




Design and Implementation of a Delta-type Parallel Robot with Artificial Vision to for object tracking

Julio César Llerena Sánchez, Br.¹, Marcelo Jaime Quispe Ccachuco, Dr.² y César Pio Castillo Cáceres, Dr.³
^{1,2,3}Universidad Católica de Santa María, Perú, 72443521@ucsm.edu.pe, mquispec@ucsm.edu.pe, ccastillo555@yahoo.es

In the industry today it is very unusual to find that a robot performs tracking operations with the help of a camera applying artificial vision, however, there are countless applications where the combination of these concepts would be fabulous, such as in the arms industry, production, selection processes, etc.




The present project consists of achieving the union of a delta-type parallel robot with an object recognition and tracking interface, which will be carried out with a camera with a resolution of 640 x 480 pixels, which will be more than enough for our application. Artificial vision will have a program that will allow us to recognize objects and discriminate them by color and size. It should be noted that, if greater precision is desired regarding the position of our object, it is possible to change the camera to one with higher resolution, since we will use the pixels as coordinate points in the camera plane. The equations that govern the kinematics of the entire system were calculated, these equations were implemented in an Arduino program, which receives data from the coordinate points in ASCII format through the serial port of the computer at a frequency of 1200 bauds. a Python program made in Visual Studio, with the aim of calculating at high speed the rotation angles of each servomotor to position our robot at the desired point.

To validate the design, simulations of the forces present in each component and the torque necessary to move the joints, the robot's work area, tracking tests, connectivity tests and data transmission speed were carried out.

Keywords—*Servomotor, Delta Robot, Computer Vision, Parallel Robot, Tracking.*

Digital Object Identifier: (only for full papers, inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).
DO NOT REMOVE

Diseño e Implementación de un Robot Paralelo tipo Delta con Visión Artificial para seguimiento de objetos

Julio César Llerena Sánchez, Br.¹, Marcelo Jaime Quispe Ccachuco, Dr.² y César Pio Castillo Cáceres, Dr.³
^{1,2,3}Universidad Católica de Santa María, Perú, 72443521@ucsm.edu.pe, mquispec@ucsm.edu.pe, ccastillo555@yahoo.es

Resumen- En la industria actualmente es muy poco usual encontrar que un robot realice operaciones de seguimiento con ayuda de una cámara aplicando visión artificial, sin embargo, existen infinidad de aplicaciones donde la combinación de estos conceptos sería fabulosa, como en la industria armamentista, de producción, procesos de selección, etc.

El presente proyecto consiste en lograr la unión de un robot paralelo tipo delta con una interfaz de reconocimiento y seguimiento de objetos, la que se realizará con una cámara con resolución de 640 x 480 píxeles, esta será más que suficiente para nuestra aplicación. La visión artificial contará con un programa que permitirá reconocer objetos y discriminarlos por color y tamaño, cabe resaltar que si se desea mayor precisión en cuanto a la posición del objeto, es posible cambiar la cámara a una de mayor resolución, ya que se utilizó los píxeles como puntos coordenados en el plano de la cámara. Se calcularon las ecuaciones que gobiernan la cinemática de todo el sistema, estas ecuaciones fueron implementadas en un programa de Arduino, el cual recibe datos de los puntos coordenados en formato ASCII a través del puerto serie de la computadora a una frecuencia de 1200 baudios de un programa en Python realizado en Visual Studio, con el objetivo de calcular a gran velocidad los ángulos de rotación de cada servomotor para posicionar al robot en el punto deseado.

Para validar el diseño, se realizaron simulaciones de esfuerzos en cada componente y el torque necesario para mover las articulaciones, el área de trabajo del robot, ensayos de seguimiento, pruebas de conectividad y velocidad de transmisión de datos.

Palabras clave - Servomotor, Robot Paralelo, Robot Delta, Visión Artificial, Seguimiento.

I. INTRODUCCIÓN

Los robots paralelos están compuestos por dos o más cadenas cinemáticas cerradas que conectan una plataforma fija y otra móvil que definen el efector final a controlar.[1] En la intersección de la robótica avanzada y la visión artificial, la integración de robots paralelos tipo delta con sistemas de seguimiento de objetos se ha convertido en un área de investigación innovadora y prometedora. Los robots paralelos tipo delta, conocidos por su alta velocidad y precisión, se han destacado en diversas aplicaciones industriales y de fabricación. Al combinar estos robots con tecnologías de visión artificial para el seguimiento de objetos, se abren nuevas posibilidades en campos como producción automatizada o paletizado de materiales o productos[2], sin embargo, a pesar de todo el esfuerzo invertido en el estudio de estos

manipuladores, hasta el día de hoy, dichas arquitecturas continúan presentando una serie de inconvenientes, por ejemplo, un espacio de trabajo reducido, destreza limitada, arquitectura compleja, un modelo cinemático directo difícil de resolver y la presencia de múltiples configuraciones singulares, y una serie de problemas que aumentan en complejidad a medida que se agregan más cadenas cinemáticas y grados de libertad al sistema mecánico.[3]

La visión artificial en este contexto implica el uso de cámaras y algoritmos avanzados para procesar la información visual y tomar decisiones en tiempo real.[4] Esta tecnología permite a los robots identificar, seguir y manipular objetos de manera autónoma, lo que resulta crucial en entornos dinámicos y cambiantes. La capacidad de discernir entre objetos según sus características, como forma, color y tamaño, añade una capa adicional de inteligencia a la operación del robot.[5]



Fig. 1 Robot paralelo tipo delta

La combinación de robots paralelos tipo delta y visión artificial ha encontrado aplicaciones en diversas industrias, desde líneas de ensamblaje hasta almacenes automatizados, la capacidad de seguir objetos con precisión mejora la eficiencia y la versatilidad de las operaciones, además, en entornos donde la interacción humano-máquina es esencial, estos sistemas permiten una colaboración más segura y eficiente.[6]

Es por estas razones que se desea mostrar este modelo que resulta accesible y fácil de construir, con piezas realizadas a diseño propio y producidas en impresión 3D, garantizando la precisión digna de un robot paralelo con la ventaja de que este se encuentra ya en un trabajo coordinado con una cámara para

realizar el seguimiento de cualquier objeto que cumpla con las características dispuestas, además que se proporcionaran las ecuaciones comprobadas para próximas investigaciones.

