Proyecto Final
Robótica

2020
Guarise, Renzo 12262
Trubiano, Lucas 12289

Resumen

En el presente informe vamos a desarrollar aspectos técnicos referidos al robot Scara ABB IRB 910SC, este robot nosotros lo usamos para resolver una aplicación de separación de residuos urbanos para su posterior tratamiento y reciclaje. Lo que encontraremos a lo largo del informe será una introducción a la problemática, una especificación de las características del robot empleado (IRB 910SC) y una serie de conceptos necesarios para poder planificar las tareas y trayectorias que le permitirán al robot resolver su tarea.

Índice

Introducción

La problemática planteada fue la automatización de la recolección de residuos en una planta de reciclaje. Dado que nuestra aplicación está orientada a los residuos domésticos decidimos que lo mejor para la aplicación era un robot del tipo scara, ya que son robots rápidos, ágiles y de tamaño medianos.

De estos robot elegimos un robot comercial, el ABB IRB910 SC, dado que es un robot conocido y de una marca líder.

Para este problema propusimos que nuestro robot se posicionara en frente de una cinta transportadora de aproximadamente unos 40 cm de ancho, sobre la cual se depositan los residuos a separar, y sobre la periferia del robot se encuentran los recipientes correspondientes a cada tipo de residuo, de esta manera el entorno de trabajo de nuestro robot se vería como se aprecia en la figura 1.

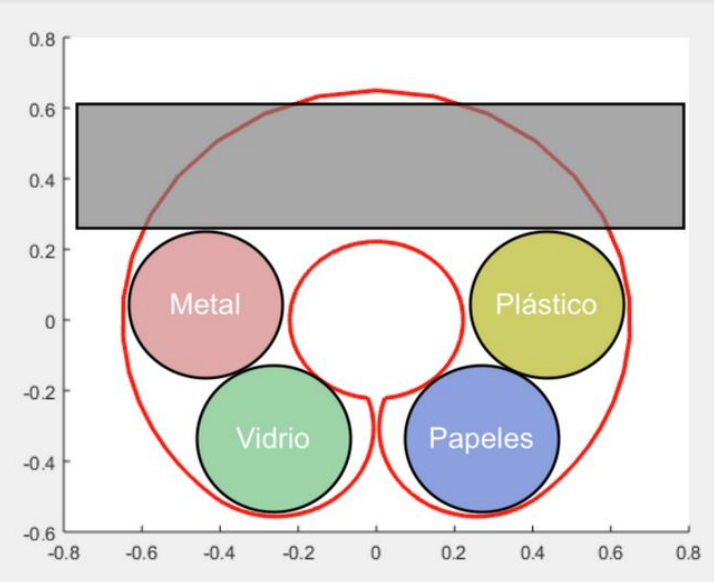


Figura 1.

La curva roja denota el espacio de trabajo del robot, los círculos de colores, los diferentes recipientes para el depósito de residuos y el rectángulo, la cinta transportadora.

ABB IRB910 SC

Como ya dijimos este es un robot del tipo scara (en la figura 2 se puede apreciar) . Algunas de sus características extraídas del datasheet que creemos importantes son:

- Peso: 25.5 kg
- Capacidad máxima de carga: 6 kg
- Longitud máxima: 0.650 mm
- GDL: 4
- Voltaje de entrada: 200-600 V 50/60 Hz
- Potencia consumida: 200 W

figura 2.

Como vemos estos son robots livianos, que alcanzan grandes velocidades de trabajo y fáciles de integrar en espacios de trabajos con otros robots. También son robots muy precisos y están diseñados para trabajar en ambientes industriales por lo que son resistentes al polvo y al agua. Además tienen un buen espacio de trabajo como se ve en la figura 3, ideal para nuestra aplicación.

Como vemos también estos presentan consumo de potencia bajo.

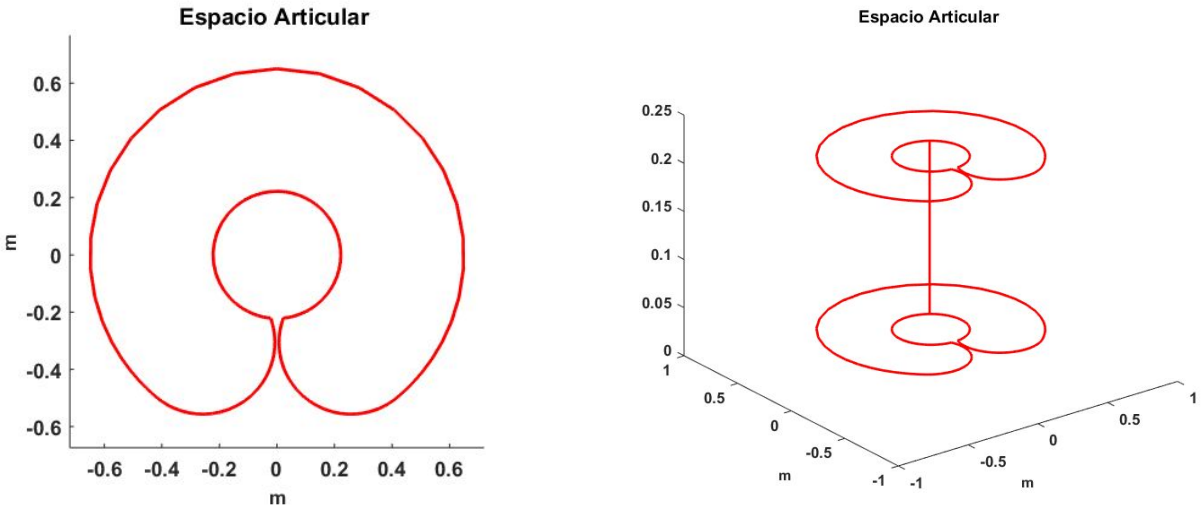


figura 3.

Como efector final para nuestra aplicación elegimos un gripper ya que requerimos que nuestro robot tenga la capacidad de agarrar objetos de distintas formas y materiales, por lo que necesitamos un componente con buena sujeción y adaptabilidad.

