}{dt}q2(t)\right)^2 lc2^2\right) + \frac{1}{2}\frac{I1\,dq1^2}{dt^2} + \frac{1}{2}I2\left(\frac{dq1}{dt} + \frac{dq2}{dt}\right)^2 - m1\,g\,lc1\cos(q1(t)) - m2\,g\,(lc2\cos(q2(t)) + l1\cos(q1(t)))$$

>

Lagrangiano 2 : PLq2p = Derivada parcial de L con respecto a q2 punto,
PLq2 = Derivada parcial de L con respecto a q2

> $PLq2p := m2 \cdot \cos(q1(t) - q2(t)) \cdot a \cdot lc2 \cdot l1 + m2 \cdot b \cdot lc2 + I2 \cdot (b + a);$

$$PLq2p := m2 \cos(q1(t) - q2(t)) \left(\frac{d}{dt} q1(t) \right) lc2\, l1 + m2 \left(\frac{d}{dt} q2(t) \right) lc2 + I2 \left(\frac{d}{dt} q1(t) + \frac{d}{dt} q2(t) \right)$$

> $PLq2 := m2 \cdot \sin(q1(t) - q2(t)) \cdot a \cdot lc2 \cdot l1 + m2 \cdot g \cdot lc2 \cdot \sin(q2(t));$

$$PLq2 := m2 \sin(q1(t) - q2(t)) \left(\frac{d}{dt} q1(t) \right) lc2\, l1 + m2\, g\, lc2 \sin(q2(t))$$

> $diff(PLq2p, t) - PLq2 + bm2 \cdot b = tao2;$

$$-m2 \sin(q1(t) - q2(t)) \left(\frac{d}{dt} q1(t) - \left(\frac{d}{dt} q2(t) \right) \right) \left(\frac{d}{dt} q1(t) \right) lc2\, l1 + m2 \cos(q1(t) - q2(t)) \left(\frac{d^2}{dt^2} q1(t) \right) lc2\, l1 + m2 \left(\frac{d^2}{dt^2} q2(t) \right) lc2 + I2 \left(\frac{d^2}{dt^2} q1(t) + \frac{d^2}{dt^2} q2(t) \right) - m2 \sin(q1(t) - q2(t)) \left(\frac{d}{dt} q1(t) \right) lc2\, l1 - m2\, g\, lc2 \sin(q2(t)) + bm2 \left(\frac{d}{dt} q2(t) \right) = tao2$$

> $ec2 := simplify(\%, size);$

$$ec2 := -m2 \left(\frac{d}{dt} q1(t) \right) lc2\, l1 \left(\frac{d}{dt} q1(t) - \left(\frac{d}{dt} q2(t) \right) + 1 \right) \sin(q1(t) - q2(t)) + m2 \cos(q1(t) - q2(t)) \left(\frac{d^2}{dt^2} q1(t) \right) lc2\, l1 + I2 \left(\frac{d^2}{dt^2} q1(t) \right) + (I2 + lc2\, m2) \left(\frac{d^2}{dt^2} q2(t) \right) - m2\, g\, lc2 \sin(q2(t)) + bm2 \left(\frac{d}{dt} q2(t) \right) = tao2$$

>

> $ec1; ec2;$

$$
\begin{aligned}
& m2\left(\left(-lc2+lc1\right)\left(\frac{\mathrm{d}}{\mathrm{d}t}\,q1(t)\right)\right. \\
& \quad \left.+lc2\left(\frac{\mathrm{d}}{\mathrm{d}t}\,q2(t)\right)\right)\left(\frac{\mathrm{d}}{\mathrm{d}t}\,q2(t)\right) l1\sin(q1(t)-q2(t)) \\
& \quad +m2\cos(q1(t)-q2(t))\left(\frac{\mathrm{d}^2}{\mathrm{d}t^2}\,q2(t)\right) lc2\,l1+\left(m1\,lc1^2\right. \\
& \quad \left.+m2\,l1^2+I2+I1\right)\left(\frac{\mathrm{d}^2}{\mathrm{d}t^2}\,q1(t)\right)+I2\left(\frac{\mathrm{d}^2}{\mathrm{d}t^2}\,q2(t)\right) \\
& \quad +bm1\left(\frac{\mathrm{d}}{\mathrm{d}t}\,q1(t)\right)-g\sin(q1(t))\left(lc1\,m1+m2\,l1\right)=tao1
\end{aligned}
$$

