UNIVERSIDAD POPULAR AUTÓNOMA DEL ESTADO DE PUEBLA

FACULTAD DE INGENIERÍAS

MEC319-REBÓTICA INDUSTRIAL

Proyecto Final

"ROBOT PIKZA PIKZA - ABRE CAJA"

CATEDRÁTICO:

Dr. JOSÉ MIGUEL GUTIÉRREZ RAMÍREZ

PRESENTADORES:

LICENCIATURA EN INGENIERÍA MECATRÓNICA

- 5801965 | JUAN ALFREDO SERRATO VALLE
- 5802074 | KATHIA PAOLA BUSTAMANTE CLIMACO
- 5801988 | DENNIS IVÁN PÉREZ MONTIEL
- 5801819 | ALDO ÁLVAREZ ZAVALETA
- 5801724 | EDUARDO HUERTA CERVANTES
- 5802023 | AXEL ARRIOLA FONSECA

OTOÑO 2021

ÍNDICE

1. INTRODUCCIÓN

La robótica se está apoderando del mundo. Considerando el entorno de un ingeniero mecatrónico a cualquier lado donde observamos, los encontraremos. De diferentes tamaños, versiones, estructuras etc. Esto nos ha despertado una inquietud, como futuros ingenieros, de conocer el detrás de éstas maravillosas creaciones. Para iniciar un desarrollo de éste tipo, comenzamos con la estrategia de un planteamiento matemático capaz de representar y llevar al dimensionamiento un conjunto de eslabones con un diseño más allá de un par de líneas.

Los robots desde el prototipado se diseñan para desempeñar tareas específicas, de forma individual o en conjunto para que el resultado de uno dependa de otro, formando una cadena o línea sistemática. Podemos encontrar una amplia gama en los catálogos de proveedores o comenzar de cero con un diseño que vaya más allá de una idea común.

Para introducirnos al mundo de la robótica, hemos planteado el desarrollo de un robot con dimensiones estratégicas para una actividad específica. A lo largo de este reporte se encuentra la esencia del proyecto el cual consiste en primera instancia en un diagrama cinemático del cual partimos para construir el diagrama de parámetros por el popular método de Denavit - Hartenberg. Posteriormente se emplea la herramienta digital matlab para calcular la matriz de transformación homogénea con la que le daremos definición al robot.

2. OBJETIVOS

2.1. Objetivo general

Construir un robot que forme parte de una celda de servido automatizado de pizza.

2.2. Objetivos específicos

2.2.1 Diseñar un robot con cuatro grados de libertad capaz de abrir una caja de pizza medida estándar tomando en consideración todas las características matemáticas aprendidas en el curso de Rebótica otoño 2021.
2.2.2 Adaptar el robot diseñado para ser controlado implementando actuadores mediante una programación específica.
2.2.3 Construir el robot diseñado realizando análisis de materiales y considerando estabilidad en el proceso.

3. FUNCIONAMIENTO

Al ser un ensamble electro-mecánico con cinco ejes de libertad , éste robot pudiese cumplir alrededor de XXX combinaciones de movimientos, considerando un solo eslabón, un par y sucesivamente. Pero su funcionalidad se centra en una acción específica guiada por una receta previamente programada, con la cual no requiere de sensores que soporten el funcionamiento. La funcionalidad del robot se puede enunciar en las siguientes líneas:

- Mantener estabilidad al posicionarse sobre una superficie plana en la posición HOME y al aplicar cualquier movimiento en cualquier eslabón.
- Al tomar la posición inicial, el robot adoptará la posición HOME.
- El robot tendrá la capacidad de adoptar una posición en la que consiga sujetar una superficie de una caja de cartón como la que se muestra en la Figura 3.1 que contenga una pizza medida estándar.

Figura 3.1 Dimensión estándar de una caja de pizza.

- Una vez sujetada la caja, el robot procederá a abrir la misma de manera que sea posible extraer el producto de forma sencilla para la siguiente estación. Pasando de una posición A a una B, como se muestra en la Figura 3.2.

Figura 3.2 Dimensión estándar de una caja de pizza.

4. DIAGRAMA CINEMÁTICO

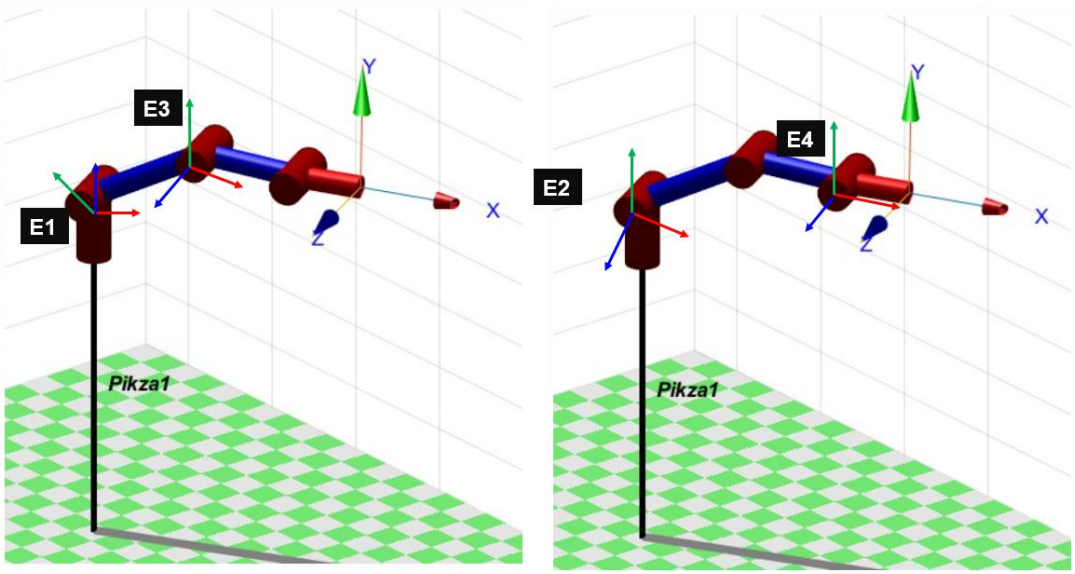


Figura 4.1 Diagrama cinemático. Donde el eje de coordenadas “x” está representado en color rojo, “y” en verde y “z” en azúl.