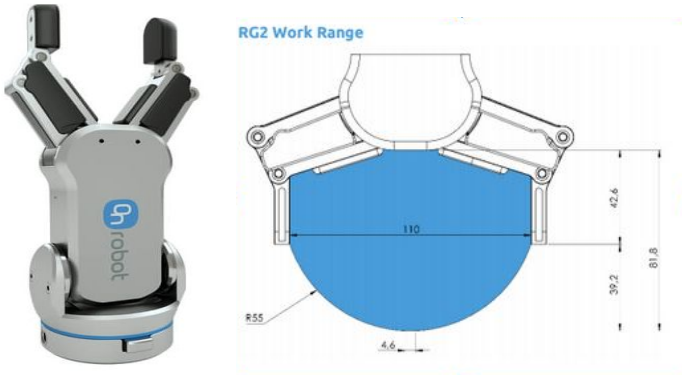
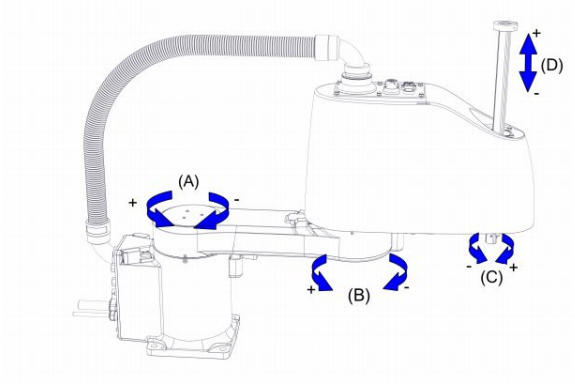


figura 4.

En la figura 4 se muestra el gripper con su capacidad de agarre.

Denavit-Hartenberg:

Para obtener la matriz de parámetros de DH seguimos la convención propuesta por la cátedra al robot seleccionado y nos apoyamos en los documentos de las especificaciones técnicas del robot que se pueden encontrar en la página de ABB. Las articulaciones son las siguientes:



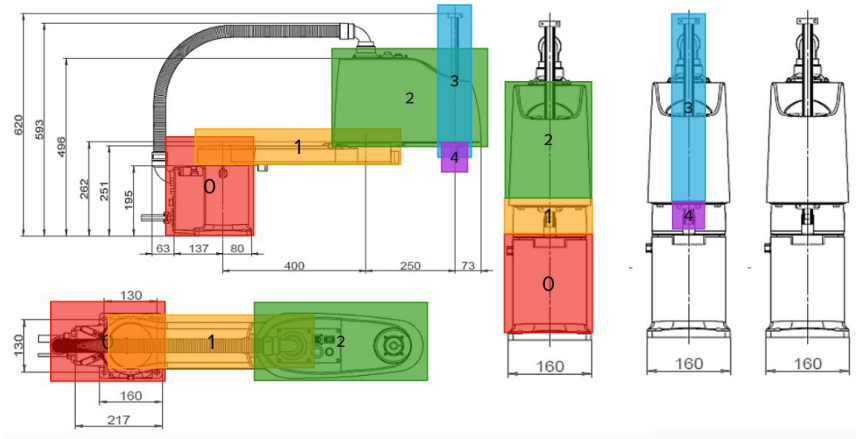
xx1500002631

Posi-tion	Description	Posi-tion	Description
A	Axis 1	B	Axis 2
C	Axis 4	D	Axis 3

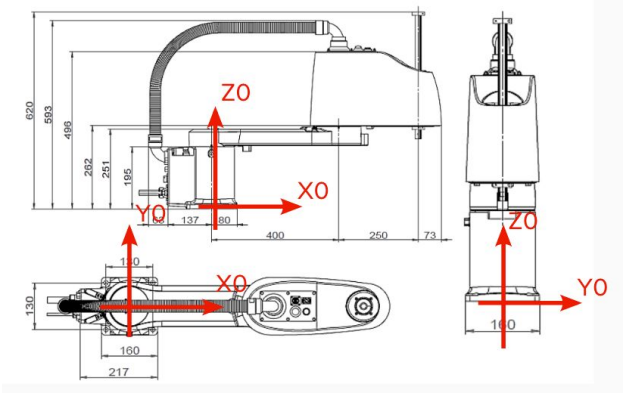
figura 5.

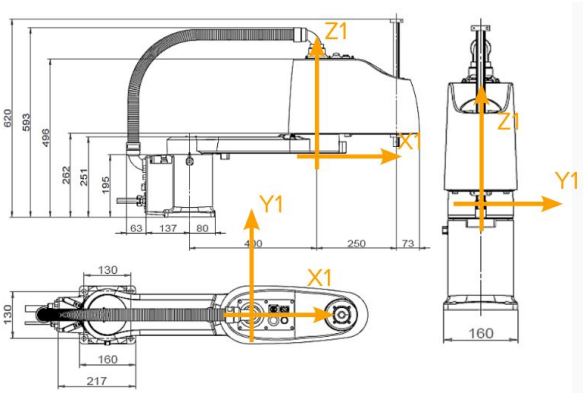
Y los ejes y eslabones que planteamos según la convención quedaron de la siguiente forma:

- **Eslabones**

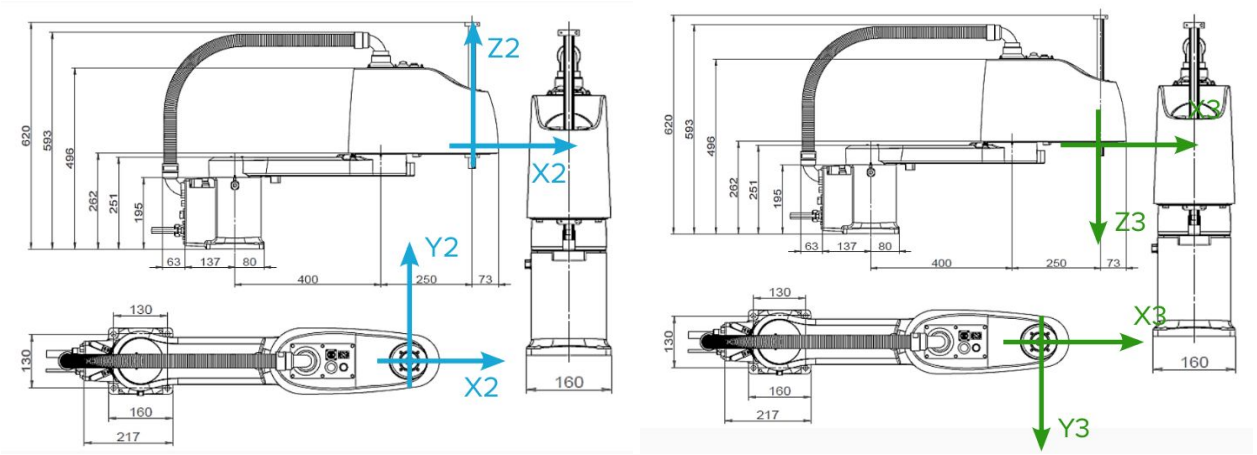


- **Sistema 0** **Sistema 1**





- **Sistema 2** **Sistema 3**



Sistema 4

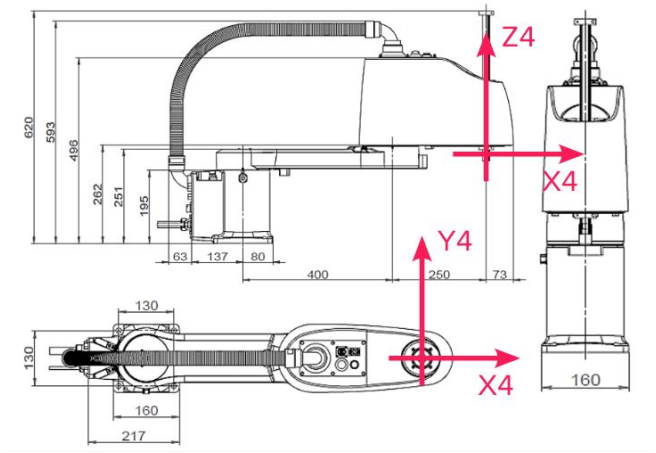


figura 6.

Así llegamos a obtener la siguiente tabla de DH

- **Tabla de parámetros de DH:**

Sistema	Tita	d	a	alfa	tipo
1	q1	199.2 mm	400 mm	0°	0 (rot)
2	q2	58.5 mm	250 mm	180°	0 (rot)
3	0°	d3	0 mm	180°	1 (tras)
4	q4	0 mm	0 mm	0°	0 (rot)

Nuestro robot tiene en cada articulación los siguientes límites articulares dados por el fabricante:

Robot motion

Axis	Type of motion	Working range
Axis 1	Rotation motion	-140° to +140°
Axis 2	Rotation motion	-150° to +150° [i]
Axis 3	Linear motion	-180 mm to 0 mm
Axis 4	Rotation motion	Default: -400° to +400° Maximum revolutions: -864 to +864 [ii]

figura 7.

Teniendo en cuenta nuestro efector final establecimos las matrices de **base** y **tool** las cuales son:

```
Tbase =

     0   -1.0000         0         0
1.0000         0         0         0
     0         0    1.0000    0.2130
     0         0         0    1.0000
```

```
Ttool =

1.0000         0         0         0
     0    1.0000         0         0
     0         0    1.0000   -0.2500
     0         0         0    1.0000
```

Se quiso que la distancia del sistema 4 de nuestro robot al piso fuera la misma que la que tenía este sin el efecto final. Cabe destacar que las magnitudes de las distancias en z de ambas matriz no son iguales porque nosotros la distancia d4 no la tuvimos en cuenta en la tabla dh, la podríamos tener poniendo d4=-0.037, si no que la tuvimos en cuenta en la matriz tool.

Con todo este análisis modelamos el robot y lo mostramos con el método de matlab **.teach** que no solo nos permite visualizar el robot sino que también nos permite mover sus distintas articulaciones y visualizar el movimiento.

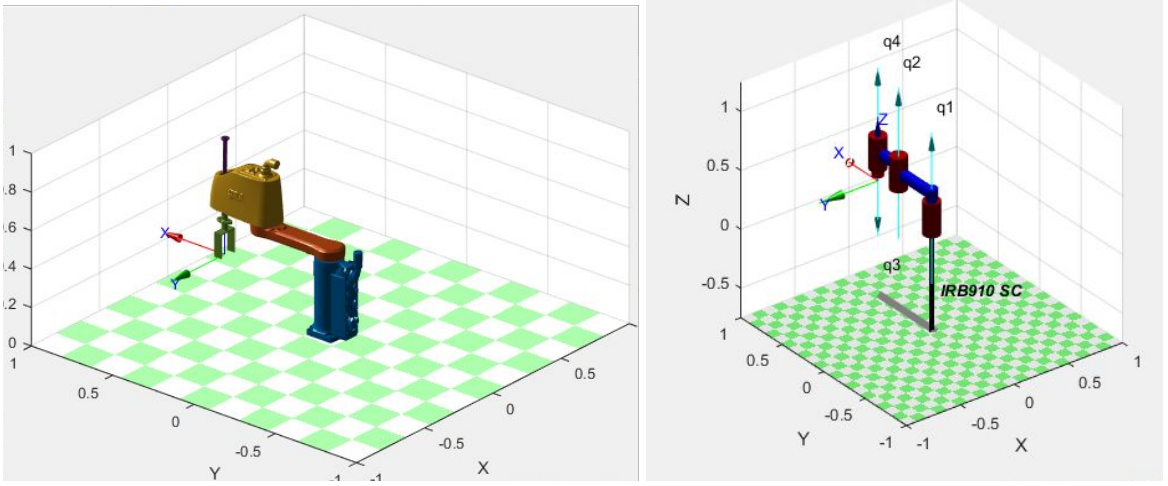


figura 8.

En la figura 8 podemos ver a la derecha el modelo por defecto que plotea el toolbox con los ejes articulares y en la izquierda un modelo al que le cargamos los stl para una vista más realista. Nótese que la base del modelo de la izquierda está sobre dimensionada para que esta no se grafique en el aire, ya que en nuestra aplicación el robot necesita un suplemento en la base, esta modificación se hizo únicamente con fines de tener una vista más agradable de las simulaciones pero no se modificó ninguna variable física del robot por lo que los cálculos no varían.