II. MARCO TEÓRICO

A. Fundamentos de la robótica

Todo robot presenta en su estructura dos partes fundamentales, los eslabones y las articulaciones, las que le permiten realizar movimientos teniendo en consideración un área o volumen de trabajo donde desarrolle sus actividades. El movimiento de cada articulación puede ser de desplazamiento, de giro o una combinación de ambos, teniendo que cada uno de los movimientos independientes que puede realizar cada articulación con respecto a la anterior se denomina grado de libertad [7], el cual determinaremos con la fórmula de Grübler vista en la ecuación 1.

$$NGDL = \lambda \cdot (n - j - 1) + \sum_{i=1}^j f_i - f_p \quad (1)$$

Donde:

λ : Grados de libertad del espacio de trabajo, 3 en el plano y 6 en el espacio.

n : Número de eslabones.

j : Número de articulaciones.

f_i : Grados de libertad permitidos en cada articulación (Tabla 1).

f_p : Numero de grados de libertad pasivos (no producen torque).

Según Lynch debemos tomar en consideración los siguientes grados de libertad permitidos por articulación mostrados en la tabla I[8]:

TABLA I
NÚMERO DE GRADOS DE LIBERTAD PARA UNIONES COMUNES

Joint type	Dof	Constraints c between two planar rigid bodies	Constraints c between tow spatial rigid bodies
Revolute (R)	1	2	5
Prismatic (P)	1	2	5
Helical (H)	1	N/A	5
Cylindrical (C)	2	N/A	4
Universal (U)	2	N/A	4
Spherical (S)	3	N/A	3

MODERN ROBOTICS DE KEVIN M. LYNCH Y FRANK C. PARK 3 DE MAYO DE 2017

Los movimientos de un robot quedan definidos por los siguientes elementos matemáticos: Las coordenadas cartesianas (x, y, z) y las coordenadas angulares (α, β, γ), cabe resaltar que estos puntos de referencia están directamente relacionados con el *Tool Center Point (TCP)*, el cual deberá tomarse en cuenta si se emplea alguna herramienta o actuador en el extremo del robot.

La cinemática de un robot es la relación matemática entre él mismo y un sistema de origen, sin tomar en cuenta las fuerzas y pares implicados con el objetivo de encontrar la posición y orientación de cada eslabón, especialmente del extremo del robot, así podemos decir que la cinemática es la descripción analítica de los movimientos espaciales del robot en función del tiempo, teniendo tres aspectos a resolver, la cinemática directa, la cinemática inversa y el modelo diferencial.

El desarrollo de la cinemática directa relaciona el hallar la orientación y posición cartesiana en función de los ángulos de rotación dispuestos en el robot, este método posee una única solución, pero es posible hallarlo de diversas maneras, entre las que se tiene la solución geométrica como se muestra en la figura 2, solución por Matriz de Transformación Homogénea (MTH) o solución por Denavit Hartenberg (DH).[9]

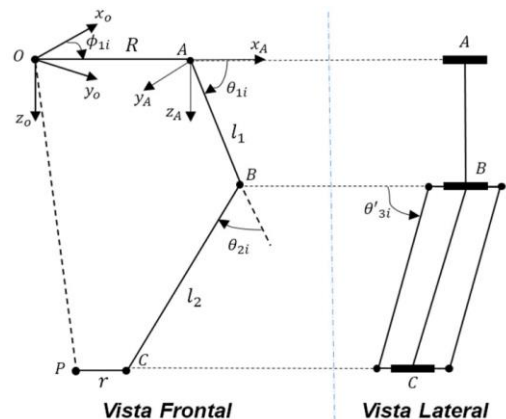


Fig. 2 Parámetros geométricos del manipulador delta[10]

El desarrollo de la cinemática inversa representa lo contrario a la cinemática directa, se emplea la posición cartesiana para hallar los ángulos de rotación de cada articulación, además, se debe saber que podemos tener más de una solución posible con diferentes orientaciones de los eslabones, las cuales satisfagan la posición cartesiana deseada, se pueden aplicar métodos como solución geométrica, solución por MTH o solución por desacople cinemático en caso tengamos un robot de más de tres variables articulares y sus tres últimos grados de libertad se corten en un punto.

El desarrollo del modelo diferencial consta de dos partes a solucionar, el Jacobiano Analítico y el Jacobiano Geométrico, donde hallaremos las velocidades de localización y las velocidades lineales y angulares del extremo del robot respectivamente, a partir de las velocidades articulares, para así conocer a qué velocidad es que nuestra herramienta o extremo del robot se desplaza.

B. Fundamentos de la visión artificial

La visión artificial es una disciplina científica formada por un conjunto de técnicas que permiten la captura, el procesamiento y el análisis de imágenes, con el objetivo de extraer información útil para responder a preguntas sobre el contenido. Todo programa capaz de correr un código de visión

artificial o cualquier otro, está apoyado en librerías, que facilitan las aplicaciones en el área.[5]

OpenCV (Open Source Computer Vision Library) es una biblioteca de código abierto diseñada para el procesamiento de imágenes y visión por computadora. Se desarrolló para proporcionar un conjunto de herramientas y funciones que permiten a los desarrolladores trabajar con imágenes y videos de manera eficiente. OpenCV está escrito en C++ y cuenta con interfaces para varios lenguajes de programación, incluyendo Python.

NumPy es una biblioteca de Python que proporciona soporte para arreglos y matrices multidimensionales, junto con una amplia colección de funciones matemáticas para operar en estos arreglos. Esta biblioteca es esencial para la computación científica en Python debido a su eficiencia en la manipulación de datos numéricos y su capacidad para integrarse con otras bibliotecas y herramientas científicas.

La librería serial en Python se refiere a la interfaz de programación que permite la comunicación serie, especialmente a través de puertos seriales. Los puertos seriales son interfaces de hardware que permiten la transmisión de datos bit a bit entre dispositivos. Esta comunicación serie se utiliza comúnmente para conectar dispositivos electrónicos, como microcontroladores, sensores, módems, impresoras y otros dispositivos periféricos.