5. PARÁMETROS DE DENAVIT-HARTENBERG

Pikza1:: 4 axis, RRRR, stdDH, slowRNE

j	theta	d	a	alpha	offset
1	q1	5	0	1.5708	0
2	q2	0	20	0	0
3	q3	0	15	0	0
4	q4	0	10	0	0

Figura 5.1 Parámetros de Denavit-Hartenberg (Datos obtenidos en Matlab).

6. MATRIZ DE TRANSFORMACIÓN HOMOGÉNEA SIMBÓLICA

[0.5*cos(t2 - 1.0*t1 + t3 + t4) + 0.5*cos(t1 + t2 + t3 + t4), - 0.5*sin(t2 - 1.0*t1 + t3 + t4) - 0.5*sin(t1 + t2 + t3 + t4),

sin(t1), 7.5*cos(t1 + t2 + t3) + 5.0*cos(t2 - 1.0*t1 + t3 + t4) + 10.0*cos(t1 - 1.0*t2) + 5.0*cos(t1 + t2 + t3 + t4) + 10.0*cos(t1 + t2) + 7.5*cos(t2 - 1.0*t1 + t3)]

[0.5*sin(t1 + t2 + t3 + t4) - 0.5*sin(t2 - 1.0*t1 + t3 + t4), 0.5*cos(t1 + t2 + t3 + t4) - 0.5*cos(t2 - 1.0*t1 + t3 + t4), -

1.0*cos(t1), 7.5*sin(t1 + t2 + t3) - 5.0*sin(t2 - 1.0*t1 + t3 + t4) + 10.0*sin(t1

– 1.0*t2) + 5.0*sin(t1 + t2 + t3 + t4) + 10.0*sin(t1 + t2) – 7.5*sin(t2 – 1.0*t1 + t3)]

[sin(t2 + t3 + t4),

cos(t2 + t3 + t4),

0.00000000000000000612323399572792216556419396025 47,

10.0*sin(t2 + t3 + t4) + 15.0*sin(t2 + t3) + 20.0*sin(t2) + 5.0]

[0, 0, 0,

1.0]

Matriz 6.1 MTHS (Matriz de transformación homogénea simbólica) (Datos obtenidos en Matlab).

7. MATRIZ DE TRANSFORMACIÓN HOMOGÉNEA EN HOME

Se utilizó la siguiente matriz para la posición de home: **qh=[0,0,2.7,-1.3]**

```
MTH2 =

    0.1700   -0.9854         0    8.1386
    0.0000    0.0000   -1.0000    0.0000
    0.9854    0.1700    0.0000   21.2652
         0         0         0    1.0000
```

Matriz 6.2 MTH2 (Matriz de transformación homogénea HOME) (Datos obtenidos en Matlab).

8. JACOBIANO, VELOCIDAD X,Y,Z

XYZ =

```
 7.5*cos(t1 + t2 + t3) + 5.0*cos(t2 - 1.0*t1 + t3 + t4) + 10.0*cos(t1 - 1.0*t2) +
5.0*cos(t1 + t2 + t3 + t4) + 10.0*cos(t1 + t2) + 7.5*cos(t2 - 1.0*t1 + t3)

 7.5*sin(t1 + t2 + t3) - 5.0*sin(t2 - 1.0*t1 + t3 + t4) + 10.0*sin(t1 - 1.0*t2) +
5.0*sin(t1 + t2 + t3 + t4) + 10.0*sin(t1 + t2) - 7.5*sin(t2 - 1.0*t1 + t3)

 10.0*sin(t2 + t3 + t4) + 15.0*sin(t2 + t3) + 20.0*sin(t2) + 5.0
```

Matriz 8.1 Jacobiano (Velocidad lineal) (Datos obtenidos en Matlab).

t1d =

```
-5.0*sin(t1)*(2.0*cos(t2 + t3 + t4) + 3.0*cos(t2 + t3) + 4.0*cos(t2))
 5.0*cos(t1)*(2.0*cos(t2 + t3 + t4) + 3.0*cos(t2 + t3) + 4.0*cos(t2))
```

Matriz 8.2 Jacobiano articulación 1 (Velocidad lineal) (Datos obtenidos en Matlab).

```
0
```

t2d =

```
-5.0*cos(t1)*(2.0*sin(t2 + t3 + t4) + 3.0*sin(t2 + t3) + 4.0*sin(t2))
-5.0*sin(t1)*(2.0*sin(t2 + t3 + t4) + 3.0*sin(t2 + t3) + 4.0*sin(t2))
```

```
10.0*cos(t2 + t3 + t4) + 15.0*cos(t2 + t3) + 20.0*cos(t2)
```

Matriz 8.3 Jacobiano articulación 2 (Velocidad lineal) (Datos obtenidos en Matlab).

```
t3d =

-5.0*cos(t1)*(2.0*sin(t2 + t3 + t4) + 3.0*sin(t2 + t3))

-5.0*sin(t1)*(2.0*sin(t2 + t3 + t4) + 3.0*sin(t2 + t3))

      10.0*cos(t2 + t3 + t4) + 15.0*cos(t2 + t3)
```

Matriz 8.3 Jacobiano articulación 3 (Velocidad lineal) (Datos obtenidos en Matlab).

