IMPLEMENTACIÓN DE UN SISTEMA DE ACCIONAMIENTO EN UN BRAZO ROBÓTICO BASADO EN CINEMÁTICA INVERSA DE CUATRO GRADOS DE LIBERTAD OPERADO CON INTERFAZ MANUAL

LUIS HERNÁN GONZÁLEZ CORREA
JORGE ANDRÉS PADILLA PÁJARO

UNIVERSIDAD PONTIFICIA BOLIVARIANA

ESCUELA DE INGENIERÍAS Y ARQUITECTURA

INGENIERÍA ELECTRÓNICA

MONTERÍA

2023

IMPLEMENTACIÓN DE UN SISTEMA DE ACCIONAMIENTO EN UN BRAZO ROBÓTICO BASADO EN CINEMÁTICA INVERSA DE CUATRO GRADOS DE LIBERTAD OPERADO CON INTERFAZ MANUAL

LUIS HERNÁN GONZÁLEZ CORREA
JORGE ANDRÉS PADILLA PÁJARO

Trabajo de grado para optar al título de Ingeniero Electrónico

Asesor

ANA MILENA LÓPEZ LÓPEZ

Ingeniera Electrónica

UNIVERSIDAD PONTIFICIA BOLIVARIANA

ESCUELA DE INGENIERÍAS Y ARQUITECTURA

INGENIERÍA ELECTRÓNICA

MONTERÍA

2023

DEDICATORIAS

A Dios, a mis padres Hernán Alberto González Ávila y Lidis Correa Berrocal, Jaime Eugenio Padilla Mesperuza y Luzneira Rosa Pájaro Hernández y toda nuestra familia que siempre confiaron en nosotros y nos apoyaron en todo este proceso, también a nuestra directora Ana Milena López por su orientación y acompañarnos durante la ejecución del proyecto.

AGRADECIMIENTOS

Deseamos expresar nuestro sincero agradecimiento a Dios, a nuestra familia, Docentes, compañeros y demás personas que estuvieron presentes durante esta etapa tan importante de nuestras vidas, brindándonos apoyo e intercambiando conocimientos.

ÍNDICE GENERAL

ÍNDICE DE ECUACIONES

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN:

El trabajo de grado tuvo como objetivo principal desarrollar un sistema de control para un prototipo robótico en el laboratorio de Electrónica de la Universidad Pontificia Bolivariana Seccional Montería para fortalecer la enseñanza de la robótica en el Programa. Se diseñó e implementó un mecanismo de control, tomando como parámetro de entrada valores angulares en grados en las junturas del robot y así establecer nuevos parámetros que permitan obtener nuevas posiciones para el prototipo. Además de ello se desarrolló un código en Arduino, que permite el accionamiento del robot mediante instrucciones basados en los resultados obtenidos del procedimiento matemático realizado. Además, se diseñó una interfaz de usuario en MATLAB para facilitar la interacción en tiempo real con el dispositivo, en la cual se encuentran elementos como cuadros de texto y botones de inicio y origen mediante los cuales el usuario pueda controlar el robot. La metodología empleada incluyó el análisis de requerimientos, el diseño de software, la implementación y la validación del sistema.

El resultado fue un brazo robótico con un sistema de control que permite posicionar el prototipo en diversos puntos mediante valores angulares en grados para sus articulaciones, que puede ser integrado al currículo académico. Este trabajo también puede contribuir al fortalecimiento del área de control en el ámbito académico y fomentó la innovación en robótica educativa. Se espera que esta herramienta educativa sea valiosa para la universidad y sus estudiantes, promoviendo el aprendizaje y la experimentación en robótica y sistemas de control.

ABSTRACT:

The main objective of the thesis was to improve a robotic prototype in the Electronics Laboratory of the Pontifical Bolivarian University, Monteria Campus, in order to strengthen the teaching of robotics in the faculty. A control system was designed and implemented, taking angular values in degrees at the robot's joints as input parameters, to establish new parameters that allow obtaining new positions for the prototype. In addition to this, an Arduino code was developed to enable the robot to be operated based on the results obtained from the mathematical procedure. Furthermore, a user interface was designed in MATLAB to facilitate real-time interaction with the device, which includes elements such as text boxes and start and home buttons through which the user can control the robot. The methodology employed included requirement analysis, software design, implementation, and system validation.

The result was a robotic arm with a control system that allows positioning the prototype at various points using angular values in degrees for its joints, which can be integrated into the academic curriculum. This work can also contribute to the strengthening of the control area in the academic field and foster innovation in educational robotics. It is expected that this educational tool will be valuable for the university and its students, promoting learning and experimentation in robotics and control systems.

INTRODUCCIÓN

El avance continuo de la tecnología en diversos ámbitos, como la industria, la educación y la investigación, ha llevado al desarrollo de elementos automatizados que desempeñan funciones importantes. Estos avances incluyen actuadores industriales, computadoras electrónicas, transmisión de energía mediante mecanismos y engranajes, así como sensores y transductores. A partir de 1950, se empezaron a crear los primeros robots debido a estas investigaciones en inteligencia artificial, que permitieron poner a prueba teorías que anteriormente solo se podían comprobar en humanos, ahora aplicadas a computadoras y dispositivos electrónicos.

Las industrias hoy día se centran en la automatización de una gran parte de sus operaciones, lo que les permite minimizar los posibles peligros para la seguridad del personal y también reducir los costos asociados con la contratación de una gran cantidad de trabajadores para realizar tareas industriales (Medrano Trujillo, 2020). Para lograr esto, utilizan dispositivos automatizados como cintas transportadoras, grúas industriales, robots manipuladores y otros, que se encargan de realizar actividades peligrosas. En esta situación, los robots manipuladores desempeñan diversas tareas, como el transporte de objetos, seguimiento de tareas, trazos y vigilancia, entre otras. Un manipulador robótico, conocido también como brazo robótico antropomórfico debido a su similitud con un brazo humano, se basa en el principio de articular segmentos de su estructura para lograr la flexibilidad necesaria y así poder mover elementos dentro de un espacio geométrico determinado (Millán Castrillón, J. K, 2019).

La robótica tiene una gran variedad de aplicaciones en la actualidad, por ende, se brinda como objeto de estudio por parte de centros de investigación y universidades, desarrollando proyectos robóticos para fomentar un aprendizaje práctico sobre el control aplicado a esta ciencia. Para esto es necesaria la disposición de una variedad de equipos electrónicos con los que se puedan poner en práctica las diferentes técnicas que comprende la robótica y validar los modelos matemáticos que están establecidos. El laboratorio de Ingenierías y Arquitectura de la Universidad Pontificia Bolivariana actualmente cuenta con equipos con los cuales es posible la práctica de teorías propuestas en algunos cursos del plan educativo. En lo referente al programa de Ingeniería Electrónica se cuenta con equipos para llevar a cabo prácticas del área de automatización y control, tales como tableros con PLC integrados, software de simulación Matlab, suministros de energía para alimentar los dispositivos utilizados y un prototipo robótico propuesto por (Heller, 2017), el cual funcionó correctamente en ese momento, cumpliendo con los parámetros establecidos. Debido al desuso del equipo por alrededor de 2 años en el contexto de la pandemia y la falta de documentación del brazo, no se encontraba funcional. Por ello se propuso implementar un nuevo sistema de control de este, partiendo de la estructura del prototipo, con el propósito de accionar 3 grados de libertad a través de una interfaz de usuario.

En el presente trabajo de grado se realizó el proceso de reestructuración del sistema de control del brazo robótico, siguiendo una metodología necesaria para el desarrollo del control, desde métodos matemáticos hasta el diseño de un software y una interfaz de usuario donde se asignaron parámetros que permiten al usuario ingresar unos ángulos que definirán el posicionamiento final que tendrá el robot. Además de ello este dispositivo se brinda como herramienta para la enseñanza teórica y práctica de la robótica dentro del plan académico de la facultad de Ingeniería Electrónica, aportando nuevas técnicas de control que pueden ser aplicadas en los distintos proyectos que se realicen a futuro.

1. Marco teórico/estado del arte

La robótica como tecnología da forma al conocimiento y el comportamiento de los robots, lo que significa usar el conocimiento de diferentes campos para diseñar, construir, ensamblar y operar robots para propósitos específicos. Asimismo, la robótica se estructura en la sinergia de ejes de contenidos considerados en la educación técnica, como electricidad y electrónica, mecánica, energía, sensores y tecnologías de la información. Desde sus inicios en la década de 1960, la robótica ha crecido enormemente y ha tenido un impacto en diversas industrias, como la fabricación, la medicina, la logística, la exploración espacial y más.
Cualquier sistema robótico debe tener ciertas mejoras para funcionar sin problemas ante las incertidumbres y perturbaciones que pueden ocurrir durante la operación (Conejo Benítez, 2021).

1.1. Dispositivos

Entre los dispositivos se encuentran los elementos que conforman la estructura física del brazo robot, los cuales son:

1.1.1. Arduino

Arduino es una placa de desarrollo de hardware gratuita que pueden utilizar tanto los aficionados como los fabricantes para diseñar y construir dispositivos que interactúen con el mundo real utilizando la amplia variedad de sensores y otros componentes electrónicos disponibles en el mercado. (Peña, 2020).

En cuanto al hardware, Arduino tiene un microcontrolador integrado que permite programar en un lenguaje de alto nivel. Es el elemento encargado de realizar procesos matemáticos y lógicos, así como de gestionar los recursos de cada componente externo que se conecte a la placa base. Una placa Arduino contiene varias entradas analógicas y digitales para que se pueda conectar diferentes sensores y otras placas o shield. Todo esto permite añadir nuevas funciones sin cambiar el diseño original de la placa (Millahual, 2020).

Figura 1

Placa de desarrollo Arduino

Nota: Estos son los diferentes tipos de placa Arduino. Tomado de *Placa de desarrollo Arduino* [Imagen] E. Vega, 2020.

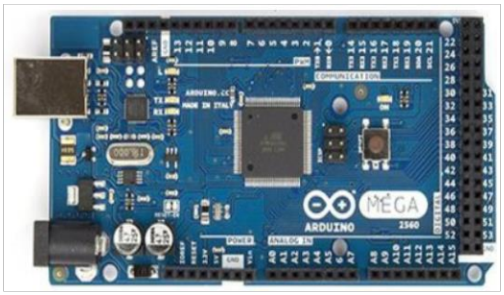
Un elemento importante dentro del hardware de Arduino son sus puertos de entrada/salida, mediante los que es posible conectarla a la computadora para integrar el trabajo con el software.

- **Arduino Mega 2560**

  Arduino Mega 2560 es una placa de desarrollo basada en el microcontrolador ATmega2560. Dispone de 54 entradas y salidas digitales, 15 de las cuales pueden utilizarse como salidas PWM (Pulse Width Modulation), 4 puertos seriales hardware (UART), cristal de 16 MHz, conexión USB, enchufe de alimentación, conector ICSP para programación y un botón de reset (Arduino Mega 2560, 2020).

Figura 2

Placa Arduino Mega 2560



Nota: La placa de desarrollo Arduino Mega 2560. Tomado de *Placa Arduino Mega 2560* [Imagen] Milanés Hermosilla, 2016, http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S1815-59282016000300006

1.1.2 Motores paso a paso

Un motor paso a paso es un motor de corriente continua sin escobillas que puede ser un motor de imán permanente o de resistencia variable. Tiene características tales como rotación bidireccional, movimiento angular preciso, sostener un torque de retención a velocidad cero y controlarse con circuitos digitales.
La cantidad de rotación es directamente proporcional al número de pulsos y la velocidad de rotación es relativa a la frecuencia de los pulsos. Estos motores son simples de operar en una configuración de lazo cerrado y debido a su tamaño proporcionan un excelente torque a baja velocidad (Jennings, 2002).

El motor paso a paso consta de dos partes:

- **El estator** es la parte fija del motor donde en sus cavidades van depositadas las bobinas.

- **El rotor** es la parte móvil del motor construido por un imán permanente.

Estas dos partes van montadas sobre un eje (Medrano Trujillo, 2020).

Figura 3

Motor Paso a Paso

Nota: El motor utilizado para accionar los grados de libertad del brazo robótico. Tomado de *Motor Paso a Paso* [Imagen] Medrano Trujillo, 2020.

Los motores paso a paso son comúnmente utilizados en sistemas de control digital, donde reciben comandos de lazo abierto en forma de secuencia de pulsos para girar su eje en un ángulo específico. Una característica clave de los motores paso a paso es que rotan de forma incremental, lo que significa que cada paso representa un desplazamiento angular fijo del eje. Estos motores se encuentran principalmente en electrodomésticos pequeños, sistemas de control y medición, y servomecanismos especiales, con potencias inferiores a los 700 W. Por esta razón, también se les conoce coloquialmente como motores de baja potencia (Francisco, 2005).

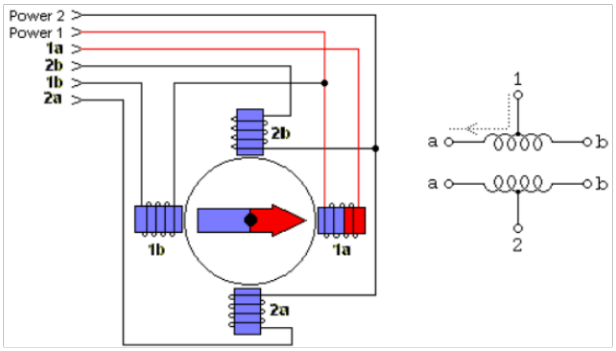
1.1.2.1 Tipos de motores paso a paso

Hay dos tipos de motores paso a paso: los unipolares y los bipolares.

- **Motores Unipolares:** Estos motores poseen dos bobinas que tienen un punto central del cual se originan los cables hacia el exterior. Estos cables se enlazan a la fuente de energía, mientras que los extremos de las bobinas se conectan a tierra para completar el circuito. La existencia de líneas comunes independientes o interdependientes varía según el tipo de motor.

Figura 4

Modelo conceptual de motor unipolar

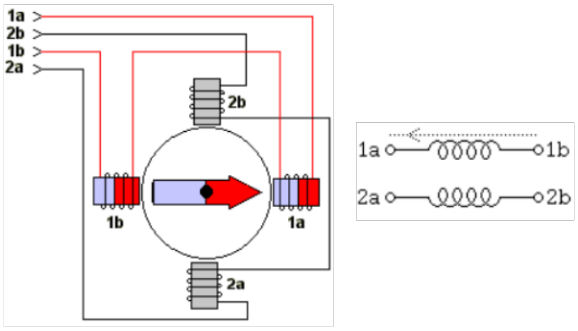


Nota: Representación de la estructura eléctrica de un motor unipolar. Tomado de *Motores Unipolares* [Imagen] 330ohms, 2016.

- **Motores Bipolares:** Tienen dos bobinas sin un punto medio donde salga un cable, lo que resulta en cuatro cables, correspondiendo cada par a las terminales de una bobina. Debido a la configuración de la bobina, la corriente puede fluir en dos direcciones, lo cual requiere un control que sea bidireccional o bipolar. En términos generales, con relación al sentido de giro de los motores paso a paso bipolares, es importante tener en cuenta que dicho sentido de giro depende de la dirección del flujo de corriente a través de las bobinas, ya que este flujo induce un campo magnético en el embobinado que genera un polo magnético norte y sur. Por lo tanto, el rotor se mueve de manera que uno de los polos del rotor sea opuesto al de la bobina (Borja Conde, 2018).

Figura 5

Modelo conceptual de motor bipolar



Nota: Representación de la estructura eléctrica de un motor bipolar. Tomado de *Motores Unipolares* [Imagen] 330ohms, 2016.

1.1.2.3 Motor paso a paso bipolar NEMA17

Estos motores tienen la particularidad de contar con dos bobinas que requieren cambios en la dirección de las corrientes que fluyen a través de ellas para generar pasos. Cada bobina tiene dos terminales. Al energizarlas de manera ordenada a través de estos terminales, se logra que el motor dé un paso o micro paso (que representa la mitad, un cuarto o un octavo de un paso completo). Por lo tanto, es necesario generar una secuencia repetitiva de activación de las bobinas.

