Description: http://www.siga.frba.utn.edu.ar/imag/utnba.png

**ROBOTICA**

**Proyecto robot SCARA**

Curso: R6055

Año: 2013

**Docente:**

Ing. HernanGiannetta

**Ayudante:**

Ing. DamianGranzella

**Alumnos:**

Ignacio Tamayo

Javier Garbini

ÍNDICE

[1 Introducción 4](#_Toc363081115)

[2 Memoria descriptiva 4](#_Toc363081116)

[Funcionamiento 4](#_Toc363081117)

[Modelo adoptado 4](#_Toc363081118)

[Medidas del robot 5](#_Toc363081119)

[3 Cinematica del robot 5](#_Toc363081120)

[3.1 Cinemática directa 5](#_Toc363081121)

[Algoritmo Denavit-Hartenberg 5](#_Toc363081122)

[Desplazamiento de los ejes 5](#_Toc363081123)

[Coordenadas articulares válidas 7](#_Toc363081124)

[3.2 Cinemática Inversa 8](#_Toc363081125)

[Selección de método 8](#_Toc363081126)

[Método geométrico 8](#_Toc363081127)

[Ecuaciones 9](#_Toc363081128)

[4 Espacios de trabajo 11](#_Toc363081129)

[Proyecciones sobre los planos 11](#_Toc363081130)

[Generación de trayectorias articulares 12](#_Toc363081131)

[5 Análisis dinámico 15](#_Toc363081132)

[Modelo físico tridimensional 15](#_Toc363081133)

[Modelo de mecánica dinámica 16](#_Toc363081134)

[Trayectoria polinomial 19](#_Toc363081135)

[6 Control PWM de motores 21](#_Toc363081136)

[Tabla de estados 22](#_Toc363081137)

[7 Control pwm en VHDL 23](#_Toc363081138)

[Diagrama en bloques 23](#_Toc363081139)

[Tabla de estados 23](#_Toc363081140)

[Sensores de efecto Hall 24](#_Toc363081141)

[FPGA 25](#_Toc363081142)

[8 lenguaje de programacion y compilador 26](#_Toc363081143)

[Pseudolenguaje 26](#_Toc363081144)

[Simulación 29](#_Toc363081145)

[9 Conclusiones 31](#_Toc363081146)

**Revision**

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Nombre | Descripción | Versión |
| 03/06/2013 | TP1 | Cinemática Directa e Inversa | 1.0 |
| 22/06/2013 | TP2 | Análisis dinámico | 2.0 |
| 08/07/2013 | TP2 | Simulación en VHDL | 2.1 |
| 01/08/2013 | TP3 | Compilador | 3.0 |

# Introducción

A fin de cumplimentar los requerimientos de cursada para la cátedra de la asignatura electiva de 6° nivel Robótica se realiza la presentación del Proyecto un robot SCARA con 5 grados de libertad. Se toma como base el trabajo realizado en el año 2012 con esta misma arquitectura, agregando un grado más de libertad.

# Memoria descriptiva

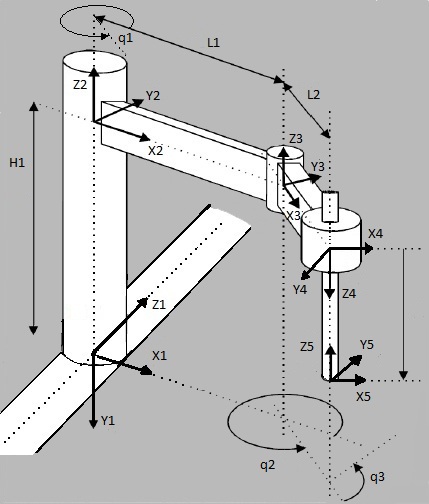
### Funcionamiento

SCARA es un tipo de configuración de robot manipulador las siglas en inglés significan Selective Compliance Assembly Robot Arm. Este robot fue creado por un grupo de industrias electrónicas japonesas, en colaboración con dos universidades, para insertar los componentes de forma vertical.

La configuración seleccionada para realizar el proyecto está formada por dos articulaciones de rotación con respecto a dos ejes paralelos entre sí y perpendiculares al plano de trabajo, dos de desplazamiento una dirección paralela a la de los ejes de rotación y la otra dirección en sentido perpendicular a los ejes de rotación.

### Modelo adoptado

Para el proyecto de la cátedra se eligió un SCARA con el agregado de un grado más de libertad, permitiendo que el robot se desplace linealmente sobre el eje Y del sistema de referencia inicial, como se muestra en la siguiente figura:



### Medidas del robot

H1 = 25cm Altura de la barra.

L1 = 30cm Longitud de la primera articulación.

L2 = 20cm Longitud de la segunda articulación.

# Cinematica del robot

## Cinemática directa

### Algoritmo Denavit-Hartenberg

Para realizar el cálculo de la cinemática directa utilizaremos el algoritmo de Denavit-Hartenberg, el cual define los movimientos de las articulaciones de la siguiente manera:

* Rotación alrededor del eje zi-1 un ángulo θi
* Traslación a lo largo de zi-1 una distancia di ; vector di (0,0,di).
* Traslación a lo largo de xi una distancia ai ; vector ai (0,0,ai).
* Rotación alrededor del eje xi un ángulo αi.

### Desplazamiento de los ejes

Se muestra el gráfico se la secuencia de sistemas para el algoritmo

Art1 d1

Art2 q2

Art3 q3

Art4 d2

Art5 q3

S0

X0

Z0

Y0

S1

Y1

X1

Z1

S2

X2

Y2

Z2

S3

X3

Y3

Z3

S4

X4

Z4

Y4

S5

X5

Y5

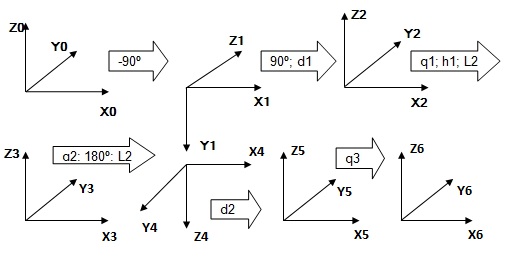
Z5

S6

X6

Z6

Y6



Para el caso de nuestro robot los parámetros de la matriz son los siguientes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **θ** | **d** | **a** | **α** |
| **T0** | -90 ° | 0 | 0 | 0 |
| **T1** | 90 ° | 0 | 0 | **d1** |
| **T2** | 0 | L1 | **q1** | H1 |
| **T3** | 180 ° | L2 | **q2** | 0 |
| **T4** | -180 ° | 0 | 0 | **d2** |
| **T5** | 0 | 0 | **q3** | 0 |

Se tiene una aparente articulación extra (6 filas en lugar de 5) pero esta articulación sirve solo para orientar el eje de coordenadas en el sentido del desplazamiento de la primera articulación.