Cinemática Directa

En cuanto a la cinemática directa del robot primero ploteamos el espacio de trabajo y verificamos que fuera adecuado para la aplicación, la gráfica obtenida se ve en la figura 3 y ese gráfico aplicado a nuestra aplicación se ve en la figura 1.

Una vez que verificamos el espacio de trabajo y tenemos una visión general de cómo es, pasamos a elegir el efector final para nuestra aplicación y para así poder resolver la cinemática directa teniendo en cuenta este. El efector que elegimos fue un gripper ya que es muy versátil para agarrar objetos de formas variadas y se puede ver en la figura 4. También tuvimos que tener en cuenta la matriz base.

Luego con toda esta información pudimos resolver la cinemática directa del robot, es decir partiendo de determinadas coordenadas articulares llegamos a las coordenadas cartesianas del efector final. En la figura 9 se puede apreciar las ecuaciones para obtener cada elemento de la matriz homogénea que nos dice la posición y orientación del efector final para una determinada coordenada cartesiana, pero nosotros este cálculo lo realizamos generalmente con el método **.fkine** característicos de las instancias de la clase **SerialLInk.**

```
[ -sin(q1 + q2 + q4), -cos(q1 + q2 + q4), 0, - a2*sin(q1 + q2) - a1*sin(q1)]
[  cos(q1 + q2 + q4), -sin(q1 + q2 + q4), 0,   a2*cos(q1 + q2) + a1*cos(q1)]
[                  0,                  0, 1,         d1 + d2 - q3 - 37/1000]
[                  0,                  0, 0,                              1]
```

figura 9.

También implementamos una función de cinemática directa a la que llamamos **verificación**. Esta función tiene como entradas una posición articular y un objeto robot, y devuelve un booleano que nos indica si las posiciones articulares son correctas, si estas son correctas las mismas coordenadas y si no las coordenadas articulares más cercanas dentro de los límites articulares y la matriz homogénea correspondiente a la posición articular pasada.

Por ejemplo, si nosotros le pasamos a la función **verificación** la posición articular q1=[pi/2,pi,0,1], esta nos da como resultado lo mostrado en la figura 10:

```
bool =

     0


coord_articulares =

    1.5708    2.6180         0    1.0000


n =

     2


T =
    0.8887   -0.4586         0   -0.1835
    0.4586    0.8887         0    -0.125
         0         0         1    0.2207
         0         0         0         1
```

figura 10.

Vemos que el valor de la articulación 2 no es pi dado que este está fuera de los límites articulares y por lo tanto nuestra función nos da el valor más cercano a este, nos dice que el booleano es falso, el error se produce en la segunda articulación y nos devuelve la matriz homogénea.

Cinemática Inversa:

Para llegar a la cinemática inversa del robot seguimos el método geométrico explicado en clases. Para un mejor análisis partimos de los eslabones 1 y 2 donde tenemos un clásico problema de codo arriba y codo abajo, como se ve en la figura 11. De esta manera procedimos a calcular q1 y q2.

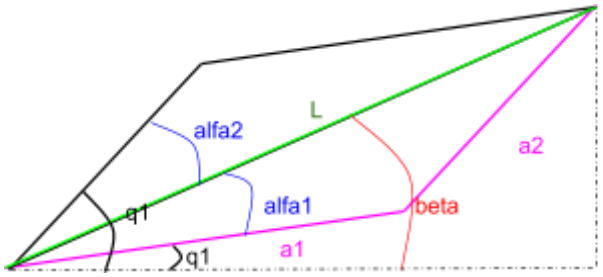


figura 11.

Como podemos apreciar q1 tiene dos soluciones posibles y para su cálculo procedimos de la siguiente manera:

$$\beta = arctan(y_4/x_4) \quad (1)$$

$$\alpha_{1,2} = arcos((a_2^2 - l^2 - a_1^2)/(-2.a_1.l)) \quad (2)$$

$$q_1 = \beta - \alpha_{1,2} \quad (3)$$

Donde x4 ,y4 y z4 son las coordenadas del efector final con respecto a la base del robot.

Luego dado que tenemos q1 podemos calcular la matriz de transformación homogénea del sistema 1 con respecto a la base y así para cada q1 obtenemos:

$$(x_4^1, y_4^1) = inv(T_1^0) * (x_4^0, y_4^0, 0, 1)' \quad (4)$$

$$q2 = artan(y_4^1/x_4^1) \quad (5)$$

Luego procedimos al cálculo de q3:

$$q_3 = Z_{max} - z_4 \quad (6)$$

Donde Zmax es el máximo valor alcanzable por el efector en el eje z.

Y para el cálculo de q4 dado que ya tenemos q1,q2 y q3 y teniendo la matriz de transformación consigna lo podemos calcular de la siguiente manera:

$$T_4^0 = T_1^0 * T_2^1 * T_3^2 * T_4^3 \quad (7)$$

$$T_4^3 = T_0^3 * T_4^0 \quad (8)$$

Y así con la función tr2rpy() del toolbox obtenemos el ángulo yaw de esta matriz que va a ser nuestro ángulo q4.

Hay que notar que nuestro robot puede tener como máximo 8 soluciones para una misma coordenada cartesiana, dado que para q1 pueden existir dos soluciones siempre que se esté dentro del límite articular y para cada q4 dependiendo el ángulo exigido puedo tener hasta 3 soluciones, ya que estos tienen un límite articular de +-400°.

Para el cálculo de la cinemática inversa nosotros creamos una función a la que llamamos **inversa,** esta función recibe como parámetros un objeto Robot, una matriz de transformación homogénea y un vector semilla, como resultado devuelve un booleano que es verdadero si las posiciones cartesianas deseadas son alcanzables por el robot y falso si no lo son, y las coordenadas articulares más cercanas al vector semilla si estas son alcanzables y si no devuelve las coordenadas articulares más próximas a las deseadas.

