

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES
INGENIERIA ELECTRÓNICA

TESIS FINAL

PUMA560

Análisis Cinemático, dinámico y
implementación de un compilador
sobre el brazo robot PUMA560



ROBÓTICA

INTEGRANTES:

PEREZ, Julián Gabriel [LEG : 1203307]
DE ANGELIS, Fernando [LEG: 1190842]

Profesor: **GIANNETTA**, Hernán
Jefe de TP: **GRANZELLA**, Eduardo Damián
Fecha de Entrega: 27 -08 -2010
Curso: R6055
Año: 2010

TESIS FINAL

Análisis cinemático, dinámico y compilador sobre brazo robot PUMA

OBJETIVO

- Introducción al PUMA
- Análisis de la cinemática sobre PUMA560
 - Introducción teórica de la cinemática del robot
 - Cinemática directa
 - Cinemática inversa
 - Modelo diferencial (Matriz Jacobiana)
- Análisis de la dinámica sobre PUMA560
 - Introducción teórica de la dinámica del robot
 - Dinámica
- Compilador
 - Introducción teórica.
 - Capturas de pantalla
 - Código
- FPGA
 - Esquema de Control
 - Tabla de Estados
 - Código EN VHDL
 - Resultado de la simulación
- Conclusiones finales.

INTRODUCCIÓN AL PUMA

En 1975 nace el PUMA (Programmable Universal Machine for Assembly, or Programmable Universal Manipulation Arm). Es un brazo robot industrial desarrollado por Víctor Scheinman en la empresa pionera en robótica Unimation. Inicialmente desarrollado para General Motors en 1978, el brazo robot PUMA nació de los diseños iniciales inventados por Scheinman mientras se encontraba en el MIT y en la Stanford University.

Unimation produjo PUMAs durante algunos años hasta que fue absorbida por Westinghouse en 1980, y posteriormente por la empresa suiza Staubli en 1988. Nokia Robotics manufacturó cerca 1500 brazos robots PUMA durante los años 1980, siendo el PUMA650 el modelo más popular entre los clientes.

Pionero en la utilización de robots en cirugía, el robot PUMA560, en 1985, fue utilizado para introducir una aguja en el cerebro.

Fue usado también para investigación, por las posibilidades que ofrecía su lenguaje de programación.

El PUMA es un brazo robot con 6 grados de libertad, capaz de mover un objeto y colocarlo en cualquier orientación en un lugar deseado que estuviera a su alcance. El concepto básico multiarticulado del PUMA es la base de la mayoría de los robots actuales.

INTRODUCCIÓN TEÓRICA DE LA CINEMÁTICA DEL ROBOT

La cinemática es la parte de la mecánica clásica que estudia las leyes del movimiento de los cuerpos sin tener en cuenta las causas que lo producen, limitándose esencialmente, al estudio de la trayectoria en función del tiempo. Cinemática deriva de la palabra griega κινεω (kineo) que significa mover.

En la cinemática se utiliza un sistema de coordenadas para describir las trayectorias y se le llama sistema de referencia.

CINEMÁTICA DEL ROBOT

Estudio de su movimiento con respecto a un sistema de referencia:

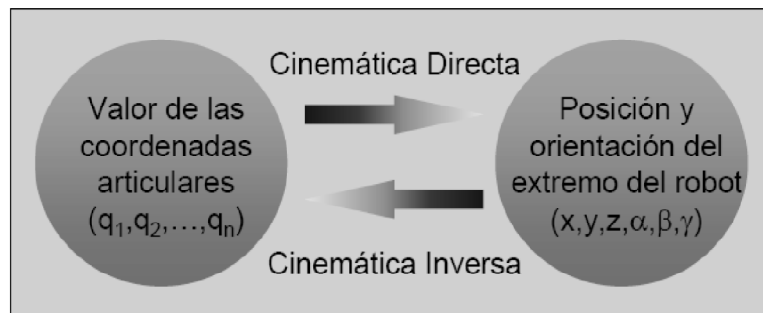
- Relación entre la localización del extremo del robot y los valores de sus articulaciones.
- Descripción analítica del movimiento espacial en función del tiempo

Problema cinemática directo: Determinar la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas de referencia, conocidos los ángulos de las articulaciones y los parámetros geométricos de los elementos del robot.

Problema cinemática inverso: Determinar la configuración que debe adoptar el robot para alcanzar una posición y orientación del extremo conocidas.

Modelo diferencial (matriz Jacobiana): Relaciones entre las velocidades de las N articulaciones y las del extremo del robot.

RELACION ENTRE CINEMÁTICA DIRECTA E INVERSA



MODELO CINEMÁTICO

Método basado en relaciones geométricas (Trigonometría)

- No sistemática
- Es válido para robots de pocos grados de libertad

Método basado en matrices de transformación homogéneas

- Sistemático
- Es válido para robots con muchos grados de libertad.

MÉTODO BASADO EN MATRICES DE TRANSFORMACIÓN HOMOGÉNEAS

Se utiliza fundamentalmente el álgebra vectorial y matricial para representar y describir la localización de un objeto en el espacio tridimensional con respecto a un sistema de referencia fijo.

Dado que un robot se puede considerar como una cadena cinemática formada por objetos rígidos o eslabones unidos entre sí mediante articulaciones, se puede establecer un sistema de referencia fijo situado en la base del robot y describir la localización de cada uno de los eslabones con respecto a dicho sistema de referencia.

De esta forma, el problema cinemática directo se reduce a encontrar una matriz homogénea de transformación T que relacione la posición y orientación del extremo del robot respecto del sistema de referencia fijo situado en la base del mismo. Esta matriz T será función de las coordenadas articulares.

La resolución del problema cinemática directo consiste en encontrar las relaciones que permiten conocer la localización espacial del extremo del robot a partir de los valores de sus coordenadas articulares.