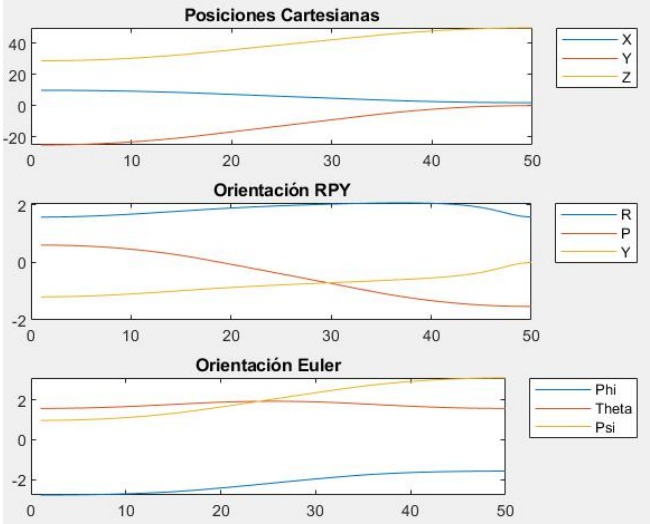
```
t4d =

-10.0*sin(t2 + t3 + t4)*cos(t1)

-10.0*sin(t2 + t3 + t4)*sin(t1)

      10.0*cos(t2 + t3 + t4)
```

Matriz 8.4 Jacobiano articulación 4 (Velocidad lineal) (Datos obtenidos en Matlab).



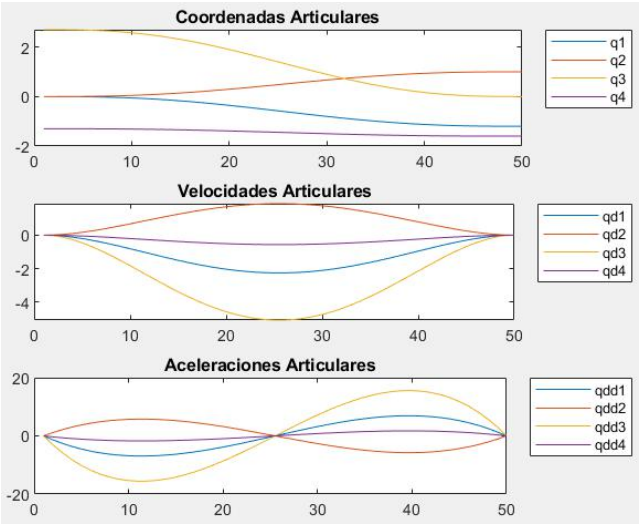


Figura 8.1 Gráficos de posición y velocidad.

9. SIMULACIÓN DE ROBOT (HOME)

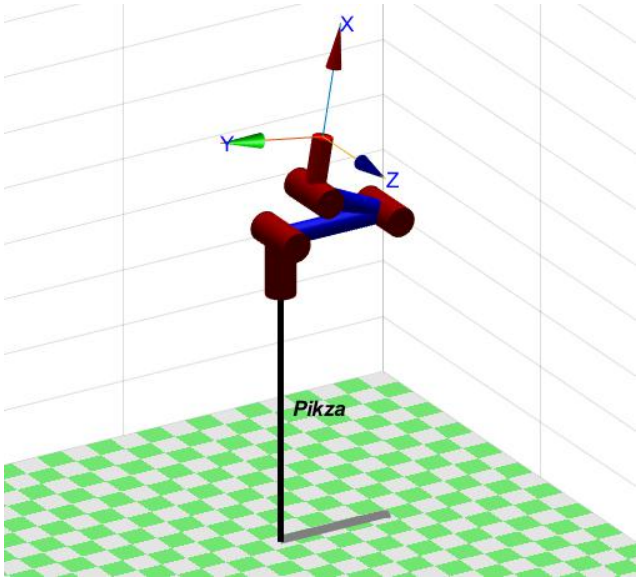


Figura 9.1 Robot en posición HOME.

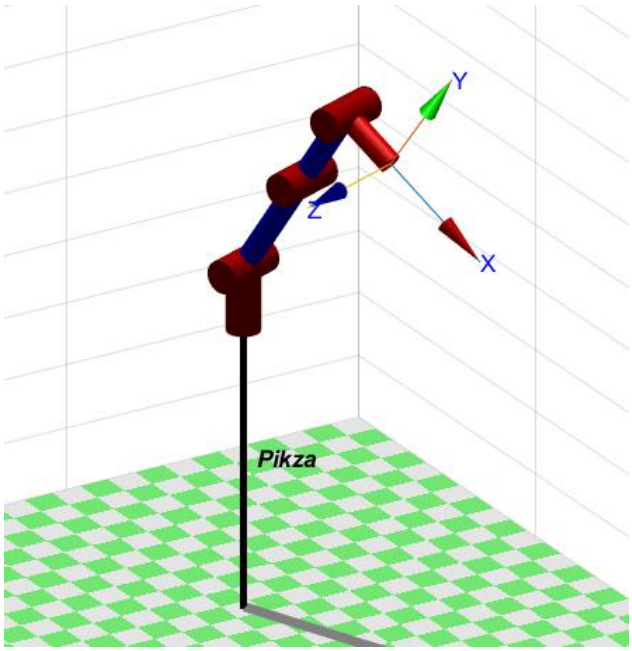


Figura 9.2 Robot en posición A

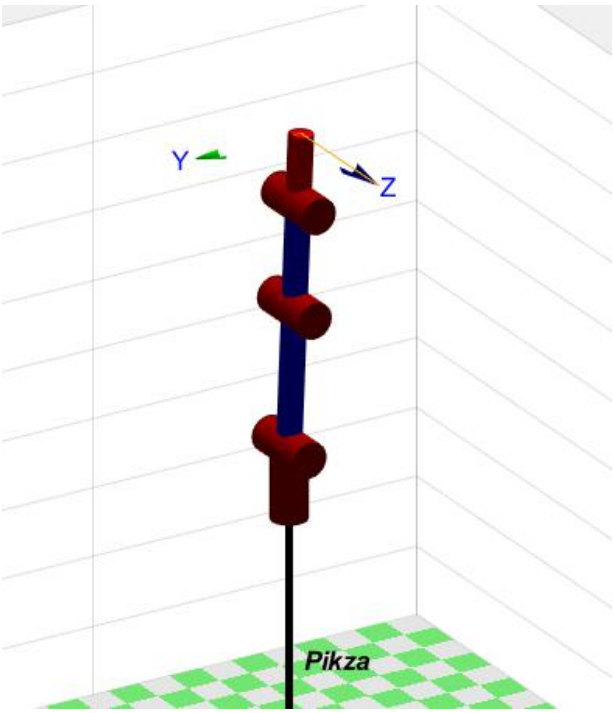


Figura 9.3 Robot en posición B

10. MATERIAL PARA FABRICACIÓN

El material que seleccionamos para la fabricación del robot Pikza Pikza es PETG, el cual es uno de los polímeros más utilizados dentro de la impresión en 3D, esta selección es debido a sus superiores características frente al PLA, como lo son mayor resistencia a impactos y a mayores temperaturas y su mayor flexibilidad.