Es importante resaltar que OpenCV trabaja los colores en niveles de luminosidad, tono y saturación, a continuación, se puede apreciar cómo podemos extraer dichos colores de la paleta de la figura 3.

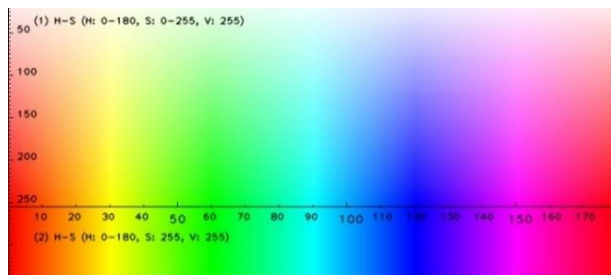


Fig. 3 Colores en niveles de luminosidad, tono y saturación

El término "ASCII" (American Standard Code for Information Interchange) se refiere a un conjunto de códigos de caracteres que representan caracteres y símbolos comunes en inglés. Cada carácter en el conjunto ASCII está asociado con un número único, que se expresa en forma de valor numérico. El estándar ASCII utiliza valores numéricos de 0 a 127 para representar caracteres, y cada número tiene una correspondencia con un carácter específico.

C. Software empleado

El entorno de desarrollo integrado (IDE) más popular en la industria del software es Microsoft Visual Studio. Es una herramienta completa que proporciona un conjunto de características poderosas para el desarrollo de aplicaciones en varios lenguajes de programación, como C++, C# y F#, entre

otros. Visual Studio ofrece un ambiente completo que incluye herramientas de depuración, gestión de versiones, editor de código y capacidades para construir y desplegar aplicaciones.

El Arduino IDE (Entorno de desarrollo integrado) es un programa gratuito y de código abierto diseñado para programar y cargar código en placas de desarrollo basadas en microcontroladores de la plataforma Arduino.

III. MATERIALES Y MÉTODOS

A. Generación del modelamiento robótico

Se empezó definiendo las restricciones físicas del robot, donde era sabido que este tendría un volumen de trabajo esférico, sin embargo, para cubrir un área base de trabajo, esta esfera tendría que estar cortada en alguna región conveniente para su propósito, en este caso, en cualquier plano de la mitad de la esfera hacia abajo como se muestra en la figura 4.

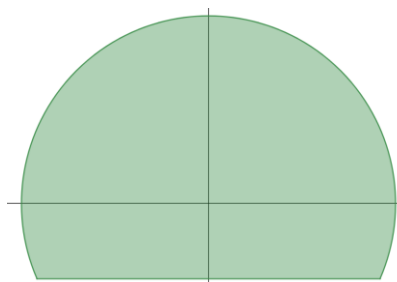


Fig. 4 Volumen de trabajo

Se realizó la cinemática directa del robot teniendo en cuenta sus ángulos y articulaciones presentes en el, se tomaron una articulación rotacional y dos esféricas por cada brazo, donde la única que ejercía torque es la rotacional, ubicada en la base superior del robot, se aplicó el método de solución de Denavit Hartenberg, donde se tomaron los siguientes ángulos mostrados en la figura 5.

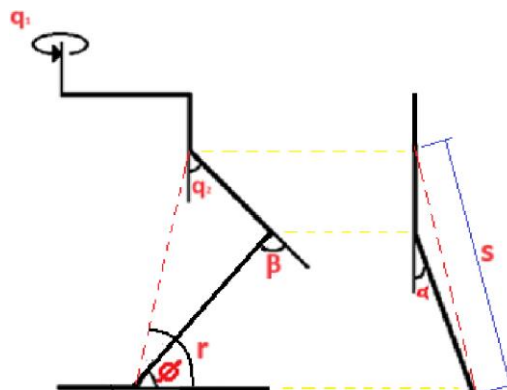


Fig. 5 Ángulos representativos del robot

Planteando así la solución matemática por Denavit Hartenberg en referencia a la figura 6 con las siguientes posiciones y orientaciones de cada articulación, que a su vez se ven reflejadas en la tabla II con las rotaciones en Z, desplazamiento en Z, rotación en X y desplazamiento en X.

TABLA II
PARAMETROS DE DENAVID HARTENBERG

ART	θ	d	a	α
1	$q_1 + \frac{\pi}{2}$	$-L_2$	L_1	$\frac{\pi}{2}$
2	$q_2 - \frac{\pi}{2}$	0	L_3	0
3	$-\beta$	$L_4 \sin(\alpha)$	$L_4 \cos(\alpha)$	0
4	$-\phi$	0	0	$\frac{\pi}{2}$
5	0	L_5	0	0

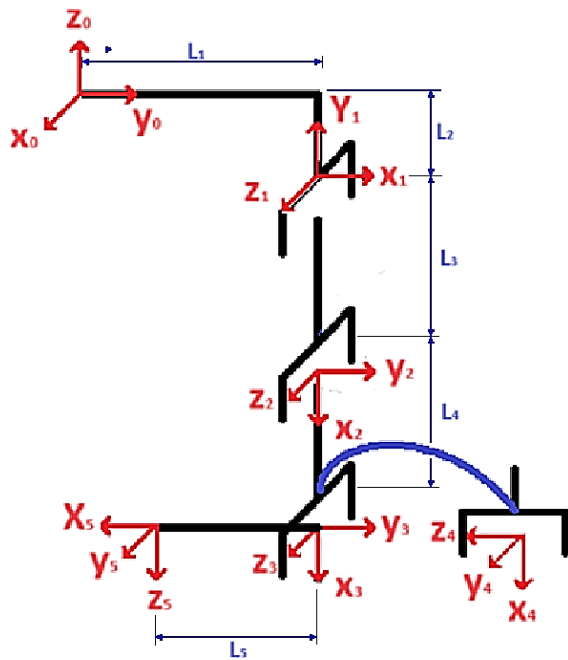


Fig. 6 Denavid Hartenberg

Seguidamente se procede a realizar la cinemática inversa del robot, aplicando nuevamente el método trigonométrico, se debe tomar en cuenta que q_1 representa la posición del brazo alrededor del robot, teniendo valores fijos, siendo 0° , 120° y 240° , logrando así la forma triangular representativa del robot.