$$
\begin{aligned}
& -m2\left(\frac{\mathrm{d}}{\mathrm{d}t}\,q1(t)\right) lc2\,l1\left(\frac{\mathrm{d}}{\mathrm{d}t}\,q1(t)-\left(\frac{\mathrm{d}}{\mathrm{d}t}\,q2(t)\right)+1\right)\sin(q1(t) \\
& \quad -q2(t))+m2\cos(q1(t)-q2(t))\left(\frac{\mathrm{d}^2}{\mathrm{d}t^2}\,q1(t)\right) lc2\,l1 \\
& \quad +I2\left(\frac{\mathrm{d}^2}{\mathrm{d}t^2}\,q1(t)\right)+(I2+lc2\,m2)\left(\frac{\mathrm{d}^2}{\mathrm{d}t^2}\,q2(t)\right) \\
& \quad -m2\,g\,lc2\sin(q2(t))+bm2\left(\frac{\mathrm{d}}{\mathrm{d}t}\,q2(t)\right)=tao2
\end{aligned}
$$

b. Programa principal main.m

```
% clear all
% Close all

global m1 m2 lc1 lc2 l1 l2 I2 I1  g bm1 bm2 i1 i2 i3 i4 km km1 kd kd1 D tao1
tao2;


%INICIALIZACION
% peso de un eslabon de 30 gr
% peso en kg
m1= 0.03;
m2=0.02;
% longitudes en metros m
l1=6/100; %45cm
lc1=l1/2

l2=3/100;
lc2=l2/2

g=9.8;
bm1=0.1;
bm2=0.1;
```

```
% Inercia del eslabon
a=1/100;
I1=(1/12)*m1*(a*a+l1*l1)+(1/4)*m2*(l1*l1);

I2=(1/12)*m1*(a*a+l2*l2)+(1/4)*m2*(l1+l2)^2;


tao1=0;
tao2=0;
% Corrientes de las fases del motor

i1=0;
i2=0;

%--------------------------------
km=0.1;
km1=0.1;
kd=0.005;
kd1=0.005;
D=0.002;

tts = [0];
t0 =0;
xo=[(0)*pi/180 0 (0)*pi/180 0];
%xo=[0 0];
xx=xo;
Deltat=1;
tf=1;
tspan=[t0 tf];
pap=1;
k=0;
%-------Simulacion-------2
for i=1:pap

    i1=0;
    i2=-0.5;

    i3=0;
    i4=-0.5;

    options = odeset('RelTol',1e-6,'AbsTol',1e-6);
    [t,x] = ode23s('Dinamica3',tspan, xo,options);
    xo=[x(length(x),1) x(length(x),2) x(length(x),3) x(length(x),4)];
    t0=t(length(t));
    tf=t0+Deltat;
    tspan=[t0 tf];
    tts=[tts;t];
    xx=[xx;x];

    i1=0;
    i2=-0.5;
```

```
    i3=0.5;
    i4=0;
   options = odeset('RelTol',1e-6,'AbsTol',1e-6);
    [t,x] = ode23s('Dinamica3',tspan, xo,options);
    xo=[x(length(x),1) x(length(x),2) x(length(x),3) x(length(x),4)];
    t0=t(length(t));
    tf=t0+Deltat;
    tspan=[t0 tf];

    tts=[tts;t];
    xx=[xx;x];


     i1=0;
    i2=0.5;

end

figure(1)

subplot(2,1,1)
plot(tts,xx(:,1)*180/pi,'r');
xlabel('Tiempo(s)'), ylabel('°');
title('Angulo del primer eslabon');
grid on;


subplot(2,1,2)
 plot(tts,xx(:,3)*180/pi,'r');
xlabel('Tiempo(s)'), ylabel('°');
title('Angulo del segundo eslabon');
grid on;
```