A continuación podemos observar los datos y propiedades recuperados de la ficha técnica del PETG.

Nombre comercial	Prusament PETG
Nombre químico	Copolyester
Uso	FDM Impresión 3d
Diámetro	1.75 ± 0.02 mm
Fabricante	Prusa Polymers, Praga, Republica Checa

Figura 10.1 Identificación del PETG.

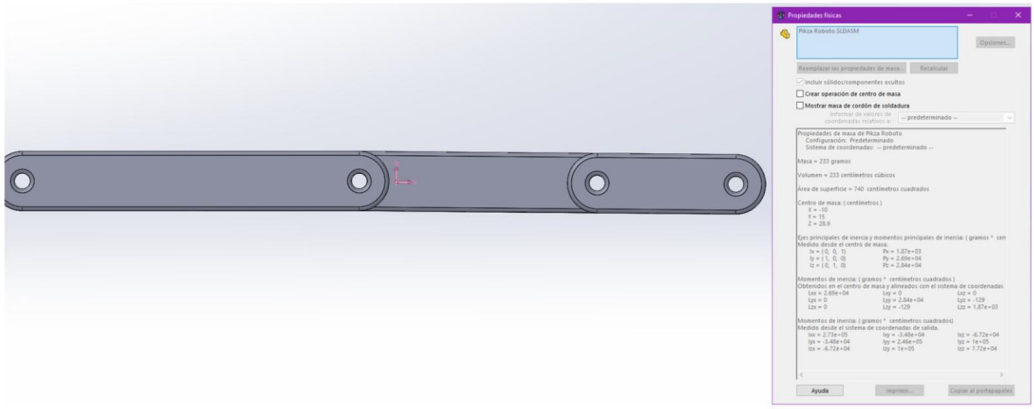
Temperatura del Nozzle [°C]	250 ± 10
Temperature de la Base Calefactable [°C]	80 ± 10
Velocidad de Impresión [mm/s]	hasta 200

Figura 10.2 Parámetros para la impresión del PETG.

Propiedades Físicas	**Valor Típico**	**Método**
Gravedad Específica [g/cm3]	1.27	ISO 1183
Absorción de humedad en 24 horas [%](1)	0.2	Prusa Polymers
Absorción de humedad 7 días [%](1)	0.3	Prusa Polymers
Absorción de humedad 4 semanas [%](1)	0.3	Prusa Polymers
Temperatura de Deflexión Térmica (0,45 MPa) [°C]	68	ISO 75
Resistencia a la Tracción del Filamento [MPa]	46 ± 1	ISO 527

Figura 10.3 Propiedades del PETG.

Así como el material también se propusieron los servomotores a partir del cálculo de masa de las articulaciones del robot, masa de los motores y masa de la caja, quedando los cálculos de la siguiente manera:



M=233g*1.27=295.91g Centro de masa: Z=28.9cm Torque: 8.55kg*cm

Servo1: 55g en Z=17cm Torque: 0.935kg*cm

Servo2: 47g en Z=29cm Torque: 1.363kg*cm

Servo3: 13.4g en Z= 36cm Torque: 0.48kg*cm

Peso estimado tapa Pizza: 100g en Z=45cm Torque: 4.5Kg*cm

Torque Total: 15.828 Kg*cm

Figura 10.4 Cálculo de torque.

Con un torque requerido de 15.828 kg*cm, como se puede observar en la Figura 10.4, se seleccionó el motor Tower Pro mg995, mostrado en la Figura 10.5 ya que ofrece un torque nominal de 15 kg*cm a 6V, teniendo un rango de operación de 4.8V a 7.2V consideramos que podría soportar el torque máximo calculado, ya que en ningún momento de la operación del robot, éste llegaría a tener las articulaciones extendidas completamente horizontales al suelo. Se seleccionaron 3 servomotores diferentes, Tower Pro mg995 (2 unidades), Tower Pro SG-5010 (2 unidades) y Tower Pro mg90s (1 unidad), teniendo torques a 6V de 15kg*cm, 6.5kg*cm y 2.2kg*cm respectivamente. Los motores mg995 serán los encargados de mover la primera y segunda articulación del brazo por el torque requerido, los SG-5010 estarán encargados de la tercera articulación y de la base giratoria ya que requieren menor torque, y finalmente el motor mg90s se encargará de la apertura y cierre del actuador final (Gripper).

Por cuestiones de presupuesto y defectos de fábrica en un servomotor, se tuvo que cambiar la configuración y selección de motores, siendo ahora 2 motores mg995 encargados del giro y

primer eslabón, un mg90s para el segundo eslabón y finalmente un sg90 para el último eslabón. Todos los motores de la marca Tower Pro. La diferencia entre los modelos sg90 y m90s es que el segundo cuenta con engranes metálicos y el motor cuenta con mayor torque a diferencia del sg90, ya que sus dimensiones son iguales.

Figura 10.5 Selección de servo motores.

11. DISEÑO CAD

Para el diseño del brazo articulado se consideró tener una caja de almacenamiento inferior para colocar dentro de esta el cableado, el controlador y a su vez la fuente de alimentación, dándole así un centro de gravedad relativamente bajo para tener estabilidad y poder realizar movimientos suaves y firmes. Se realizaron algunas modificaciones al archivo CAD original para poder reducir el peso de los eslabones y optimizar la cantidad de material de impresión utilizado. El resultado final se muestra en las Figuras 11.1 a 11.4.