Fácilmente hallamos el ángulo α para el Brazo I, que es la inclinación lateral de cada brazo respecto de su posición de origen como se ve en la ecuación 2, las ecuaciones de α para los demás brazos se mostraran posteriormente como a_2 y a_3 .

$$\alpha = \text{asin}\left(\frac{x}{L_4}\right) \quad (2)$$

Seguidamente se calculó q_2 , que será la única articulación que genere torque y por ende donde serán colocados los servomotores, teniendo esta mayor relevancia que las anteriores ya que sabiendo este ángulo, podemos llevar al robot a cualquier posición del volumen de trabajo, quedando definida en la ecuación 3.

$$q_2 = \text{atan}\left(\frac{L_1 - P_y \cdot C_{q_1} - L_5 + P_x \cdot C_{q_1} - L_4 \cdot C_\alpha \cdot C_\phi}{L_2 + P_z + L_4 \cdot C_\alpha \cdot S_\phi}\right) \quad (3)$$

A demás se debe tener en cuenta determinadas restricciones y áreas de trabajo del robot para no causar conflicto en el mismo, estas restricciones las veremos en el brazo II y III correspondientes a un q_1 igual a 120° y 240° respectivamente.

En el caso del Brazo II y III, se debe tomar en cuenta un ángulo i , que no es más que la suma del ángulo del punto deseado en valor absoluto respecto al eje x y un ángulo de 30° , ya que este es el que se forma con la proyección posterior de 120° y 240° desplazados 90° por comodidad como se muestra en la figura 7.

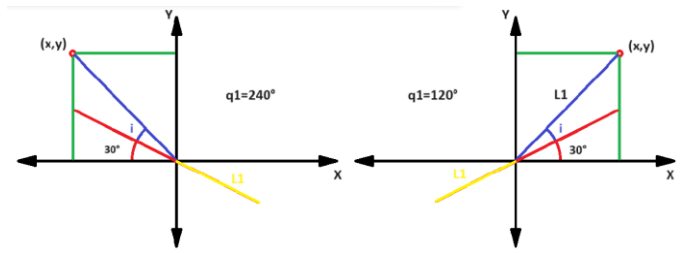


Fig. 7 Definición de ángulo i

Se define la ecuación 4 de i para el Brazo II y Brazo III respectivamente:

$$i = \left| \text{atan}\left(\frac{P_y}{P_x}\right) \right| + 30 \quad (4)$$

Para el Brazo II, existe un comportamiento distinto según las siguientes regiones:

TABLA III
ECUACIONES EN FUNCIÓN DE SU POSICIÓN DEL BRAZO II

```

if y>x/sqrt(3)
    a2=asin((sin(i2)*(sqrt(x^2+y^2))/L4));
end
if y<x/sqrt(3)
    a2=-asin((sin(i2)*(sqrt(x^2+y^2))/L4));
end
if y==x/sqrt(3)
    a2=0;
end
L6=abs(L5+y*cos(q12)-x*sin(q12));

```



```

if y>-sqrt(3)*x+2*L5
    s2=sqrt((L1+L6)^2+(z+L2)^2);
    r2=atan(abs((z+L2)/(L1+L6)));
end
if y<=-sqrt(3)*x+2*L5
    s2=sqrt((L1-L6)^2+(z+L2)^2);
    r2=atan(abs((z+L2)/(L1-L6)));
end
if y<=-sqrt(3)*x+2*L5 && L1<L6
    r2=pi-atan(abs((z+L2)/(L1-L6)));
end
if y<=-sqrt(3)*x-2*L5 && L6==L1
    r2=0;
end

```

Para el Brazo III, existe un comportamiento distinto según las siguientes regiones.

TABLA IV
ECUACIONES EN FUNCIÓN DE SU POSICIÓN DEL BRAZO III

```

if y>-x/sqrt(3)
    a3=-asin((sin(i3)*(sqrt(x^2+y^2))/L4));
end
if y<-x/sqrt(3)
    a3=asin((sin(i3)*(sqrt(x^2+y^2))/L4));
end
if y==x/sqrt(3)
    a3=0;
end
L6=abs(L5+y*cos(q13)-x*sin(q13));
if y>sqrt(3)*x+2*L5
    s3=sqrt((L1+L6)^2+(z+L2)^2);
    r3=atan(abs((z+L2)/(L1+L6)));
end
if y<=sqrt(3)*x+2*L5
    s3=sqrt((L1-L6)^2+(z+L2)^2);
    r3=atan(abs((z+L2)/(L1-L6)));
end
if y<=sqrt(3)*x+2*L5 && L1<L6
    r3=pi-atan(abs((z+L2)/(L1-L6)));
end
if y>sqrt(3)*x+2*L5 && L1<L6
    r2=pi-atan(abs((z+L2)/(L1+L6)));
end
if y<=sqrt(3)*x-2*L5 && L6==L1
    r3=0;
end

```

Todas las restricciones del sistema anteriormente mencionadas, se ven afectadas por cada una de las regiones sombreadas vistas en la figura 8, que corresponden a la geometría de la base inferior, la cual es un triángulo equilátero con una altura media de 5.2 cm, adicionalmente, el modelo podemos apreciarlo en la figura 9, donde también se muestra una cavidad media donde es

posible insertar algún actuador y 3 agujeros laterales para colocar las 6 rótulas correspondientes a la base inferior.