Por ejemplo, si le pasamos a nuestra función la posición p=[0,-0.5,0.1] expresada como matriz homogénea y un vector semilla q=[0 0 0 0] nos devuelve:

```
coord_art1 =

    2.4435    1.6236    0.1207    0.6454


bool =

     0
```

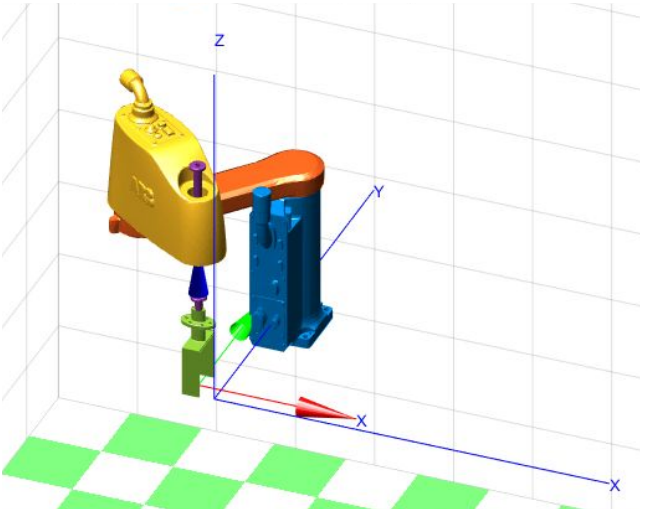


figura 12.

Vemos que el bool es falso por lo que el robot no puede llegar a esa posición debido a sus límites articulares, y por lo tanto nos devuelve la posición más cercana posible. En la figura 12 se puede ver como el robot queda posicionado cerca de la posición deseada pero no la alcanza.

Jacobiano:

Para hallar el jacobiano simbólico de nuestro robot utilizamos el motor de cálculo simbólico nativo de matlab. En la figura 13 se muestra el **jacobiano** obtenido (**J**), que vemos que depende únicamente de de a1, a2, q1 y q2, el **jacobiano reducido (Jr)**, obtenido al sacar las

filas de ceros y el **determinante de Jr (detJr).** Vemos que el **determinante** depende únicamente de a1,a2 y q2, además este va a ser **cero** para **q2= n*pi**, siendo n un número entero.

Además podemos notar que nuestro robot va a estar en una singularidad únicamente sobre el límite del espacio de trabajo, dentro de este el robot no se va a encontrar con ninguna, ya que q2 nunca va a ser cero.

Si bien en el Tp6 dijimos que nuestro robot no iba a trabajar en una singularidad luego de un análisis más profundo del área de trabajo del robot notamos que nuestro robot puede caer en singularidades como se ve en la figura 15, y en la figura 14 se ve al robot en las cercanías de una singularidad, vemos que para estos puntos un eje del elipsoide tiende a cero lo que indica que nuestro robot no puede alcanzar velocidades en esa dirección. Aclaramos que en la imagen de la derecha de la figura 14 graficamos la elipse sin tener en cuenta la dimensión z, ya que la velocidad nunca se ve limitada en esta dirección y por lo tanto no es relevante su análisis porque permanece invariante.

Para conocer cuando se está en una singularidad se puede recurrir a distintas magnitudes escalares como el número de condición, el determinante y la manipulabilidad. El número de condición cuando se está cerca de una singularidad nos va a dar muy grande, el determinante muy chico y la manipulabilidad muy chica.

```
J =

[ - a2*sin(q1 + q2) - a1*sin(q1), -a2*sin(q1 + q2),  0, 0]
[   a2*cos(q1 + q2) + a1*cos(q1),  a2*cos(q1 + q2),  0, 0]
[                              0,                0, -1, 0]
[                              0,                0,  0, 0]
[                              0,                0,  0, 0]
[                              1,                1,  0, 1]


Jr =

[ - a2*sin(q1 + q2) - a1*sin(q1), -a2*sin(q1 + q2),  0, 0]
[   a2*cos(q1 + q2) + a1*cos(q1),  a2*cos(q1 + q2),  0, 0]
[                              0,                0, -1, 0]
[                              1,                1,  0, 1]


detJr =

-a1*a2*sin(q2)
```

figura 13.

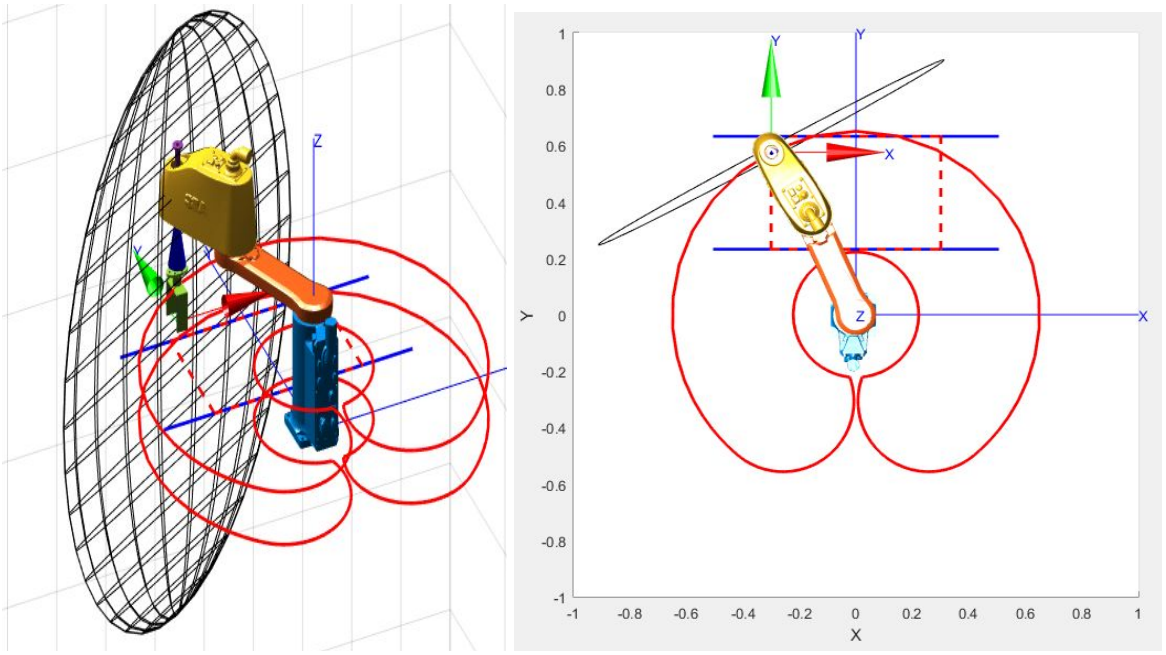


figura 14.