c. Programa Dinamica3.m

```
function dx = Dinamica(t,x);

global m1 m2 lc1 lc2 l1 l2 I2 I1  g bm1 bm2 i1 i2 i3 i4 km km1 kd kd1 D tao1
tao2;

%Para m1
% x1= q1
% x2= dq1=x1p
%   d^2q1= x2p

% x3= q2
% x4= dq2=x3p
%   d^2q2= x4p
```

```
%Definicion de la entrada

tao1=-km*(i1*sin(14*x(1))+i2*sin(14*x(1)-pi/2))- kd*sin(4*14*x(1))-D*x(2);
tao2=-km1*(i3*sin(14*x(3))+i4*sin(14*x(3)-pi/2))- kd1*sin(4*14*x(3))-D*x(4);


%--------------------------
dx(1)= x(2);

dx(2)=(l1*(m2*l1*lc2^2*cos(x(1)-x(3))+lc2*I2)*m2*sin(x(1)-x(3))*(x(2))^2+(-
l1*(lc2*(-lc2+lc1)*m2+I2*(lc1-2*lc2)-m2*l1*lc2^2*cos(x(1)-x(3)))*m2*sin(x(1)-
x(3))*(x(4))-bm1*(I2+lc2*m2)-l1*(m2*l1*lc2^2*cos(x(1)-
x(3))+lc2*I2)*m2*sin(x(1)-x(3)))*x(2)-l1*(x(4))^2*lc2*(I2+lc2*m2)*m2*sin(x(1)-
x(3))+(m2*lc2*l1*bm2*cos(x(1)-x(3))+bm2*I2)*(x(4)) )/(-m2^2*cos(x(1)-
x(3))^2*lc2^2*l1^2-2*m2*cos(x(1)-
x(3))*lc2*l1*I2+m2^2*l1^2*lc2+((m1*lc1^2+I2+I1)*lc2+l1^2*I2)*m2+I2*(m1*lc1^2+I
1))+(lc2*m2*tao1+g*(lc1*m1+m2*l1)*(I2+lc2*m2)*sin(x(1))-
m2*lc2*l1*(m2*g*lc2*sin(x(3))+tao2)*cos(x(1)-x(3))+I2*(tao1-tao2)-
m2*g*lc2*sin(x(3))*I2)/(-m2^2*cos(x(1)-x(3))^2*lc2^2*l1^2-2*m2*cos(x(1)-
x(3))*lc2*l1*I2+m2^2*l1^2*lc2+((m1*lc1^2+I2+I1)*lc2+l1^2*I2)*m2+I2*(m1*lc1^2+I
1));

dx(3)= x(4);

dx(4)=(lc2*(m1*lc1^2+m2*l1^2+I2+I1)*l1*m2*sin(x(1)-
x(3))*(x(2))^2+((l1*lc2*m2*(-lc2+lc1)*cos(x(1)-x(3))-l1^2*m2*lc2+(-m1*lc1^2-
I1-2*I2)*lc2+lc1*I2)*l1*m2*sin(x(1)-
x(3))*(x(4))+lc2*(m1*lc1^2+m2*l1^2+I2+I1)*l1*m2*sin(x(1)-
x(3))+l1*bm1*lc2*m2*cos(x(1)-x(3))+I2*bm1)*x(2)+(m2*l1*lc2^2*cos(x(1)-
x(3))+lc2*I2)*l1*m2*sin(x(1)-x(3))*(x(4))^2-
bm2*(m1*lc1^2+m2*l1^2+I2+I1)*(x(4)))/(-m2^2*cos(x(1)-x(3))^2*lc2^2*l1^2-
2*m2*cos(x(1)-
x(3))*lc2*l1*I2+m2^2*l1^2*lc2+((m1*lc1^2+I2+I1)*lc2+l1^2*I2)*m2+I2*(m1*lc1^2+I
1))+(g*lc2*m2*(m1*lc1^2+m2*l1^2+I2+I1)*sin(x(3))+tao2*m2*l1^2-
I2*g*(lc1*m1+m2*l1)*sin(x(1))-
l1*(g*sin(x(1))*(lc1*m1+m2*l1)+tao1)*lc2*m2*cos(x(1)-x(3))+(tao2-
tao1)*I2+tao2*(m1*lc1^2+I1))/(-m2^2*cos(x(1)-x(3))^2*lc2^2*l1^2-2*m2*cos(x(1)-
x(3))*lc2*l1*I2+m2^2*l1^2*lc2+((m1*lc1^2+I2+I1)*lc2+l1^2*I2)*m2+I2*(m1*lc1^2+I
1));
dx = dx';
```