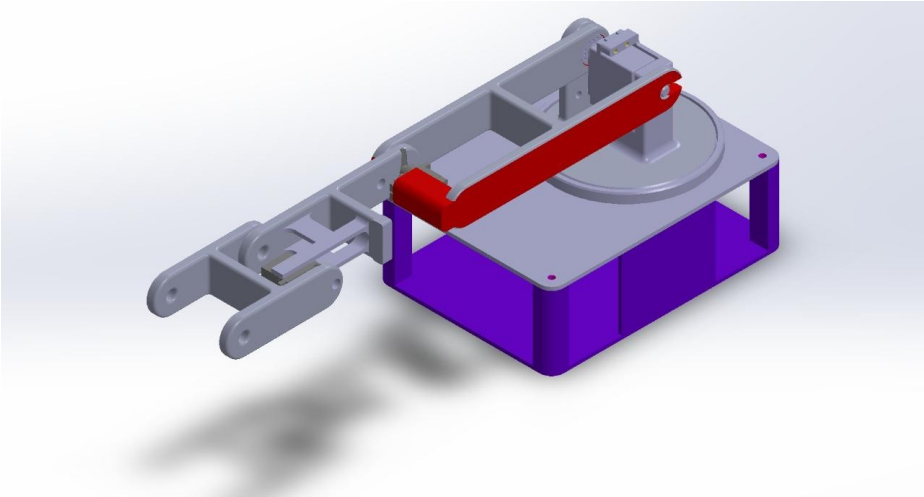


Figura 11.1 Vista isométrica

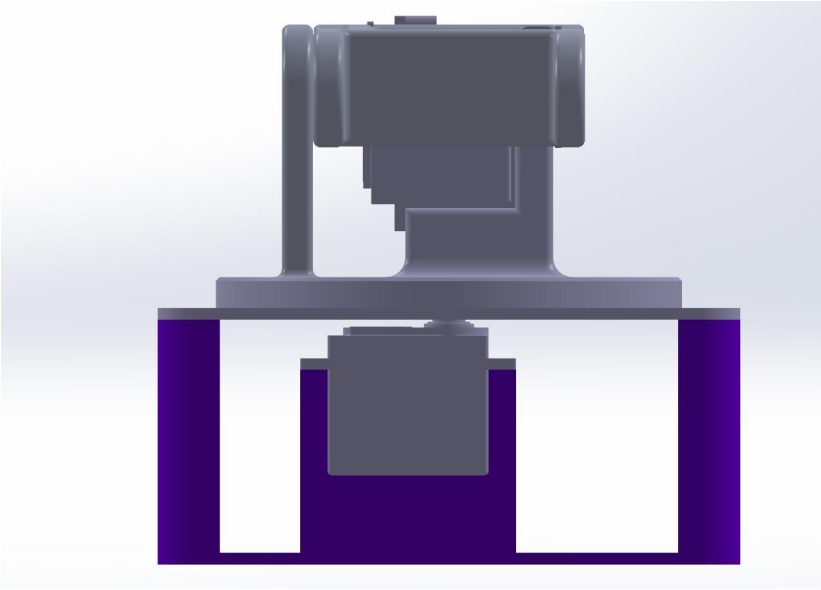


Figura 11.2 Vista frontal

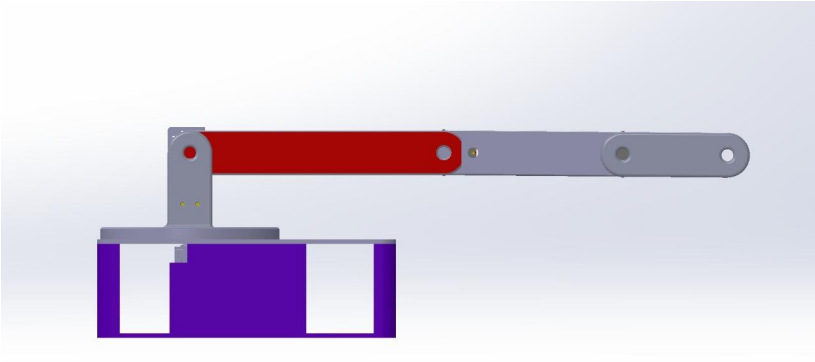


Figura 11.3 Vista lateral izquierda

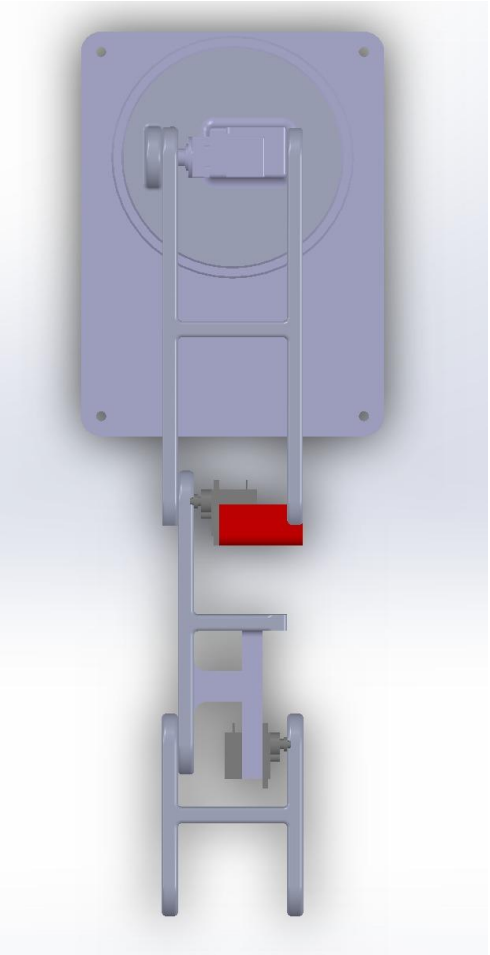


Figura 11.4 Vista superior

12. MEMORIA FOTOGRÁFICA DE CONSTRUCCIÓN

En éste apartado encontrarán una serie de fotografías donde se muestra parte del proceso de construcción de nuestro robot Pikza Pikza. Durante el proceso de ensamblado se pudo observar que uno de los motores mg90s venía defectuoso de fábrica, por lo que ese motor que se encontraba en el eslabón final se cambió por un motor de la misma

marca pero modelo sg90, que a pesar de tener menos torque, es compatible con el diseño CAD por sus dimensiones similares al mg90s.