Por medio de Matlab se obtiene la siguiente matriz de transformaciónhomogénea:

Matriz DH del robot

[cos(q1+q2), -sin(q1+q2), 0, L2\*cos(q1+q2)+L1\*cos(q1) ]

[sin(q1+q2), cos(q1+q2), 0, L2\*sin(q1+q2)+L1\*sin(q1)+d1 ]

[0, 0, 1, H1-d2 ]

[0, 0, 0, 1 ]

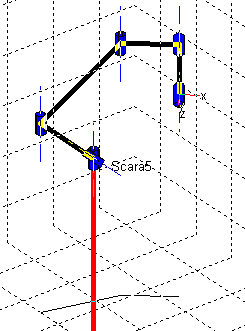
También usando MatLab se obtienes su matriz Jacobiana correspondiente

[ 0, -L2\*sin(q1+q2)-L1\*sin(q1), -L2\*sin(q1+q2), 0 ]

[ 1, L2\*cos(q1+q2)+L1\*cos(q1), L2\*cos(q1+q2), 0 ]

[ 0, 0, 0, -1 ]

Se valida el diseño y las matrices obtenidas usando el Toolbox Corke, del cual se valida el modelo obtenido



L0 = link([-pi/2 0 0 0]) ;

L1 = link([pi/2 0 0 0 1]);

L2 = link ([0 l1 0 h1]);

L3 = link([pi l2 0 0]);

L4 = link([-pi 0 0 0 1]);

L5 = link([0 0 0 0]);

Scara5 = robot({L0,L1,L2,L3,L4,L5},'Scara5');

drivebot(Scara5);

### Coordenadas articulares válidas

Usando la matriz Jacobiana obtenida, se usa un proceso recursivo para encontrar las coordenadas articulares [d1 q1 q2 d2 q3] que causan que el determinante del Jacobiano sea nulo, y son estas las coordenadas articulares que se deben evitar

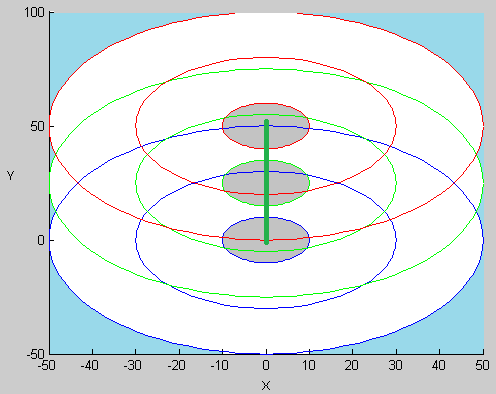
Al no ser el Jacobiano una matriz cuadrada, se usa el siguiente determinante

Det(inv(Jacob\*Jacob') )

Los límites del espacio Externos son los puntos que exceden el radio de los brazos del robot estirados por completo

Los límites del espacio Internos de encuentran son para todo q1, con q2 entre **175-190** grados. Es decir, cuando el segundo brazo se rebate sobre el primero. Estos valores se obtienen del proceso iterativo de anular el determinante.

Estos espacios terminan generando un círculo interior y exterior que no alcanza el robot y son los límites de la zona de trabajo. La siguiente figura muestra los círculos de las zonas de trabajo para distintos valores de desplazamiento d1 sobre el eje Y (vista superior, plano XY)



Sin embargo, los círculos interiores se pueden alcanzar de maneras más cómodas desplazando la articulación d1 de manera que el punto espacial sea alcanzable con q2 fuera de los ángulos críticos

En esta premisa se basa el algoritmo cinemático inverso usado para nuestro robot Scara

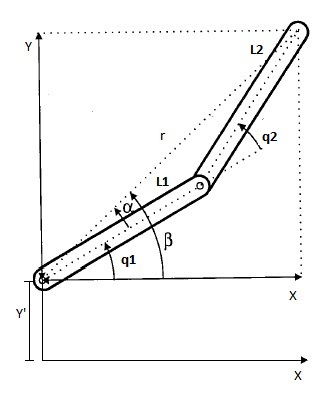
## Cinemática Inversa

### Selección de método

Para el cálculo de la cinemática inversa se utilizó el método geométrico, ya que resulta ser más sencillo de implementar. Como el robot posee un grado de libertad sobre el plano XY es muy tedioso utilizar la matriz de transformación inversa incluyendo esta articulación y a los fines didácticos el método geométrico es mucho más académico al momento de analizarlo.

### Método geométrico

Vista superior del plano XY:



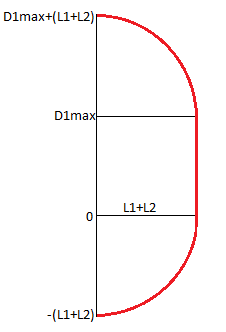
Como se puede observar en la figura el robot se desplaza Y’ para llegar al punto deseado, realizando el anélisis geométrico obtenemos las siguientes ecuaciones:

Analíticamente el desplazamiento Y’ es muy difícil despejar por lo que se eligió para el proyecto trabaja con diferentes zonas de trabajo, dependiendo de la ubicación del punto (px, py, pz).

Los límites de la zona se definieron de la siguiente manera:

De 0 a D1max es el desplazamiento máximo que realiza el robot sobre el eje Y, llegando con las 2 articulaciones extendidas a las posiciones [0-(L1+L2)] y[D1max +(L1+L2)] sobre dicho eje

Siendo las dimensiones del robot L1 + L2 = D1max = 50 cm se tiene una zona de trabajo principal de geometría cuadrada. Se tiene acceso a los puntos de la zona cuadrada de 50cm x 50cm, más un cuarto de circunferencia de radio 50cm sobre esta área y otro de iguales características en la parte inferior. En la siguiente figura se observa la zona de trabajo sobre el plano XY.



### Ecuaciones

Para el cálculo de las ecuaciones se parte de las coordenadas donde se quiere posicionar el extremo del robot (px;py;pz).

Como primer aproximación se determina la posición sobre el eje “y” dependiendo de los valores del punto.

* Para tener el primer brazo flexionado hacia arriba (q1 > 0)

Si py>= D1max+L1: Se desplaza al robot sobre el eje “y” hasta el punto D1max.

Sino Si py>= L1: Se desplaza al robot sobre el eje “y” hasta el punto py – L1.

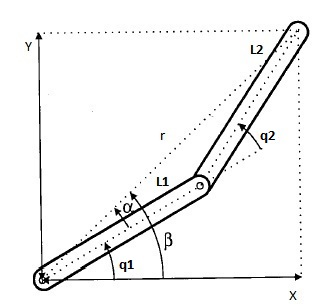
Sino: Se desplaza al robot sobre el eje “y” hasta el punto 0

* Para tener el primer brazo flexionado hacia abajo (q1 < 0)

Si py<= -L1: Se desplaza al robot sobre el eje “y” hasta el punto 0.

Sino Si py<= D1max -L1: Se desplaza al robot sobre el eje “y” hasta el punto py + L1.  
Sino: Se desplaza al robot sobre el eje “y” hasta el punto D1max

Con estos posicionamientos se logran las ecuaciones geométricas sin tener en cuenta el desplazamiento Y’ siendo de este modo mucho más simple el análisis de la cinemática inversa.



Para el cálculo de q2

*;*

Por el teorema del coseno sabemos que r es:

;

Sabiendo

*; ;*

Siendo

Para el cálculo de q1

)-

De esta manera se tiene el algoritmo cinemático inverso, que a partir de una posición (px; py; pz) entrega las coordenadas articulares [d1 q1 q2 d2]

Adicionando un cálculo más, el algoritmo entrega la coordenada articular q3 para obtener una orientación α en la última articulación

Tomando en cuenta que conforme se mueve el robot se debe mantener la orientación de los brazos para evitar saltos bruscos en las coordenadas articulares, se tiene finalmente el algoritmo completo implementado en Matlab:

if (q1\_actual > 0)

if (q2\_actual > 0)

[d1 q1 q2 d2 q3] = MoveScara5\_up(X,Y,Z,A,0); % Curvatura q2>0

else

[d1 q1 q2 d2 q3] = MoveScara5\_up(X,Y,Z,A,1); % Curvatura q2<0

end

else

if (q2\_actual > 0)

[d1 q1 q2 d2 q3] = MoveScara5\_down(X,Y,Z,A,0); % Curvatura q2>0

else

[d1 q1 q2 d2 q3] = MoveScara5\_down(X,Y,Z,A,1); % Curvatura q2<0

end

end

MoveScara5\_up/down() toma las coordenadas espaciales y el ángulo deseado para el muñón, primero calcula la coordenada D1 según se explicó anteriormente. La denominación Up/Down responde a la orientación que se le da al primer brazo del robot. Luego determina los valores angulares Q1 y Q2 según las ecuaciones trigonométricas antes indicadas. Finalmente con una resta simple se obtiene la coordenada D2 y Q3, que solo completa el ángulo de orientación deseado