Apéndice D

Implementación de la Interfaz Gráfica

- Form1 : Ventana de Área de Trabajo

```
Imports System.IO
Imports System.IO.Ports
Imports System.Threading

Imports System.Data.OleDb
Imports System.Data

Imports System
Imports System.ComponentModel
Imports System.Windows.Forms

Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

    End Sub

    Private Sub InicioToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles InicioToolStripMenuItem.Click
        Dim form As New Form2
        form.MdiParent = Me
        GraficasToolStripMenuItem.Enabled = True
        MovimientoTiempoRealToolStripMenuItem.Enabled = True
        form.Show()

    End Sub

    Private Sub SalirToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles SalirToolStripMenuItem.Click
        Application.Exit()
    End Sub

    Private Sub GraficasToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles GraficasToolStripMenuItem.Click
        Dim form1 As New Form3
        form1.MdiParent = Me
        form1.Show()
    End Sub

    Private Sub MovimientoTiempoRealToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MovimientoTiempoRealToolStripMenuItem.Click
        Dim form4 As New Form4
        form4.MdiParent = Me
        form4.Show()
```

```
    End Sub


    Private Sub ConexionesToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles ConexionesToolStripMenuItem.Click
        Dim form5 As New Form5
        form5.MdiParent = Me
        form5.Show()
    End Sub

    Private Sub InformacionToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles InformacionToolStripMenuItem.Click
        Dim form6 As New Form6
        form6.MdiParent = Me
        form6.Show()
    End Sub
End Class
```