Así, si se han escogido coordenadas cartesianas y ángulos de Euler para representar la posición y orientación del extremo de un robot de seis grados de libertad, la solución al problema cinemática directo vendrá dada por las relaciones:

$$\begin{aligned}x &= f_x(q_1, q_2, q_3, q_4, q_5, q_6) \\y &= f_y(q_1, q_2, q_3, q_4, q_5, q_6) \\z &= f_z(q_1, q_2, q_3, q_4, q_5, q_6) \\\alpha &= f_\alpha(q_1, q_2, q_3, q_4, q_5, q_6) \\\beta &= f_\beta(q_1, q_2, q_3, q_4, q_5, q_6) \\\gamma &= f_\gamma(q_1, q_2, q_3, q_4, q_5, q_6)\end{aligned}$$

En general, un robot de n grados de libertad está formado por n eslabones unidos por “ n ” articulaciones, de forma que cada par articulación-eslabón constituye un grado de libertad.

A cada eslabón se le puede asociar un sistema de referencia solidario a él y, utilizando las transformaciones homogéneas, es posible representar las rotaciones y traslaciones relativas entre los distintos eslabones que componen el robot.

Normalmente, la matriz de transformación homogénea que representa la posición y orientación relativa entre los sistemas asociados a dos eslabones consecutivos del robot se suele denominar matriz ${}^{i-1}A_i$.

Así pues, 0A_1 describe la posición y orientación del sistema de referencia solidario al primer eslabón con respecto al sistema de referencia solidario a la base, 1A_2 describe la posición y orientación del segundo eslabón respecto del primero, etc.

Del mismo modo, denominando 0A_k a las matrices resultantes del producto de las matrices ${}^{i-1}A_i$ con i desde 1 hasta k , se puede representar de forma total o parcial la cadena cinemática que forma el robot.

Cuando se consideran todos los grados de libertad, a la matriz 0A_n se le suele denominar **T**. Así, dado un robot de seis grados de libertad, se tiene que la posición y orientación del eslabón final vendrá dada por la matriz **T**:

$$\mathbf{T} = {}^0A_6 = {}^0A_1 {}^1A_2 {}^2A_3 {}^3A_4 {}^4A_5 {}^5A_6$$

Aunque para describir la relación que existe entre dos elementos contiguos se puede hacer uso de cualquier sistema de referencia ligado a cada elemento.

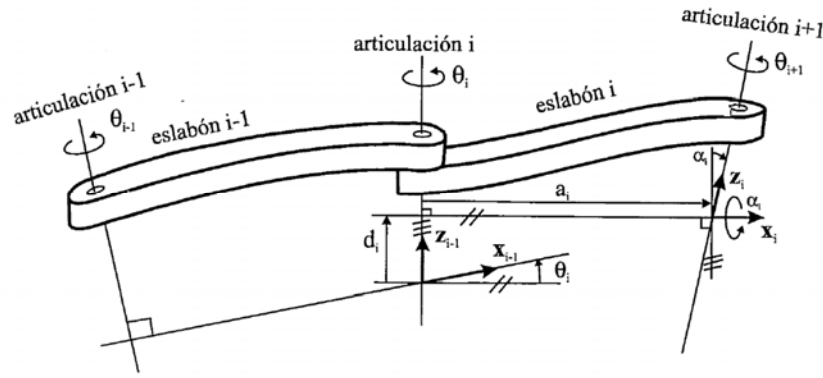
La forma habitual que se suele utilizar en robótica es la representación de *Denavit- Hartenberg (D-H)*. Denavit y Hartenberg propusieron en 1955 un método matricial que permite establecer de manera sistemática un sistema de coordenadas $\{S_i\}$ ligado a cada eslabón i de una cadena articulada, pudiéndose determinar a continuación las ecuaciones cinemáticas de la cadena completa.

Según la representación de D-H, escogiendo adecuadamente los sistemas de coordenadas asociados a cada eslabón, será posible pasar de uno al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón.

Estas transformaciones básicas consisten en una sucesión de rotaciones y traslaciones que permiten relacionar el sistema de referencia del elemento i con el sistema del elemento $i-1$.

Las transformaciones en cuestión son las siguientes (es importante recordar que el paso del sistema $\{S_{i-1}\}$ al $\{S_i\}$ mediante estas 4 transformaciones está garantizado solo si los sistemas $\{S_{i-1}\}$ al $\{S_i\}$ han sido definidos de acuerdo a unas normas determinadas que se expondrán posteriormente):

1. Rotación alrededor del eje z_{i-1} un ángulo θ_i
2. Traslación a lo largo de z_{i-1} una distancia d_i - vector \mathbf{d}_i **(0,0,d_i)**.
3. Traslación a lo largo de x_i una distancia a_i - vector \mathbf{a}_i **(0,0,a_i)**.
4. Rotación alrededor del eje x_i un ángulo θ_i .



$${}^{i-1}A = T(z, \theta_i) T(0, 0, d_i) T(a_i, 0, 0) T(x, \alpha_i)$$

$${}^{i-1}A_i = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C\theta_i & -C\alpha_i C\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ -S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

donde a_i , d_i , θ_i , α_i son los parámetros D-H del eslabón i . De este modo, basta con identificar los parámetros a_i , d_i , θ_i , α_i para obtener las matrices A y relacionar así todos y cada uno los eslabones del robot.

Como se ha indicado, para que la matriz ${}^{i-1}A_i$, definida anteriormente, relacione los sistemas $\{S_{i-1}\}$ y $\{S_i\}$, es necesario que los sistemas se hayan escogido de acuerdo a unas determinadas normas. Estas, junto con la definición de los 4 parámetros de Denavit-Hartenberg, conforman el siguiente algoritmo para la resolución del problema cinemática directo:

Se define el Algoritmo de Denavit-Hartenberg para la obtención del modelo cinemática directo.

ALGORITMO DE DENAVIT-HARTENBERG

D-H 1. Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (último eslabón móvil). Se numerara como eslabón 0 a la base fija del robot.

D-H 2. Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad) y acabando en n .

D-H 3. Localizar el eje de cada articulación. Si esta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.

D-H 4. Para i de 0 a $n-1$ situar el eje z_i sobre el eje de la articulación $i + 1$.