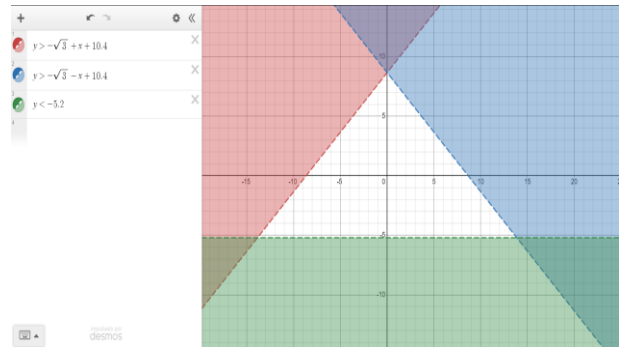


Fig. 8 Regiones de restricciones del robot

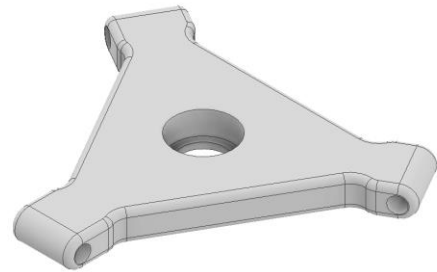


Fig. 9 Base inferior del robot paralelo

B. Diseño y análisis de partes CAD

La primera inquietud al realizar el proyecto fue el peso que tendría cada brazo, porque si queríamos que este tenga movimientos precisos y ágiles, los brazos tenían que ser lo más ligeros posibles, porque se analizaron alternativas entre polímeros y aluminio ya que es el metal comercial más ligero, entre estas alternativas se optó por la utilización del aluminio a pesar de que el polímero tenía la mitad de masa que este, porque en la zona de la rosca para conectar con las rótulas de los brazos, este sufriría un esfuerzo que llevaría a la rotura de la pieza por tener un factor de seguridad de 1 con una carga de 25 N, como se muestra en la figura 10, siendo este factor insuficiente porque buscamos al menos que sea mayor que 3 para considerar al material ideal.

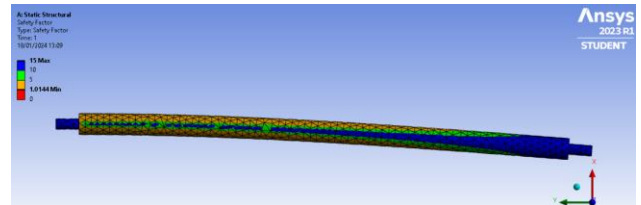


Fig. 10 Factor de seguridad de varillas de nylon

A diferencia del anterior, el aluminio presenta un factor de seguridad de 3.9 visto en la figura 1 y una deformación máxima de 0.91 mm con la misma carga visto en la figura 12, siendo este ideal para el trabajo a ejecutar.

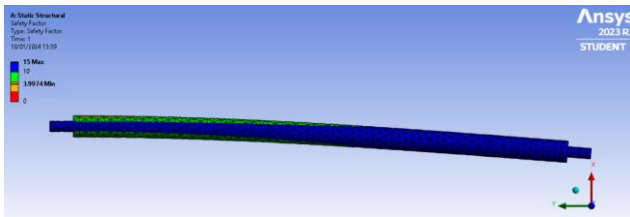


Fig. 11 Factor de seguridad de varillas de aluminio

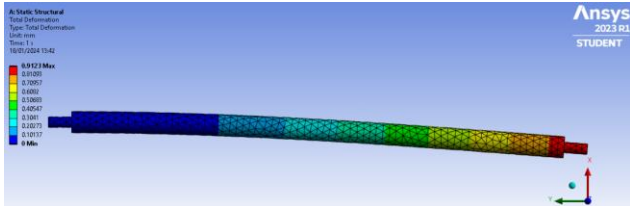


Fig. 12 Deformación máxima de varillas de aluminio

Es necesario también realizar un análisis de los brazos, a los que cada motor dará directamente todo el torque necesario para mover cada brazo, en este caso se consideró el peso de las varillas de aluminio con un adicional de 800 g, a causa de la fuerza de tracción ejercida al momento de levantar la base inferior ya que todos los brazos convergen en ella, tirando cada uno para su lado.

Teniendo como resultante una fuerza de 28.2 N que afectaran directamente al brazo en el punto más alejado, logrando determinar así una deformación máxima de 0.29 mm que se ve en la figura 13, y un factor de seguridad de 2.56, siendo óptimo para el propósito deseado visto en la figura 14.

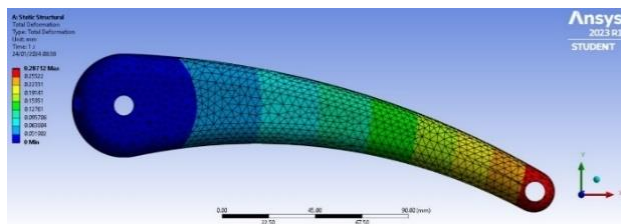


Fig. 13 Deformación máxima del brazo principal

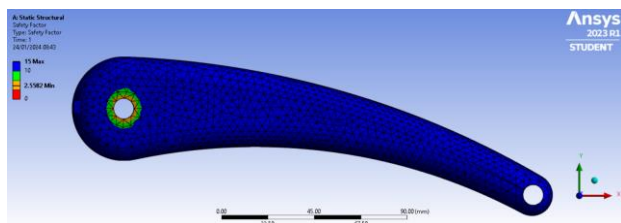


Fig. 14 Factor de seguridad del brazo principal

Es importante mencionar que cada brazo fue impreso en 3D con PLA (Acido Poliláctico) en una impresora Artillery X1 como se muestra en la figura 15, con una calidad alta de paso 0.12 mm y una densidad del 80% para garantizar que soporte las cargas mencionadas.



Fig. 15 Impresión en 3D del brazo principal

C. Sistemas electrónicos y ensamble

En función de los productos que hay en el mercado y el cálculo realizado de torques, se vio por conveniente emplear dos motores en paralelo para doblar el torque aplicado, en este caso empleamos motores MG996R, los cuales tienen una fuerza de 11 Kg-f, teniendo en total 22 Kg-f, suficiente para cubrir la demanda de fuerza necesaria.

A partir de lo mencionado anteriormente, teniendo en cuenta que se trabajará con Arduino, el ideal sería un Arduino Nano porque no aplicaremos muchos puertos y la memoria a utilizar será mínima, sin embargo, algunas pruebas de movimiento también fueron realizadas por medio de Matlab, por lo que necesariamente se utilizó un Arduino Mega por la memoria que este puede manejar.

Las pruebas realizadas en Matlab se realizaron tomando unas coordenadas cartesianas iniciales, las cuales ingresaban en un código que permitía realizar la cinemática inversa, y con los resultados de esta, realizar la cinemática directa para graficar la posición y orientación del extremo del robot con ayuda del *tool box* de robótica de Peter Core como se ve en la figura 16, obteniendo la MTH que mostraría en su última columna, la posición obtenida, la cual tendría que ser la inicialmente ingresada.