- Form2 : Ventana Principal

```
Imports System.IO
Imports System.IO.Ports
Imports System.Threading

Imports System.Data.OleDb
Imports System.Data

Imports System
Imports System.ComponentModel
Imports System.Windows.Forms
Module Module1
    'Esta variable es global. Podremos utlizarlo desde cualquier
    'formulario de nuestro proyecto.
    Public ang_actual As Double ' Angulo Actual
    Public I As Long ' Contador

End Module
Public Class Form2

    Inherits Form

    Dim cambio12 As Boolean = True
    Dim cambio13 As Boolean = True
    Dim led1 As Integer = 12
    Dim count_t As Integer = 1
    Dim led2 As Boolean = True
    Dim angS As Double = 0.0
    Dim angI As Double = 0.0
    Private trd As Thread
    Dim newval As Integer
    Dim tiempo_sos As Integer
    Dim TiempoTerapia As Integer = 100
    Delegate Sub SetTextCallback(ByVal [text] As String)
    Private Thread As Thread = Nothing
```

```
    Dim Tiempo As String  'Tiempo total transcurrido.
    Dim aux As Integer

    Private WithEvents setTextSafeBtn As Button
    Private WithEvents setTextBackgroundWorkerBtn As Button
    Private WithEvents backgroundWorker1 As BackgroundWorker

    Private Sub Form2_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        SerialPort1.Close()
        SerialPort1.PortName = "com13" 'CAMBIA EL PUERTO COM
        SerialPort1.BaudRate = 9600
        SerialPort1.DataBits = 8
        SerialPort1.Parity = Parity.None
        SerialPort1.StopBits = StopBits.One
        SerialPort1.Handshake = Handshake.None
        SerialPort1.Encoding = System.Text.Encoding.Default
        SerialPort1.Open()

        Me.Thread = New Thread( _
        New ThreadStart(AddressOf Me.ThreadProcSafe))
        Me.Thread.Start()


    End Sub

    Private Sub ThreadProcSafe()

        Do While True
            ang_actual = CDbl(SerialPort1.ReadLine)
            SetText(SerialPort1.ReadLine)

            If I >= TiempoTerapia Then
                'Deshabilita el reloj
                Timer1.Enabled = False
                SerialPort1.Write("f")
                cambio12 = Not (cambio12)
                SetText1("Inicio")
                SetText2("n")
            End If

        Loop

    End Sub

    Private Sub SetText(ByVal [text] As String)

        If Me.TextBox1.InvokeRequired Then
            Dim d As New SetTextCallback(AddressOf SetText)
            Me.Invoke(d, New Object() {[text]})
        Else
            Me.TextBox1.Text = [text]
        End If

    End Sub

    Private Sub SetText1(ByVal [text] As String)
```

```
        If Me.Button3.InvokeRequired Then
            Dim d As New SetTextCallback(AddressOf SetText1)
            Me.Invoke(d, New Object() {[text]})
        Else
            Me.Button3.Text = [text]
        End If

    End Sub

    Private Sub SetText2(ByVal [text] As String)

        If Me.PictureBox1.InvokeRequired Then
            Dim d As New SetTextCallback(AddressOf SetText2)
            Me.Invoke(d, New Object() {[text]})
        Else
            PictureBox1.Image = My.Resources._stop
        End If

    End Sub



    'Boton de inicio y parada de la aplicacion

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button3.Click

        ' Comprueba que los valores pedidos sean correctos
        If CDbl(TextBox2.Text) * 1000 > 550 And CDbl(TextBox6.Text) > 0 And
CDbl(TextBox3.Text) <= 60 And CDbl(TextBox4.Text) <= 80 Then
            TiempoTerapia = Val(TextBox6.Text) * 60 * 10
            ' Asigna el valor del angulo superior deseado
            angS = CDbl(TextBox3.Text)
            ' Asigna el valor del angulo inferior deseado
            angI = CDbl(TextBox4.Text)

            If cambio12 = True Then
                'Habilita el reloj
                Timer1.Enabled = True
                Button3.Text = "Stop"
                'Cambia el color del ovalo a verde'
                OvalShape1.BackColor = Color.Green
                I = 0 'Inicializar el contador del tiempo.
                TextBox5.Text = " "
                Timer1.Interval = 1    'Iniciar el cronometro en un segundo

            Else
                'Deshabilita el reloj
                Timer1.Enabled = False
                SerialPort1.Write("f")
                Button3.Text = "Inicio"
                'Cambia el color del ovalo a rojo
                OvalShape1.BackColor = Color.Red
                'Cambia la imagen
                PictureBox1.Image = My.Resources._stop
```

```
            End If

            cambio12 = Not (cambio12)

        Else
            MsgBox("Por favor complete la informacion pedida")
        End If

    End Sub

    'Boton de salida de la aplicacion

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
        SerialPort1.Write("f")
        SerialPort1.Close()
        Me.Thread.Abort()
        Application.Exit()

    End Sub

    'Asigna el valor de la barra a el texbox del tiempo de sostenimiento
    Private Sub TextBox2_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TextBox2.TextChanged
        tiempo_sos = Val(TextBox2.Text) * 10
    End Sub

    'Asigna el valor de la barra a el texbox del angulo superior
    Private Sub TrackBar1_Scroll(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TrackBar1.Scroll
        TextBox3.Text = TrackBar1.Value
    End Sub

    'Asigna el valor de la barra a el texbox del angulo inferior
    Private Sub TrackBar2_Scroll(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TrackBar2.Scroll
        TextBox4.Text = -TrackBar2.Value
    End Sub


    'Asigna el valor de la barra a el texbox del tiempo de sostenimiento
    Private Sub TrackBar3_Scroll(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TrackBar3.Scroll
        TextBox2.Text = TrackBar3.Value
    End Sub


    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Timer1.Tick
        'Cuenta los segundos transcurridos una vez habilitado el reloj
        I = I + 1

        ' Asigna a la variable tiempo el valor del tiempo en el formato
horas/minutos/segundos
        Tiempo = Format(Int(I / 36000) Mod 24, "00") & ":" & _
                 Format(Int(I / 600) Mod 60, "00") & ":" & _
                 Format(Int(I / 10) Mod 60, "00")
```

```
        'Actualiza el tiempo actual de la terapia
        TextBox5.Text = Tiempo
        ' Ya cada paso se realiza segun el tiempo de sostenimiento deseado
        ' Se mueve el motor una vez cumplido dicho tiempo de sostenimiento
        ' el count_t sirve para actializar el ciclo en el cual debe ser movido

        If I = count_t * tiempo_sos Then

            If cambio13 And ang_actual <= angS Then
                SerialPort1.Write("i")
            Else
                cambio13 = False
            End If

            If Not (cambio13) And ang_actual >= angI Then
                SerialPort1.Write("d")
            Else
                cambio13 = True
            End If
            count_t = count_t + 1

        End If

    End Sub

End Class
```