Figura 12.1 Puesta en marcha para construcción

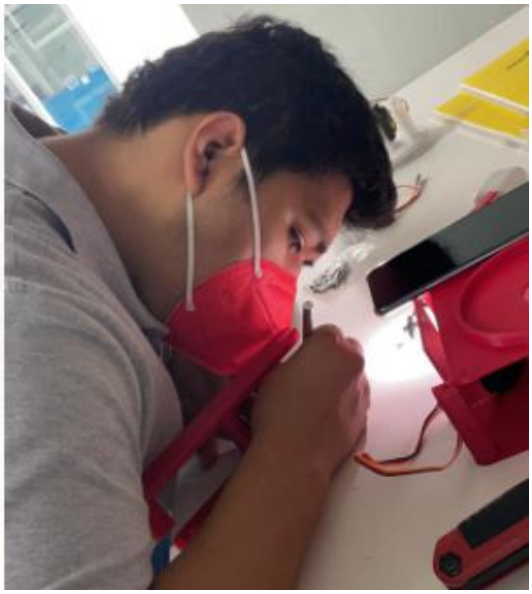


Figura 12.2 Ing. Alfredo haciendo un ajuste fino en los componentes.

Figura 12.3 Motores ensamblados en bases.

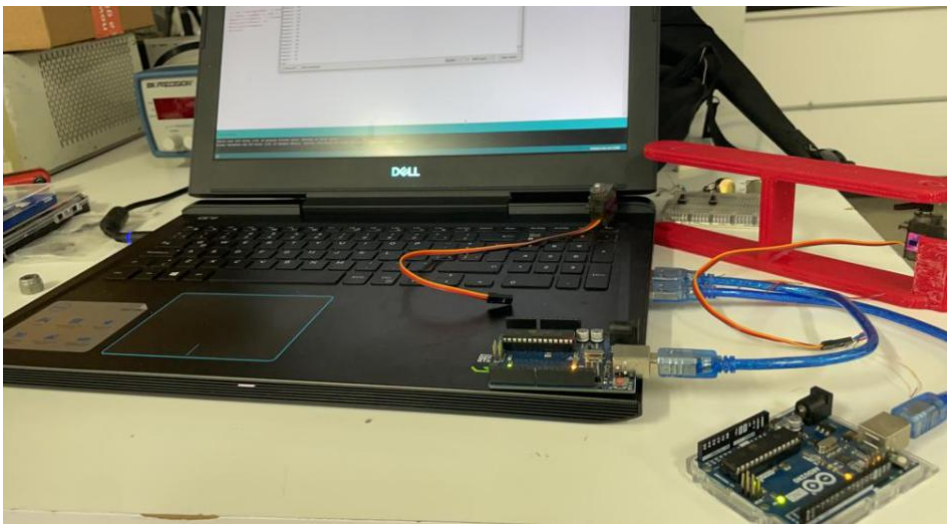


Figura 12.4 Conexión con software.

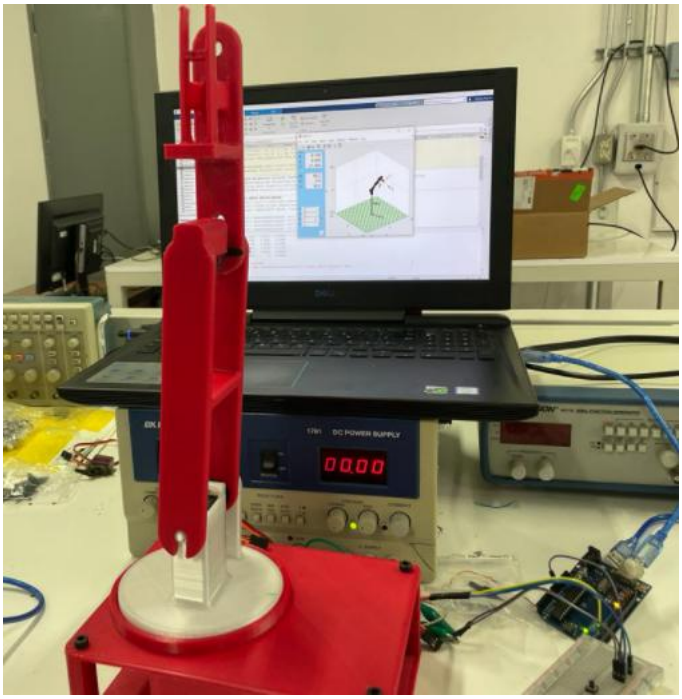


Figura 12.5 Pruebas de movimiento en ejes.

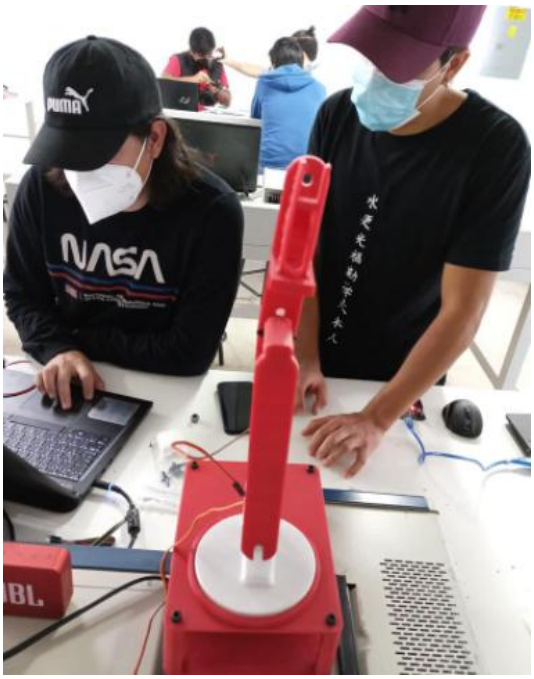


Figura 12.6 Ingenieros Dennis y Axel optimizando el código.