TABLA V

VERIFICACIÓN DE POSICIÓN Y ORIENTACIÓN DE BRAZOS

```
clear all; close all; clc;
L1=15.0;
L2=2.1;
L3=20.2;
L4=40.5
L5=5.2;
x=15;
y=15;
z=-35;
[Q] = paraleloD(x,y,z)
b=q2-p+pi/2;
l=L4*sin(a);
o=L1+L3*sin(q2)-L4*cos(p)*cos(a);
T1=Revolute('d',-L2,'a',L1,'alpha',pi/2);
```

```

T2=Revolute('d',0,'a',L3,'alpha',0);
T3=Revolute('d',L4*sin(a),'a',L4*cos(a),'a
lpha',0);
T4=Revolute('d',0,'a',0,'alpha',pi/2);
T5=Revolute('d',0,'a',-L5,'alpha',pi/2);
bot1=SerialLink([T1 T2 T3 T4
T5],'name','Robot_1');
ws=(-100 100 -100 100 -70 20]);
bot1.fkine([pi/2 q2-pi/2 -b -p pi])

```

```

ans =
     0     0     1    15
     1     0     0    15
     0     1     0   -35
     0     0     0     1

```

```

figure(1)
bot1.plot([pi/2 q2-pi/2 -b -p
pi],'workspace',ws,'noname')
hold on
b=q2-p+pi/2;
l=L4*sin(a);
o=L1+L3*sin(q2)-L4*cos(p)*cos(a);
T1=Revolute('d',-L2,'a',L1,'alpha',pi/2);
T2=Revolute('d',0,'a',L3,'alpha',0);
T3=Revolute('d',L4*sin(a),'a',L4*cos(a),'a
lpha',0);
T4=Revolute('d',0,'a',0,'alpha',pi/2);
T5=Revolute('d',0,'a',-L5,'alpha',pi/2);
bot2=SerialLink([T1 T2 T3 T4
T5],'name','Robot_2');
ws=(-100 100 -100 100 -70 20]);
bot2.fkine([7*pi/6 q2-pi/2 -b -p pi])

```

```

ans =
 -0.8660     0  -0.5000    15
 -0.5000     0   0.8660    15
     0     1     0   -35
     0     0     0     1

```

```

bot2.plot([7*pi/6 q2-pi/2 -b -p
pi],'workspace',ws,'noname')
hold on
b=q2-p+pi/2;
l=L4*sin(a);
o=L1+L3*sin(q2)-L4*cos(p)*cos(a);

```

```

T1=Revolute('d',-L2,'a',L1,'alpha',pi/2);
T2=Revolute('d',0,'a',L3,'alpha',0);
T3=Revolute('d',L4*sin(a),'a',L4*cos(a),'a
lpha',0);
T4=Revolute('d',0,'a',0,'alpha',pi/2);
T5=Revolute('d',0,'a',-L5,'alpha',pi/2);
bot3=SerialLink([T1 T2 T3 T4
T5],'name','Robot_3');
ws=(-100 100 -100 100 -70 20]);
bot3.fkine([11*pi/6 q2-pi/2 -b -p pi])

```

```

ans =
  0.8660     0  -0.5000    15
 -0.5000     0  -0.8660    15
     0     1     0   -35
     0     0     0     1

```

```

bot3.plot([11*pi/6 q2-pi/2 -b -p
pi],'workspace',ws,'noname')

```

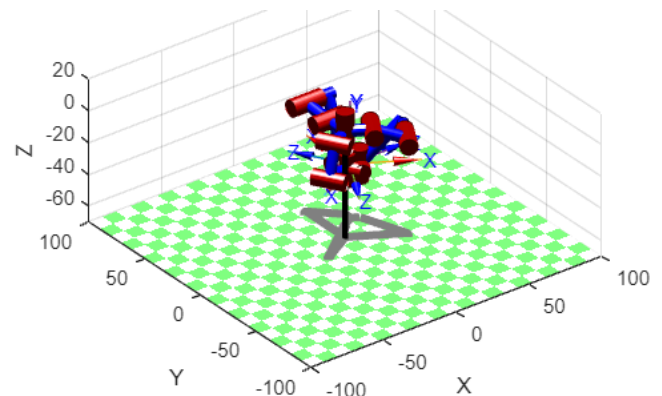


Fig. 16 Denavid Hartenberg de los tres brazos unidos en una sola toma

Como se pudo apreciar en las líneas de código anteriores, el procedimiento se realizó 3 veces, cada una de estas representa cada brazo calculado independientemente, claramente no todos los brazos tendrán los mismos ángulos de rotación, es así que con la ayuda de las restricciones mencionadas y con el ángulo q_1 correspondiente, podemos calcular los diferentes ángulos de rotación que corresponden a cada brazo para determinada posición cartesiana.

En total se emplearon 6 puertos digitales para el manejo de 6 servomotores como se muestra en la figura 17, además de adicionar un puerto para endosar un módulo Bluetooth, para lo cual se realizó el diseño y fabricación de un circuito impreso para tener una mejor organización, control y espacio en el proyecto.