D-H 5. Situar el origen del sistema de la base $\{S_0\}$ en cualquier punto del eje z_0 . Los ejes x_0 e y_0 se situaran de modo que formen un sistema dextrógiro con z_0 .

D-H 6. Para i de 1 a $n-1$, situar el sistema $\{S_i\}$ (solidario al eslabón i) en la intersección del eje z_i con la línea normal común a z_{i-1} y z_i . Si ambos ejes se cortasen se situaría $\{S_i\}$ en el punto de corte. Si fuesen paralelos $\{S_i\}$ se situaría en la articulación $i + 1$.

D-H 7. Situar x_i en la línea normal común a z_{i-1} y z_i .

D-H 8. Situar y_i de modo que forme un sistema dextrógiro con x_i y z_i .

D-H 9. Situar el sistema $\{S_n\}$ en el extremo del robot de modo que z_n , coincida con la dirección de z_{n-1} y x_n sea normal a z_{n-1} y z_n .

D-H 10. Obtener θ_i como el ángulo que hay que girar en torno a z_{i-1} para que x_{i-1} y x_i , queden paralelos.

D-H 11. Obtener d_i , como la distancia, medida a lo largo de z_{i-1} , que habría que desplazar $\{S_{i-1}\}$ para que x_i y x_{i-1} quedasen alineados.

DH 12. Obtener a_i como la distancia medida a lo largo de x_i (que ahora coincidiría con x_{i-1}) que habría que desplazar el nuevo $\{S_{i-1}\}$ para que su origen coincidiese con $\{S_i\}$.

DH 13. Obtener α_i como el ángulo que habría que girar entorno a x_i , (que ahora coincidiría con x_{i-1}), para que el nuevo $\{S_{i-1}\}$ coincidiese totalmente con $\{S_i\}$.

DH 14. Obtener las matrices de transformación ${}^{i-1}A_i$ definidas anteriormente.

DH 15. Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot $T = {}^0A_1, {}^1A_2, \dots, {}^{n-1}A_n$

DH 16. La matriz T define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base en función de las n coordenadas articulares.

Los cuatro parámetros de D-H ($a_i, d_i, \theta_i, \alpha_i$) dependen únicamente de las características geométricas de cada eslabón y de las articulaciones que le unen con el anterior y siguiente. En concreto estos representan:

- θ_i Es el ángulo que forman los ejes x_{i-1} y x_i , medido en un plano perpendicular al eje z_{i-1} , utilizando la regla de la mano derecha. Se trata de un parámetro variable en articulaciones giratorias.
- d_i Es la distancia a lo largo del eje z_{i-1} desde el origen del sistema de coordenadas $(i-1)$ -ésimo hasta la intersección del eje z_{i-1} con el eje x_i . Se trata de un parámetro variable en articulaciones prismáticas.
- a_i Es la distancia a lo largo del eje x_i que va desde la intersección del eje z_{i-1} con el eje x_i hasta el origen del sistema i -ésimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas, se calcula como la distancia más corta entre los ejes z_{i-1} y z_i .
- d_i Es el ángulo de separación del eje z_{i-1} y el eje z_i , medido en un plano perpendicular al eje x_i , utilizando la regla de la mano derecha.

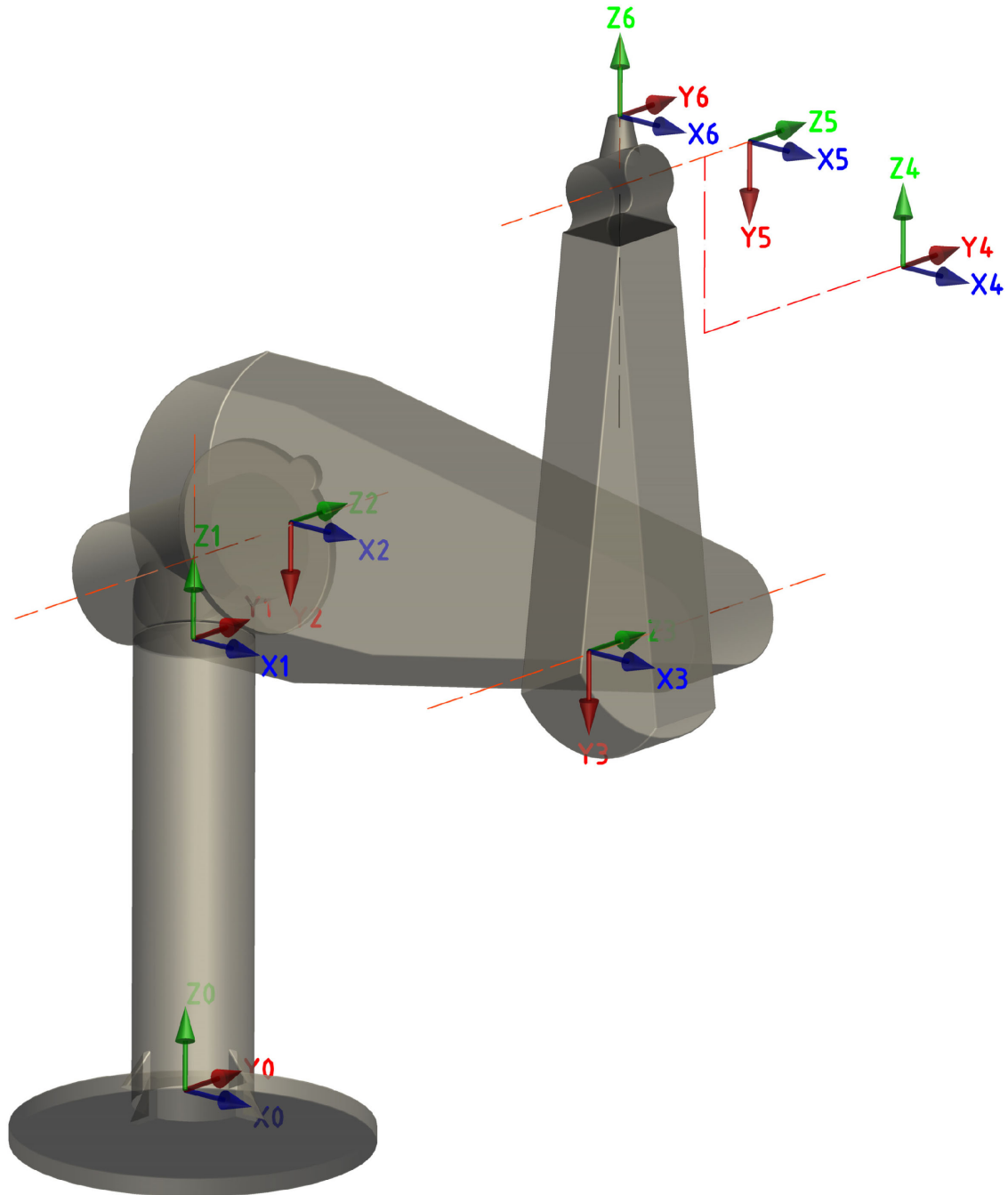
Para obtener el modelo cinemática directo de un robot procedamos de la siguiente manera:

- Establecer para cada elemento del robot un sistema de coordenadas cartesianas ortonormal (x_i, y_i, z_i) donde $i=1,2,\dots,n$ (n = número de gdl). Cada sistema de coordenadas corresponderá a la articulación $i+1$ y estará fijo en el elemento i (algoritmo D-H).
- Encontrar los parámetros D-H de cada una de las articulaciones.
- Calcular las matrices ${}^{i-1}A_i$
- Calcular la matriz $T_n = {}^0A_1 {}^1A_2 \dots {}^{n-1}A_n$

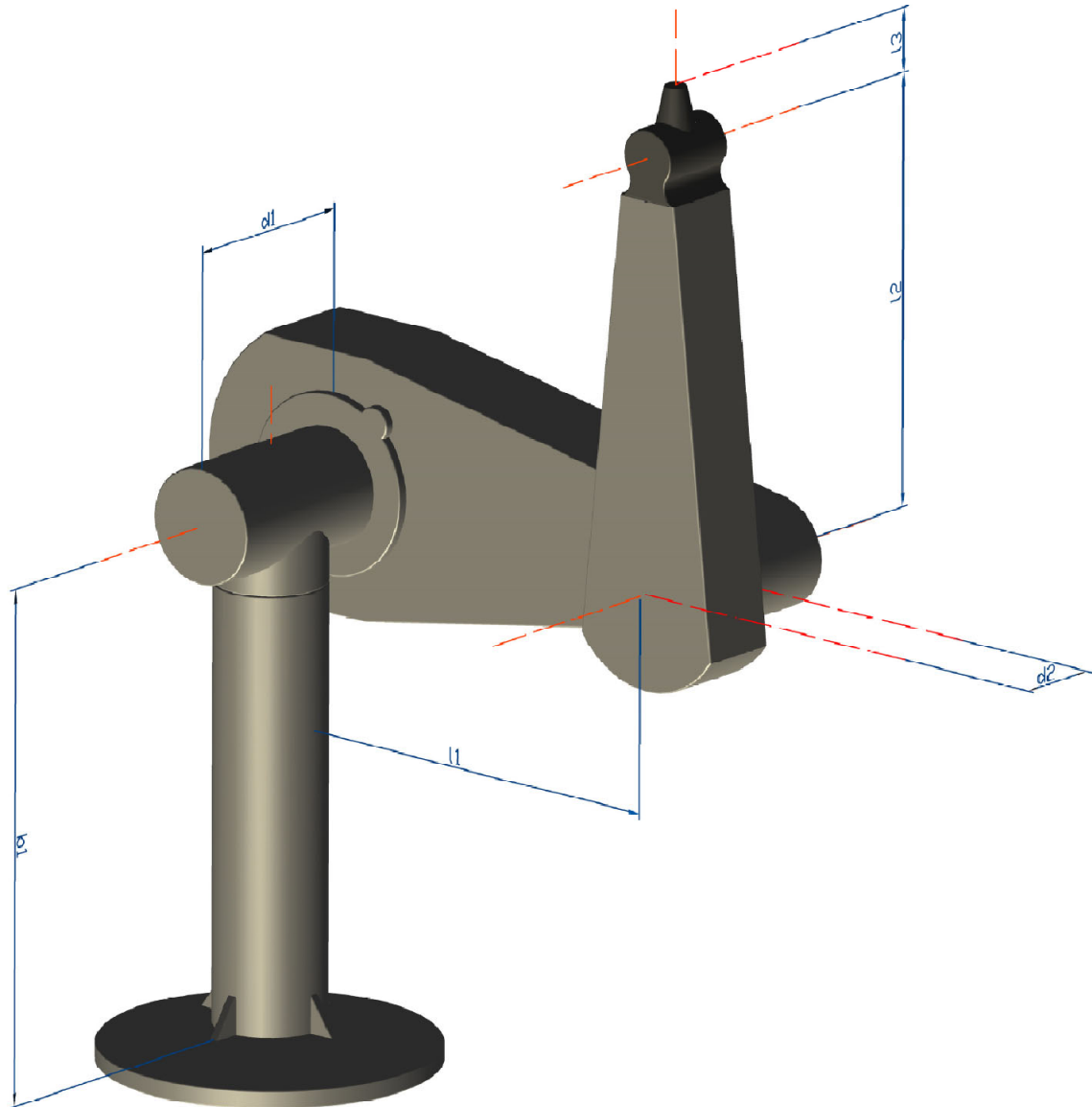
CINEMÁTICA DIRECTA

MATRIZ DENAVIT-HARTENBERG DEL MANIPULADOR PUMA560

A través del algoritmo de *Denavit-Hartenberg* se halla la matriz dh del robot PUMA560. A continuación puede verse como quedan los ejes en cada articulación y su matriz correspondiente.