# Espacios de trabajo

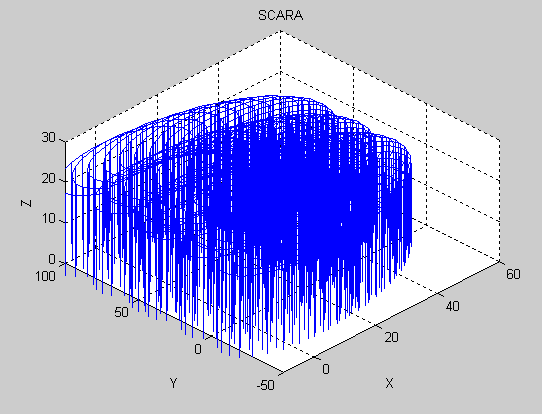
Considerando que los giros de las articulaciones están físicamente limitados por los motores que los van a realizar, además de evitar las coordenadas articulares inválidas antes halladas, se le definen excursiones máximas a las articulaciones y se itera para tener el espacio de trabajo del robot SCARA

Articulación d1: [0 a D1max] cm, Valor de D1max = 50 cm

Articulación d2: [0 a H1] cm, Valor de H1 = 25 cm

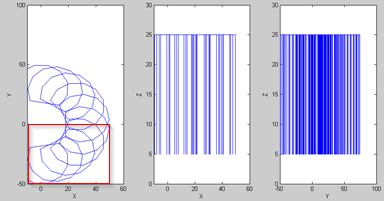
Articulaciones q1,q2,q3: [-145° a 145°]

A continuación se muestra el espacio de trabajo para 3 valores de la articulación d1, sobre el eje Y

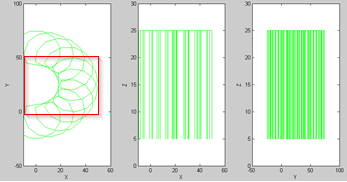


### Proyecciones sobre los planos

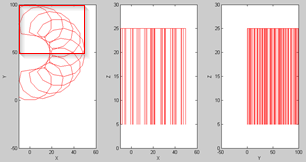
Zona inferior de 0 a -(L1+L2)



Zona media de 0 a D1max



Zona superior de D1max a D1max+(L1+L2)



### Generación de trayectorias articulares

Para poder controlar efectivamente el Robot, se necesita además del algoritmo cinemático una determinada forma de atravesar en el tiempo las coordenadas indicadas. Para nuestro robot SCARA se usa un generador de trayectorias polinómicas de 3er orden, de manera de evitar tener aceleraciones articulares bruscas.

Si usásemos una trayectoria lineal, efectivamente las aceleraciones tendrían valores infinitos en cada arranque de movimiento y esto exigiría demasiado a los motores.

Se implementa el siguiente flujo para la generación de trayectorias y las curvas temporales de velocidad y aceleraciones articulares que los motores deben proveer para conseguir el movimiento deseado

Trayectoria espacial

Espacio de trabajo

Trayectoria espacial alcanzable

Algoritmo de cinemática inversa

Valores articulares reales

Trayectoria articular

Matriz Transferencia

Trayectoria espacial real

Velocidades articulares

Matriz Jacobiana

Velocidades espaciales

Generamos las siguientes coordenadas a las que debe acceder el robot

[X Y Z] coordenada espacial para el muñón del robot

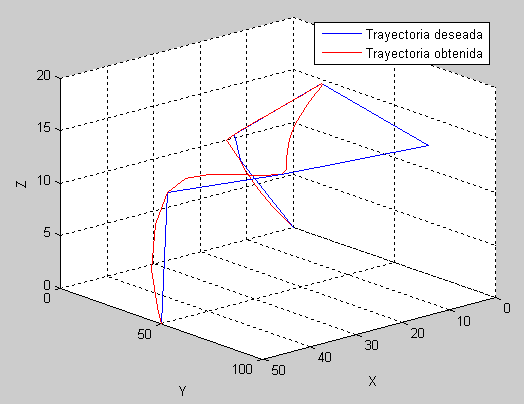
A ángulo de orientación que se desea dar al muñón

T tiempo donde ocurre la coordenada

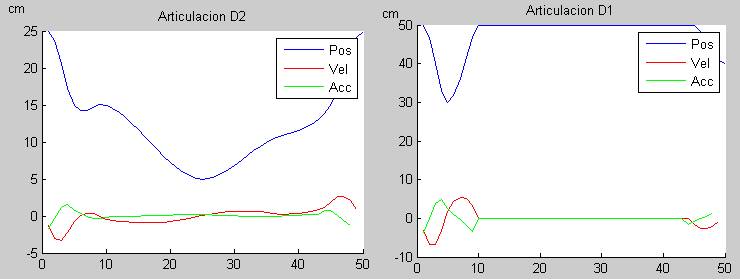
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | 50 | 40 | 20 | 10 | 20 | 30 | 20 | 0 |
| Y | 50 | 30 | 40 | 90 | 60 | 40 | 20 | 0 |
| Z | 0 | 10 | 10 | 15 | 20 | 15 | 10 | 0 |
| A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| T | 1 | 5 | 10 | 15 | 25 | 35 | 45 | 50 |

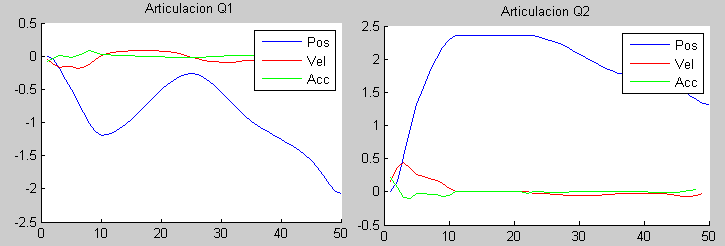
El único punto fuera del área de trabajo es (10, 90, 20). Este punto es descartado de la trayectoria

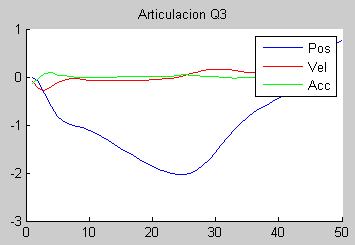
Efectivamente, el robot sigue la trayectoria deseada, salgo el punto descartado



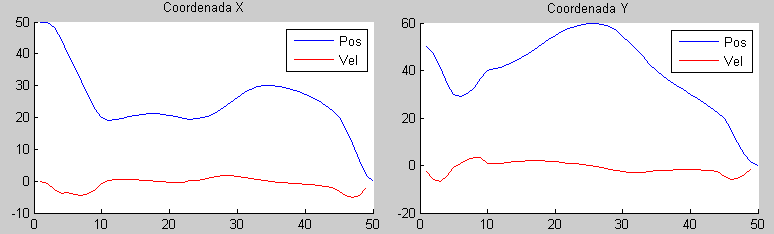
Se analizan las velocidades y aceleraciones articulares necesarias para este movimiento, donde se verifica que no hay saltos bruscos en la velocidad

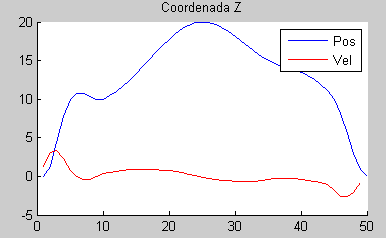






Finalmente se verifican las velocidades espaciales que adquiere el muñón del robot SCARA. Debido a la trayectoria polinomial, se obtienen velocidades sin transiciones buscas