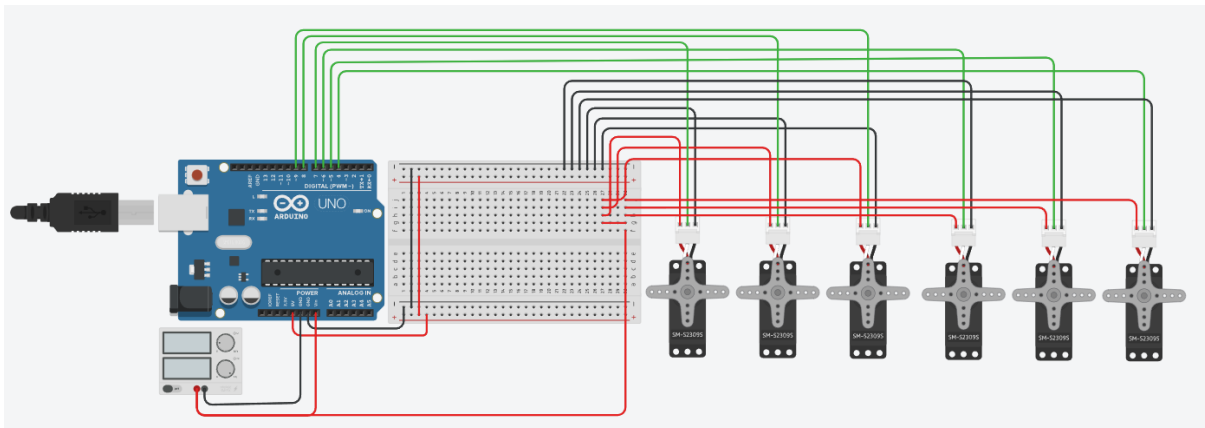


Fig. 17 Conexiones directas de los servomotores al Arduino

Se procedió a realizar todas las conexiones necesarias y a cargar el código para verificar el funcionamiento primeramente con ingreso de coordenadas fijas para verificar el área de trabajo, posteriormente, se realizó una variante que consiste en obtener coordenadas a través del puerto serie del Arduino desde el programa en Python en formato ASCII, en esta parte se presentó la mayor dificultad, debido a que no se lograba la comunicación entre estos dos programas y se determinó que la velocidad de comunicación era demasiado elevada por lo que se redujo de 9600 a 1200 baudios, además, se creó una función de extracción de coordenadas, donde una vez se haya cumplido el punto de llegada anterior, recién podía pasar al siguiente, además, se creó una condición que consistía en que si la pelota no se desplazaba una distancia mínima de 5 pixeles, el robot permanecería estático y así de esta manera se logró comunicar ambos programas a tiempo real y realizar la lectura correcta de la posición de la pelota como se ve en la imagen 18.



Fig. 18 Ensayo de reconociendo y seguimiento de objetos en función del color del objeto.

Para que se reconozcan colores específicos fue necesario la utilización de la paleta de colores en niveles de luminosidad, tono y saturación de la figura 3, además que el tamaño del objeto también se ve comprometido, si no cubre determinada cantidad de pixeles, este no será reconocido.

IV. RESULTADOS Y DISCUSIÓN

Se logro el movimiento coordinado de cada servomotor que se trabajó en pares, invirtiendo la polaridad tanto del potenciómetro como del motor dentro del servomotor, con el objetivo de digitar un mismo ángulo y que estos giren en direcciones contrarias, dando como resultado un giro coordinado de los motores frente a frente aplicando un fuerza de 22 Kg-f.

Dentro de todo el volumen de trabajo admisible matemáticamente, se añadieron algunas restricciones más, a consecuencia de las limitaciones físicas de los actuadores, que únicamente podían girar un máximo de 180°, por lo que además se ubicó la posición 0 en 10° de cada servomotor, con el objetivo de que este pudiera ascender un poco más y tenga mayor libertad en el movimiento.



Fig. 19 Máximo punto en altitud correspondiente a 10° respecto a la horizontal

La combinación serial se dio a una tasa de 1200 baudios porque el robot se saturaba de información, provocando vibraciones en él, además se dio una subrutina para lograr que el robot se moviese únicamente si el objeto era desplazado 5 pixeles a partir de su posición origen.

El desarrollo de la cinemática directa se dio siguiendo los pasos de Denavid Hartenberg y la inversa se logró mediante métodos geométricos donde se definieron además todas las

restricciones del robot, todos los datos fueron corroborados en un programa de Matlab y Arduino, donde se ingresaban datos al azar en cinemática directa, y los resultados ingresaban directamente a la cinemática inversa, obteniendo como resultado final los datos ingresados inicialmente.

Dentro de las pruebas físicas, se optó por darle formas geométricas como coordenadas, tales como, circunferencias, medias circunferencias, triángulos y cuadrados, evidenciando así que el robot era capaz de realizar tareas de secuencias y movimientos definidos por funciones matemáticas, además de tener la seguridad de que este podría cumplir con su función básica de seguimiento de objetos dentro del área especificada y mostrada en la figura 20.

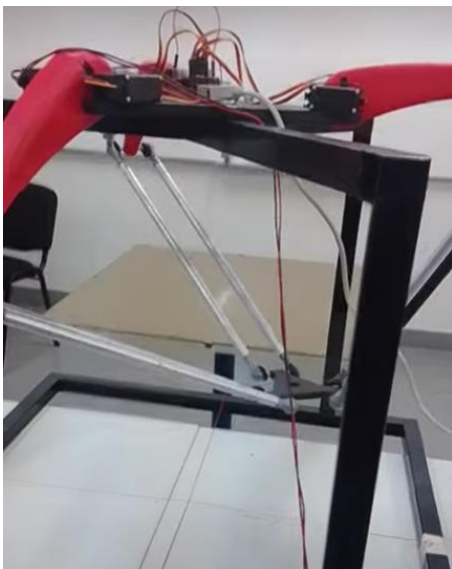


Fig. 20 Prueba de circunferencia completa dirigido por Arduino

Adicional a lo mencionando anteriormente, se buscó que Matlab grafique la trayectoria seguida por el robot como se ve en la figura 21, la cual tuvo un total de 600 puntos alrededor de la circunferencia de radio 15 cm, para garantizar el movimiento suave y continuo del robot, evitando desplazamientos bruscos.

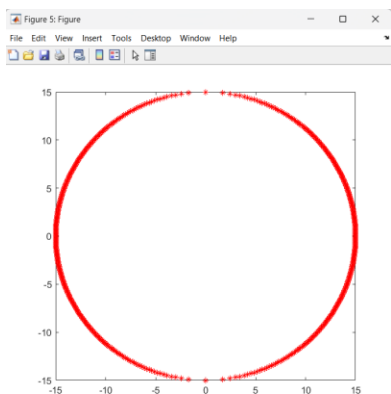


Fig. 21 Trayectoria circular del robot paralelo.



Fig. 22 Robot en físico, con webcam y Arduino mega listo para operar.

La prueba final como se muestra en la figura 23 fue el seguimiento de una pelota de color amarillo, la resultado un éxito, con una precisión de $\pm 1\text{cm}$ y una velocidad de 100 ms por grado rotado, teniendo así suavidad en los movimientos para evitar el temblor ocasionado por la sobre carga de datos de posición enviadas por el puerto serial desde una laptop que también recibía datos de la webcam como se ve en la figura 23.