Figura 6

Motor Nema 17

Nota: Referencia específica del motor utilizado en el brazo robótico. Tomado de *Motor Nema 17* [Imagen] Borja Conde, J.A, 2018.

Estos motores se caracterizan por tener alta precisión y potencia. Sus especificaciones se encuentran en la Tabla 1:

Tabla 1

Características principales de los motores paso a paso

Características	Medidas y unidades
Corriente nominal	1,2 A DC
Tensión de alimentación	12-36 V
Ángulo de avance	1.8°±5%
Par de fricción	12mN.m
Par de mantenimiento	≥300mN.m(I = 1.5 A)
Máx. Frecuencia de arranque sin carga	≥1500pps
Máx. Frecuencia de ejecución sin carga	≥8000pps
Peso del motor	0.23Kg
Tamaño (L*W*H)	42*42*34 mm

Nota: Estas son las características principales de los motores paso a paso. Tomado de Borja Conde, J.A. (2018). *Desarrollo de un robot auto equilibrado basado en Arduino con motores paso a paso*. (Trabajo Fin de Grado Inédito). Universidad de Sevilla, Sevilla.

1.1.3 servomotor

Un servomotor es un tipo de motor de corriente continua que puede posicionarse en cualquier punto dentro de su rango de funcionamiento y mantenerse estable en esa posición. Un servomotor es un motor eléctrico que puede ser controlado tanto en velocidad como en posición (Millán Castrillón, 2019).

Figura 7

Servomotor

Tomado de *Servomotor* [Imagen] Millán Castrillón, 2019.

Dentro del servomotor, se encuentra un motor de corriente continua común. El eje del motor está conectado a una caja de engranajes similar a una transmisión, lo cual amplifica el torque del motor y permite mantener una posición fija cuando sea necesario. Al igual que en un automóvil, a menor velocidad, mayor es el torque. El circuito electrónico es responsable de controlar el movimiento y la posición del motor (Alberto, 2015).

1.1.3.1 Tipos de Servomotores

Es importante destacar que, entre los diversos tipos de servomotores, es posible categorizarlos según las particularidades de su movimiento rotativo.

- **Servomotores de rango de giro limitado:** son ampliamente utilizados y representan el tipo más común de servomotor. Tienen la capacidad de girar hasta 180 grados, lo que significa que no pueden completar una rotación completa de 360 grados.

- **Servomotores de rotación continua:** presentan la capacidad de realizar una rotación completa de 360 grados. Aunque su funcionamiento es similar al de un motor convencional, poseen las propiedades distintivas de un servo. Esto implica que se puede regular su posición y velocidad de giro en cualquier momento determinado.

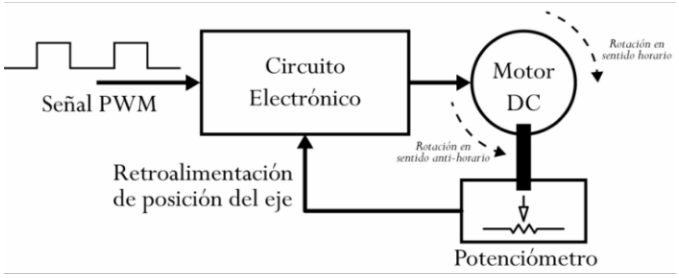
Es posible modificar los servomotores de rango de giro limitado para que operen como servomotores de rotación continua. No obstante, si se requiere un servo con una rotación de 360 grados, es recomendable adquirir uno específicamente diseñado para ese propósito (García González, 2016).

1.1.3.2 Funcionamiento de los Servomotores

Para operar el Servomotor AC de Imanes permanentes se requiere un dispositivo de conducción o driver. Este recibe la señal de control del ordenador central y la dirige hacia la bobina del motor, regulando la corriente de activación y desactivación de la bobina para crear un campo magnético. Este campo magnético asegura que el polo de Imán permanente en el rotor se mantenga en un ángulo eléctrico de 90 grados. A través de la utilización de mecanismos de acople como el codificador fotoeléctrico y el transformador giratorio, se detecta la interrelación entre el estator y el rotor, permitiendo un control parcial o total del bucle cerrado para manejar con exactitud el desplazamiento angular. (Hernández Orozco, 2015).

Figura 8

Diagrama de bloque del Servomotor



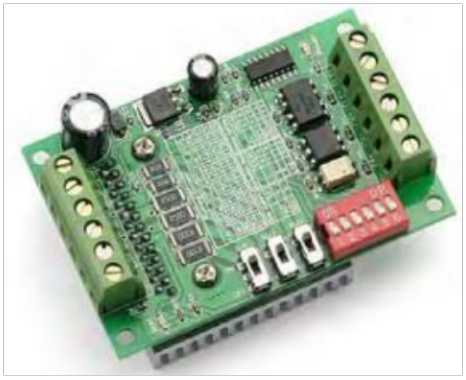
Nota: El diagrama de bloques representa la estructura de funcionamiento de un servomotor. Tomado de *Diagrama de bloque del Servomotor* [Imagen] González, A. G, 2016.

1.1.4 Driver TB6560

El driver TB6560 es un controlador de motor paso a paso que se utiliza comúnmente en sistemas de control de movimiento. Se trata de un aparato económico y sencillo de manejar que posibilita el manejo de motores bipolares y unipolares de hasta 3A de corriente. La placa que incorpora el circuito integrado TB6560 es un controlador de motores que opera mediante la emisión de pulsos digitales de trabajo y dirección.

Figura 9

Driver TB6560.



Nota: Driver utilizado para el control de los motores paso a paso. Tomado de *Driver TB6560* [Imagen] Caballero Durán, 2022.

Este controlador emplea el chip TOSHIBA TB6560, que utiliza tecnología de control de corriente sinusoidal pura, permitiendo que distintos tipos de motores operen con menos ruido y evitando el sobrecalentamiento. Además, proporciona movimientos suaves y mejora el desempeño a altas velocidades en comparación con otros controladores disponibles en el mercado (Ortiz González & Vintimilla Ávila, 2021).

Tabla 2

Hoja de datos del driver TB6560

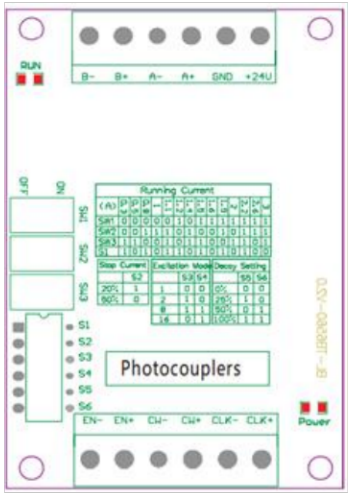
Símbolo de la terminal	Descripción
B- B+	Fase B Motor a pasos
A- A+	Fase A motor a pasos
GND +24V	Alimentación
EN- EN+	Enable – Activación
CW- CW+	Dirección
CLK- CLK+	Pulso de trabajo

Nota: Esta tabla contiene las especificaciones de los pines de la tarjeta TB6560. Tomado de *Driver TB6560*, Caballero Durán, 2022.

El controlador de motores paso a paso bipolares TB6560, con un máximo de 3A y 24V, es fácil de usar y se puede utilizar en proyectos que requieren el control de motores. Cuenta con 2 terminales grandes en sus extremos, 6 entradas, 6 salidas y, posteriormente, los interruptores de control.

Figura 10

Esquemático de la tarjeta TB6560



Nota: Pines de conexión de la tarjeta TB6560. Tomado de Esquemático de la tarjeta TB6560 [Imagen] Benne, 2019.

Los interruptores de control, cuenta con “SW1 SW2 SW3 S1 S2 S3 S4 S5 S6”, como se muestra en la Figura 11.

- “SW1 SW2 SW3 S1” controlan el Amperaje, se recomienda colorarlo debajo del Amperaje de opresión de nuestro Motor a pasos (NEMA 17 1.8A) lo colocamos a 1.6Amp.
  - (SW1 “ON” SW2 “OFF” SW3 “OFF” S1 “OFF”)

- “S1 20% y S2 50%” Stop Current, donde si el Amp aumenta un 20% se detendrá.
  - (S1 “OFF” y S2 “OFF”)

- “S3 S4” son el paso por Pulso e Opresión, a menor paso, más precisión, menor potencia.
  - (Experimenten S3 S4: 1 “OFF OFF”, ½ “ON OFF”, 1/8 “ON ON”, 1/16 “OFF ON”)

- “S5 S6” Decay Setting
  - (RECOMENDADO S5 “OFF” S6 “OFF”)

Tabla 3

Switches de control del Driver TB6560

Corriente de trabajo														
(A)	0.3	0.5	0.8	1.0	1.1	1.2	1.4	1.5	1.6	1.9	2.0	2.2	2.6	3.0
SW1	OFF	OFF	OFF	OFF	OFF	ON	OFF	ON	ON	ON	ON	ON	ON	ON
SW2	OFF	OFF	ON	ON	ON	OFF	ON	OFF	OFF	OFF	OFF	ON	ON	ON
SW3	ON	ON	OFF	OFF	ON	OFF	ON	ON	OFF	ON	ON	ON	OFF	ON
S1	ON	OFF	ON	OFF	ON	ON	OFF	ON	OFF	OFF	OFF	ON	OFF	OFF

Corriente Max	
	S2
20%	ON
50%	OFF

Control de pasos		
STEP	S3	S4
Completo	OFF	OFF
Medio	ON	OFF
1/8	ON	ON
1/16	OFF	ON

Control de decline		
	S5	S6
0%	OFF	OFF
25%	ON	OFF
50%	OFF	ON
100%	ON	ON

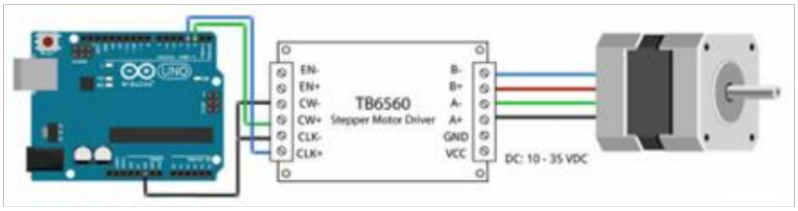
Nota: Esta tabla contiene las especificaciones de los switches de control de la tarjeta TB6560. Tomado de *Switches de control del Driver TB6560*, Pedroza Fernández, F. J., & Araque Mora, 2019.

1.1.4.1 Cableado - Conexión del TB6560 al motor paso a paso y al Arduino

El siguiente diagrama de cableado muestra cómo se puede conectar el controlador de motor paso a paso TB6560 al Arduino y a un motor paso a paso.

Figura 11

Conexión del Driver TB6560



Nota: Controlador de motor paso a paso TB6560 con Arduino UNO y diagrama de cableado del motor paso a paso. Tomado de *Conexión del Driver TB6560* [Imagen] Benne, 2019.

1.1.5 Convertidor DC – DC XL6009 tipo Boost

El XL6009 Boost Converter es un regulador de conmutación que suministra un voltaje de salida constante y más alto que el voltaje de entrada, incluso ante fluctuaciones en los voltajes de entrada y carga. Este dispositivo puede operar en un rango de voltajes de entrada de 3V a 32V, ofreciendo un voltaje de salida ajustable de 5V a 35V. Además, puede proporcionar una corriente de salida máxima de 4A, sujeta a la tensión de entrada y salida (Galindo Mejía & Rodríguez Peláez 2019).

Figura 12

Convertidor DC – DC XL6009 tipo Boost

Nota: El convertidor DC-DC es utilizado para regular el voltaje de los motores. Tomado de *Convertidor DC – DC XL6009 tipo Boost* [Imagen] Galindo Mejía & Rodríguez Peláez, 2019.

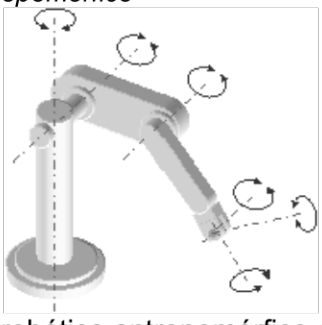
1.2 Sistema Robótico

Un brazo robótico es un dispositivo programable cuyas funciones principales y comportamiento se asemejan a las de un brazo humano. Las diversas partes que lo constituyen se enlazan y se interconectan, permitiendo al robot realizar movimientos rotativos y de desplazamiento. Al extremo del brazo, se encuentra una mano robótica, la cual puede ser de pinza, ventosa o garra, dependiendo de la tarea a realizar, como sujetar, mover objetos, seleccionar o ensamblar componentes.

Principalmente, estos dispositivos se utilizan para emular o reemplazar las capacidades del brazo humano en procesos de producción o logística. Al ser completamente automatizado, el brazo robótico industrial lleva a cabo las mismas labores que un trabajador, pero con la ventaja de poder manipular cargas más pesadas, a mayor velocidad y sin el desgaste que supone para un ser humano la repetición constante de movimientos (Mecalux, 2021).

Figura 13

Morfología de un brazo robótico antropomórfico



Nota: La morfología de un brazo robótico antropomórfico es similar a la de un brazo humano. Tomado de Morfología de un brazo robótico antropomórfico [Imagen] Trejo Peraza, 2018.

La forma de un robot industrial varía según la tarea específica que se busca llevar a cabo. Las articulaciones, que son los componentes que conectan los ejes del robot, son responsables de generar el movimiento de este. Cada acción independiente que una articulación puede ejecutar se conoce como grado de libertad (GDL). El número total de GDL de un robot indica la cantidad de movimientos independientes que puede realizar. En un espacio tridimensional, el máximo de GDL es seis, incluyendo tres desplazamientos y tres rotaciones, como se muestra en la Figura 15 (Trejo Peraza, 2018).

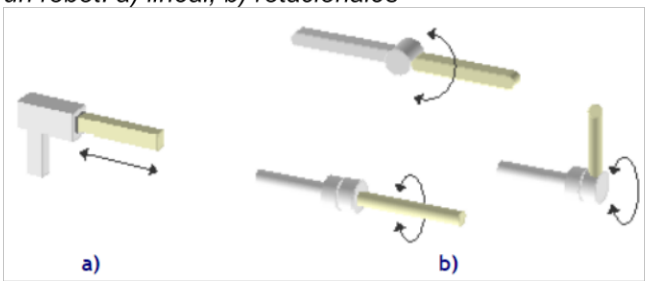
Una articulación puede ser:

- **Lineal** (deslizante, traslacional o prismática), si un eslabón desliza sobre un eje solidario al eslabón anterior.

- **Rotacional**, en caso de que un eslabón gire en torno a un eje solidario al eslabón anterior.

Figura 14

Tipos de articulaciones de un robot: a) lineal, b) rotacionales

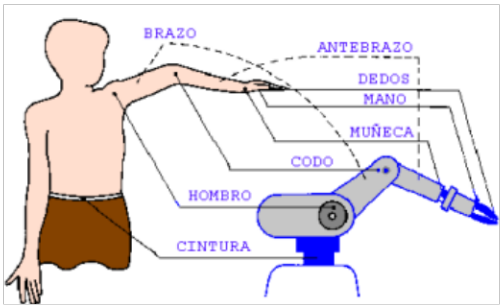


Nota: Las articulaciones de un robot pueden ser lineales o rotacionales. Tomado de Tipos de articulaciones de un robot [Imagen] 5.4 Robots Industriales, 2004.

El brazo robótico antropomórfico utiliza coordenadas angulares para determinar la posición de su extremo, ya que sus tres principales articulaciones son del tipo R. Se le llama antropomórfico porque este brazo simula los movimientos de un brazo humano, donde el primer eje representa el cuerpo, el segundo el brazo, el tercero el antebrazo y el resto la muñeca y la mano. La primera articulación corresponde al movimiento de la cintura, la segunda al hombro, la tercera al codo y las restantes se encuentran en la muñeca (Trejo Peraza, 2018).

Figura 15

Relación brazo robótico con anatomía humana



Nota: Se puede observar la relación de las partes del brazo robótico con el brazo humano. Tomado de *Relación brazo robótico con anatomía humana* [Imagen] Pedroza Fernández & Araque Mora, 2019.