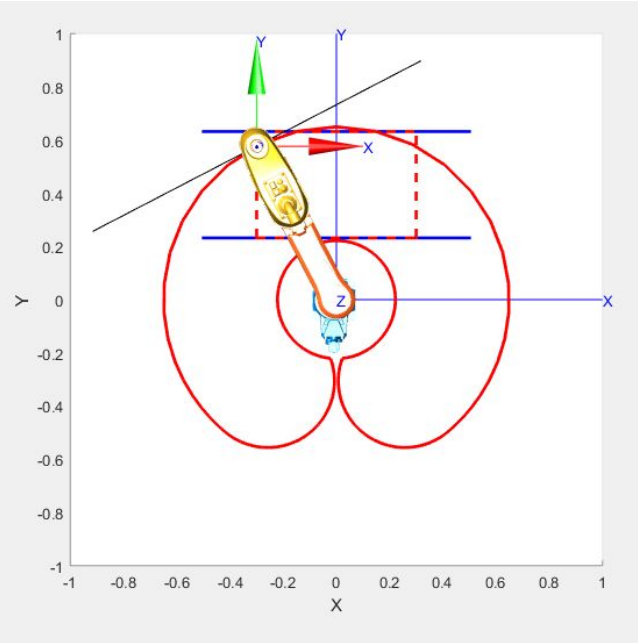


figura 15.

Sensores y Actuadores:

Actuadores

Algunos **motores**/actuadores que consideramos a **priori** fueron **motores eléctricos** ya que neumáticos tienen el problema de no ser tan precisos en las posiciones que va tomando, e hidráulicos serían muy lentos para la aplicación y si bien tienen bastante torque no necesitamos tanto.

Por ende de este primer filtro pensamos en motores eléctricos y de esos analizamos motores que pudieran dirigirse o moverse a posiciones específicas por lo cual consideramos **motores pap** o **servomotores** como segunda instancia.

Y ya que nosotros seleccionamos un **robot comercial** que ya trae motores y drivers incluidos tratamos de deducir qué tipo de motores podría tener nuestro robot. En internet no encontramos mucho acerca de qué motores tenía en cada articulación, ni qué tipos de motor usaba ya que eso entendemos que es información reservada de la empresa.

Pero si pensamos que los motores **pap no** serían **convenientes** ya que son **lentos** en comparación con las velocidades que logra nuestro robot. Asique casi por descarte decidimos que nuestro **robot utiliza servomotores**, estos servomotores nosotros los elegimos **brushless** y de **AC** porque los creemos mejores para nuestra aplicación ya que se trata de una aplicación industrial entonces tiene sentido hablar de motores AC y ya no tanto en CC, además que los brushless son de **altas velocidades**, y combinado con alguna **caja reductora** podrían **moverse rápidos**, **precisos** con pares o **torques considerables**.

Algunos servomotores que encontramos y que nos podrían servir para nuestra aplicación son los siguientes (solo considerando aspectos cinemáticos, no dinámicos, ni económicos):

Servomotor brushless - BS35 series - Mini Motor - AC / 220V / IP65

Servomotorreductor brushless - BSE35 series - Mini Motor - trifásico / planetario / coaxial

Y a esos motores debemos manejarlos con servo drivers capaces de controlar posición y velocidad de los motores en función de las consignas dadas, un servo driver que encontramos fue el siguiente:

iPOS4850 BX-CAN-STO - Servo-variador inteligente by Technosoft

Los servomotores cuentan con sensores encoders y resolvers que le mandan información al servo driver para cerrar el lazo de control, pero de los sensores internos hablaremos en el apartado de sensores.

Sensores

En cuanto al apartado de sensores primero hablaremos de los sensores externos o que están más relacionados con la tarea del robot y para nuestro caso pensamos en un sistema de **cámaras de visión artificial** para **reconocer** en primer lugar los **objetos** que queremos separar según su tipo de residuo y en segundo lugar además de reconocer el objeto, poder conocer su **posición** dentro del **espacio de trabajo.** A continuación detallamos un sensor que encontramos de este estilo con su aplicación incluida.

Sensores Vision 2D InspectorP62x

App Sensores Vision 2D InspectorP62x

En caso de ser necesario también podríamos incluir algún pequeño sensor de proximidad para detectar si efectivamente agarramos el residuo o no, pero depende de qué tan completo sea nuestro sistema de visión artificial quizás no necesitemos de este sensor.

Sensores de proximidad capacitivos | CM | SICK

Respecto a los sensores internos tenemos los sensores **encoders** y **resolvers** que vienen incluidos en los servomotores para poder controlar la **posición** y **velocidad** de los servo motores, además de esto el robot viene con algunas protecciones de temperaturas y corrientes excesivas que son implementados internamente pero no aclara cuántos son esos sensores ni de qué tipos.

Los encoders y resolver si bien están físicamente acoplados al motor, sus señales son enviadas a los servodrives y ahí es donde se cierra el lazo de control propio de ese motor. A esos servodrives nosotros deberíamos enviarle las consignas con los perfiles de posición, velocidad y aceleración deseados.

Fuente de alimentación

Según las especificaciones del fabricante del robot ABB IRB 910SC, este robot se alimenta con una tensión de **220V** y su potencia de consumo está en el orden de **71 W** (para frenos activados) o **127,6 W** (para frenos desacoplados). Con esto cubrimos lo que es la alimentación tanto de los motores (servomotores), como de los servo drives internos para el control de posición y velocidad.

Además debemos alimentar el sistema de **visión artificial** que se puede alimentar con corriente continua de **12V** a **24V** y tiene un consumo promedio de **4W.**

Para alimentar la lógica de nuestro robot proponemos la siguiente fuente:

Fuente siemens

Planificación y Generación de Trayectorias:

Para la planificación de trayectorias lo primero que hicimos fue plantear escenarios posibles que tuviera que resolver nuestro robot y uno de ellos es el que se ilustra en la figura 16.