Como referencia se tiene:



MATRIZ $RZ - TZ - TX - RX$ A UTILIZAR EN CADA UNA DE LAS FILAS PARA OBTENER $I-1A_i$

$C\theta_i$	$-C\alpha_i S\theta_i$	$S\alpha_i S\theta_i$	$a_i.C\theta_i$
$S\theta_i$	$C\alpha_i C\theta_i$	$-S\alpha_i C\theta_i$	$a_i.S\theta_i$
0	$S\alpha_i$	$C\alpha_i$	d_i
0	0	0	1

MATRICES PARCIALES

MATRIZ $0A_1$

$C1$	$-S1$	0	0
$S1$	$C1$	0	0
0	0	1	$B1$
0	0	0	1

MATRIZ 1A2

C2	0	-S2	0
S2	0	C2	0
0	-1	0	d1
0	0	0	1

MATRIZ 2A3

C3	-S3	0	l1.C3
S3	C3	0	l1.S3
0	0	1	-d2
0	0	0	1

MATRIZ 3A4

C4	0	S4	0
S4	0	-C4	0
0	1	0	l2
0	0	0	1

MATRIZ 4A5

C5	0	-S5	0
S5	0	C5	0
0	-1	0	0
0	0	0	1

MATRIZ 5A6

C6	0	S6	0
S6	0	-C6	0
0	1	0	l3
0	0	0	1

MATRIZ HOMOGENEA FINAL

$$T = {}^0A_6 = {}^0A_1 {}^1A_2 {}^2A_3 {}^3A_4 {}^4A_5 {}^5A_6$$

nx	ox	ax	Px
ny	oy	ay	Py
nz	oz	az	Pz
0	0	0	1

$$\begin{aligned} n_x &= C12 * C34 * C5 - S12 * S5) * C6 - C12 * S34 * S6 \\ o_x &= -(C12 * C34 * S5 + S12 * C5), \\ a_x &= (C12 * C34 * C5 - S12 * S5) * S6 + C12 * S34 * C6 \\ p_x &= -(C12 * C34 * S5 + S12 * C5) * l3 - S12 * l2 + C12 * l1 * C3 + S12 * d2 \end{aligned}$$

$$\begin{aligned} n_y &= S12 * C34 * C5 + C12 * S5) * C6 - S12 * S34 * S6 \\ o_y &= S12 * C34 * S5 + C12 * C5 \\ a_y &= (S12 * C34 * C5 + C12 * S5) * S6 + S12 * S34 * C6 \\ p_y &= (-S12 * C34 * S5 + C12 * C5) * l3 + C12 * l2 + S12 * l1 * C3 - C12 * d2 \end{aligned}$$

$$\begin{aligned} n_z &= S34 * C5 * C6 + C34 * S6) \\ o_z &= S34 * S5 \\ a_z &= -S34 * C5 * S6 + C34 * C6 \\ p_z &= S34 * S5 * l3 - l1 * S3 + d1 + b1 \end{aligned}$$

Donde $S_{xy} = \text{sen}(x + y)$, $C_{uv} = \text{cos}(u + v)$, $S_z = \text{sen}(z)$, $C_w = \text{cos}(w)$

MATRIZ JACOBIANA

$$J = \begin{bmatrix} \frac{\partial p_x}{\partial q_1} & \frac{\partial p_x}{\partial q_2} & \frac{\partial p_x}{\partial q_3} & \frac{\partial p_x}{\partial q_4} & \frac{\partial p_x}{\partial q_5} & \frac{\partial p_x}{\partial q_6} \\ \frac{\partial p_y}{\partial q_1} & \frac{\partial p_y}{\partial q_2} & \frac{\partial p_y}{\partial q_3} & \frac{\partial p_y}{\partial q_4} & \frac{\partial p_y}{\partial q_5} & \frac{\partial p_y}{\partial q_6} \\ \frac{\partial p_z}{\partial q_1} & \frac{\partial p_z}{\partial q_2} & \frac{\partial p_z}{\partial q_3} & \frac{\partial p_z}{\partial q_4} & \frac{\partial p_z}{\partial q_5} & \frac{\partial p_z}{\partial q_6} \end{bmatrix}$$

$$p_x = -(\cos(q1 + q2) * \cos(q3 + q4) * \sin(q5) + \sin(q1 + q2) * \cos(q5)) * l3 - \sin(q1 + q2) * l2 + \cos(q1 + q2) * l1 * \cos(q3) + \sin(q1 + q2) * d2$$

$$\frac{\partial p_x}{\partial q_1} = d2 * \cos(q1 + q2) - l2 * \cos(q1 + q2) - l3 * (\cos(q1 + q2) * \cos(q5) - \cos(q3 + q4) * \sin(q1 + q2) * \sin(q5)) - l1 * \sin(q1 + q2) * \cos(q3)$$

$$\frac{\partial p_x}{\partial q_2} = d2 * \cos(q1 + q2) - l2 * \cos(q1 + q2) - l3 * (\cos(q1 + q2) * \cos(q5) - \cos(q3 + q4) * \sin(q1 + q2) * \sin(q5)) - l1 * \sin(q1 + q2) * \cos(q3)$$

$$\frac{\partial p_x}{\partial q_3} = l3 * \cos(q1 + q2) * \sin(q3 + q4) * \sin(q5) - l1 * \cos(q1 + q2) * \sin(q3)$$

$$\frac{\partial p_x}{\partial q_4} = l3 * \cos(q1 + q2) * \sin(q3 + q4) * \sin(q5)$$

$$\frac{\partial p_x}{\partial q_5} = l3 * (\sin(q1 + q2) * \sin(q5) - \cos(q1 + q2) * \cos(q3 + q4) * \cos(q5))$$

$$\frac{\partial p_x}{\partial q_6} = 0$$

$$p_y = (-\sin(q1 + q2) * \cos(q3 + q4) * \sin(q5) + \cos(q1 + q2) * \cos(q5)) * l3 + \cos(q1 + q2) * l2 + \sin(q1 + q2) * l1 * \cos(q3) - \cos(q1 + q2) * d2$$

$$\frac{\partial p_y}{\partial q_1} = d2 * \sin(q1 + q2) - l2 * \sin(q1 + q2) - l3 * (\sin(q1 + q2) * \cos(q5) + \cos(q1 + q2) * \cos(q3 + q4) * \sin(q5)) + l1 * \cos(q1 + q2) * \cos(q3)$$