Para lograr una precisión óptima en la ejecución de tareas por parte de un robot, se necesitan varios componentes esenciales. Estos incluyen los actuadores, responsables de generar la fuerza necesaria para el movimiento de las partes mecánicas del manipulador. También se requieren sensores que permitan medir las características del entorno en el que se encuentra el robot. Además, se utilizan modelos cinemáticos para evaluar la posición y orientación del extremo del robot. Los generadores de trayectorias desempeñan un papel crucial al establecer los valores de posición, velocidad, tiempo y tipo de trayectoria, según las preferencias del usuario. Por último, el tipo de control implementado, como el control PID, adaptivo, predictivo o basado en cálculos de torque, también es fundamental en este proceso (Fernando, 2019).

1.2.1 Modelado cinemático del Brazo Robótico

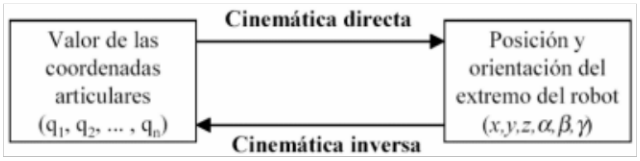
La cinemática en la robótica es una disciplina que se dedica al estudio y análisis del desplazamiento de los sistemas mecánicos, sin considerar las fuerzas que los generan. En este contexto, el análisis cinemático de un brazo robótico antropomórfico de 4 grados de libertad (4-DOF) se enfoca en comprender y representar el movimiento y las posiciones que puede alcanzar este tipo de brazo robótico, el cual emula la estructura y funciones del brazo humano.

Un brazo robótico antropomórfico que tiene cuatro grados de libertad incluye cuatro articulaciones separadas que posibilitan movimientos en diversas direcciones y planos. Estas articulaciones típicamente engloban la rotación en su base, el desplazamiento del hombro, el movimiento del codo y la articulación de la muñeca. Al combinar estos movimientos, el brazo robótico puede lograr distintas posiciones y orientaciones en el espacio de trabajo.

El análisis cinemático de un brazo robótico de 4-DOF se divide en dos partes principales: cinemática directa e inversa.

Figura 16

Relación Cinemática Directa vs Cinemática Inversa



Nota: La cinemática directa e inversa son fundamentales en el modelamiento de un brazo robótico. Tomado de Relación Cinemática Directa vs Cinemática Inversa [Imagen] Millán Castrillón, 2019.

1.2.1.1 Modelado cinemático Directo

La cinemática directa implica determinar la posición y orientación del extremo del robot con relación a un sistema de coordenadas de referencia. Esto se logra al conocer los valores de las articulaciones y los parámetros geométricos de los componentes del robot. En la cinemática, las variables con las que se trabaja son los ángulos entre los eslabones en el caso de una articulación rotacional, y las longitudes formadas por la extensión del eslabón en el caso de una articulación prismática. Estas variables se representan mediante 4 elementos (θ_i, d_i, a_i y α_i).

1.2.1.1.1 Cinemática Directa: método Denavit-Hartenberg

El proceso de la cinemática directa inicia con la aplicación del método de Denavit-Hartenberg, el cual posibilita determinar la posición de los sistemas de referencia de los componentes articulados en los robots, tanto aquellos con movimientos prismáticos como de revolución, en cadenas cinemáticas abiertas. A continuación, se detalla la secuencia de pasos necesarios para obtener dichos parámetros:

- Numerar los eslabones: desde la base hasta el efector final.
- Numerar las articulaciones o grados de libertad.
- Localizar el eje de cada articulación: Para pares de revolución, será el eje de giro. Para prismáticos será el eje a lo largo del cual se mueve el eslabón.
- Sistema de coordenadas: Se sitúa el punto origen en cada articulación.
- Determinar la normal común entre las articulaciones y ubicar como eje Z.
- El último sistema se coloca en el extremo del robot, con su eje Z paralelo al anterior.
- Ejes X: Cada uno va en la dirección de la normal común a Z. El sentido del eje es indistinto.
- Ejes Y: Se colocan para que los ejes X y Z formen un sistema dextrogiro (regla de la mano derecha) (Ramos Rojas & Pertuz Sánchez, 2018).

Con este algoritmo cada transformación homogénea entre cada articulación se representa con un producto de cuatro transformaciones:

$$A_i = Rot_{z,\theta_i} * Trans_{Z,di} * Trans_{X,ai} * Rot_{X,\alpha i} \quad (1)$$

$$\begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & 0 \\ sin\theta_i & cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & di \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & ai \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -sin\theta_i & 0 & 0 \\ 0 & cos\alpha_i & -sin\alpha_i & 0 \\ 0 & sin\alpha_i & cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$Ai = \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & aicos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Donde los valores θi, di, ai y αi son los parámetros que se asocian a la interacción entre las i articulaciones (grados de libertad). Estos parámetros se describen como:

θi: Ángulo formado por la articulación.

ai: Longitud que tiene el enlace.

αi: Ángulo de torsión del enlace.

di: El desplazamiento u offset del enlace.

1.2.1.2 Modelado Cinemático Inverso

El objetivo es determinar los valores de las coordenadas articulares del robot ($q = [q1, q2, \dots, qn]^T$) para que su extremo se posicione y oriente según una ubicación espacial específica (p, [n, o, a]). Mediante la cinemática directa, se utiliza de manera sistemática matrices homogéneas que representan cada grado de libertad del brazo robótico, sin considerar una configuración particular. Esto permite definir la cinemática inversa, que determina la relación de movimiento y orientación del robot en función de una configuración dada del brazo robótico.

El método inverso ofrece múltiples soluciones en contraste con el método directo, lo que significa que existen varias combinaciones para posicionar la parte final del brazo de la misma manera. Esto brinda la oportunidad de seleccionar la solución más adecuada mediante un enfoque matemático cerrado.

Sin embargo, si se requiere interpretar el movimiento del brazo robótico en tiempo real, como en el caso de seguir una trayectoria específica, cualquier

otra solución no garantizaría un seguimiento preciso en el momento y lugar adecuados.

El problema inverso puede descomponerse en dos también:

- Encontrar las transformaciones necesarias para obtener la ubicación del extremo del efector. De manera más precisa, considerando un objetivo {O} establecido, se realizarían las siguientes acciones:

  a. Determinar la ubicación requerida de la herramienta {H} basándose en el objetivo {O}.
  b. Calcular la ubicación de la muñeca a partir de {H}.

- Determinar los valores de las articulaciones a partir de la posición y la dirección en el espacio tridimensional especificadas por la muñeca (Medrano Trujillo, 2020).

1.2.1.2.1 Solución del problema cinemático inverso

El problema se explica mediante una serie de ecuaciones que utilizan una matriz 4x4 para representar la transformación homogénea final de los marcos de coordenadas H. Esta transformación depende de los n grados de libertad (q1,..., qn). (Ecuación 7)

$$H = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \in SE(3) \quad (4)$$

$$H = \begin{bmatrix} R & O \\ 0 & 1 \end{bmatrix} \in SE(3) \quad (5)$$

$$T_n^0(q_1, \ldots, q_n) = H \quad (6)$$

$$T_n^0(q_1, \ldots, q_n) = A_1(q_1) \ldots A_n(q_n) \quad (7)$$

La matriz R en SE(3) determina una o varias soluciones posibles para la Ecuación (8) y representa la orientación final de la pinza del robot. O representa la posición final de la pinza siguiendo la estructura de los marcos de coordenadas, mientras que H es la matriz de transformación homogénea desde el marco 0 hasta el marco n de coordenadas. El objetivo es encontrar los valores de las variables $(q_1, \ldots, q_n)$ de manera que $T_n^0(q_1, \ldots, q_n)$=H (Millán Castrillón, 2019).

1.3 Control del brazo robótico

El control de un brazo robótico es fundamental para su correcto funcionamiento y la ejecución de tareas específicas. Para lograr esto, se utilizan programas o software especializados que permiten interactuar con el brazo y enviar comandos para controlar sus movimientos y acciones.
Existen diferentes tipos de software utilizados para el control de brazos robóticos, y su elección depende del tipo de brazo, sus características y los requisitos de la aplicación. Uno de los programas comúnmente utilizados es el software de control de movimiento, que permite programar y coordinar los movimientos del brazo en función de diferentes parámetros, como la posición, la velocidad y la aceleración.

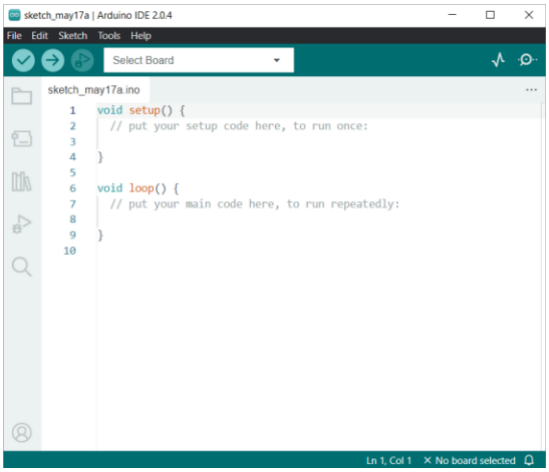
1.3.1 Arduino IDE

El entorno de desarrollo integrado o IDE de Arduino es una aplicación compatible con varias plataformas que permite escribir y cargar programas en placas Arduino y otras compatibles. Además, gracias a núcleos generados por terceros, también es posible cargar programas en placas de desarrollo de otros fabricantes.
El IDE de Arduino posibilitará no solo la redacción de un código, sino también la depuración, modificación y almacenamiento de los programas o sketches en la placa de desarrollo de una manera fácil y veloz.

Un punto clave a considerar es que, gracias a las funcionalidades proporcionadas por Arduino IDE, es posible cargar tus programas directamente en la memoria flash de Arduino de manera sencilla y rápida, evitando la necesidad de ejecutar procedimientos complicados. De este modo, la placa puede ejecutar el programa en un tiempo mínimo (Peña, C. 2020).

Figura 17

Sketch de Arduino



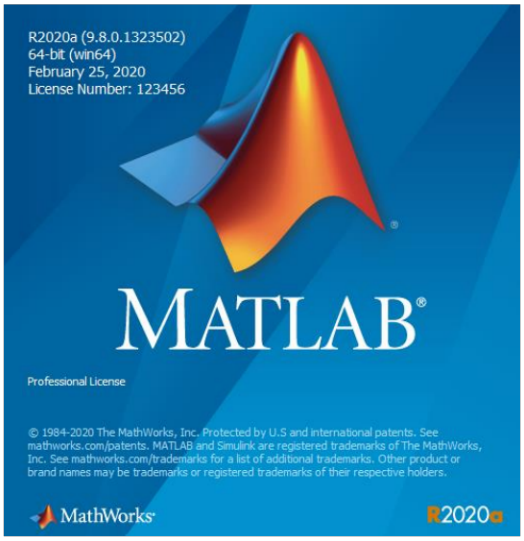
Fuente: Autor

1.3.2 Matlab

MATLAB es una plataforma interactiva y un lenguaje de programación de nivel avanzado utilizado para cálculos numéricos, visualización y desarrollo de software. Con MATLAB, es factible examinar datos, diseñar algoritmos y construir modelos o aplicaciones. Sus herramientas, funciones matemáticas incorporadas y su lenguaje permiten explorar diferentes enfoques y alcanzar una solución más rápida que con hojas de cálculo o lenguajes de programación convencionales, como C/C++ o Java (Matlab. L, 2016).

Figura 18

MATLAB R2020a



Fuente: Autor

Dentro de las funciones fundamentales que ofrece se encuentran: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y dispositivos hardware.

El software MATLAB cuenta con dos herramientas adicionales, entre muchas otras, que amplían sus funcionalidades: Simulink, una plataforma de simulación multidominio, y GUIDE, un editor de interfaces de usuario (GUI). Además, es posible mejorar las capacidades de MATLAB utilizando cajas de herramientas (toolboxes) y las de Simulink mediante paquetes de bloques (blocksets) (Corbacho, A. G. 2014).

1.3.2.1 Interfaz Gráfica

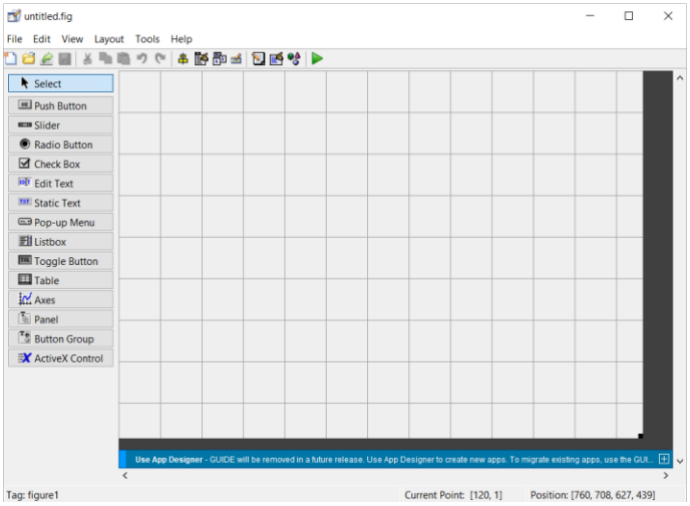
Una interfaz gráfica de usuario (GUI), es una interfaz visual que utiliza objetos gráficos como menús, botones, listas y barras de desplazamiento. Estos objetos se utilizan para definir acciones en respuesta a cambios o interacciones, como hacer

clic con el ratón o arrastrar objetos. Para crear una GUI efectiva, se deben seleccionar cuidadosamente los objetos gráficos adecuados, distribuirlos de manera lógica en la pantalla y determinar las acciones que se realizarán al activarlos.
Cuando un usuario empieza a utilizar una computadora, empieza a interactuar con la Interfaz, ya sea del sistema operativo, de un programa específico o de un sitio web. Esto marca el inicio de la interacción entre el humano y la computadora (Albornoz, M. C., Berón, M., & Montejano, G. A. 2017).

En Matlab, los gráficos se organizan en una estructura jerárquica que incluye diferentes tipos de objetos.

Figura 19

MATLAB GUI



Fuente: Autor

2. Metodología

Considerando la información presentada anteriormente, el objetivo principal de este proyecto fue implementar un sistema de control que permita programar los movimientos de un brazo robótico antropomórfico de cuatro grados de libertad.

El desarrollo del proyecto constó de varias etapas cruciales para garantizar el éxito en el resultado final. Entre estas etapas se incluye la selección del software, la documentación del modelado cinemático del brazo robótico, la implementación del sistema de control y, finalmente, una serie de pruebas realizadas. A continuación, se describen con más detalle las etapas del desarrollo del proyecto:

2.1 Selección de software

En primer lugar, se realizó un levantamiento de información previa del brazo robótico. Esto implicó el análisis de las características y especificaciones técnicas del robot en cuestión, entender las capacidades y limitaciones tanto del hardware como del software.
Es importante comprender en detalle los requisitos y funcionalidades necesarias para el control del robot. En la Figura 20 se puede observar la estructura del brazo robótico propuesto, que ya se encontraba armado, el cual comprende todos los dispositivos electrónicos necesarios para su funcionamiento, a falta del sofware que permita su control.

Figura 20

Brazo Robótico propuesto

Fuente: Autor

Una vez que se cuenta con la información necesaria, se procedió a seleccionar el software adecuado para la implementación del control. Esto implicó la investigación y evaluación de diferentes opciones de software disponibles. Al considerar los requerimientos de hardware y software del robot se eligió el software idóneo. Durante esta etapa, se compararon las características, capacidades y facilidad de uso de las distintas opciones de software para determinar cuál es la mejor elección.