- Form3 : Ventana de Gráfica t vs tita

```
Imports System.IO
Imports System.IO.Ports
Imports System.Threading

Imports System.Data.OleDb
Imports System.Data
Imports System.Windows.Forms.DataVisualization.Charting

Imports System
Imports System.ComponentModel
Imports System.Windows.Forms


Public Class Form3

    Private grafica As Thread
    Dim x As Double
    Delegate Sub SetTextCallback(ByVal x As Double, ByVal y As Double)
    Delegate Sub SetTextCallback1(ByVal [text] As String)
    Private Thread As Thread = Nothing
```

```
    Private Sub Form3_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

        Me.grafica = New Thread(AddressOf ThreadTask)
        Me.grafica.IsBackground = True
        Me.grafica.Start()

    End Sub

    Private Sub ThreadTask()

        Do While True

            SetData(I / 10, ang_actual / 10000)
            Thread.Sleep(1000)
        Loop

    End Sub

    Private Sub SetData(ByVal x As Double, ByVal y As Double)
        If Me.Chart1.InvokeRequired Then
            Dim d As New SetTextCallback(AddressOf SetData)
            Me.Invoke(d, New Object() {x, y})
        Else
            Me.Chart1.Series("Angulo").Points.AddXY(x, y)
        End If

    End Sub


    Private Sub SetText()
        Throw New NotImplementedException
    End Sub



End Class
```

- Form4 : Ventana de Mímico

```
Imports System.Math
Imports System.Threading

Public Class Form4
    Inherits System.Windows.Forms.Form

    Dim gMyGraphics As Graphics ' the drawing surface
    Dim cMyColor As Color ' used to hold colors for the pen and brush
    Dim pMyPen As Pen ' used to draw the lines
    Dim pMyPen2 As Pen
    Dim bMyBrush As SolidBrush = New SolidBrush(Color.White) ' used to draw the background
    Dim intPosition As Integer ' position to draw the animated lines
    Dim intDirection As Integer ' controls direction of the animated lines
```

```
    Dim aux As Single
    Dim ang As Single
    Dim x As Single
    Dim y As Single
    Delegate Sub SetTextCallback(ByVal [text] As String)
    Private mimico As Thread
    Delegate Sub SetTextCallback1(ByVal [text] As String)
    Delegate Sub SetTextCallback2(ByVal [text] As String)
    Private Thread As Thread = Nothing


    Private Sub Form4_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        Me.mimico = New Thread(AddressOf ThreadTask)
        Me.mimico.IsBackground = True
        Me.mimico.Start()

    End Sub

    Private Sub ThreadTask()

        Do While True
            pMyPen = New Pen(Brushes.Black, 1)


            gMyGraphics = PictureBox1.CreateGraphics()

            gMyGraphics.FillRectangle(bMyBrush, 0, 0, PictureBox1.Width,
PictureBox1.Height)

            Dim I As Integer

            For I = 0 To 400
                gMyGraphics.DrawLine(pMyPen, I, 0, I, PictureBox1.Height)
                gMyGraphics.DrawLine(pMyPen, PictureBox1.Width, I, 0, I)
                I = I + 20
            Next I


            pMyPen2 = New Pen(Brushes.Red, 5)
            ang = ((ang_actual / 10000) * 3.1416) / 180

            If ang > 0 Then
                x = Abs(Math.Cos(ang)) * 400
                y = 200 - Abs(Math.Sin(ang) * 400)
            Else
                x = Abs(Math.Cos(ang)) * 400
                y = 200 + Abs(Math.Sin(ang)) * 400
            End If

            aux = PictureBox1.Height / 2
            gMyGraphics.DrawLine(pMyPen2, 0, aux, x, y)

            Thread.Sleep(500)
        Loop

    End Sub
```

```
End Class
```