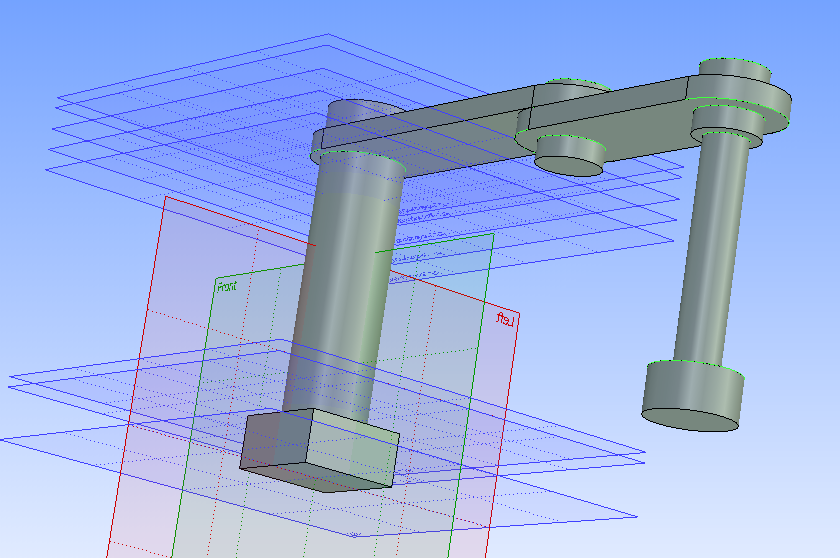
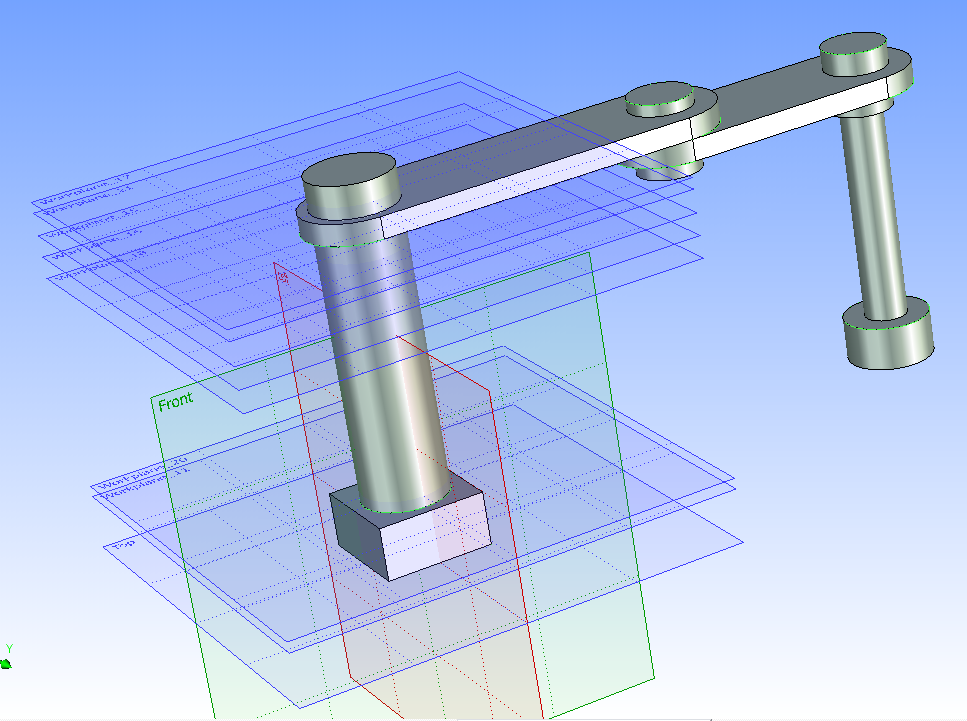
# Análisis dinámico

### Modelo físico tridimensional

Una vez obtenido el generador de trayectorias poliynomiales de orden 3, basado en la técnica de interpolación segmentada por Splines[[1]](#footnote-1), se usa el Toolbox Corke para evaluar los parámetros dinámicos del movimiento del robot, tomando en cuenta sus formas y masas, hasta lograr determinar los parámetros de los actuadores necesarios

Primeramente, se modela el robot en T-Flex para obtener sus parámetros dinámicos

Con las dimensiones mencionadas anteriormente, y dándole un material metálico como construcción, se tiene este modelo en 3 dimensiones



Los parámetros dinámicos por cada eslabón son

Matriz Inercia Eslabón 1 = Centro de Masa Eslabón 1 = Masa Eslabón 1 =

[ 0.1394 0 0] [0 , 0 , 0.0975] mts 6.65675 Kg

[ 0 0.1394 0]

[ 0 0 0.0104] mts

Matriz Inercia Eslabón 2 = Centro de Masa Eslabón 2 = Masa Eslabón 2 =

[ 0.3654 0 0.0001] [0.2162 , 0 , 0] mts 3.88378 Kg

[ 0 0.5842 0]

[ 0.0001 0 0.2238] mts

Matriz Inercia Eslabón 3 = Centro de Masa Eslabón 3 = Masa Eslabón 3 =

[ 0.2729 0 0.0005] [0.1433, 0 , 0] mts 3.05477 Kg

[ 0 0.8889 0]

[ 0.0005 0 0.6198] mts

Matriz Inercia Eslabón 4 = Centro de Masa Eslabón 4 = Masa Eslabón 4 =

[ 0.0634 0 0] [0 , 0 , 0.1700] mts 1.96035 Kg

[ 0 0.5535 0]

[ 0 0 0.4905] mts

Matriz Inercia Eslabón 5 = Centro de Masa Eslabón 5 = Masa Eslabón 5 =

[ 0.0048 0 0] [0 , 0 , 0.0400] mts 1.56828 Kg

[ 0 0.3968 0]

[ 0 0 0.3933] mts

Tomamos una trayectoria suave, una cuasi diagonal en el espacio, y se le da trayectoria con polinomio de 3er grado. Luego de usa esta trayectoria y el modelo del ToolBox Corke para tener los Torques/Fuerzas necesarias en las articulaciones. El ToolBox usa el modelo mecánico de Newton-Euler para sus cálculos.

### Modelo de mecánica dinámica

Optamos por usar el Toolbox de Corke pues en el trabajo anterior se demostró por comparación que los resultados del Toolbox son muy similares a los conseguidos usando las ecuaciones de Newton-Euler directamente

Luego, internamente usamos este procedimiento codificado en MATLAB, usando matrices

Trayectoria espacial discreta

Espacio de trabajo

Algoritmo de cinemática inversa

Trayectoria articular discreta

Interpolación polinomial

Torques articulares

Algoritmo dinámico Newton-Euler

Trayectoria articular continua

Aquí se muestra un extracto de las funciones del Toolbox Corke usadas

%% Construccion del robot

Link0 = link([-pi/2 0 0 0]) ;

Link1 = link([pi/2 0 0 0 1]);

Link2 = link ([0 L1 0 H1]);

Link3 = link([pi L2 0 0]);

Link4 = link([-pi 0 0 0 1]);

Link5 = link([0 0 0 0]);

%% Asignación de las propiedades dinámicas de los eslabones,

LinkX.I = Ix;

LinkX.m = mx;

LinkX.r = cmx;

LinkX.Jm = 1e-6; % Rozamiento minimo

LinkX.G = 1; %constante unitaria

%% Objeto Robot

Scara5 = robot({Link0,Link1,Link2,Link3,Link4,Link5},'Scara5');

Scara5 = nofriction(Scara5, 'all' );

%% Barrido de trayectoria articular y obtención de Torques/Fuerzas

for n=1:length(Ti)

Torc(n,:) = rne(Scara5,[ 0 Qi(n,:)] , [0 dQi(n,:)], [0 ddQi(n,:)] );

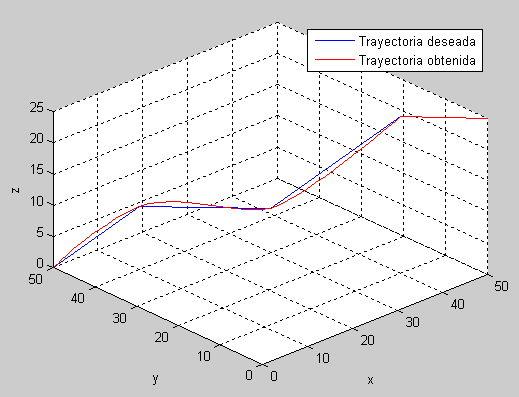
Despl = Qi(n,:) - PrevQ ;