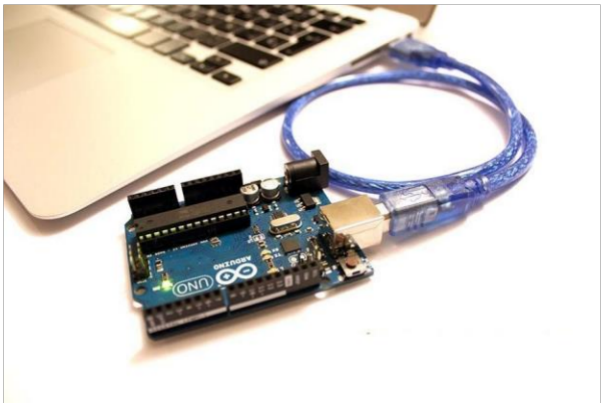
Teniendo en cuenta que el brazo robótico contiene una placa Arduino 2560, se eligió el IDE Arduino como software para implementar el sistema de control, ya que este proporciona una biblioteca de funciones específicas de Arduino que simplifican la programación de los motores y otras partes del brazo robótico. Además, el IDE Arduino permite cargar el código directamente en la placa Arduino, lo que facilita la iteración y prueba del sistema de control.
Al programar en el IDE Arduino, se utiliza un lenguaje de programación basado en C++ con algunas funciones y bibliotecas adicionales proporcionadas por Arduino,

permitiendo aprovechar su experiencia en programación y aplicarla al desarrollo del sistema de control del brazo robótico.

Una vez seleccionado el software, se procedió a establecer la comunicación entre el brazo robótico y el software de control. Esta etapa implicó la configuración de las conexiones físicas necesarias para permitir la comunicación entre el robot y el software. Esto incluye la conexión de cables y la configuración de interfaces de comunicación. Para esto se tiene un cable USB que conecta el Arduino con el ordenador desde el cual se está realizando el control, para establecer una conexión directa con la placa y poder cargar el código realizado.

Figura 21

Comunicación entre brazo robótico y software



Nota: La comunicación entre el brazo robótico y el software se realiza a través de la placa Arduino. Tomado de *Comunicación entre brazo robótico y software* [Imagen] Geek Factory, 2017.

Además de la configuración física, también es necesario establecer los protocolos de comunicación adecuados. Esto implica definir cómo el robot y el software de control intercambiarán información y comandos, por ejemplo, que seleccionar el valor de baudios correctos con los que se va a trabajar.

2.2 Solución modelo cinemático del brazo robótico

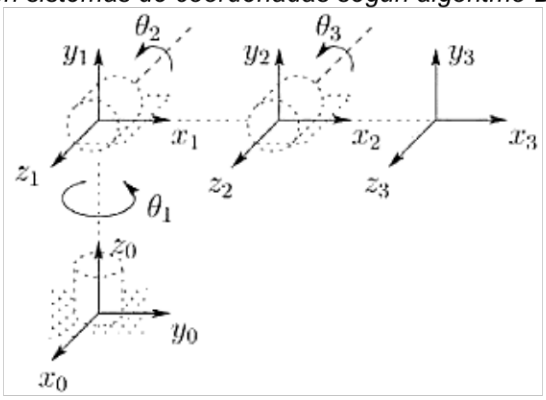
En el ámbito de la robótica industrial, existen diversas categorías de brazos robóticos. En este trabajo de grado se enfocará en el análisis y desarrollo de un brazo robótico antropomórfico de 4 grados de libertad. Este tipo de brazo se caracteriza por contar con articulaciones rotacionales que permiten posicionar el elemento final en distintos puntos dentro de su espacio de trabajo. Para comprender y evaluar el comportamiento de este brazo robótico, es fundamental obtener los modelos cinemáticos. Estos modelos proporcionan información sobre el movimiento del brazo al considerar exclusivamente sus características físicas y geométricas, sin tener en cuenta las fuerzas y pares que intervienen en su movimiento.

2.2.1 Solución del modelo cinemático directo mediante parámetros D-H

Cumpliendo con cada uno de los pasos mencionados anteriormente resulta un planteamiento tal que:

Figura 22

Representación del robot en sistemas de coordenadas según algoritmo D-H



Nota: Así es la representación en sistema de coordenadas del brazo robótico teniendo en cuenta el método D-H. Tomado de *Representación del robot en sistemas de coordenadas según algoritmo de Denavit-Hartenberg* [Imagen] Milanés Hermosilla, D., & Castilla Pérez, 2016.

Al emplear los sistemas de coordenadas, se adquieren los valores que indicarán la rotación de cada una de las articulaciones del brazo robótico. Los datos correspondientes se encuentran detallados en la Tabla 3.

Tabla 4

Parámetros D-H del brazo robótico

Eslabón	θ_i	a_i	α_i	d_i
1	θ_1	0	$\pi/2$	d_1
2	θ_2	a_2	0	0
3	θ_3	a_3	0	0
4	θ_4	a_4	0	0

Fuente: Autor

Tomando en cuenta el planteamiento de la matriz de transformación mencionada en la ecuación 3 y la estructura física del brazo robótico, a través del método de Denavit-Hartenberg se obtienen las siguientes matrices homogéneas para cada uno de los eslabones, las cuales están dadas por:

$$A_1 = \begin{bmatrix} cos\theta_1 & 0 & sin\theta_1 & 0 \\ sin\theta_1 & 0 & -cos\theta_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (8)$$

$$A_2 = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & a_2 cos\theta_2 \\ sin\theta_2 & cos\theta_2 & 0 & a_2 sin\theta_2 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

$$A_3 = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & a_3 cos\theta_3 \\ sin\theta_3 & cos\theta_3 & 0 & a_3 sin\theta_3 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

Una vez obtenidas las matrices homogéneas de cada articulación se procede al siguiente paso estipulado en el método D-H, el cual consta del cálculo de la matriz de transformación homogénea general del sistema (ecuación 3), la cual contiene las ecuaciones necesarias para resolver el problema cinemático directo.
Dicho esto,

$$A_g = A_1 * A_2 * A_3 * \dots * A_n \quad (11)$$

$$A_g = \begin{bmatrix} cos(\theta_2+\theta_3)\,cos\theta_1 & -sin(\theta_2+\theta_3)\,cos\theta_1 & sin\theta_1 & sin\theta_1(a_3\,cos(\theta_2+\theta_3)+a_2 cos\theta_2) \\ cos(\theta_2+\theta_3)sin\theta_1 & -sin(\theta_2+\theta_3)sin\theta_1 & -cos\theta_1 & sin\theta_1(a_3\,cos(\theta_2+\theta_3)+a_2 cos\theta_2) \\ sin\,(\theta_2+\theta_3) & cos\,(\theta_2+\theta_3) & 0 & d_1+a_3\,sin(\theta_2+\theta_3)+a_2 sin\theta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (12)$$

Definida la matriz homogénea genérica en la Ecuación 4, la posición final del efector será:

$$p_x = sin\theta_1(a_3\,cos(\theta_2+\theta_3)+a_2 cos\theta_2) \quad (13)$$

$$p_y = sin\theta_1(a_3\,cos(\theta_2+\theta_3)+a_2 cos\theta_2) \quad (14)$$

$$p_z = d_1 + a_3\,sin(\theta_2+\theta_3)+a_2 sin\theta_2 \quad (15)$$

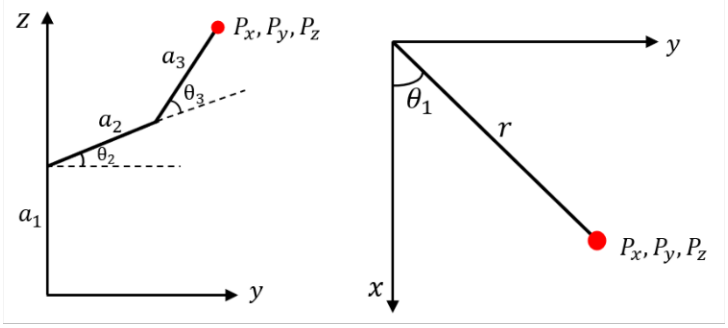
2.2.2 Solución del modelado cinemático inverso

Para obtener el modelo cinemático inverso de este primer modelo, se siguió empleando la trigonometría. Aprovechando la posición supuesta del brazo en el modelo cinemático inverso y los diferentes triángulos que se pueden formar a partir de esa posición, se calcularon los diferentes ángulos, esta vez expresados en función de los parámetros conocidos.

El robot utilizará la siguiente configuración para realizar el cálculo del modelo cinemático inverso:

Figura 23

Perspectiva del brazo robótico en los planos ZY y XY



Fuente: Autor

En la dimensión ZY se logra observar cómo se comportan los ángulos θ_2 y θ_3, mientras que en el plano XY se aclara que el brazo no se encuentra dentro del plano ZY, sino que tiene un ángulo θ_1 que lo posiciona entre los ejes X e Y.

- En el plano XY, se puede percibir un valor llamado "r", el cual representa la perspectiva de l2 más l3 desde ese plano.

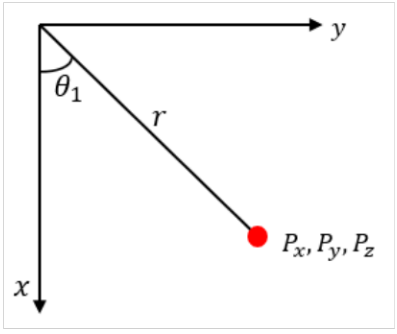
- La ubicación del punto final deseado para la muñeca se indica con un punto rojo, identificado como "Px, Py, Pz".

Una vez que se ha comprendido la posición del brazo, se trazan los triángulos relacionados y se emplea la trigonometría correspondiente para resolver el problema de la cinemática inversa.

Solución de θ_1

Figura 24

Perspectiva del brazo robótico en el plano XY



Fuente: Autor

El ángulo θ_1 se resuelve utilizando el plano XY como referencia, ya que desde este punto se puede observar su efecto. Para lograr esto, simplemente se plantea el arco tangente del ángulo en cuestión, que se forma a partir de las coordenadas Px y Py.

$$\theta_1 = \text{atan}\left(\frac{Py}{Px}\right) \tag{16}$$

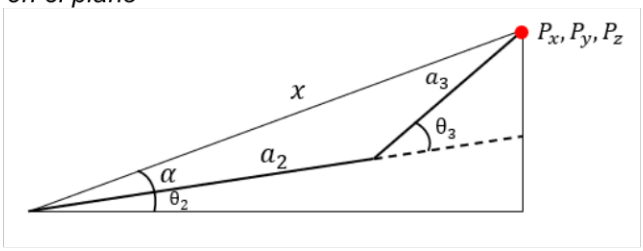
Además, se despeja "r" que será necesaria para el cálculo de los otros ángulos:

$$r^2 = P_x^2 + P_y^2 \tag{17}$$

Solución de θ_2 y θ_3

Figura 25

Perspectiva de a_2 y a_3 en el plano



Nota: Esta imagen es el resultado de observar a_2 y a_3 en el plano que lo contiene. Fuente: Autor

Como se puede apreciar la imagen muestra cuatro triángulos en total, los cuales, al combinarse entre sí, permiten obtener todos los parámetros deseados.

Se utiliza el teorema del coseno para resolver el triángulo compuesto por a_2, a_3 y x, para así obtener θ_3.

$$x^2 = a_2^2 + a_3^2 - 2 * a_2 * a_3 * \cos(180° - \theta_3) \tag{18}$$

Si se grafica la circunferencia trigonométrica de θ_3 y 180-θ_3, se puede notar que se verifica la misma relación:

$$\cos(180° - \theta_3) = -\cos(\theta_3) \tag{19}$$

Sustituyendo en la ecuación 14

$$x^2 = a_2^2 + a_3^2 - 2 * a_2 * a_3 * \cos(\theta_3) \tag{20}$$

Ahora se obtiene una ecuación que depende de parámetros conocidos, excepto x. Para resolver esto, se utiliza un triángulo rectángulo que engloba a los otros tres. Aplicando el teorema de Pitágoras en este triángulo, obtenemos:

$$x^2 = r^2 + P_z^2 \tag{21}$$

Previamente se había despejado r en la ecuación 13, por tanto, se reemplaza en la ecuación 17

$$r^2 + P_z^2 = a_2^2 + a_3^2 - 2 * a_2 * a_3 * \cos(\theta_3) \tag{22}$$

$$P_x^2 + P_y^2 + P_z^2 = a_2^2 + a_3^2 - 2 * a_2 * a_3 * \cos(\theta_3) \tag{23}$$

Ordenando la ecuación se obtiene

$$\cos(\theta_3) = \frac{P_x^2+P_y^2+P_z^2-a_2^2-a_3^2}{2*a_2^2*a_3^2} \tag{24}$$

Luego

$$\theta_3 = cos^{-1}\left(\frac{P_x^2+P_y^2+P_z^2-a_2^2-a_3^2}{2*a_2^2*a_3^2}\right) \tag{25}$$

Para la obtención de θ_2 se agrega un nuevo parámetro β, el cual representa la suma del ángulo θ_2 y α.

$$\beta = \theta_2 + \alpha \tag{26}$$

Con lo cual se procede al cálculo de ambos ángulos:

- Angulo β

$$\beta = atan\left(\frac{P_z}{r}\right) \tag{27}$$

Despejando r igual que en casos anteriores:

$$\beta = atan\left(\frac{P_z}{\pm\sqrt{P_x^2+P_y^2}}\right) \tag{28}$$

$$\alpha = atan\left(\frac{a_{3*}sin\theta_3}{a_2+a_3*cos\theta_3}\right) \tag{29}$$

Una vez calculados α y β, se reemplazan en la ecuación de θ_2:

$$\theta_2 = \beta - \alpha \tag{30}$$

$$\theta_2 = atan\left(\frac{P_z}{\pm\sqrt{P_x^2+P_y^2}}\right) - atan\left(\frac{a_{3*sin\theta_3}}{a_2+a_3*cos\theta_3}\right) \tag{31}$$

2.3 Sistema de control

El sistema de control del brazo robótico es una parte fundamental en el funcionamiento y la operación del robot. Este sistema es responsable de coordinar y controlar los movimientos de los diferentes actuadores del brazo, permitiendo que el robot realice tareas específicas de manera precisa y eficiente.

2.3.1 Modelamiento en Matlab

Contar con un proceso matemático para modelar un sistema real puede brindar ventajas al momento de observar y analizar el comportamiento de un robot. Asimismo, ofrece la posibilidad de realizar pruebas de funcionamiento y llevar a cabo experimentos controlados. En el caso específico del modelado cinemático de un brazo robótico, el objetivo primordial radica en comprender en profundidad el movimiento de este mecanismo, lo cual resulta fundamental para la predicción y planificación precisa de las trayectorias del sistema. Al desarrollar este modelo se puede optimizar el rendimiento y la eficiencia del brazo robótico.

En este caso se utilizó el software MATLAB y su herramienta Toolbox, del profesor Peter Corke para desarrollar una Interfaz Gráfica de Usuario (GUI), con el fin de aplicar la teoría de Denavit-Hartenberg y simplificar la interacción y el control del sistema, permitiendo programar movimientos, supervisar el funcionamiento en tiempo real y ajustar la configuración de manera más accesible.

2.3.1.1 Modelamiento cinemático

El modelamiento cinemático se enfoca en analizar la geometría y las relaciones de posición y orientación de sus articulaciones e implica la determinación de las ecuaciones matemáticas que describen la cinemática directa e inversa del sistema. En este caso se ha utilizado el Toolbox de MATLAB, el cual es apropiado para este proceso.

Se procede a realizar la simulación del robot en el entorno de MATLAB utilizando el comando Link, mediante el cual se pueden establecer los parámetros de cada uno de los eslabones del brazo robótico, empleando la notación D-H.