$$\frac{\partial p_y}{\partial q_2} = d2 * \sin(q1 + q2) - l2 * \sin(q1 + q2) - l3 * (\sin(q1 + q2) * \cos(q5) + \cos(q1 + q2) * \cos(q3 + q4) * \sin(q5)) + l1 * \cos(q1 + q2) * \cos(q3)$$

$$\frac{\partial p_y}{\partial q_3} = l3 * \sin(q1 + q2) * \sin(q3 + q4) * \sin(q5) - l1 * \sin(q1 + q2) * \sin(q3)$$

$$\frac{\partial p_y}{\partial q_4} = l3 * \sin(q1 + q2) * \sin(q3 + q4) * \sin(q5)$$

$$\frac{\partial p_y}{\partial q_5} = -l3 * (\cos(q1 + q2) * \sin(q5) + \cos(q3 + q4) * \sin(q1 + q2) * \cos(q5))$$

$$\frac{\partial p_y}{\partial q_6} = 0$$

$$p_z = b1 + d1 - l1 * \sin(q3) + l3 * \sin(q3 + q4) * \sin(q5)$$

$$\frac{\partial p_z}{\partial q_1} = 0$$

$$\frac{\partial p_z}{\partial q_2} = 0$$

$$\frac{\partial p_z}{\partial q_3} = l_3 * \cos(q_3 + q_4) * \sin(q_5) - l_1 * \cos(q_3)$$

$$\frac{\partial p_z}{\partial q_4} = l_3 * \cos(q_3 + q_4) * \sin(q_5)$$

$$\frac{\partial p_z}{\partial q_5} = l_3 * \sin(q_3 + q_4) * \cos(q_5)$$

$$\frac{\partial p_z}{\partial q_6} = 0$$

Por razones de espacio no se puede detallar como quedara la matriz Jacobiana, pero basta con reemplazar las ecuaciones diferenciales obtenidas en las respectivas posiciones de la matriz para observar el resultado final

CINEMÁTICA INVERSA

El objetivo del problema cinemático inverso consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot $q = [q_1, q_2, \dots, q_n]$, para que su extremo se posicione y oriente según una determinada localización espacial.

MÉTODO DE DESACOPLO CINEMÁTICO

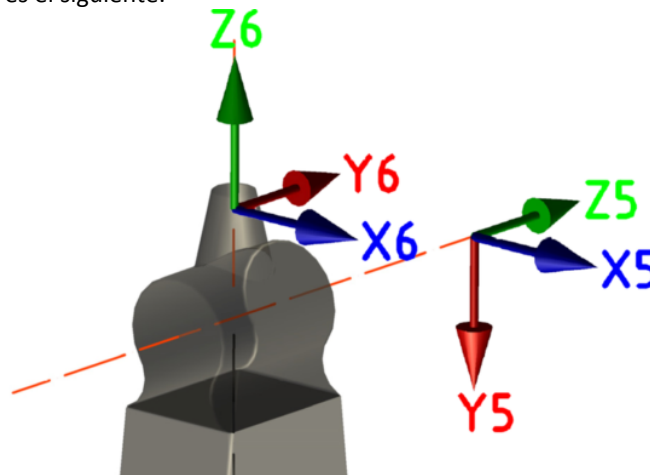
En todo brazo robot, en general, es preciso conseguir posicionar el extremo del robot en un punto del espacio y que la herramienta que aquél porta se oriente de una manera determinada.

El método de desacoplo cinemática separa ambos problemas, posición y orientación. Para ello, dada una posición y orientación final deseadas, establece las coordenadas del punto de corte de los 3 últimos ejes (que informalmente se denomina muñeca del robot) y calcula los valores de las tres primeras variables articulares (q_1, q_2, q_3) que consiguen posicionar este punto.

Para la orientación de la herramienta, los robots cuentan con otros tres grados de libertad adicionales, situados al final de la cadena cinemática y cuyos ejes, generalmente, se cortan en un punto (muñeca del robot). Si bien la variación de estos tres últimos grados de libertad origina un cambio en la posición final del extremo real del robot, su verdadero objetivo es poder orientar la herramienta del robot libremente en el espacio.

Entonces, a partir de los datos de orientación y de los calculados para el posicionamiento (q_1, q_2, q_3) obtiene los valores del resto de las variables articulares.

El proceso de cálculo es el siguiente:



El punto central de la muñeca del robot corresponde al origen del sistema $\{S5\}$:O5.

Por su parte, el punto final del robot será el origen del sistema $\{S6\}$:O6.

Definimos entonces los siguientes vectores:

$$\begin{aligned} P_m &= O_0 - O_5 = (X_m, Y_m, Z_m) - (0, 0, 0) && \text{Coordenada de la muñeca} \\ P_f &= O_0 - O_6 = (X_f, Y_f, Z_f) - (0, 0, 0) && \text{Coordenada de la posición final} \end{aligned}$$

Que van desde el origen del sistema asociado a la base del robot $\{S0\}$ hasta los puntos centro de la muñeca y fin del robot, respectivamente.

El vector director z_6 es el vector correspondiente a la orientación deseada y la distancia entre O5 y O6 medida a lo largo de z_6 , es l_3 (un parámetro asociado con el robot). Por lo tanto, las coordenadas del punto central de la muñeca (X_m, Y_m, Z_m) son obtenibles de la siguiente forma:

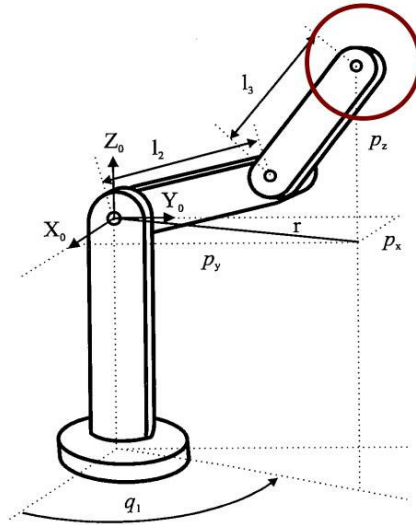
$$P_m = P_f - l_3 z_6 \rightarrow (X_m, Y_m, Z_m) = (X_f, Y_f, Z_f) - l_3 * (X_o, Y_o, Z_o) / |X_o, Y_o, Z_o|$$

Donde (X_o, Y_o, Z_o) son las coordenadas que determinan la orientación del extremo.