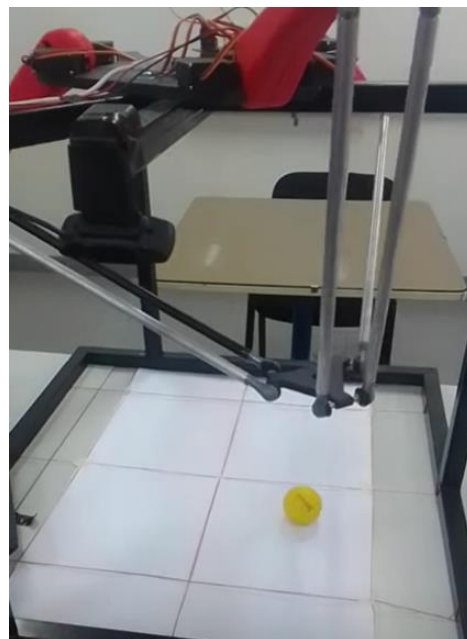


Fig. 23 Prueba de seguimiento de una pelota.

Durante las pruebas realizadas pudimos apreciar que la respuesta del robot era instantánea, imposible de medir manualmente, sin embargo, sabemos que el robot responde en

un bucle de 100 ms por grado rotado, lo que nos daría la velocidad de respuesta que este tiene.

Es importante mencionar además que la velocidad de este puede ser alterada, sin embargo, en pruebas realizadas, se vio que estaríamos perdiendo precisión y performance al momento realizar el seguimiento, ya que el robot entraría en estado de inestabilidad por la alta velocidad a la que este se movería, ocasionando, además, un consumo excesivo de corriente, desgaste mecánico y sobrecalentamiento prematuro de los motores, reduciendo su tiempo de vida útil.

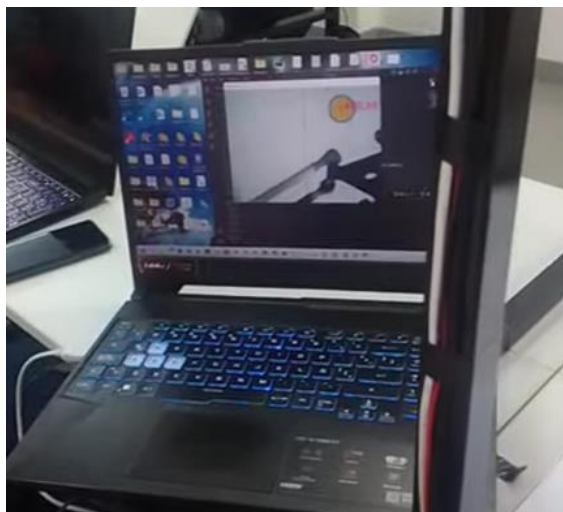


Fig. 24 Recopilación de datos de la webcam.

IV. CONCLUSIONES

Se logro realizar el seguimiento continuo de un objeto, independientemente de su forma, tamaño y color, ya que estos son variables en el modelo de visión artificial.

Se implemento de manera eficiente el sistema, logrando un seguimiento controlado, con una velocidad de respuesta instantánea o como máximo 100 ms, que es el tiempo correspondiente a la velocidad por grado rotado.

El diseño se validó estructuralmente con el software Ansys Work Bench y con pruebas físicas de funcionamiento, llegando a la conclusión que se debía emplear dos servomotores en paralelo para alcanzar el torque requerido por el robot.

Se diseñó y construyó un robot paralelo tipo delta que se puede desplazar en una circunferencia con un radio de 15 cm y en una altura de 10 cm, siendo capaz de posicionarse en cualquier lugar con una precisión de 5 pixeles.

El análisis de la cinemática directa e inversa sirvieron para corroborar y ensayar las diferentes posiciones que el robot podía alcanzar mediante un programa en Matlab para el cálculo numérico y Simulink para comandar los servomotores en función de los cálculos realizados, debiendo obtener la misma posición en ambos.

Si trabajaría con movimientos físicos, en función de reconocimiento de objetos por un computador, se recomienda

disminuir la frecuencia de envío de datos para no sobrecargar el sistema.

El modelo puede seguir desarrollándose para lograr la predicción de posición de objetos sobre una faja transportadora y realizar procesos de paletización.

REFERENCIAS

- [1] J. Brinker, N. Funk, P. Ingenlath, Y. Takeda, and B. Corves, "Comparative Study of Serial-Parallel Delta Robots with Full Orientation Capabilities," *IEEE Robot Autom Lett*, vol. 2, no. 2, pp. 920–926, Apr. 2017, doi: 10.1109/LRA.2017.2654551.
- [2] C. Augusto, P. Cortés, E. M. Oviedo, P. Fabián, and C. Herrera, "Optimización dimensional de un robot paralelo tipo delta basado en el menor consumo de energía," 2011.
- [3] M. A. Garcia-Murillo, R. E. Sanchez-Alonso, and J. Gallardo-Alvarado, "Kinematics and dynamics of a translational parallel robot based on planar mechanisms," *Machines*, vol. 4, no. 4, Dec. 2016, doi: 10.3390/machines4040022.
- [4] A. Sharida and I. Hashlamon, "Real-time vision-based controller for delta robots," *International Journal of Intelligent Systems Technologies and Applications*, vol. 20, no. 4, p. 271, 2021, doi: 10.1504/ijista.2021.10045532.
- [5] E. Técnica *et al.*, "Universidad Politécnica de Madrid," 2015.
- [6] C. A. Jara *et al.*, "Análisis del espacio de trabajo de un robot paralelo 3RRR."
- [7] Lung-Wen-Tsai. "Robot Analysis The Mechanics of Serial and Parallel Manipulators" 1999.
- [8] K. (Kevin M.) Lynch and F. C. Park, *Modern robotics : mechanics, planning, and control*.
- [9] A. Barrientos, L. Peñin, C. Balaguer y R. Aracil "Fundamentos de la Robótica" 1997, pp 30-144.
- [10] M. Felipe pedraza, pedro F. cárdenas, F. José rodríguez, and eugenio YiMe, "IV Congreso internacional de ingeniería mecatrónica y automatización-CIIMA 2015."