Figura 26

Parámetros Denavit-Hartenberg en MATLAB

```
>> l1=13;
l2=15;
l3=16;

L1 = Link('d', l1 , 'a', 0 , 'alpha', pi/2);
L2 = Link('d', 0 , 'a', l2 , 'alpha', 0);
L3 = Link('d', 0 , 'a', l3 , 'alpha', 0);

R=SerialLink([L1,L2,L3])

R =

noname:: 3 axis, RRR, stdDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|         13|          0|     1.5708|          0|
|  2|         q2|          0|         15|          0|          0|
|  3|         q3|          0|         16|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

Fuente: Autor

En la Figura 27 se denotan los parámetros de los eslabones del brazo robot siguiendo el método D-H, lo que permite la obtención del modelado cinemático directo e inverso del robot. Haciendo uso del script de MATLAB nuevamente, se procede a definir las ecuaciones de cinemática inversa, que permiten obtener los ángulos que deben adoptar las articulaciones para posicionar el extremo del brazo robótico en el punto predeterminado. Para esta etapa se realizaron tres pruebas, introduciendo diferentes posiciones para comprobar el modelo cinemático inverso.

Figura 27

Obtención de variables articulares para la posición (10,20,30) mediante Cinemática Inversa

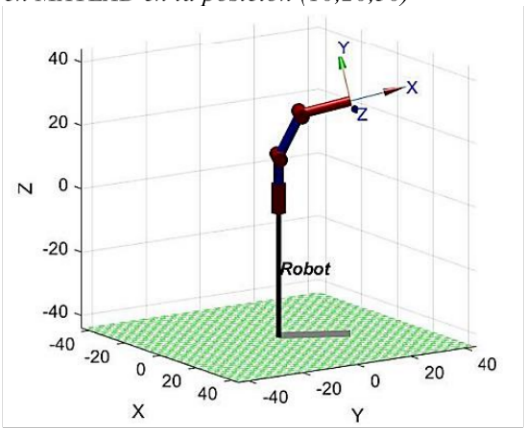
```
>> L1=13;
L2=15;
L3=16;
px=10;
py=20;
pz=30;
D=(px^2+py^2+(pz-L1)^2-L2^2-L3^2)/(2*L2*L3);
theta1=atan2(py,px);
theta3=atan2(-sqrt(1-D^2),D);
theta2=atan2(pz-L1,sqrt(px^2+py^2))-atan2(L3*sin(theta3),L2+L3*cos(theta3));
fprintf('Las variables articulares son:')
q=[theta1*180/pi;theta2*180/pi;theta3*180/pi]

Las variables articulares son:
q =

   63.4349
   63.1497
  -50.0838
```

Fuente: Autor

En la figura 28 se puede apreciar la obtención de las variables angulares en grados de las articulaciones del brazo robótico para posicionarse en un punto específico en el plano, para este caso en el punto (10,20,30) empleando las ecuaciones de cinemática inversa. Así mismo, se realiza una simulación del robot para observar su ubicación luego de conocer estas variables.

Figura 28

Simulación del brazo robótico en MATLAB en la posición (10,20,30)



Fuente: Autor

Seguidamente se realizó una segunda prueba ingresando la posición (5,0,10), calculando las variables angulares que debe tomar las articulaciones del brazo robot para posicionarse en ese punto.

Figura 29

Obtención de variables articulares para la posición (5,0,10) mediante Cinemática Inversa

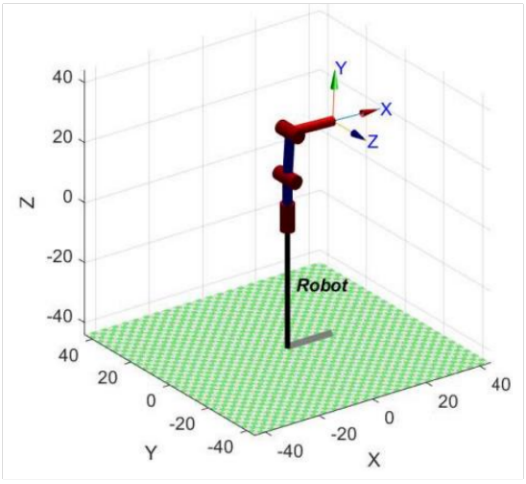
```
>> L1=13;
L2=15;
L3=16;
px=5;
py=0;
pz=10;
D=(px^2+py^2+(pz-L1)^2-L2^2-L3^2)/(2*L2*L3);
theta1=atan2(py,px);
theta3=atan2(-sqrt(1-D^2),D);
theta2=atan2(pz-L1,sqrt(px^2+py^2))-atan2(L3*sin(theta3),L2+L3*cos(theta3));
fprintf('Las variables articulares son:')
q=[theta1*180/pi;theta2*180/pi;theta3*180/pi]

Las variables articulares son:
q =

         0
   58.0536
 -158.6305
```

Fuente: Autor

También se realiza la simulación del brazo robótico, para observar el posicionamiento de este en el punto definido.

Figura 30

Simulación del brazo robótico en MATLAB en la posición (5,0,10)



Fuente: Autor

Finalmente se establece la coordenada (45,60,90) y se realiza el mismo procedimiento en MATLAB para hallar el valor de los ángulos de las articulaciones para esta posición.

Figura 31

Obtención de variables articulares para la posición (15,7,23) mediante Cinemática Inversa

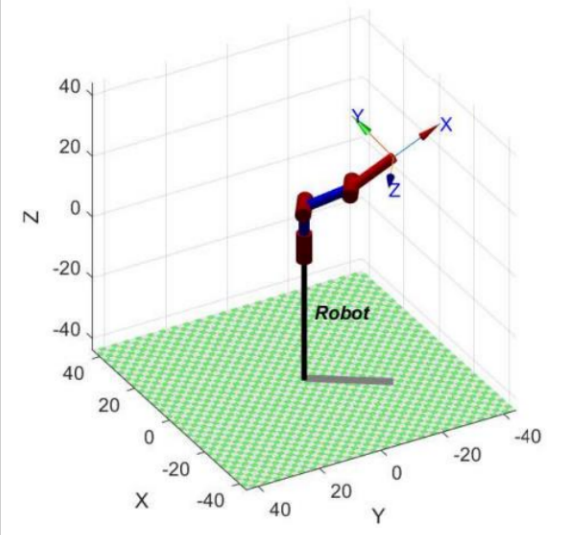
```
>> L1=13;
L2=15;
L3=16;
px=21;
py=15;
pz=7;
D=(px^2+py^2+(pz-L1)^2-L2^2-L3^2)/(2*L2*L3);
theta1=atan2(py,px);
theta3=atan2(-sqrt(1-D^2),D);
theta2=atan2(pz-L1,sqrt(px^2+py^2))-atan2(L3*sin(theta3),L2+L3*cos(theta3));
fprintf('Las variables articulares son:')
q=[theta1*180/pi;theta2*180/pi;theta3*180/pi]