Una vez obtenidas estas coordenadas es posible obtener los valores de q_1, q_2 y q_3 para lograr el posicionamiento de la herramienta. Elegimos para ello el método geométrico.

MÉTODO GEOMÉTRICO PARA LA OBTENCIÓN DE Q1, Q2 Y Q3

Este método es adecuado para el caso de que se consideren sólo los primeros grados de libertad, dedicados a posicionar el extremo. El procedimiento en sí se basa en encontrar suficiente número de relaciones geométricas en las que intervendrán las coordenadas del extremo del robot, sus coordenadas articulares y las dimensiones físicas de sus elementos.



El dato de partida son la coordenadas $P_m = (X_m, Y_m, Z_m) = (P_x, P_y, P_z)$ referidas a $\{S_0\}$ en las que se quiere posicionar su extremo.

Este robot posee una estructura planar, quedando este plano definido por el ángulo de la primera variable articular (q_1). El valor de q_1 se obtiene inmediatamente como:

$$\tan(q_1) = \frac{P_y}{P_x} \rightarrow q_1 = \arctan\left(\frac{P_y}{P_x}\right)$$

Considerando ahora únicamente los elementos 2 y 3 que están situados en un plano, y utilizando el teorema del coseno, se tendrá:

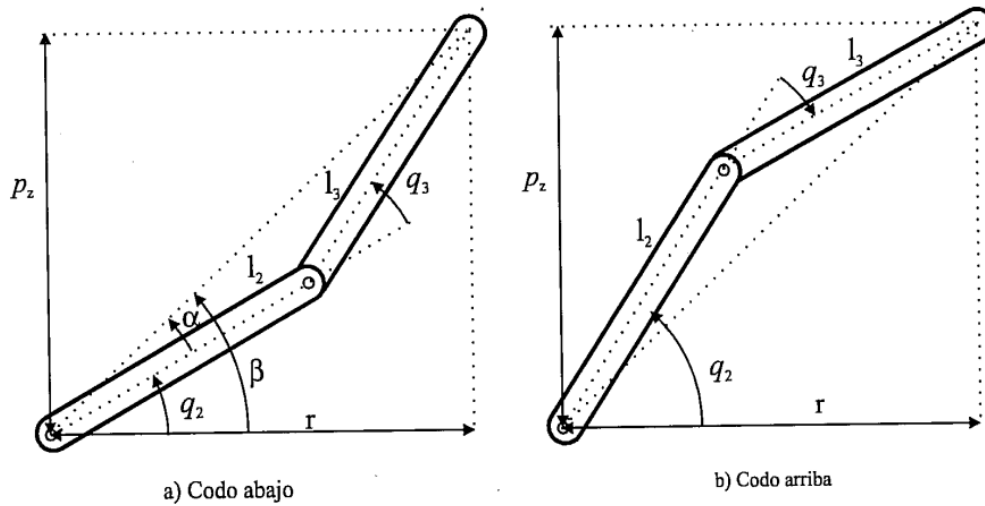
$$r^2 = P_x^2 + P_y^2$$

$$P_z^2 + r^2 = l_2^2 + l_3^2 + 2l_2l_3 \cos(q_3)$$

$$\cos(q_3) = \frac{P_x^2 + P_y^2 + P_z^2 - l_2^2 - l_3^2}{2l_2l_3}$$

Por motivos de velocidad de procesamiento es conveniente la utilización de \arctg en vez de \arccos

$$\sin(q_3) = \sqrt{1 + \cos^2(q_3)} \rightarrow q_3 = \arctan\left[\frac{\pm \sqrt{1 + \cos^2(q_3)}}{\cos(q_3)}\right]$$



Existen dos posibles soluciones para q_3 según se tome el signo positivo o el signo negativo en la raíz. Estas corresponden a las configuraciones de codo arriba y codo abajo del robot. q_2 se obtiene a partir de la diferencia entre β y α

$$q_2 = \beta - \alpha$$

donde

$$\beta = \arctan(P_z / r) = \arctan\left(\frac{P_z}{\sqrt{P_x^2 + P_y^2}}\right)$$

$$\alpha = \arctan\left(\frac{l_3 \cdot \sin(q_3)}{l_2 + l_3 \cdot \cos(q_3)}\right)$$

$$q_2 = \beta - \alpha = \arctan\left(\frac{P_z}{\sqrt{P_x^2 + P_y^2}}\right) - \arctan\left(\frac{l_3 \cdot \sin(q_3)}{l_2 + l_3 \cdot \cos(q_3)}\right)$$

De nuevo los dos posibles valores según la elección del signo dan lugar a dos valores diferentes de q_2 correspondientes a las configuraciones codo arriba y abajo. Las expresiones anteriores resuelven el problema de la cinemática inversa para la parte del posicionamiento del robot.

Queda ahora obtener los valores de q_4, q_5, q_6 que consiguen la orientación deseada.

Para ello, denominando 0R_6 a la submatriz de rotación de 0T_6 (la cual es en si la orientación buscada) se tendrá:

$${}^0R_6 = {}^0R_3 \cdot {}^3R_6$$

Donde 0R_3 es igual a la multiplicación de las matrices rotación de cada parte, ${}^0R_3 = {}^0R_1 \cdot {}^1R_2 \cdot {}^2R_3$ la cual puede determinarse ya que ya habíamos calculado los valores de q_1, q_2 y q_3 .