13. GRÁFICO EN MATLAB DE TODOS LOS ROBOTS EN LA CELDA PIZZERA

14. CONCLUSIONES

- Alfredo Serrato

  Durante todo el proceso de reportar la creación de este robot pizzero nos fuimos dando cuenta de todas las dificultades que conlleva la creación de un proyecto de esta magnitud, ya que no solo debemos realizar nuestro proyecto, si no que debe cumplir con un proceso para que todo el circuito de robots puedan realizar sus acciones si algo llega a fallar con nosotros (que somos el primer equipo en realizar su proceso) todo el proceso conjunto fallaria, entre otras revelaciones me pude dar cuenta lo necesario que es el análisis de los materiales que debemos utilizar como los servos que son fundamentales para los movimientos y estos mismos deben solapar el peso de los eslabones.

- Kathia Bustamante

Al finalizar éste proyecto hemos logrado involucrar diversos factores tanto de diseño asistido por computadora como elementos matemáticos. Notamos que la complejidad del proyecto no está en el diseño mecánico o en la programación en algún lenguaje conocido; la complejidad la hemos encontrado en el planteamiento del funcionamiento puro de nuestro sistema. Acoplar eso al resto del sistema se vuelve sencillo si hacemos las simulaciones y cálculos correctos previamente. En conclusión, crear un brazo robótico desde cero, involucra todos los conocimientos aprendidos durante éste curso de robótica industrial. Para completar nuestro proyecto, a manera de mejora, sería interesante realizar ajustes en las tablas y mejoras mecánicas.

- Dennis Pérez

  He de concluir que el diseño de un robot desde cero es una tarea bastante compleja, desde que se plantea el problema a resolver con un robot, hasta su posible solución, como posible solución me refiero a que es muy diferente las primeras fases donde la creatividad está al máximo, a cuando ya se empiezan a contemplar más detalles y particularidades, sin embargo entre más dificultad tenga un proyecto es muy recomendable tener un equipo con habilidades variadas, y eso es lo que ayudó en este equipo, por lo que al final de mi conclusión recalcó que es difícil diseñar un robot pero un correcto estudio del problema y diferentes opiniones acercan al diseño a la mejor solución posible. La complejidad de este proyecto no radicó en su en la parte de requerido de la materia, si no, en la implementación de todo lo aprendido a lo largo de la carrera, para ofrecer un proyecto acorde a los requerimientos.

- Aldo Álvarez

  Este tipo de proyecto es una aplicación de todos los conocimientos adquiridos a lo largo de la carrera, que van desde álgebra lineal hasta la programación en diferentes lenguajes de programación y comunicación IP. Durante la elaboración de este proyecto han habido diferentes complicaciones que se han ido afrontando poco a poco con la experiencia de previos proyectos, uno de estos es el consumo de corriente por la alta demanda eléctrica de los servomotores, así como los protocolos de comunicación de comunicación entre MATLAB y

ROS en una máquina virtual. Más adelante esperamos poder optimizar el diseño del robot para hacerlo no solo más estético sino económico y funcional.

- Eduardo Huerta

  Como conclusión se puede comentar que hemos aprendido el proceso inicial para el desarrollo de un robot sencillo. Ha sido un conjunto de actividades en equipo de las cuales hemos aprendido grandes lecciones. Desde cálculos, uso de software hasta modelado.

- Axel Arriola

  El presente proyecto de la creación de un robot desde cero es una buena forma de poner en la mesa todos los conocimientos adquiridos durante la carrera y la materia, ya que se tiene que considerar cada aspecto del robot, primero, su finalidad, para saber cuánto peso va a cargar, el diseño del robot, cuantos eslabones, rotacionales o de traslación, su diagrama cinemático y la forma de cada eslabón. Posteriormente tomar todos los datos y realizar la programación en MATLAB para definir los cálculos de la matriz de transformación homogénea, matriz Jacobiana (velocidades XYZ) y las matrices de posición que son las más importantes para mandar la trayectoria traducida al microcontrolador y servomotores. En resumen, cada parte del diseño y construcción de este robot son de suma importancia, ya que el error de una produce una cadena de errores, fue un verdadero reto este proyecto pero con un aprendizaje teórico y práctico.

15. ANEXOS

ANEXO 1:
CÓDIGO MATLAB CÁLCULOS MTH, JACOBIANO, TRAYECTORIA Y GRÁFICAS:

```
clear all, clc
%% ---ROBOT PIZZA (RRRR)---
syms t1 t2 t3 t4; %Variables para qs
qh=[0,0,2.7,-1.3]; %Posición HOME
qs=[t1,t2,t3,t4]; %Posición para MTH simbólica
q1=[-1.2,1,0,-1.6]; %Posición media
q2=[0,1.53,0,0]; %Posición final
```

```
l1=5; l2=20; l3=15; l4=10; %Longitudes eslabones

%% Robot
E(1)=Link('revolute', 'd', l1, 'a', 0, 'alpha', pi/2);  %Eslabón rotacional 1
E(2)=Link('revolute', 'd', 0, 'a', l2, 'alpha', 0);     %Eslabón rotacional 2
E(3)=Link('revolute', 'd', 0, 'a', l3, 'alpha', 0);     %Eslabón rotacional 3
E(4)=Link('revolute', 'd', 0, 'a', l4, 'alpha', 0);     %Eslabón rotacional 4
E(1).qlim=[-pi pi]; E(2).qlim=[-pi pi]; E(3).qlim=[-pi pi]; E(4).qlim=[-pi pi]; %Rango de movimiento
eslabones

Lirusisa=SerialLink(E,'name','Pikza') %Creación robot con eslabones E
Lirusisa.teach(qh) %Posición robot inicial HOME (qh)

%% Trayectoria de posición articular (joint)
[q,qd,qdd]=jtraj(qh,q1,50) %Trayectoria de posición articular y matrices articulares (posición, velocidad y
aceleración)
Lin={'b','LineWidth',1}; %Linea trayectoria
Lirusisa.plot(q) %Plot trayectoria posición
q_matlab=abs(q*0.318309886) %Conversion q a Matlab-servos

%% Trayectoria cartesiana
T=simplify(Lirusisa.fkine(q1)); %Matriz de posición 1 (q1) en coordenadas cartesianas
T2=simplify(Lirusisa.fkine(q2)); %Matriz de posición 2 (q2) en coordenadas cartesianas
TC=ctraj(T,T2,50); %Trayectoria cartesiana entre 2 puntos

qi=Lirusisa.ikcon(TC); %Iteraciones
qii=Lirusisa.ikunc(TC);
qiii=Lirusisa.fkine(qi);

Lirusisa.plot(qi) %Trayectoria plot

%% Matriz Transformación Homogénea simbólica y Jacobiano
MTH=double(vpa(Lirusisa.fkine(qs),4)) %MTH(qs)
XYZ=simplify(transl(MTH)) %Jacobiano
t1d=diff(XYZ,t1) %Jacobiano articulacion 1
t2d=diff(XYZ,t2) %Jacobiano articulacion 2
t3d=diff(XYZ,t3) %Jacobiano articulación 3
t4d=diff(XYZ,t4) %Jacobiano articulación 4

MTH2=double(vpa(Lirusisa.fkine(qh),4))%Matriz Transformación Homogénea (qh) HOME

%% Gráficas
figure
t=1:1:50;
subplot(3,1,1)
plot(t,q)
title('Coordenadas Articulares')
```

```
legend('q1','q2','q3','q4','Location','bestoutside')

subplot(3,1,2)
plot(t,qd)
title('Velocidades Articulares')
legend('qd1','qd2','qd3','qd4','Location','bestoutside')

subplot(3,1,3)
plot(t,qdd)
title('Aceleraciones Articulares')
legend('qdd1','qdd2','qdd3','qdd4','Location','bestoutside')

figure
subplot(3,1,1)
plot(t,TC.transl)
title('Posiciones Cartesianas')
legend('X','Y','Z','Location','bestoutside')

subplot(3,1,2)
plot(t,TC.torpy)
title('Orientación RPY')
legend('R','P','Y','Location','bestoutside')

subplot(3,1,3)
plot(t,TC.toeul)
title('Orientación Euler')
legend('Phi','Theta','Psi','Location','bestoutside')
```