Las variables articulares son:
q =

   35.5377
   19.3278
  -62.5860
```

Fuente: Autor

También se realizó la respectiva simulación del robot adoptando esta nueva posición.

Figura 32

Simulación del brazo robótico en MATLAB en la posición (5,0,10)



Fuente: Autor

Es importante aclarar que este modelado solo se hará a nivel de simulación y se dejará propuesto como objeto de estudio, debido que hubo que cambiar la forma de controlar el brazo robótico, tomando los grados de ángulos en cada juntura como parámetro a intervenir, debido que no había relación entre las posiciones ingresadas mediante este método con los movimientos que realizaba el prototipo.

2.3.2 Algoritmo de control

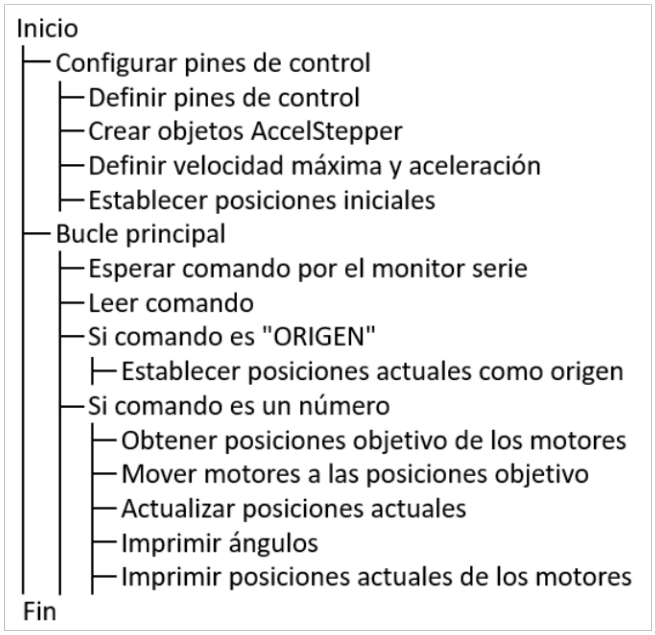
El software de control aplicado a un brazo robótico es una pieza fundamental para su funcionamiento y la operación eficiente. Este tipo de software se encarga de coordinar y controlar los movimientos del brazo robótico, permitiendo ejecutar una amplia variedad de tareas de forma precisa y autónoma.

Para este software se han importado librerías propias de Arduino como lo son: AccelStepper, la cual se utiliza para el control de los motores paso a paso ofreciendo varias características útiles, como aceleración, desaceleración y velocidad constante al momento de la puesta en marcha del sistema.

A continuación, se presentará un diagrama de flujo mediante el cual se mencionarán los procesos más relevantes del software diseñado para el sistema de control del brazo robótico, el cual fue desarrollado en Arduino.

Figura 33

Diagrama de flujo de la estructura del software de control



Fuente: Autor

En la Figura 30 se presenta un diagrama de flujo mediante el cual se describe la estructura del código diseñado para el control del brazo robótico.

El programa comienza con la configuración de los pines de control para los motores. Se definen los pines de paso y dirección para cada motor, y se crean los objetos AccelStepper correspondientes a cada motor. Además, se establece la velocidad máxima y la aceleración para los motores, y se inicializan las posiciones actuales de los motores.

Luego, el programa entra en un bucle principal donde espera la recepción de comandos a través del monitor serie. Una vez que un comando está disponible, se lee y se realiza una serie de acciones dependiendo del tipo de comando.

Si el comando es "ORIGEN", el programa establece las posiciones actuales de los motores como el origen (posición cero) para reiniciar el sistema.

Si el comando es un número, se interpreta como un ángulo para los motores. Luego, los motores se mueven teniendo en cuenta los ángulos ingresados utilizando la función moveTo() de la librería AccelStepper, y se espera a que los motores alcancen dichas posiciones utilizando la función runToPosition(). Después de mover los motores, se actualizan las posiciones actuales.

Después de realizar las acciones según el comando recibido, se imprime la posición actual de cada motor por el monitor serie.

El programa vuelve al inicio del bucle principal y espera la llegada de nuevos comandos.

Así, el diagrama de flujo muestra la estructura general del programa de control, facilitando la comprensión del proceso y las acciones realizadas en cada etapa.

Es importante decir que se tuvo que ajustar los parámetros de control del brazo robótico, debido que no se pudo realizar a través de modelado cinemático inverso, sino que se realizó mediante grados en juntura, lo que significa que se deben ingresar valores angulares para sus articulaciones moviéndose cada una de estas la cantidad de ángulos correspondientes a la cantidad requerida por el usuario. Además, cabe aclarar que el cálculo del modelo cinemático realizado se dejará propuesto a nivel de simulación ofrecido como objeto de estudio.

2.3.3 Desarrollo de Interfaz Gráfica

Luego de haber diseñado el software de control para el brazo robótico se procedió a la elaboración de la interfaz gráfica usando la herramienta MATLAB GUI, mediante la cual se mostrará una serie de parámetros y se establecerá la comunicación con el robot desde un ordenador.
En la interfaz se incluyó unas casillas donde se introducirán los valores angulares de las junturas del brazo robot y dos botones, uno para enviar los valores ingresados y otro que al oprimirse regresa al robot a su posición original (posición cero). Además, cuenta con una imagen que muestra la estructura del brazo robótico y donde se encuentran sus ejes de acción, los cuales se van a intervenir para mover los eslabones del prototipo.

En la Figura 34 se puede observar la interfaz de usuario diseñada:

Figura 344

Interfaz de usuario diseñada en MATLAB

Fuente: Autor

Es importante mencionar que los valores que se envíen al robot a través de la interfaz están definidos en grados, debido a que hubo que establecer una relación entre valores arbitrarios que se debían ingresar para que el robot moviera sus articulaciones considerablemente y los grados que se movían las junturas cuando se asignaban estos valores, por tanto, no había un parámetro válido o justificado para realizar tal acción. Por ello se estableció una relación ingresando valores progresivamente e ir tabulando los grados obtenidos de las mediciones realizadas manualmente. Este proceso se detalla a continuación:

Figura 355

Relación entre los valores ingresados y los ángulos generados

Fuente: Autor

Como se puede observar en la Figura 34, se establece una relación lineal entre las variables, lo que permite la aplicación de la fórmula de la pendiente para hallar el valor de x que en este caso sería el que representa la entrada para cada una de las junturas. Este proceso es igual para las articulaciones 2 y 3 del sistema robótico.

$$y = mx + b \tag{32}$$

Despejando x

$$x = \frac{y-b}{m} \tag{33}$$

Pero como se sabe que la recta interceptará al eje y en el origen, entonces se asume que b=0, por lo que se obtiene

$$x = \frac{y}{m} \tag{34}$$

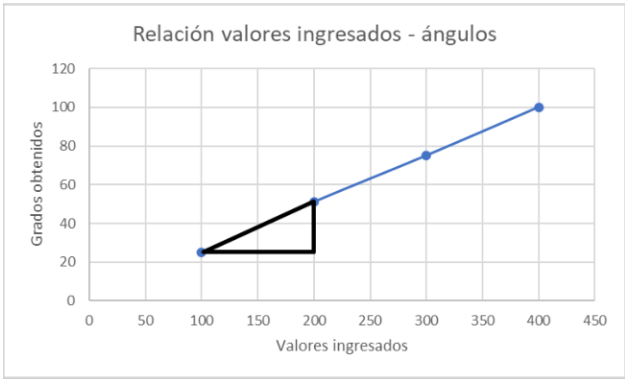
Y como se conoce que

$$m = \frac{y_2-y_1}{x_2-x_1} \tag{35}$$

Ahora se toma un ejemplo en la gráfica obtenida para calcular el valor de la pendiente.

Figura 366

Selección de puntos para cálculo de pendiente



Fuente: Autor

Reemplazando los valores en la Ecuación 34 se tiene:

$$m = \frac{20-10}{200-100} = 0{,}1 \tag{36}$$

Por tanto

$$y = 0{,}1x \tag{37}$$

$$x = 100y \tag{38}$$

De esta forma se transforman los valores que se debían ingresar inicialmente a valores angulares representados en grados que asumirán cada una de las articulaciones del brazo robótico.

3. Resultados y discusión

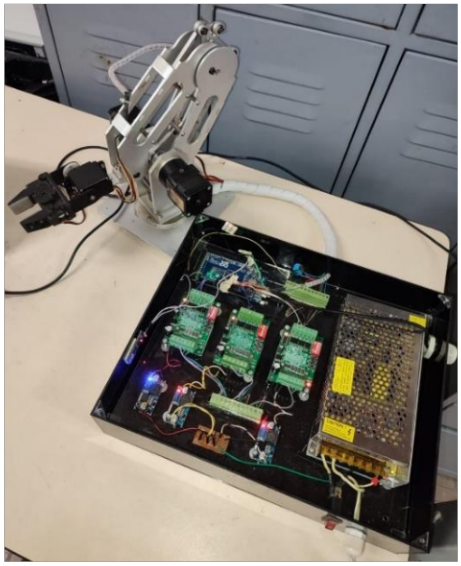
Luego de haber realizado las actividades estipuladas en cada una de las etapas propuestas en el proceso de reestructuración del sistema de control del brazo robótico perteneciente al laboratorio de Electrónica de la Universidad Pontificia Bolivariana, se presentarán los resultados obtenidos de cada una de ellas, justificando la razón de estos.

Inicialmente se realizó un levantamiento de información del brazo robótico propuesto, verificando el estado en que se encontraba cada uno de los elementos y conexiones que conforman el robot, realizando diversas pruebas alimentando el sistema y probando los motores con códigos genéricos a ver si respondían a

instrucciones dadas, de lo cual se pudo comprobar que todo se encontraba en correcto estado, a falta del sistema de control.

Figura 37

Comprobación de estado del brazo robótico



Fuente: Autor

3.1 Brazo Robótico con Interfaz de usuario

Finalmente se obtiene el brazo robótico físico controlado mediante la implementación del sistema de control diseñado y la interfaz de usuario. En el cual se han realizado diversas pruebas con el fin de garantizar el cumplimiento de parámetros de movimiento establecidos para sus junturas.

Actualmente la posición en reposo del brazo robótico es la siguiente.

Figura 38

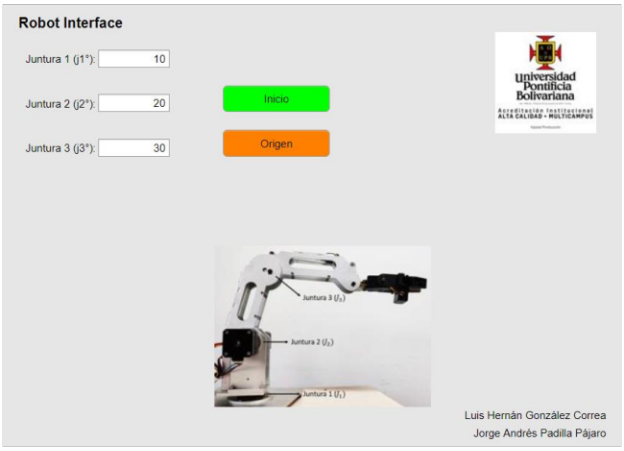
Posición en reposo del brazo robótico

Fuente: Autor

Para la primera prueba realizada se eligen los ángulos (10,20,30) para las junturas correspondientes, es importante resaltar que los valores angulares ingresados corresponden a las variables (j1,j2,j3) respectivamente y se ingresan a través de la interfaz de usuario diseñada, obteniendo lo siguiente:

Figura 39

Valores angulares (10,20,30) ingresados en la interfaz de usuario



Fuente: Autor

Figura 40

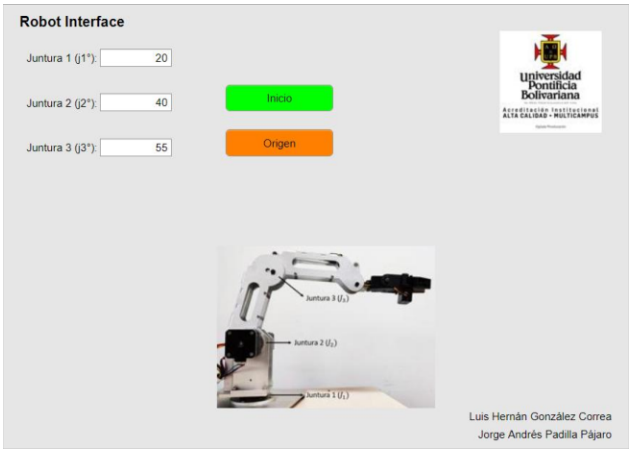
Brazo robótico con ángulos (10,20,30) en sus junturas

Fuente: Autor

Luego se realiza otra prueba partiendo de la posición que tiene luego de haber ingresado el primer parámetro. Para este caso se ingresarán los ángulos (20,40,55), obteniendo lo siguiente.

Figura 41

Valores angulares (20,40,55) ingresados en la interfaz de usuario



Fuente: Autor

Figura 42

Brazo robótico con ángulos (20,40,55) en sus junturas

Fuente: Autor

Una vez el brazo robótico se encuentra en esa posición si se desea regresarlo a su posición cero, se le puede ingresar a través de las casillas de valores angulares o presionar el botón denominado “Origen”, el cual está diseñado para esta función.

Figura 43

Posición del brazo robótico al pulsar el botón "Origen"

Fuente: Autor

Es importante mencionar que todos los movimientos que ha realizado el brazo robótico han sido verificados manualmente mediante uso de un transportador como herramienta para comprobar que en realizad la juntura se halla movido los grados correspondientes al valor ingresado por el usuario. De este procedimiento se ha obtenido que los ángulos medidos físicamente coinciden con los valores ingresados mediante la interfaz gráfica.

Conclusiones y recomendaciones

En cuanto a lo abordado con anterioridad es posible concluir que se obtuvo un sistema de control que permite el movimiento del brazo robótico a partir de valores angulares en sus junturas, validando físicamente el cumplimiento de dichos parámetros.

En el Trabajo de grado presente se ha realizado un algoritmo de control en Arduino, el cual permite establecer parámetros de posicionamiento del brazo robótico, mediante el movimiento de sus junturas que a su vez serán accionadas a través de instrucciones angulares. Además, se ha diseñado un programa en MATLAB para el análisis de estos modelos matemáticos que rigen el movimiento del robot y una interfaz de usuario en MATLAB que permite el control del prototipo, la cual es la encargada de establecer la comunicación del usuario con el prototipo.

Finalmente se puede decir que este proceso ha generado grandes aprendizajes tanto académicamente como en lo personal, ya que ha sido necesaria una amplia investigación acerca de procedimientos para realizar un sistema de control para un brazo robótico y se espera que este prototipo pueda ser utilizado como herramienta de investigación y enseñanza para el área de la robótica en el plan académico del programa.

Bibliografía

Albornoz, M. C., Berón, M., & Montejano, G. A. (2017, September). Interfaz gráfica de usuario: el usuario como protagonista del diseño. In XIX Workshop de Investigadores en Ciencias de la Computación (WICC 2017, ITBA, Buenos Aires).

Arduino Mega 2560 el hermano mayor de Arduino UNO. (2020, June 9). Programarfacil.com. https://programarfacil.com/blog/arduino-blog/arduino-mega-2560/

Benne. (2019, August 22). TB6560 Stepper Motor Driver con Arduino Tutorial (2 Ejemplos). Makerguides.com. https://www.makerguides.com/es/tb6560-stepper-motor-driver-arduino-tutorial/

Borja Conde, J. A. (2018). Desarrollo de un robot autoequilibrado basado en Arduino con motores paso a paso. Depósito de Investigación Universidad de Sevilla. https://idus.us.es/bitstream/handle/11441/84332/TFG_JoseABorjaConde.pdf?sequence=3&isAllowed=y

Caballero Durán, D. (2022). Desarrollo de sistema de control cinemático para el BCN3D Moveo. Universidad de los Andes.

Carrillo Carrasco, P. (2018). Implementación de módulo de movimiento de motores paso a paso con controlador integrado.

Conejo Benitez, C. E. (2021, March). MODELADO Y CONTROL DE UN BRAZO ROBÓTICO DE 3 GRADOS DE LIBERTAD

Corbacho, A. G. (2014). Desarrollo de una interfaz para el control del robot irb120 desde matlab. Trabajo Fin de Grado.

Duque Betancur, J. A., Hernández Orozco, J. F., & Vargas Ramírez, J. C. (2015). Módulo de servomotor PLC por pulsos.

E.Vegas. (2020, December 23). Placas de Arduino. Draco-Robotic. https://draco-robotic.com/placa-de-desarrollo-arduino/

Galindo Mejía, M. F., & Rodríguez Peláez, J. A. (2019). Diseño e implementación de un prototipo a escala que permita el seguimiento del punto de máxima potencia de un sistema de generación solar fotovoltaica. Retrieved from https://ciencia.lasalle.edu.co/ing_electrica/267

González, A. G. (2016, December 2). ¿Qué es un servomotor y cómo funciona? Panama Hitek. https://panamahitek.com/que-es-y-como-funciona-un-servomotor/ https://revistas.ucr.ac.cr/index.php/educacion/article/view/10628/10298

Jennings, S. (2002). MOTORES PASO A PASO (J. Huntley, Ed.) [Review of MOTORES PASO A PASO]. INFORMADOR TECNICO 65 2 002. https://revistas.sena.edu.co/index.php/inf_tec/article/view/899/988

Lopez Ramirez, P. A., & Andrade Sosa, H. (2013, June 30). Aprendizaje de y con robótica, algunas experiencias [Review of Aprendizaje de y con robótica, algunas experiencias]. Revista Educación.

Matlab, L. (2016). MATLAB & Simulink.

Medrano Trujillo, J. I. (2020). Brazo robot de bajo coste con motores paso a paso que escribe.

Milanés Hermosilla, D., & Castilla Pérez, A. (2016). Generación de trayectorias para el brazo robótico (ArmX). Ingeniería Electrónica, Automática y Comunicaciones, 37(3), 58-71.

Millahual, C. P. (2020). Descubriendo Arduino. RedUsers.

Millán Castrillón, J. K. (2019, August 22). SISTEMA DE MOVIMIENTOS PROGRAMADOS PARA UN BRAZO ROBOTICO MINIATURA DE 6 GRADOS DE LIBERTAD. https://red.uao.edu.co/bitstream/handle/10614/11680/T08840.pdf?sequence=5&isAllowed=y

Motores paso a paso. (2020, September 8). MOTOR PASO A PASO – TIPOS Y EJEMPLOS DEL USO DE MOTORES PASO A PASO. Electronic Components. https://www.tme.com/co/es/news/library-articles/page/41861/Motor-paso-a-paso-tipos-y-ejemplos-del-uso-de-motores-paso-a-paso/

Ortiz González, A. F., & Vintimilla Avila, R. X. (2021). Diseño e implementación de un sistema automatizado para optimizar la fase de impregnación de tinta en el proceso de serigrafía aplicado al estampado de tafiletes para sombreros de paja toquilla en la microempresa Jo & Mi Confecciones (Bachelor's thesis).

Pedroza Fernández, F. J., & Araque Mora, A. F. (2019, May 27). DESARROLLO DE UN ROBOT MANIPULADOR DE 6 GRADOS DE LIBERTAD CON TECNOLOGÍAS ABIERTAS. https://red.uao.edu.co/bitstream/handle/10614/11016/T08578.pdf

Peña, C. (2020). Introducción a Arduino (Vol. 88). RedUsers.

Ramírez Benavides, K. D. (2012, April 18). Cinemática Directa del Robot [Review of Cinemática Directa del Robot]. https://www.kramirez.net/wp-content/uploads/2012/04/CinematicaDirectaRobot.pdf

Ramos Rojas, J. A. (2018, August). DISEÑO Y CONSTRUCCIÓN DE UN MANIPULADOR ANTROPOMÓRFICO DE 4 GRADOS DE LIBERTAD PARA MANIPULACIÓN DE ALIMENTOS (I. S. Pertuz Sanchez, Ed.) https://repository.unimilitar.edu.co/bitstream/handle/10654/18040/RamosRojasJorgeAndr%C3%A9s2018.pdf?sequence=2&isAllowed=y

Rodríguez Pérez, C. A. (2013). Diseño de un robot antropomórfico de propósito general.

Trejo Peraza, J. M. (2018, January). DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO DE ROBOT CON TRES GRADOS DE LIBERTAD PARA POSICIONAMIENTO DE OBJETOS. https://www.itca.edu.sv/wp-content/uploads/2018/10/Dise%C3%B1o-y-construcci%C3%B3n-de-un-prototipo-de-robot-con-tres-grados-de-libertad-para-posicionamiento-de-objetos.pdf

330ohms. (2016, February 9). Motores a pasos... ¿unipolares o bipolares? 330ohms. https://blog.330ohms.com/2016/02/09/motores-a-pasos-unipolares-o-bipolares/

5.4 Robots industriales. (n.d.). Platea.pntic.mec.es. http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.4.htm

7. Anexos

Código en Matlab modelado cinemática inversa

```
function q=CIantropomorfico(pos)
L1=13
L2=15;
L3=16;
px=10;
py=20;
pz=30;
D=(px^2+py^2+(pz-L1)^2-L2^2-L3^2)/(2*L2*L3);
theta1=atan2(py,px);
theta3=atan2(-sqrt(1-D^2),D);
theta2=atan2(pz-L1,sqrt(px^2+py^2))-
atan2(L3*sin(theta3),L2+L3*cos(theta3));
fprintf('Las variables articulares son:')
q=[theta1*180/pi;theta2*180/pi;theta3*180/pi]
%q=[theta1;theta2;theta3]
end

%% Gráfica

clear all; clc;
l1=13;l2=15;l3=16;

L1 = Link('d', l1 , 'a', 0 , 'alpha', pi/2);
L2 = Link('d', 0 , 'a', l2 , 'alpha', 0);
L3 = Link('d', 0 , 'a', l3 , 'alpha', 0);

R=SerialLink([L1,L2,L3],'name','Robot') %unir eslabones
p=[5,5,0];
q=CIantropomorfico(p)
R.plot([q(1),q(2),q(3)])
```

Código Arduino

```
#include <AccelStepper.h>

// Define los pines para controlar los motores
#define STEP_PIN_1 9
#define DIR_PIN_1 8

#define STEP_PIN_2 13
#define DIR_PIN_2 12
```

```
#define STEP_PIN_3 11
#define DIR_PIN_3 10

// Crea tres objetos AccelStepper para controlar los motores
AccelStepper stepper1(AccelStepper::DRIVER, STEP_PIN_1,
DIR_PIN_1);
AccelStepper stepper2(AccelStepper::DRIVER, STEP_PIN_2,
DIR_PIN_2);
AccelStepper stepper3(AccelStepper::DRIVER, STEP_PIN_3,
DIR_PIN_3);

// Define la velocidad máxima y la aceleración de los motores
const float MAX_SPEED = 2000;
const float ACCELERATION = 1000;

// Define las posiciones actuales de los motores
long currentPosition1 = 0;
long currentPosition2 = 0;
long currentPosition3 = 0;

// Define las relaciones entre los valores ingresados y los grados
de movimiento para cada motor
const float ANGLE_PER_STEP_1 = 0.25; // 360° / 1000
const float ANGLE_PER_STEP_2 = 0.25; // 360° / 2000
const float ANGLE_PER_STEP_3 = 0.25; // 25° / 100

void setup() {
  // Inicializa la comunicación serie
  Serial.begin(9600);

  // Configura los motores
  stepper1.setMaxSpeed(MAX_SPEED);
  stepper1.setAcceleration(ACCELERATION);
  stepper2.setMaxSpeed(MAX_SPEED);
  stepper2.setAcceleration(ACCELERATION);
  stepper3.setMaxSpeed(MAX_SPEED);
  stepper3.setAcceleration(ACCELERATION);

  // Establece las posiciones iniciales de los motores
  stepper1.setCurrentPosition(currentPosition1);
  stepper2.setCurrentPosition(currentPosition2);
  stepper3.setCurrentPosition(currentPosition3);
}

void loop() {
  // Espera a que llegue un comando a través del monitor serie
  while (Serial.available() == 0)
    ;
```

```
  // Lee el comando
  String command = Serial.readStringUntil('\n');

  // Si el comando es "ORIGEN", establece las posiciones actuales
como el origen
  if (command == "ORIGEN") {
    currentPosition1 = 0;
    stepper1.setCurrentPosition(currentPosition1);
    currentPosition2 = 0;
    stepper2.setCurrentPosition(currentPosition2);
    currentPosition3 = 0;
    stepper3.setCurrentPosition(currentPosition3);
  }

  // Si el comando es un número, mueve los motores a esas
posiciones
  else if (command.indexOf(",") != -1) {
    int firstCommaIndex = command.indexOf(",");
    int secondCommaIndex = command.indexOf(",", firstCommaIndex +
1);
    long targetPosition1 = -abs(command.substring(0,
firstCommaIndex).toInt());
    long targetPosition2 = -abs(command.substring(firstCommaIndex
+ 1, secondCommaIndex).toInt());
    long targetPosition3 = -abs(command.substring(secondCommaIndex
+ 1).toInt());

    // Calcula el número de pasos para cada motor basado en los
valores de ángulo ingresados
    long steps1 = targetPosition1 * (1.0 / ANGLE_PER_STEP_1);
    long steps2 = targetPosition2 * (1.0 / ANGLE_PER_STEP_2);
    long steps3 = targetPosition3 * (1.0 / ANGLE_PER_STEP_3);

    stepper1.moveTo(steps1);
    stepper2.moveTo(steps2);
    stepper3.moveTo(steps3);
    stepper1.runToPosition();
    stepper2.runToPosition();
    stepper3.runToPosition();
    currentPosition1 = targetPosition1;
    currentPosition2 = targetPosition2;
    currentPosition3 = targetPosition3;

    // Calcula los ángulos correspondientes a los valores
ingresados para cada motor
    float angle1 = targetPosition1 * ANGLE_PER_STEP_1;
    float angle2 = targetPosition2 * ANGLE_PER_STEP_2;
    float angle3 = targetPosition3 * ANGLE_PER_STEP_3;
```

```
    Serial.print("Ángulo 1: ");
    Serial.print(angle1);
    Serial.println(" grados");
    Serial.print("Ángulo 2: ");
    Serial.print(angle2);
    Serial.println(" grados");
    Serial.print("Ángulo 3: ");
    Serial.print(angle3);
    Serial.println(" grados");
  }