PrevQ = Qi(n,:);

Pot(n,:) = [0 Despl] .\* Torc(n,:);

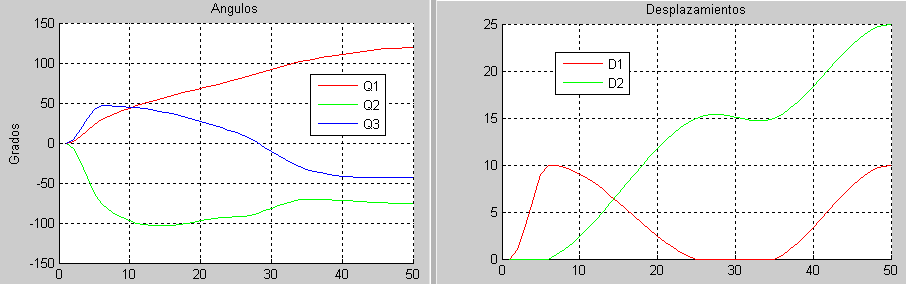
end;

Finalmente se grafican tanto la trayectoria espacial obtenida como los gráficos de aceleración, velocidad, torque y potencia de las articulaciones



*Trayectoria realizada en 50 segundos*

Si bien las trayectorias son de curva suave, al ser el polinomio de 3er orden, las aceleraciones son lineales y los cambios de tramos generan cambios bruscos de aceleración y por ende de torques y fuerzas



Los valores de Fuera y Torque calculados son

Potencias medias:

D1=4.559028 [W]

Q1=6.290679 [W]

Q2=4.090474 [W]

D2=0.182755 [W]

Q3=0.000030 [W]

Torques/Fuerzas máx:

D1=2276.040633 [N]

Q1=621.742307 [Nm]

Q2=159.446726 [Nm]

D2=34.622761 [N]

Q3=0.002786 [Nm]

Torques/Fuerzas RMS:

D1=452.035790 [N]

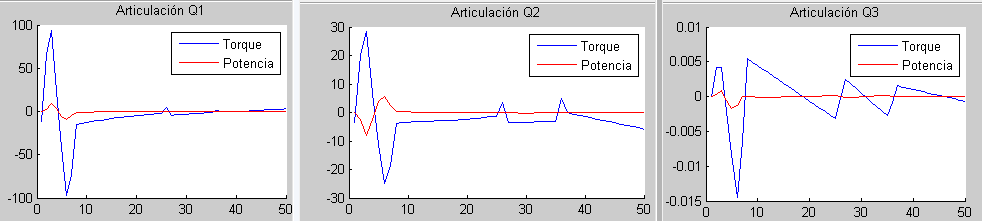
Q1=123.549475 [Nm]

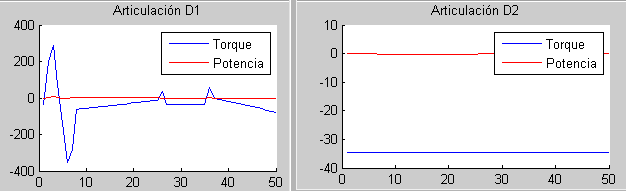
Q2=31.618331 [Nm]

D2=0.003761 [N]

Q3=0.000902 [Nm]

Sin embargo se tienen picos de fuerza y torque por las aceleraciones bruscas



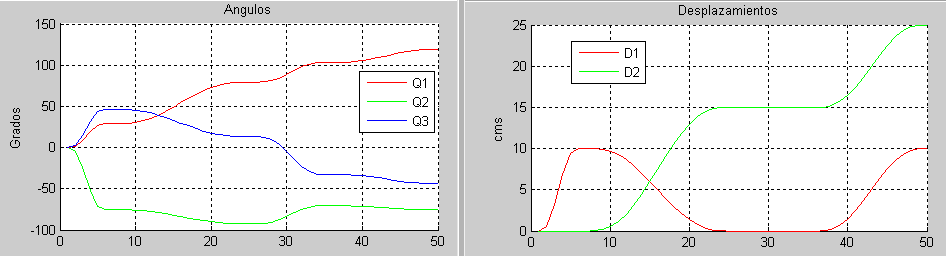


La articulación D2 tiene una componente de fuerza constante pues debe contrarrestar la fuerza peso de la última articulación, pues el peso se ejerce directamente en la dirección del eje.

### Trayectoria polinomial

Para reducir estos cambios violentos de torques se usa ahora una trayectoria polinomial de 5to orden. Si bien se complejiza el cálculo de la trayectoria, el comportamiento pedido a los motores es menos brusco y evita maltratos y el acortamiento de su vida útil

La trayectoria espacial es muy similar



Los valores de Fuerzas/Torques y Potencias tienen valores menores, siendo la misma trayectoria

Potencias medias:

D1=0.728995 [W]

Q1=1.068372 [W]

Q2=0.697858 [W]

D2=0.173075 [W]

Q3=0.000000 [W]

Torques/Fuerzas máx:

D1=483.187836 [N]

Q1=136.078460 [Nm]

Q2=40.847976 [Nm]

D2=34.626095 [N]

Q3=0.000000 [Nm]

Torques/Fuerzas RMS:

D1=109.833923 [N]

Q1=32.023366 [Nm]

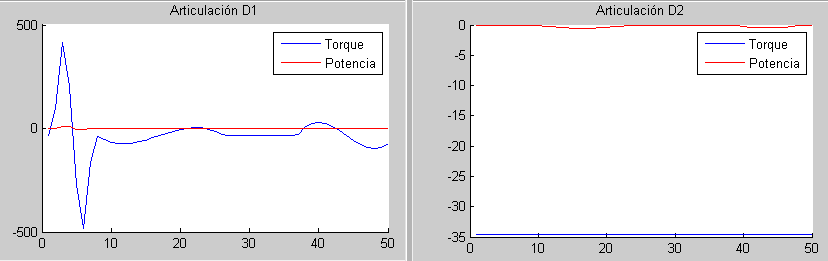
Q2=9.086058 [Nm]

D2=0.005641 [N]

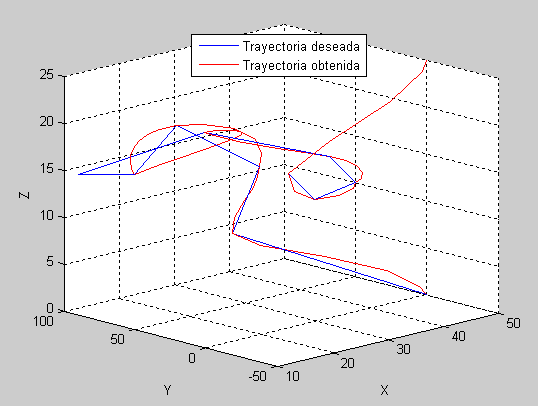
Q3=0.000000 [Nm]

Cambian las formas de las señales de Torque y Potencia de los motores, con transiciones más suaves





Finalmente, para completar el análisis, se propone una trayectoria mucho más exigente para obtener parámetros máximos para los motores



*Trayectoria ejecutada en 60 segundos, con puntos fuera del espacio de trabajo*

Los valores obtenidos para el dimensionamiento de los actuadores son

Potencias medias:

D1=8.315069 [W]

Q1=6.349446 [W]

Q2=3.394626 [W]

D2=0.324975 [W]

Q3=0.000891 [W]

Torques/Fuerzas máx:

D1=1461.339613 [N]

Q1=328.843058 [Nm]

Q2=133.946575 [Nm]