- Form5 : Ventana de Conexiones

```
Public Class Form5

    Private Sub Form5_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

    End Sub

    Private Sub RadioButtons_CheckedChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles RadioButton1.CheckedChanged, RadioButton2.CheckedChanged
        If RadioButton1.Checked Then
            PictureBox1.Image = My.Resources.computadoraUsb
        ElseIf RadioButton2.Checked Then
            PictureBox1.Image = My.Resources.la_llave
        Else
        End If
    End Sub
End Class
```

- Programa para el Arduino Uno


```
int motorPins[] = {8, 9, 10, 11};          // PIN-es del motor
 int count = 0;                            // Contador 1
 long delayTime = 5000;      // Delay que determina la velocidad de giro
 float tita=0.0;
 int pin=13;
 int val=0;
 int i =0;
 boolean b= true;
// Declaraciones para el comp
char inData[20]; // Allocate some space for the string
char inChar=-1; // Where to store the character read
byte index = 0; // Index into array; where to store the character
char dato = 'f';
int tiempo;
char cadena[24];
byte contador=0;

void setup() {
  Serial.begin(9600);
  pinMode(13, OUTPUT);

  for (count = 0; count < 4; count++) {          // Configuración de los PIN-es como salida digital
    pinMode(motorPins[count], OUTPUT);
  }
}
```

```
void moveForward(int count) {

  switch (count) {
    case 0:
      digitalWrite(motorPins[3], LOW);
      digitalWrite(motorPins[2], LOW);
      digitalWrite(motorPins[0], HIGH);
      digitalWrite(motorPins[1], HIGH);
      break;
    case 1:
       digitalWrite(motorPins[0], LOW);
       digitalWrite(motorPins[3], LOW);
       digitalWrite(motorPins[1], HIGH);
       digitalWrite(motorPins[2], HIGH);
      break;
    case 2:
         digitalWrite(motorPins[1], LOW);
         digitalWrite(motorPins[0], LOW);
         digitalWrite(motorPins[2], HIGH);
         digitalWrite(motorPins[3], HIGH);
      break;
    case 3:
       digitalWrite(motorPins[2], LOW);
       digitalWrite(motorPins[1], LOW);
       digitalWrite(motorPins[3], HIGH);
       digitalWrite(motorPins[0], HIGH);

     break;
    default:
      count =0 ;
    }

}
void moveBackward(int count) {

  switch (count) {
    case 0:
     digitalWrite(motorPins[2], LOW);
     digitalWrite(motorPins[1], LOW);
     digitalWrite(motorPins[3], HIGH);
     digitalWrite(motorPins[0], HIGH);
      break;
    case 1:
     digitalWrite(motorPins[1], LOW);
     digitalWrite(motorPins[0], LOW);
     digitalWrite(motorPins[2], HIGH);
     digitalWrite(motorPins[3], HIGH);

      break;
```

```
   case 2:
    digitalWrite(motorPins[0], LOW);
    digitalWrite(motorPins[3], LOW);
    digitalWrite(motorPins[1], HIGH);
    digitalWrite(motorPins[2], HIGH);
      break;

   case 3:
     digitalWrite(motorPins[3], LOW);
     digitalWrite(motorPins[2], LOW);
     digitalWrite(motorPins[0], HIGH);
     digitalWrite(motorPins[1], HIGH);
        break;

   default:
     count =0 ;
   }

}

void loop() {

     if(count>3){
     count = 0;
     }

  if(Serial.available()>0){
    dato = Serial.read();
  }else{
    dato = 'n';
    }

   if(dato=='d'){
     moveForward(count);
     count++;
   }else if (dato=='i'){
        moveBackward(count);
        count++;
      }else if (dato=='f'){
         digitalWrite(motorPins[1], LOW);
         digitalWrite(motorPins[0], LOW);
         digitalWrite(motorPins[2], LOW);
         digitalWrite(motorPins[3], LOW);
        }else {}
  tita =-45.23*(analogRead(A0)/204.8)+116.6;
   Serial.println(tita,4);
   delay(100);
}
```