  // Imprime las posiciones actuales de los motores
  Serial.print("Posición actual 1: ");
  Serial.println(currentPosition1);
  Serial.print("Posición actual 2: ");
  Serial.println(currentPosition2);
  Serial.print("Posición actual 3: ");
  Serial.println(currentPosition3);
}
```

Universidad Pontificia Bolivariana
SECCIONAL MONTERÍA
Vigilada Mineducación

CIDI · UPB
CENTRO DE INVESTIGACIÓN PARA EL DESARROLLO Y LA INNOVACIÓN

IMPLEMENTACIÓN DE UN SISTEMA DE ACCIONAMIENTO EN UN BRAZO ROBÓTICO BASADO EN CINEMÁTICA INVERSA DE CUATRO GRADOS DE LIBERTAD OPERADO CON INTERFAZ MANUAL

Luis Hernán González Correa, Universidad Pontificia Bolivariana, luis.gonzalezc@upb.edu.co
Jorge Andrés Padilla Pájaro, Universidad Pontificia Bolivariana, jorge.padilla@upb.edu.co

RESUMEN: El objetivo principal del trabajo de grado fue desarrollar un sistema de control para un prototipo robótico en el laboratorio de Electrónica de la Universidad Pontificia Bolivariana Seccional Montería para fortalecer la enseñanza de la robótica en la facultad. Para lograr esto, se diseñó e implementó un mecanismo de control que utiliza valores angulares en grados en las articulaciones del robot como parámetro de entrada y a partir de estos establecer nuevos movimientos para el sistema verificando físicamente que se cumpla con el valor ingresado para cada juntura.

Se diseñó un algoritmo de control en Arduino, incluyendo AccelStepper, la cual es una librería especializada en el control de motores. Además, se definieron parámetros de accionamiento para los motores, los cuales dependiendo del comando que se ingrese realizarán un proceso para mover el brazo robótico. El control de este dispositivo fue realizado mediante grados en junturas del prototipo, requiriendo así valores angulares para cada una de las articulaciones. También se desarrolló una interfaz de usuario en MATLAB que permite la comunicación con el hardware para enviar los valores de entrada y permitir controlar el robot.

La metodología empleada en el trabajo incluyó el análisis de requerimientos, el diseño de software, la implementación y la validación del sistema. Como resultado, se obtuvo un brazo robótico con un sistema de control que puede posicionarse en diferentes puntos mediante valores angulares en grados para sus articulaciones.

PALABRAS CLAVES: Robótica, sistemas de control, interfaz de usuario.

ABSTRACT: The main objective of the thesis project was to improve a robotic prototype in the Electronics laboratory at the Pontifical Bolivarian University, Monteria Campus, to enhance robotics education within the faculty. To achieve this, a control system was designed and implemented, which utilizes angular values in degrees at the robot's joints as an input parameter, and based on these values, establishes new movements for the system, physically verifying that the entered value is met for each joint.

A control algorithm was designed using Arduino, including AccelStepper, which is a specialized library for motor control. In addition, drive parameters for the motors were defined, which, depending on the entered command, will initiate a process to move the robotic arm. Control of this device was performed using degrees in the prototype's joints, requiring angular values for each of the articulations. A user interface was also developed in MATLAB, enabling communication with the hardware to send input values and control the robot.

The methodology employed in this project included requirements analysis, software design, implementation, and system validation. As a result, a robotic arm with a control system was obtained, capable of positioning itself at different points using angular values in degrees for its joints.

KEYWORDS: Robotics, control systems, user interface.

INTRODUCCIÓN. El avance continuo de la tecnología en diversos ámbitos, como la industria, la educación y la investigación, ha llevado al desarrollo de elementos automatizados que desempeñan funciones importantes. Estos avances incluyen actuadores industriales, computadoras electrónicas, transmisión de energía mediante mecanismos y engranajes, así como sensores y transductores. A partir de 1950, se empezaron a crear los primeros robots debido a estas investigaciones en inteligencia artificial, que permitieron poner a prueba teorías que anteriormente solo se podían comprobar en humanos, ahora aplicadas a computadoras y dispositivos electrónicos.

Las industrias hoy día se centran en la automatización de una gran parte de sus operaciones, lo que les permite minimizar los posibles peligros para la seguridad del personal y también reducir los costos asociados con la contratación de una gran cantidad de trabajadores para realizar tareas industriales (Medrano Trujillo, 2020). Para lograr esto, utilizan dispositivos automatizados como cintas transportadoras, grúas industriales, robots manipuladores y otros, que se encargan de realizar actividades peligrosas. En esta situación, los robots manipuladores desempeñan diversas tareas, como el transporte de objetos, seguimiento de tareas, trazos y vigilancia, entre otras. Un manipulador robótico, conocido también como brazo robótico antropomórfico debido a su similitud con un brazo humano, se basa en el principio de articular segmentos de su estructura para lograr la flexibilidad necesaria y así poder mover elementos dentro de un espacio geométrico determinado (Millán Castrillón, J. K, 2019).

La robótica tiene diversas aplicaciones en la actualidad y es objeto de estudio en centros de investigación y universidades. Se desarrollan proyectos robóticos para promover el aprendizaje práctico del control en esta disciplina. Para ello, se requiere disponer de equipos electrónicos que permitan poner en práctica las técnicas de la robótica y validar los modelos matemáticos establecidos. El Laboratorio de Ingenierías y Arquitectura de la Universidad Pontificia Bolivariana cuenta actualmente con equipos que permiten la aplicación de teorías en algunos cursos del plan educativo. En el programa de Ingeniería Electrónica, se dispone de equipos para realizar prácticas en el área de automatización y control, como tableros con PLC integrados, el software de simulación Matlab, suministros de energía y un prototipo robótico propuesto por Heller (2017), que funcionaba correctamente en ese momento y cumplía con los parámetros establecidos.
Sin embargo, debido al desuso del equipo durante aproximadamente 2 años debido a la pandemia y la falta de documentación del brazo robótico, este ya no funcionaba correctamente. Por lo tanto, se propuso implementar un nuevo sistema de control para el brazo, basado en la estructura del prototipo, con el objetivo de controlar sus 4 grados de libertad a través de una interfaz de usuario.

En este trabajo de grado, se llevó a cabo el proceso de reestructuración del sistema de control del brazo robótico, siguiendo una metodología que abarca desde métodos matemáticos hasta el diseño de un software y una interfaz de usuario. Esta interfaz permite al usuario ingresar ángulos que definirán la posición final del robot. Además, este dispositivo se ofrece como una herramienta para la enseñanza teórica y práctica de la robótica en el plan académico de la facultad de Ingeniería Electrónica, aportando nuevas técnicas de control que podrán aplicarse en futuros proyectos.

METODOLOGÍA.

El desarrollo del proyecto constó de varias etapas cruciales para garantizar el éxito en el resultado final. Entre estas etapas se incluye la selección del software, la documentación del modelado cinemático del brazo robótico, la implementación del sistema de control y, finalmente, una serie de pruebas realizadas. A

continuación, se describen con más detalle las etapas del desarrollo del proyecto:

Selección de software

Para implementar un sistema de control en el brazo robótico es fundamental conocer al detalle la estructura física de este, comprendiendo los elementos que lo conforman, funciones y conexiones entre estos.

Luego de haber realizado el levantamiento de información del dispositivo y analizado su composición se ha elegido el software Arduino para desarrollar un mecanismo de control que pueda definir los movimientos del brazo robótico.

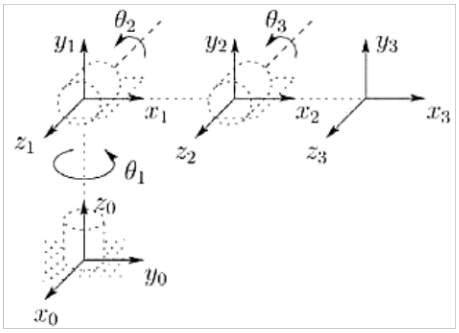
Una vez seleccionado el software, se procedió a establecer la comunicación entre el brazo robótico y el software de control. Esta etapa implicó la configuración de las conexiones físicas necesarias para permitir la comunicación entre el robot y el software. Además de la configuración física, también es necesario establecer los protocolos de comunicación adecuados. Esto implica definir cómo el robot y el software de control intercambiarán información y comandos.

Solución modelo cinemático del brazo robótico mediante parámetros D-H

Para emplear el método D-H en la solución de la cinemática inversa del brazo robótico se hace necesario el planteamiento en sistema de coordenadas de la siguiente forma:

Figura 1

Representación del robot en sistemas de coordenadas según algoritmo D-H



Nota: Así es la representación en sistema de coordenadas del brazo robótico teniendo en cuenta el método D-H. Tomado de Representación del robot en sistemas de coordenadas según algoritmo de Denavit-Hartenberg [Imagen] Milanés Hermosilla, D., & Castilla Pérez, 2016, http://scielo.sld.cu/scielo.php?pid=S1815-59282016000300006&script=sci_arttext&tlng=pt

Al emplear los sistemas de coordenadas, se adquieren los valores que indicarán la rotación de cada una de las articulaciones del brazo robótico. Los datos correspondientes se encuentran detallados en la Tabla 1.

Tabla 1

Parámetros D-H del brazo robótico

Eslabón	θ_i	a_i	α_i	d_i
1	θ_1	0	$\pi/2$	d_1
2	θ_2	a_2	0	0
3	θ_3	a_3	0	0
4	θ_4	a_4	0	0

Fuente: Autor

Tomando en cuenta los parámetros obtenidos a través del método D-H y resolviendo la matriz de transformación homogénea se obtuvieron las siguientes ecuaciones para las posiciones x,y,z.

$$p_x = sin\theta_1(a_3\, cos(\theta_2 + \theta_3) + a_2 cos\theta_2) \quad (1)$$

$$p_y = sin\theta_1(a_3\, cos(\theta_2 + \theta_3) + a_2 cos\theta_2) \quad (2)$$

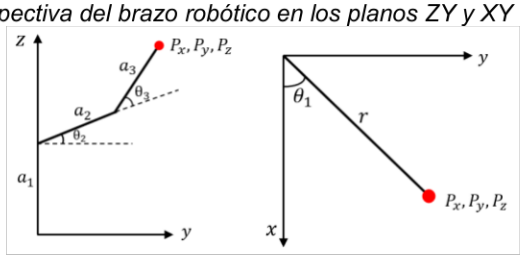
$$p_z = d_1 + a_3\, sin(\theta_2 + \theta_3) + a_2 sin\theta_2 \quad (3)$$

Solución del modelado cinemático inverso

Para obtener el modelo cinemático inverso de este primer modelo, se siguió empleando la trigonometría. Aprovechando la posición supuesta del brazo en el modelo cinemático inverso y los diferentes triángulos que se pueden formar a partir de esa posición, se calcularon los diferentes ángulos.

Figura 2

Perspectiva del brazo robótico en los planos ZY y XY



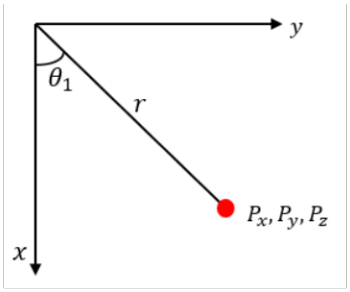
Fuente: Autor

En la dimensión ZY se logra observar cómo se comportan los ángulos θ_2 y θ_3, mientras que en el plano XY se aclara que el brazo no se encuentra dentro del plano ZY, sino que tiene un ángulo θ_1 que lo posiciona entre los ejes X e Y.

Solución de θ_1

Figura 3

Perspectiva del brazo robótico en el plano XY



Fuente: Autor

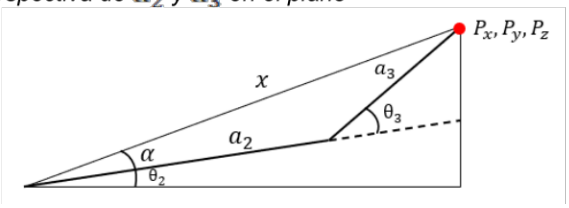
El ángulo θ_1 se resuelve utilizando el plano XY como referencia.

$$\theta_1 = \text{atan}\left(\frac{Py}{Px}\right) \quad (4)$$

Solución de θ_2 y θ_3

Figura 4

Perspectiva de a_2 y a_3 en el plano



Fuente: Autor

Como se puede apreciar la imagen muestra cuatro triángulos en total, los cuales, al combinarse entre sí, permiten obtener todos los parámetros deseados.

Se utiliza el teorema del coseno para resolver el triángulo compuesto por a_2, a_3 y x, para así obtener θ_3.

$$\theta_3 = cos^{-1}\left(\frac{P_x^2 + P_y^2 + P_z^2 - a_2^2 - a_3^2}{2 * a_2^2 * a_3^2}\right) \quad (5)$$

Para la obtención de θ_2 se agrega un nuevo parámetro β, el cual representa la suma del ángulo θ_2 y α.

$$\beta = \theta_2 + \alpha \quad (6)$$

$$\theta_2 = atan\left(\frac{P_z}{\pm\sqrt{P_x^2 + P_y^2}}\right) - atan\left(\frac{a_{3*sin\theta_3}}{a_2 + a_3 * cos\theta_3}\right) \quad (7)$$

Sistema de control

El sistema de control del brazo robótico es una parte fundamental en el funcionamiento y la operación del robot. Este sistema es responsable de coordinar y controlar los

movimientos de los diferentes actuadores del brazo, permitiendo que el robot realice tareas específicas de manera precisa y eficiente.

Modelamiento en Matlab

Contar con un proceso matemático para modelar un sistema real puede brindar ventajas al momento de observar y analizar el comportamiento de un robot. Asimismo, ofrece la posibilidad de realizar pruebas de funcionamiento y llevar a cabo experimentos controlados. En el caso específico del modelado cinemático de un brazo robótico, el objetivo primordial radica en comprender en profundidad el movimiento de este mecanismo, lo cual resulta fundamental para la predicción y planificación precisa de las trayectorias del sistema. Al desarrollar este modelo se puede optimizar el rendimiento y la eficiencia del brazo robótico.

En este caso se utilizó el software MATLAB y su herramienta Toolbox, del profesor Peter Corke para desarrollar una Interfaz Gráfica de Usuario (GUI), con el fin de aplicar la teoría de Denavit-Hartenberg y simplificar la interacción y el control del sistema, permitiendo programar movimientos, supervisar el funcionamiento en tiempo real y ajustar la configuración de manera más accesible.

Modelamiento cinemático

El modelamiento cinemático se enfoca en analizar la geometría y las relaciones de posición y orientación de sus articulaciones e implica la determinación de las ecuaciones matemáticas que describen la cinemática directa e inversa del sistema.
En este caso se ha utilizado el Toolbox de MATLAB, el cual es apropiado para este proceso.

Figura 5

Parámetros Denavit-Hartenberg en MATLAB

```
>> l1=13;
l2=15;
l3=16;

L1 = Link('d', l1 , 'a', 0 , 'alpha', pi/2);
L2 = Link('d', 0 , 'a', l2 , 'alpha', 0);
L3 = Link('d', 0 , 'a', l3 , 'alpha', 0);

R=SerialLink([L1,L2,L3])

R =

noname:: 3 axis, RRR, stdDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|         13|          0|     1.5708|          0|
|  2|         q2|          0|         15|          0|          0|
|  3|         q3|          0|         16|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

Fuente: Autor

En la Figura 5 se denotan los parámetros de los eslabones del brazo robot siguiendo el método D-H, lo que permite la obtención del modelado cinemático directo e inverso del robot. Haciendo uso del script de MATLAB nuevamente, se procede a definir las ecuaciones de cinemática inversa, que permiten obtener los ángulos que deben adoptar las articulaciones para posicionar el extremo del brazo robótico en el punto predeterminado.