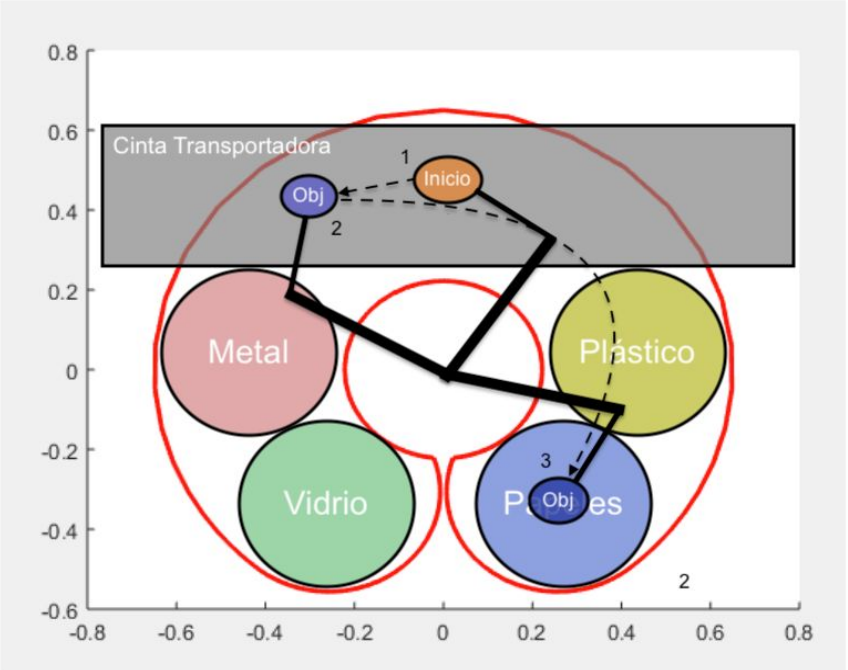


figura 16.

Este robot debe ser capaz de moverse en distintas trayectorias algunas rectas y otras curvas para lograr llevar un objeto (residuo) desde una cinta transportadora hasta un depósito de residuos correspondiente.

Los requisitos de este movimiento son los siguientes:

1) Como primer movimiento el robot debe posicionarse en el centro de la cinta transportadora y luego mediante los distintos sensores localizar un residuo objetivo.

2) Después, el robot debe **alinearse con la trayectoria** que trae el **objeto**, es decir, ya que el objeto se viene moviendo en línea recta nuestro robot debe alinearse en la misma línea.

3) Luego en **línea recta** debe buscar al objeto (que viene en la misma trayectoria recta), entonces nos aseguramos de agarrarlo y no de golpearlo ni correrlo de lugar.

4) Una vez capturado el objeto deberíamos **orientar** el robot en una posición favorable respecto del **punto final** al que debe llevar el objeto. Si no lo hiciéramos podríamos querer llegar a un tacho en los extremos y no llegaríamos porque el robot quedó muy enredado y fuera de los límites articulares.

5) La **trayectoria** hacia el punto **final**, una vez orientado el robot, puede ser en **línea curva** ya no hace falta que se mueva en línea recta. Entonces solo debemos interpolar en las coordenadas articulares.

Buscando cual era la mejor forma de interpolación para la trayectoria entre dos puntos para nuestro robot, a partir de 3 puntos en el espacio P1, P2 y P3, programamos lo siguiente:

P1 = [0, 0.432, 0.220] P2 =[0.5 -0.1,0.05] P3=[0.5, 0.3, 0.05]

Solución Interpolando en el espacio articular (jtraj):

P1 a P2 P1 a P3

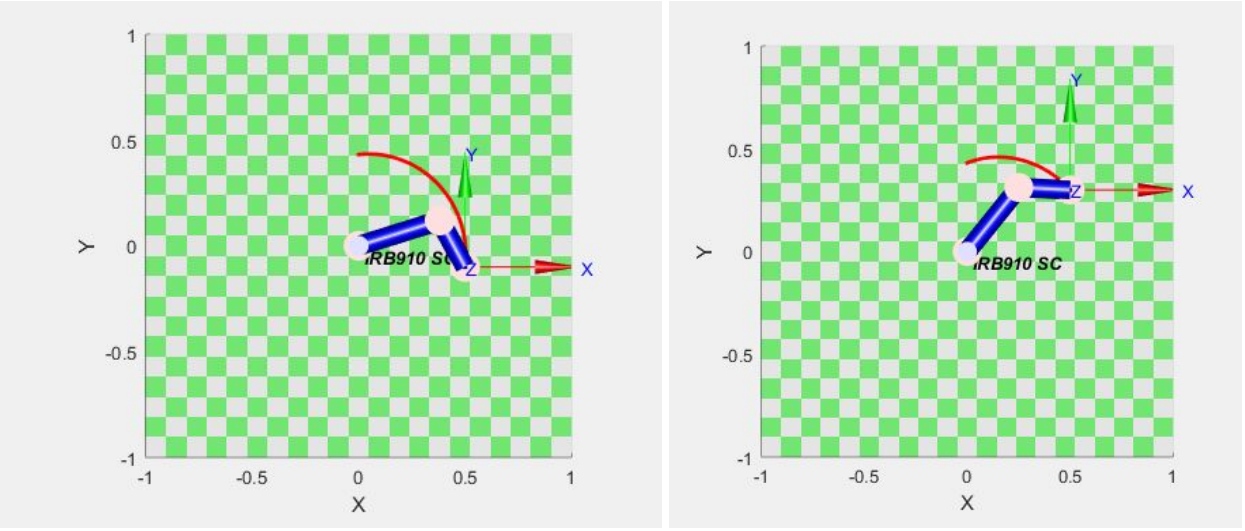


figura 17.