ANEXO 2: CÓDIGO MATLAB MOVIMIENTO ROBOT CON ARDUINO (SERVOS)

```
clear all, clc
%% Variables q's
syms t1 t2 t3 t4; %Variables para qs
qh=[0,0,2.7,-1.3]; %Posición HOME
qs=[t1,t2,t3,t4]; %Posición para MTH simbólica
q1=[-1.2,1,0,-1.6]; %Posición media
q2=[0,1.53,0,0]; %Posición final
l1=5; l2=20; l3=15; l4=10; %Longitudes eslabones

%% ---DECLARACIÓN ROBOT PIZZA (RRRR)---
E(1)=Link('revolute', 'd', l1, 'a', 0, 'alpha', pi/2);  %Eslabón rotacional 1
E(2)=Link('revolute', 'd', 0, 'a', l2, 'alpha', 0);     %Eslabón rotacional 2
E(3)=Link('revolute', 'd', 0, 'a', l3, 'alpha', 0);     %Eslabón rotacional 3
E(4)=Link('revolute', 'd', 0, 'a', l4, 'alpha', 0);     %Eslabón rotacional 4
E(1).qlim=[0 pi]; E(2).qlim=[0 pi]; E(3).qlim=[0 pi]; E(4).qlim=[0 pi]; %Rango de movimiento eslabones

Lirusisa=SerialLink(E,'name','Pikza') %Creación robot con eslabones E
```

```
Lirusisa.teach(qh) %Posición robot inicial HOME (qh)

%% Trayectoria de posición articular (joint)
[q,qd,qdd]=jtraj(qh,q1,30) %Trayectoria de posición articular y matrices articulares (posición, velocidad y
aceleración)
Lirusisa.plot(q) %Plot trayectoria posición
q_ML=abs(q*0.318309886) %Conversion matriz posición a matlab

%% CONTROL SERVOS ARDUINO-MATLAB
port = 'COM4'; % Puerto Serial Arduino
board = 'Uno'; % Modelo Arduino
arduino_board = arduino(port, board, 'Libraries', 'Servo'); %Objeto Arduino-Servo

servo_motor1 = servo(arduino_board, 'D12'); %Objeto servo 1 y PIN PWM
servo_motor2 = servo(arduino_board, 'D9');  %Objeto servo 2 y PIN PWM
servo_motor3 = servo(arduino_board, 'D10'); %Objeto servo 3 y PIN PWM
servo_motor4 = servo(arduino_board, 'D11'); %Objeto servo 4 y PIN PWM

%% Movimiento robot a partir de trayectoria articular (jtraj)

for i=1:30 %Ciclo para obtener los valores de la matriz posicion para cada articulación

   writePosition(servo_motor1, q_ML(i,1)); %Escribe valor para servo 1, columna 1, fila i
   writePosition(servo_motor2, q_ML(i,2)); %Escribe valor para servo 2, columna 2, fila i
   writePosition(servo_motor3, q_ML(i,3)); %Escribe valor para servo 3, columna 3, fila i
   writePosition(servo_motor4, q_ML(i,4)); %Escribe valor para servo 4, columna 4, fila i

   current_position1 = readPosition(servo_motor1); %Lee la posición actual del servo
   current_position2 = readPosition(servo_motor2);
   current_position3 = readPosition(servo_motor3);
   current_position4 = readPosition(servo_motor4);

   current_position1 = current_position1 * 180; %Conversión de posición actual servo a grados
   current_position2 = current_position2 * 180;
   current_position3 = current_position3 * 180;
   current_position4 = current_position4 * 180;

   fprintf('Servo #1 current position is %d\n', current_position1); %Imprime valor actual servo
   fprintf('Servo #2 current position is %d\n', current_position2);
   fprintf('Servo #3 Current position is %d\n', current_position3);
   fprintf('Servo #4 current position is %d\n', current_position4);

end

writePosition(servo_motor4, 0); %Regresa servo a posición 0
writePosition(servo_motor3, 0);
writePosition(servo_motor2, 0);
```

```
pause(0.5)
writePosition(servo_motor1, 0);
```