Figura 6

Obtención de variables articulares para la posición (10,20,30) mediante Cinemática Inversa

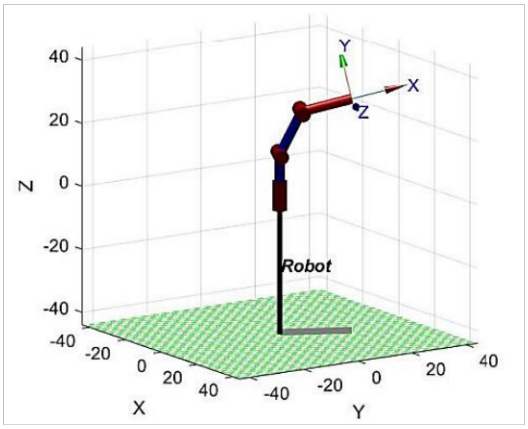
```
>> L1=13;
L2=15;
L3=16;
px=10;
py=20;
pz=30;
D=(px^2+py^2+(pz-L1)^2-L2^2-L3^2)/(2*L2*L3);
theta1=atan2(py,px);
theta3=atan2(-sqrt(1-D^2),D);
theta2=atan2(pz-L1,sqrt(px^2+py^2))-atan2(L3*sin(theta3),L2+L3*cos(theta3));
fprintf('Las variables articulares son:')
q=[theta1*180/pi;theta2*180/pi;theta3*180/pi]

Las variables articulares son:
q =

   63.4349
   63.1497
  -50.0838
```

Fuente: Autor

Así mismo, se realiza una simulación del robot para observar su ubicación luego de conocer estas variables.

Figura 7

Simulación del brazo robótico en MATLAB en la posición (10,20,30)



Fuente: Autor

Es importante aclarar que este modelado solo se hará a nivel de simulación y se dejará propuesto como objeto de estudio, debido que hubo que cambiar la forma de controlar el brazo robótico, tomando los grados de ángulos en cada juntura como parámetro a intervenir, debido que no había relación entre las posiciones ingresadas mediante este método con los movimientos que realizaba el prototipo.

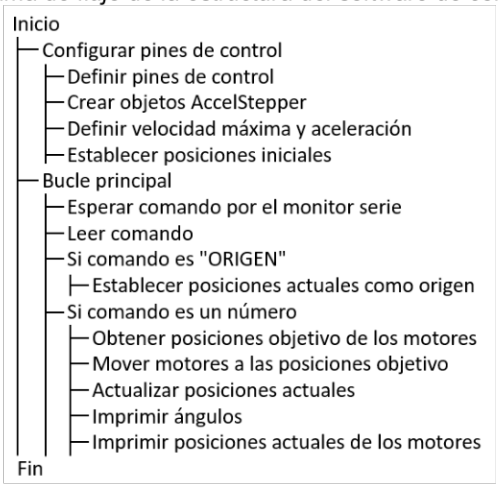
Algoritmo de control

El software de control aplicado a un brazo robótico es una pieza fundamental para su funcionamiento y la operación eficiente. Este tipo de software se encarga de coordinar y controlar los movimientos del brazo robótico, permitiendo ejecutar una amplia variedad de tareas de forma precisa y autónoma.

A continuación, se presentará un diagrama de flujo mediante el cual se mencionarán los procesos más relevantes del software diseñado para el sistema de control del brazo robótico, el cual fue desarrollado en Arduino.

Figura 8

Diagrama de flujo de la estructura del software de control



Fuente: Autor

El programa comienza con la configuración de los pines de control para los motores. Se definen los pines de paso y dirección para cada motor, y se crean los objetos AccelStepper correspondientes a cada motor. Además, se establece la velocidad máxima y la aceleración para los motores, y se inicializan las posiciones actuales de los motores.

Luego, el programa entra en un bucle principal donde espera la recepción de comandos a través del monitor serie. Una vez que un comando está disponible, se lee y se realiza una serie de acciones dependiendo del tipo de comando.

Si el comando es "ORIGEN", el programa establece las posiciones actuales de los motores como el origen (posición cero) para reiniciar el sistema.

Si el comando es un número, se interpreta como un ángulo para los motores. Luego, los motores se mueven teniendo en cuenta los ángulos ingresados utilizando la función moveTo() de la librería AccelStepper, y se espera a que los motores alcancen dichas

posiciones utilizando la función runToPosition(). Después de mover los motores, se actualizan las posiciones actuales.

Después de realizar las acciones según el comando recibido, se imprime la posición actual de cada motor por el monitor serie.

El programa vuelve al inicio del bucle principal y espera la llegada de nuevos comandos.

Así, el diagrama de flujo muestra la estructura general del programa de control, facilitando la comprensión del proceso y las acciones realizadas en cada etapa.

Es importante decir que se tuvo que ajustar los parámetros de control del brazo robótico, debido que no se pudo realizar a través de modelado cinemático inverso, sino que se realizó mediante grados en juntura, lo que significa que se deben ingresar valores angulares para sus articulaciones moviéndose cada una de estas la cantidad de ángulos correspondientes a la cantidad requerida por el usuario. Además, cabe aclarar que el cálculo del modelo cinemático realizado se dejará propuesto a nivel de simulación ofrecido como objeto de estudio.

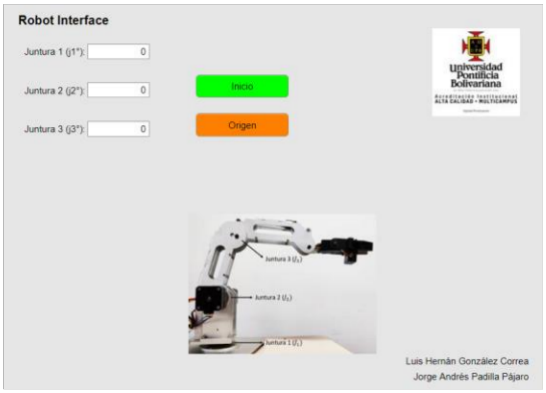
Desarrollo de Interfaz Gráfica

Luego de haber diseñado el software de control para el brazo robótico se procedió a la elaboración de la interfaz gráfica usando la herramienta MATLAB GUI, mediante la cual se mostrará una serie de parámetros y se establecerá la comunicación con el robot desde un ordenador.

En la interfaz se incluyó unas casillas donde se introducirán los valores angulares de las junturas del brazo robot y dos botones, uno para enviar los valores ingresados y otro que al oprimirse regresa al robot a su posición original (posición cero). Además, cuenta con una imagen que muestra la estructura del brazo robótico y donde se encuentran sus ejes de acción, los cuales se van a intervenir para mover los eslabones del prototipo.

Figura 9

Interfaz de usuario diseñada en MATLAB



Fuente: Autor

Es importante mencionar que los valores que se envíen al robot a través de la interfaz están definidos en grados, debido a que hubo que establecer una relación entre valores arbitrarios que se debían ingresar para que el robot moviera sus articulaciones considerablemente y los grados que se movían las junturas cuando se asignaban estos valores, por tanto, no había un parámetro válido o justificado para realizar tal acción.

CONTENIDO

El objetivo principal del trabajo de grado fue desarrollar un sistema de control para el brazo robótico, teniendo en cuenta la composición física de este.

Inicialmente se realizó un levantamiento de información acerca de este prototipo, con el fin de saber el estado de sus componentes y así proceder a realizar cualquier tipo de intervención en este. Una vez se tuvo la certeza de que todos los elementos se encontraban en funcionamiento normal se hizo el modelado cinemático directo e inverso del robot mediante el método D-H, con el fin de establecer mediante un proceso matemático la posición y

restricciones que tiene el prototipo para poder proseguir a establecer algún parámetro de movimiento. Aunque este paso anterior se dejó propuesto como objeto de estudio, debido que se cambiaron los parámetros de entrada al sistema, es decir, no se ingresarán coordenadas, sino ángulos directamente para cada una de las junturas del robot. Luego se eligió el software Arduino para desarrollar el sistema de control y establecer los parámetros que determinan el movimiento del robot y se realizó la comunicación entre este y el brazo robótico, para proceder a implementar el algoritmo diseñado en el dispositivo e ir comprobando el funcionamiento de este con los valores ingresados y establecer relaciones entre dichos datos y valores obtenidos en el robot físico de forma manual.

Además de esto se diseñó una interfaz de usuario en MATLAB GUI, en la cual se configuraron unas casillas que permitan al usuario ingresar los valores angulares que desee para las articulaciones del dispositivo; también cuenta con un botón de inicio que será el encargado de enviar los valores ingresados al sistema para que sean leídos y el brazo robótico los pueda adoptar.

Finalmente se realizaron pruebas de posicionamiento del prototipo robótico con el propósito de comprobar la veracidad del sistema de control implementado, obteniendo como resultado una buena relación entre los valores ingresados con los obtenidos de las mediciones realizadas manualmente.

RESULTADOS

Luego de haber realizado las actividades estipuladas en cada una de las etapas propuestas en el proceso de reestructuración del sistema de control del brazo robótico perteneciente al laboratorio de Electrónica de la Universidad Pontificia Bolivariana, se presentarán los resultados obtenidos de cada una de ellas, justificando la razón de estos.

Inicialmente se realizó un levantamiento de información del brazo robótico propuesto, verificando el estado en que se encontraba cada uno de los elementos y conexiones que conforman el robot, realizando diversas pruebas alimentando el sistema y probando los motores con códigos genéricos a ver si respondían a instrucciones dadas, de lo cual se pudo comprobar que todo se encontraba en correcto estado, a falta del sistema de control.

Figura 10

Comprobación de estado del brazo robótico

Fuente: Autor

Brazo Robótico con Interfaz de usuario

Finalmente se obtiene el brazo robótico físico controlado mediante la implementación del sistema de control diseñado y la interfaz de usuario. En el cual se han realizado diversas pruebas con el fin de garantizar el cumplimiento de parámetros de movimiento establecidos para sus junturas.

Actualmente la posición en reposo del brazo robótico es la siguiente.

Figura 11

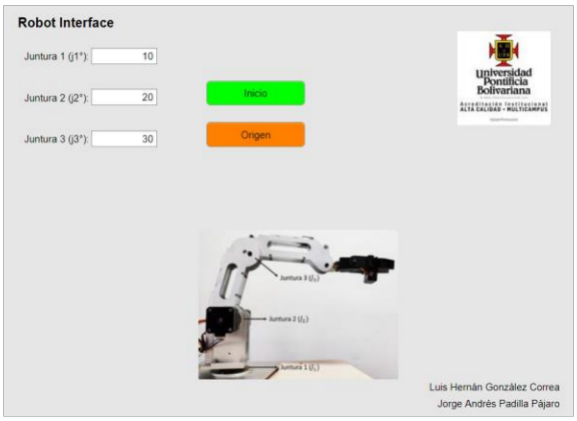
Posición en reposo del brazo robótico

Fuente: Autor

Para la primera prueba realizada se eligen los ángulos (10,20,30) para las junturas correspondientes, es importante resaltar que los valores angulares ingresados corresponden a las variables (j1,j2,j3) respectivamente y se ingresan a través de la interfaz de usuario diseñada, obteniendo lo siguiente:

Figura 12

Valores angulares (10,20,30) ingresados en la interfaz de usuario



Fuente: Autor

Figura 13

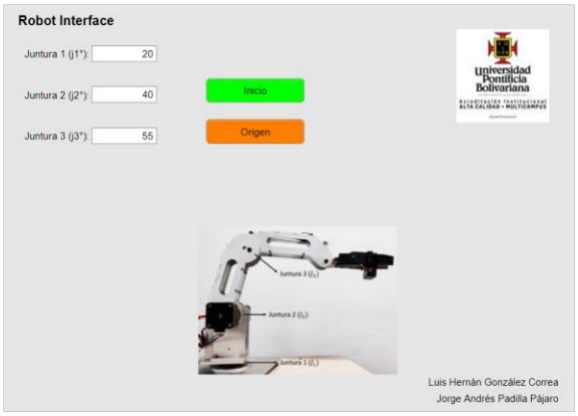
Brazo robótico con ángulos (10,20,30) en sus junturas

Fuente: Autor

Luego se realiza otra prueba partiendo de la posición que tiene luego de haber ingresado el primer parámetro. Para este caso se ingresarán los ángulos (20,40,55), obteniendo lo siguiente.

Figura 14

Valores angulares (20,40,55) ingresados en la interfaz de usuario



Fuente: Autor

Figura 15

Brazo robótico con ángulos (20,40,55) en sus junturas

Fuente: Autor

Una vez el brazo robótico se encuentra en esa posición si se desea regresarlo a su posición cero, se le puede ingresar a través de las casillas de valores angulares o presionar el botón denominado "Origen", el cual está diseñado para esta función.

Figura 16

Posición del brazo robótico al pulsar el botón "Origen"

Fuente: Autor

Es importante mencionar que todos los movimientos que ha realizado el brazo robótico han sido verificados manualmente mediante uso de un transportador como herramienta para comprobar que en realizad la juntura se halla movido los grados correspondientes al valor ingresado por el usuario. De este procedimiento se ha obtenido que los ángulos medidos físicamente coinciden con los valores ingresados mediante la interfaz gráfica.

CONCLUSIONES Y/O RECOMENDACIONES.

En cuanto a lo abordado con anterioridad es posible concluir que se obtuvo un sistema de control que permite el movimiento del brazo robótico a partir de valores angulares en sus junturas, validando físicamente el cumplimiento de dichos parámetros.

En el Trabajo de grado presente se ha realizado un algoritmo de control en Arduino, el cual permite establecer parámetros de posicionamiento del brazo robótico, mediante el movimiento de sus junturas que a su vez serán accionadas a través de instrucciones angulares. Además, se ha diseñado un programa en MATLAB para el análisis de estos modelos matemáticos que rigen el movimiento del robot y una interfaz de usuario en MATLAB que permite el control del prototipo, la cual es la encargada de establecer la comunicación del usuario con el prototipo.

Finalmente se puede decir que este proceso ha generado grandes aprendizajes tanto académicamente como en lo personal, ya que ha sido necesaria una amplia investigación acerca de procedimientos para realizar un sistema de control para un brazo robótico y se espera que este prototipo pueda ser utilizado como herramienta de investigación y enseñanza para el área de la robótica en el plan académico del programa.

BIBLIOGRAFÍA.

Albornoz, M. C., Berón, M., & Montejano, G. A. (2017, September). Interfaz gráfica de usuario: el usuario como protagonista del diseño. In XIX Workshop de Investigadores en Ciencias de la Computación (WICC 2017, ITBA, Buenos Aires).

Arduino Mega 2560 el hermano mayor de Arduino UNO. (2020, June 9). Programarfacil.com.

https://programarfacil.com/blog/arduino-blog/arduino-mega-2560/

Benne. (2019, August 22). TB6560 Stepper Motor Driver con Arduino Tutorial (2 Ejemplos). Makerguides.com. https://www.makerguides.com/es/tb6560-stepper-motor-driver-arduino-tutorial/

Borja Conde, J. A. (2018). Desarrollo de un robot autoequilibrado basado en Arduino con motores paso a paso. Depósito de Investigación Universidad de Sevilla. https://idus.us.es/bitstream/handle/11441/84332/TFG_JoseABorjaConde.pdf?sequence=3&isAllowed=y

Caballero Durán, D. (2022). Desarrollo de sistema de control cinemático para el BCN3D Moveo. Universidad de los Andes.

Carrillo Carrasco, P. (2018). Implementación de módulo de movimiento de motores paso a paso con controlador integrado.

Conejo Benitez, C. E. (2021, March). MODELADO Y CONTROL DE UN BRAZO ROBÓTICO DE 3 GRADOS DE LIBERTAD

Galindo Mejía, M. F., & Rodríguez Peláez, J. A. (2019). Diseño e implementación de un prototipo a escala que permita el seguimiento del punto de máxima potencia de un sistema de generación solar fotovoltaica. Retrieved from https://ciencia.lasalle.edu.co/ing_electrica/267

Medrano Trujillo, J. I. (2020). Brazo robot de bajo coste con motores paso a paso que escribe.

Millán Castrillón, J. K. (2019, August 22). SISTEMA DE MOVIMIENTOS PROGRAMADOS PARA UN BRAZO ROBOTICO MINIATURA DE 6 GRADOS DE LIBERTAD. https://red.uao.edu.co/bitstream/handle/10614/11680/T08840.pdf?sequence=5&isAllowed=y