Solución Interpolando en el espacio cartesiano (ctraj):

P1 a P2 P1 a P3

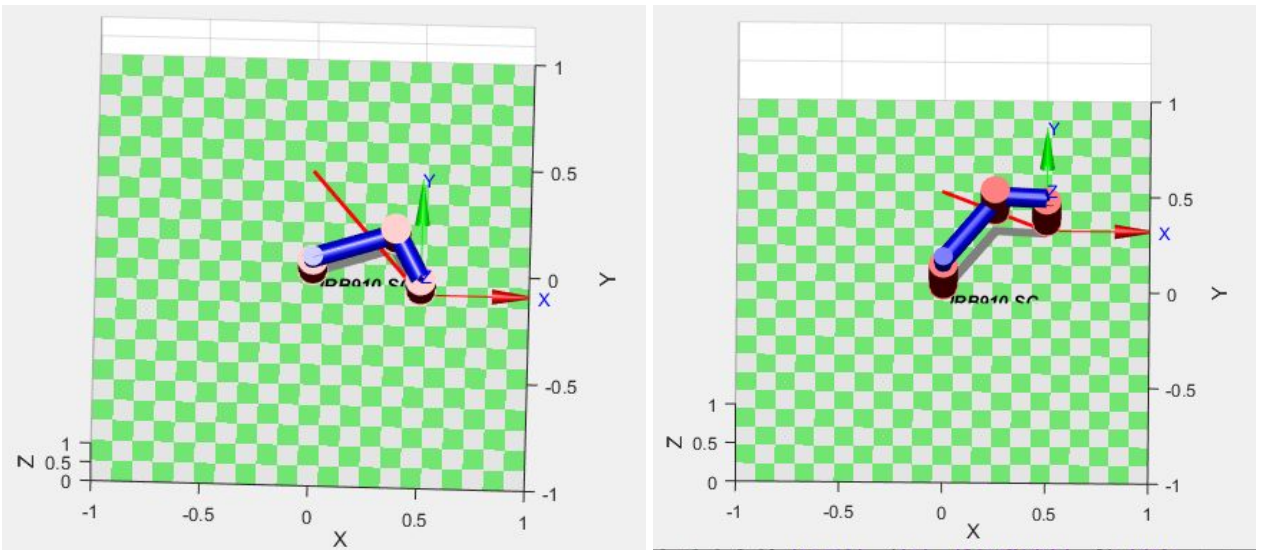


figura 18.

De estas trayectorias sacamos curvas de posiciones, velocidades y aceleraciones articulares así como también posiciones, velocidades y aceleraciones en el espacio cartesiano y analizandolas llegamos a las siguientes conclusiones.

El **robot scara** se puede mover de manera más suave, natural, más rápido y sin aceleraciones tan grandes usando **interpolación jtraj en el espacio articular**.

Esto se ve porque la mayoría de las coordenadas articulares con **ctraj** presentan **picos** de **velocidad** y **aceleración** más grandes y eso indica que exigimos más al robot. En cambio con **jtraj** las **velocidades** y las **aceleraciones** son **menores** indicando que se pueden hacer todavía **más rápido**. Y no solo son menores sino que son curvas **más suaves** tanto en velocidad como en aceleración.

Interpolación multipunto.

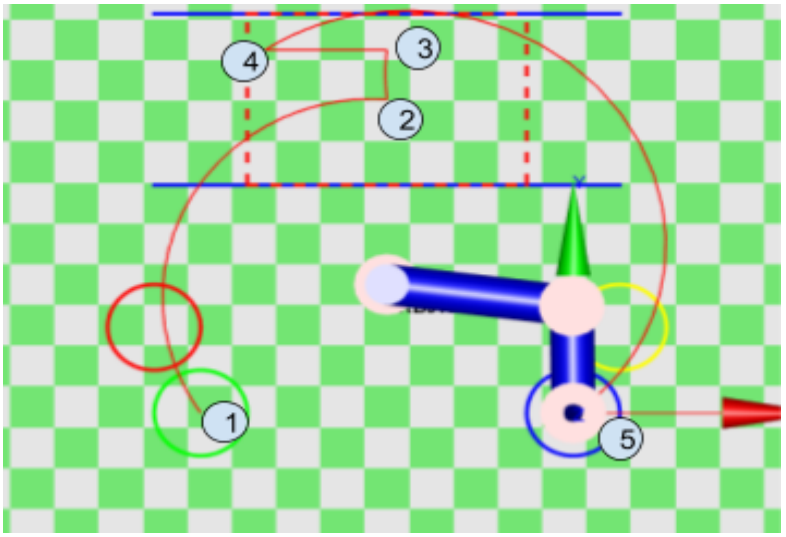


figura 19.

En la figura anterior se ve una trayectoria clásica de nuestro robot, el cual parte del **punto 1** (donde terminó la trayectoria anterior), luego va al **punto 2** que es el centro de la cinta (las líneas azules representan la cinta y el rectángulo rojo el área aproximada de trabajo del robot sobre la cinta), después se posiciona sobre la misma línea por donde se desplaza nuestro objetivo (**punto 3)** y luego va en línea recta hacia este (**punto 4**). Por último alza el efector final (**punto 4**) y lleva el objeto al recipiente correspondiente (**punto 5**).

Así resolvemos nuestras tareas combinando trayectorias rectas y curvas según se necesite.

Conclusión:

En el proyecto desarrollamos todos los temas vistos en la materia los cuales incluyen:

- Herramientas Matemáticas y numéricas para cálculos.
- Método de D-H para analizar la cadena cinemática de robots.
- Conceptos Matemáticos de la Cinemática Directa y funciones en matlab para resolverla.
- Métodos para resolver la cinemática inversa del robot (geométricos, numéricos, etc.).
- Jacobiano para detectar puntos singulares.
- Sensores y actuadores que trae el robot y que necesitamos para la aplicación.
- Generación y planificaciones de trayectorias para que el robot cumpla la tarea de forma coherente, efectiva y segura.

Nuestra aplicación fue seleccionar un robot capaz de separar residuos urbanos con el fin de poder reciclarlos posteriormente. Para esto llegamos a la conclusión que era mejor un robot del tipo Scara y elegimos uno comercial de la marca ABB en particular el IRB 910SC. Luego con todos los conocimientos anteriormente mencionados pudimos crear herramientas de cálculo para simular nuestro robot y verificar que pueda cumplir las tareas asignadas y planificar las mismas.