D2=34.669525 [N]

Q3=0.029818 [Nm]

Torques/Fuerzas RMS:

D1=375.162222 [N]

Q1=82.470954 [Nm]

Q2=29.242741 [Nm]

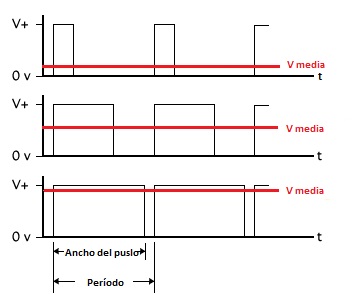
D2=0.016076 [N]

Q3=0.009119 [Nm]

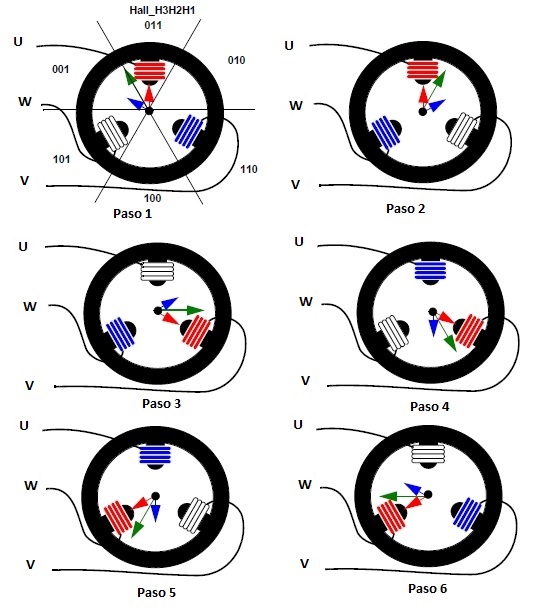
# Control PWM de motores

Para la implementación del robot se utilizaran motores BLDC (Corriente Directa Sin Escobillas), con sensores de efecto hall que determinaran la posición del rotor. El manejo de la velocidad se realizara por medio de un puente H que será controlado por medio de un PWM (pulse width modulation).

Para generar la velocidad y el torque del motor se utiliza la modulación por ancho de pulso conocida como PWM. Con este método se varía el ciclo de actividad de la señal, así variando la tensión media sobre el motor logrando los cambios de velocidad necesarios.



En la siguiente imagen se muestra un ejemplo de una secuencia con los pasos del motor



En el paso 1 por ejemplo se encuentran energizadas las bobinas U y V, siendo que los transistores Q1 y Q4 conducen y el resto están apagados. Por lo tanto se observa que se forman dos flujos uno representado con el vector rojo correspondiente a la bobina U y el otro de color azul correspondiente a la bobina V, siendo el verde la suma de ambos.

El rotor intenta seguir la secuencia del flujo, en cuanto el sensor Hall cambia su valor se energizara la bobina W, apagándose el transistor Q4 y conduciendo el Q6. De esta forma se hace girar el rotor.

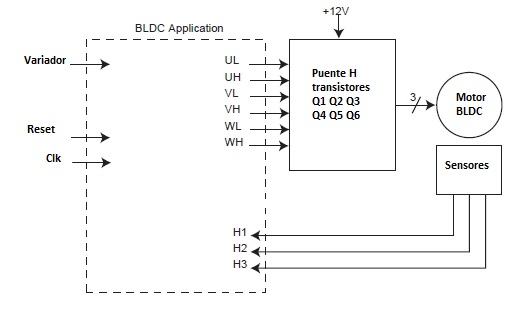
Para poder realizar variaciones de velocidad se utiliza la señal de PWM anteriormente mencionada.

### Tabla de estados

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Estado 1** | **Estado 2** | **Estado 3** | **Estado 4** | **Estado 5** | **Estado 6** |
| **S1** | 1 | 1 | 0 | 0 | 0 | 0 |
| **S2** | 0 | 0 | 0 | pwm | pwm | 0 |
| **S3** | 0 | 0 | 1 | 1 | 0 | 0 |
| **S4** | pwm | 0 | 0 | 0 | 0 | pwm |
| **S5** | 0 | 0 | 0 | 0 | 1 | 1 |
| **S6** | 0 | pwm | pwm | 0 | 0 | 0 |

# Control pwm en VHDL

### Diagrama en bloques



### Tabla de estados

library IEEE;

USE ieee.std\_logic\_1164.all;

USE ieee.std\_logic\_arith.all ;

USE work.user\_pkg.all;

ENTITY driver\_mosfet IS

PORT (

S1 :out STD\_LOGIC;

S2 :out STD\_LOGIC;

S3 :out STD\_LOGIC;

S4 :out STD\_LOGIC;

S5 :out STD\_LOGIC;

S6 :out STD\_LOGIC;

hall\_sensor\_in :in integer range 0 to 5;

reset :in STD\_LOGIC;

clock :in STD\_LOGIC;

pwm\_in :in STD\_LOGIC

);

END driver\_mosfet;

ARCHITECTURE arch\_driver OF driver\_mosfet IS

-- Internal signal declaration

BEGIN

Cntrl\_PWM\_puenteH: PROCESS (reset, hall\_sensor\_in, pwm\_in) IS

BEGIN

IF (reset = '1') THEN

s1<= '0'; s2<= '0'; s3<= '0'; s4<= '0'; s5<= '0'; s6<= '0';

ELSE

CASE hall\_sensor\_in IS

WHEN 0 => --Estado 1

s1<= '1'; s2<= '0'; s3<= '0'; s4<= pwm\_in; s5<= '0'; s6<= '0';

WHEN 1 => --Estado 2

s1<= '1'; s2<= '0'; s3<= '0'; s4<= '0'; s5<= '0'; s6<= pwm\_in;

WHEN 2 => --Estado 3

s1<= '0'; s2<= '0'; S3<= '1'; s4<= '0'; s5<= '0'; s6<= pwm\_in;

WHEN 3 => --Estado 4

s1<= '0'; s2<= pwm\_in; s3<= '1'; s4<= '0'; s5<= '0'; s6<= '0';

WHEN 4 => --Estado 5

s1<= '0'; s2<= pwm\_in; s3<= '0'; s4<= '0'; s5<= '1'; s6<= '0';

WHEN 5 => --Estado 6

s1<= '0'; s2<= '0'; s3<= '0'; s4<= pwm\_in; s5<= '1'; s6<= '0';

WHEN OTHERS =>

s1<= '0'; s2<= '0'; s3<= '0'; s4<= '0'; s5<= '0'; s6<= '0';

END CASE;

END IF;

END PROCESS;

END ARCHITECTURE arch\_driver;

### Sensores de efecto Hall

library IEEE;

USE ieee.std\_logic\_1164.all;

USE ieee.std\_logic\_arith.all ;

USE work.user\_pkg.all;

ENTITY Hall\_Sensor IS

PORT ( sig\_reset : in STD\_LOGIC;

clock : in STD\_LOGIC;

hall\_sensor : out integer range 0 to 5;

rco\_int : in STD\_LOGIC

);

END Hall\_Sensor;

ARCHITECTURE arch\_hall OF Hall\_Sensor IS

signal clock\_div10: std\_logic;

BEGIN

Gen\_clk10: PROCESS (sig\_reset, rco\_int) IS

VARIABLE hall: INTEGER range 0 to 20;

BEGIN

IF (sig\_reset = '1') THEN

hall:= 0;

clock\_div10 <= '1';

ELSE

IF rising\_edge(rco\_int) THEN

IF (hall = 20) THEN

hall:= 0;

clock\_div10 <= '1';

ELSE

hall:= hall + 1;

clock\_div10 <= '0';

END IF;

END IF;

END IF;

END PROCESS;

GenVarState: PROCESS (sig\_reset, clock\_div10) IS

VARIABLE hall: INTEGER range 0 to 5;

BEGIN

IF (sig\_reset = '1') THEN

hall:= 0;

ELSE

IF rising\_edge (clock\_div10) THEN

IF (hall = 5) THEN

hall:= 0;

ELSE

hall:= hall + 1;

END IF;

END IF;

END IF;

hall\_sensor <= hall;

END PROCESS;

END ARCHITECTURE arch\_hall;

### FPGA

library IEEE;

USE ieee.std\_logic\_1164.all;

USE ieee.std\_logic\_arith.all ;

USE work.user\_pkg.all;

ENTITY pwm\_fpga IS

PORT ( clock,reset :in STD\_LOGIC;

Data\_value :in std\_logic\_vector(7 downto 0);

pwm :out STD\_LOGIC;

rco :out STD\_LOGIC

);

END pwm\_fpga;

ARCHITECTURE arch\_pwm OF pwm\_fpga IS

SIGNAL reg\_out : std\_logic\_vector(7 downto 0);

SIGNAL cnt\_out\_int : std\_logic\_vector(7 downto 0);

SIGNAL pwm\_int, rco\_int : STD\_LOGIC

BEGIN

-- 8 BIT DATA REGISTER TO STORE THE MARKING VALUES .

-- THE MARKING VALUES WILL DETERMINE THE DUTY CYCLE OF PWM OUTPUT

PROCESS(clock,reg\_out,reset)

BEGIN

IF (reset ='1') THEN

reg\_out <="00000000";

ELSIF (rising\_edge(clock)) THEN

reg\_out <= data\_value;

END IF;

END PROCESS;

-- 8 BIT UPDN COUNTER. COUNTS UP OR DOWN BASED ON THE PWM\_INT SIGNAL AND GENERATES

-- TERMINAL COUNT WHENEVER COUNTER REACHES THE MAXIMUM VALUE OR WHEN IT TRANSISTS

-- THROUGH ZERO. THE TERMINAL COUNT WILL BE USED AS INTERRUPT TO AVR FOR GENERATING

-- THE LOAD SIGNAL.

-- INC and DEC are the two functions which are used for up and down counting. They are defined in sepearate user\_pakge library

PROCESS (clock,cnt\_out\_int,rco\_int,reg\_out)

BEGIN

IF (rco\_int = '1') THEN

cnt\_out\_int <= reg\_out;

ELSIF rising\_edge(clock) THEN

IF (rco\_int = '0' and pwm\_int ='1' and cnt\_out\_int <"11111111") THEN

cnt\_out\_int <= INC(cnt\_out\_int);

ELSE

IF (rco\_int ='0' and pwm\_int ='0' and cnt\_out\_int > "00000000") THEN

cnt\_out\_int <= DEC(cnt\_out\_int);

END IF;

END IF;

END IF;

END PROCESS;

-- Logic to generate RCO signal

PROCESS(cnt\_out\_int, rco\_int, clock,reset)

BEGIN

IF (reset ='1') THEN

rco\_int <='1';

ELSIF rising\_edge(clock) THEN

IF ((cnt\_out\_int = "11111111") or (cnt\_out\_int ="00000000")) THEN

rco\_int <= '1';

ELSE

rco\_int <='0';

END IF;

END IF;

END PROCESS;

-- TOGGLE FLIP FLOP TO GENERATE THE PWM OUTPUT.

PROCESS (clock,rco\_int,reset)

BEGIN

IF (reset = '1') THEN

pwm\_int <='0';

ELSIF rising\_edge(rco\_int) THEN

pwm\_int <= NOT(pwm\_int);

ELSE

pwm\_int <= pwm\_int;

END IF;

END PROCESS;

pwm <= pwm\_int;

rco <= rco\_int;

END arch\_pwm;

# lenguaje de programacion y compilador

A fin de dotar de un set de instrucciones al Robot SCARA, acorde al algoritmo de movimientos desarrollado y las funciones del robot, se le define un lengua con elementos Léxicos, Sintácticos y Semánticos. El objetivo es proveer un set de instrucciones reducido que pueda ser compilado y traducido en las diferentes señales a enviar a cada articulación del robot.

Rutina en pseudolenguaje

Analizador léxico

Cinemática inversa

Trayectoria articular discreta

Interpolación polinomial

Trayectoria articular continua

Analizador sintáctico

Analizador semantico

El entorno de desarrollo utilizado para el desarrollo del lenguaje es ANTLRWorks, el cual cuenta con interfaz gráfica que permite el desarrollo de gramáticas generando código JAVA. Luego, se integra la lógica del robot en código JAVA y juntos consiguen generar un flujo constante de valores para cada articulación del robot, que serían las señales enviadas a los controladores PWM de cada articulación para conseguir el movimiento programado

Si bien en este caso se usa un archivo de texto como la salida del compilador/ejecutor del programa, este archivo de texto alimenta un modelo en MATLAB que muestra gráficamente los movimientos del robot en el tiempo

### Pseudolenguaje

El lenguaje desarrollado está orientado a permitir bloques de repeticiones (loops), y las instrucciones para los movimientos se dividen en el plano XY, luego plano Z, y finalmente orientación de la última articulación utilizada para posicionar la pieza a manipular.

Cada instrucción define un tiempo para el movimiento, pues los drivers tienen la capacidad de operar en diferentes velocidades. De esta manera se puede interpolar fácilmente la trayectoria polinomial en el tiempo indicado. De no ser así, debería constantemente medirse que los motores han alcanzando la posición deseada a velocidad constante para dar por terminada la instrucción

HOME [espacios] [tiempo en segundos];

PAUSE [espacios] [tiempo en segundos];

MOVEZ [espacios] [+/- valor en mm] [ , ] [tiempo en segundos];

MOVETITA [espacios] [+/- valor en grados] [ , ] [tiempo en segundos];

MOVEZ [espacios] [+/- valor en mm] [ , ] [+/- valor en mm] [ , ] [tiempo en segundos];

LOOP [espacios] [veces]

[Instrucciones dentro del lazo]

END

El primer análisis, y de hecho las primeras definiciones del lenguaje, son las definiciones léxicas, que indican los símbolos que se reconocerán como válidos

Luego se analizan las reglas sintácticas que analizan la estructura de las instrucciones.



Función que lleva al robot a su posición inicial, tardando un tiempo en segundos dado



Función que mantiene la posición del robot durante un tiempo en segundos dado



Función que mueve la cuarta articulación del robot, alcanzando puntos en el eje Z, tardando un tiempo en segundos dado



Función que mueve la quinta articulación del robot, orientando el último eslabón, tardando un tiempo en segundos dado



Función que mueve las primeras articulaciones hasta un punto en el plano XY, tardando un tiempo en segundos dado

Finalmente el análisis semántico recorre las instrucciones ejecutando los ciclos (loops) y extrayendo las coordenadas espaciales para su procesamiento

Se muestra aquí la gramática ANTLR creada para el robot SCARA

//Reglas de léxico

COMMENT : '//' ~('\n'|'\r')\* '\r'? '\n' {$channel=HIDDEN;};

//Comentarios de una sola línea

WHITESPACE : ( '\t' | ' ' )+ ; //Espacios en blanco permitidos

ARGSPACE : ( '\t' | ' ' )\* ',' ( '\t' | ' ' )\* ; //Separación de argumentos

ENDING : ';' ; //Fin de instrucción definido

SINT : ('+' | '-') ('0'..'9')+; //Los valores signados se separan de los sin signar

INT : ('0'..'9')+;

CR : // handle newlines

( '\r\n' // DOS/Windows

| '\r' // Macintosh

| '\n' // Unix

)

{skip();}

;

Home : 'HOME';

Pause : 'PAUSE';

MoveZ : 'MOVEZ' ;

MoveT : 'MOVETITA' ;

MoveXY : 'MOVEXY';

Loop : 'LOOP';

EndLoop : 'END';

//Operandos validos

time returns [int value] : INT {$value = Integer.parseInt($INT.text);};

op returns [int value]: SINT {$value = Integer.parseInt($SINT.text);};

op1 returns [int value]: SINT {$value = Integer.parseInt($SINT.text);};

//Funciones

homeFunc : Home^ WHITESPACE! time ENDING! ;

pauseFunc : Pause^ WHITESPACE! time ENDING! ;

moveZFunc : MoveZ^ WHITESPACE! op ARGSPACE! time ENDING! ;

moveTitaFunc : MoveT^ WHITESPACE! op ARGSPACE! time ENDING! ;

moveXYFunc : MoveXY^ WHITESPACE! op ARGSPACE! op1 ARGSPACE! time ENDING! ;

expr : (orders | loop)+;

block : orders+ -> ^(BLOCK orders+);

orders : homeFunc | pauseFunc | moveZFunc | moveXYFunc | moveTitaFunc ;

loop : Loop^ WHITESPACE! time block EndLoop! ;

//Tree Grammar para analisis semántico

loop : ^(Loop

{

int cond = input.index();

int LoopCounter =0 ;

}

time

{

LoopCounter = $time.value;

int loopHead = input.index();

}

block

{

int end = input.index();

LoopCounter = LoopCounter -1;

while (LoopCounter > 0)

{input.seek(loopHead);

block();

LoopCounter = LoopCounter -1;

}

input.seek(end);

}

);

Se vuelca en una clase JAVA la lógica del algoritmo de cinemática inversa, así como los algoritmos de interpolación de trayectorias, y este programa el que genera un archivo de text que simula las señales enviadas a los drivers de los motores PWM para cada articulación. Se define que si bien las instrucciones aceptan parámetros en Segundos, el archivo de señales lo hace en intervalos de décimas de segundo (10 veces más rápido)

Un ejemplo de un programa en pseudo lenguaje valido sería

HOME 1;

MOVEXY +400,+200,5;

MOVEXY +300,+500,5;

MOVEZ +100,2;

LOOP 2

MOVETITA +90,2;

MOVETITA -90,2;

END

### Simulación

Finalmente para verificar las señales de movimiento articular generadas, se alimenta este archivo de texto generado en un modelo MATLAB del ToolBox Corke donde se visualizan los movimientos

%% Carga de archivos de movimientos articulares

% Formato esperado: |D1[mm] Q1[rad] Q2[rad] D2 [mm] Q3[rad]

% Cada Fila es 1 segundo.

Q = dlmread ('Qvalues.qs');

% Se debe agregar una fila de articulación nula Link0

Q = [ zeros(length(Q(:,1)),1), Q];

%% Inicializacion de dimensiones del Robot, em mms

global H1 L1 L2 D1max

H1 = 250; %[mm]

L1 = 300; %[mm]

L2 = 200; %[mm]

D1max = 500; %[mm]

%% Simulacion en Toolbox Corke v8

Lk0 = link([-pi/2 0 0 0]) ;

Lk1 = link([pi/2 0 0 0 1]);

Lk2 = link ([0 L1 0 H1]);

Lk3 = link([pi L2 0 0]);

Lk4 = link([0 0 0 0 1]);

Lk5 = link([0 0 0 0]);

Scara5 = robot({Lk0,Lk1,Lk2,Lk3,Lk4,Lk5},'Scara5');

Scara5.plotopt = { 'erase', 'noloop', 'nobase', 'name' , 'wrist', 'shadow', 'xyz', 'joints'};

%% Movimientos por pasos del Robot en la trayectoria

for i = [1:1:length(Q(:,1))]

plot (Scara5,Q(i,:));

end

%% Dibujo de las velocidades

dD1 = [ 0; diff(Q(:,2))];

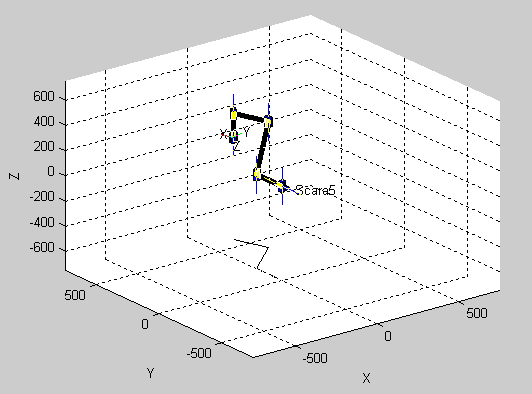
dQ1 = [ 0; diff(Q(:,3))];

dQ2 = [ 0; diff(Q(:,4))];

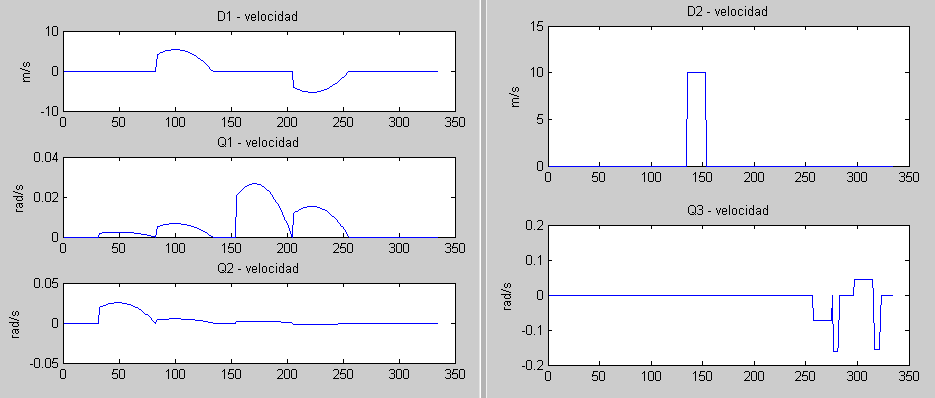
dD2 = [ 0; diff(Q(:,5))];

dQ3 = [ 0; diff(Q(:,6))];

Finalmente se muestra una gráfica del simulador en movimiento para el programa anteriormente mostrado



Las articulaciones para el movimiento XY están en trayectoria cúbica, mientras que las demás articulaciones, más simples y de menor torque, están en trayectoria lineal. La ventaja de la trayectoria lineal es que genera velocidades constantes, que son más fáciles de generar con los drivers



# Conclusiones

* Se ha desarrollado adecuadamente el modelo cinmático del Robot SCARA de 5 grados de libertad, usando el algoritmo de Denavit-Hartenberg
* Se verificó el modelo cinemático con el ToolboxCorke
* Se desarrolló un algoritmo cinemático inverso. Efectivamente, dada una trayectoria a seguir, el algoritmo cinemático inverso es capaz de seguirlo, siempre en los puntos dentro del área de trabajo válida
* Se usa un generador de trayectorias articulares polinomial de 3 er order, para suavizar las curvas de aceleraciones. Las velocidades y aceleraciones articulares no tiene discontinuidades por ser trayectorias polinomiales. Las velocidades espaciales no tienen discontinuidades.
* Se ha desarrollado el análisis dinámico logrando dimensionar las fuerzas y torques
* Se ha podido suavizar las curvas de velocidad y movimientos bruscos del robot.
* Se dimensionó el robot según la dinámica calculada.
* Se lograron hacer las simulaciones necesarias para interpretar el comportamiento del robot.
* Se logró efectivamente generar un pseudo lenguaje para programar el robot, incluyendo la generación de ciclos en el pseudo lenguaje
* Se logró programar un compilador en JAVA capaz de generar las señales que serían entregadas a los drivers de los motores de cada articulación
* Se logró migrar los algoritmos de cinemática inversa y generación de trayectoria a JAVA
* Se verificó mediante una simulación el cumplimiento de la trayectoria programada en pseudo lenguaje

1. <http://www.fi.unju.edu.ar/materias/materia/CN/document/Teoria_Calculo/INTERPOLACI%D3N__DE__SPLINES.pdf?cidReq=CN> [↑](#footnote-ref-1)