De este modo, despejando 3R_6 queda:

$${}^3R_6 = {}^0R_3^{-1} \cdot {}^0R_6 = {}^0R_3^T \cdot {}^0R_6$$

**HALLAMOS MATRIZ 0R3
MATRIZ 0R1**

C1	-S1	0
S1	C1	0
0	0	1

MATRIZ 1R2

C2	0	-S2
S2	0	C2
0	-1	0

MATRIZ 2R3

C3	-S3	0
S3	C3	0
0	0	1

$${}^0R_3 = {}^0R_1 \cdot {}^1R_2 \cdot {}^2R_3$$

MATRIZ 0R3

C12.C3	-C12.S3	-S12
S12.C3	-S12.S3	C12
-S3	-C3	0

MATRIZ 0R3 TRANSPUESTA

C12.C3	S12.C3	-S3
-C12.S3	-S12.S3	-C3
-S12	C12	0

HALLAMOS MATRIZ 0R6

Adoptamos el método de los ángulos de Euler para determinar la orientación de la herramienta. En nuestro caso en particular, la rotación en z, seguida de la rotación en u' (x inicial), seguida de la rotación en w'' (z inicial). Con estos tres ángulos queda determinada la orientación de la herramienta donde la matriz 0R6 correspondiente es:

$$\begin{aligned} \mathbf{R} = \mathbf{R}_{z,\phi} \mathbf{R}_{u',\theta} \mathbf{R}_{w'',\psi} &= \begin{bmatrix} C\phi & -S\phi & 0 \\ S\phi & C\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix} \begin{bmatrix} C\psi & -S\psi & 0 \\ S\psi & C\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \\ &= \begin{bmatrix} C\phi C\psi - S\phi C\theta S\psi & -C\phi S\psi - S\phi C\theta C\psi & S\phi S\theta \\ S\phi C\psi + C\phi C\theta S\psi & -S\phi S\psi + C\phi C\theta C\psi & -C\phi S\theta \\ S\theta S\psi & S\theta C\psi & C\theta \end{bmatrix} \end{aligned}$$

Donde ϕ , θ y ψ son respectivamente el ángulo de rotación en z, en u' y en w''. Recordamos que 3R6 la obtenemos:

$${}^3R_6 = {}^0R_3^{-1} \cdot {}^0R_6 = {}^0R_3^T \cdot {}^0R_6$$

RELACIÓN ENTRE 3R6 Y Q4, Q5 Y Q6

3R6 puede obtenerse, además, como $3R6 = 3R4 \cdot 4R5 \cdot 5R6$

MATRIZ 3R4

C4	0	S4
S4	0	-C4
0	1	0

MATRIZ 4R5

C5	0	-S5
S5	0	C5
0	-1	0

MATRIZ 5R6

C6	0	S6
S6	0	-C6
0	1	0

MATRIZ 3R6

$3R6 = 3R4 \cdot 4R5 \cdot 5R6$

$C4.C5.C6 - S4.S6$	$-C4.S5$	$C4.C5.S6 + S4.C6$
$S4.C5.C6 + C4.S6$	$-S4.S5$	$S4.C5.S6 - C4.C6$
$S5.C6$	$C5$	$S5.S6$

De las ecuaciones, para la obtención de q4, q5 y q6 utilizaremos

$$R12 = -C4 \cdot S5$$

$$R22 = -S4 \cdot S5$$

$$R22 / R12 = \tan(q4) / \cos(q4) = \tan(q4) \rightarrow q4 = \arctg(R22 / R12)$$

$$R23 = \cos(q5) \rightarrow q5 = \arccos(R23)$$

$$R13 = S5 \cdot C6$$

$$R33 = S5 \cdot S6$$

$$R33 / R13 = \tan(q6) / \cos(q6) = \tan(q6) \rightarrow q6 = \arctg(R33 / R13)$$

INTRODUCCIÓN SOBRE LA DINÁMICA DEL ROBOT

En la sección anterior se ha considerado la geometría de los robots, así como el movimiento sin analizar las fuerzas que lo producen. Como se sabe, las velocidades lineales y angulares vienen dadas por las fuerzas y pares que se aplican a la estructura mecánica y dependen también de las magnitudes de las masas y su distribución. Las relaciones involucradas constituyen el modelo dinámico del manipulador.

Este modelo, establece la relación matemática entre:

- La localización del robot definida por sus variables articulares o por las coordenadas de localización de su extremo, y sus derivadas: velocidad y aceleración.
- Las fuerzas y pares aplicados en las articulaciones (o en el extremo del robot)
- Los parámetros dimensionales del robot, como longitud, masas e inercias de sus elementos.

La obtención de este modelo para mecanismos de uno o dos grados de libertad no es excesivamente compleja, pero a medida que el número de grados de libertad aumenta, el planteamiento y obtención del modelo dinámico se complica enormemente. Por este motivo no siempre es posible obtener un modelo dinámico expresado de una forma cerrada, esto es, mediante una serie de ecuaciones, normalmente de tipo diferencial de segundo orden, cuya integración permita conocer que movimiento surge al aplicar unas fuerzas o qué fuerzas hay que aplicar para obtener un movimiento determinado. El modelo dinámico debe ser resuelto entonces de manera iterativa mediante la utilización de un procedimiento numérico.

Es por esto que en este trabajo práctico se hará hincapié en la utilización de herramientas informáticas, tales como ToolKit HEMERO para Matlab. Este calcula el modelo dinámico completo de forma analítica empleando para ello el método de Newton-Euler recurrente.

A modo introductorio diremos que la formulación de Newton-Euler utiliza las ecuaciones de equilibrio de fuerzas y torques.

$$\sum F = m\dot{v}$$

$$\sum T = I\dot{\omega} + \omega \times (I\omega)$$

Un adecuado desarrollo de estas ecuaciones conduce a una formulación recursiva en la que se obtienen la posición, velocidad y aceleración del eslabón i referidos a la base del robot a partir de los correspondientes valores de (posición, velocidad y aceleración) del eslabón $i-1$ y del movimiento relativo de la articulación i . De este modo, partiendo del eslabón 1 se llega al eslabón n .

Con estos datos se procede a obtener las fuerzas y torques actuantes sobre el eslabón i referidos a la base del robot a partir de los correspondientes valores del eslabón $i + 1$, recorriéndose de esta forma todos los eslabones desde el eslabón n al eslabón 1.

El algoritmo se basa en operaciones vectoriales (con productos escalares y vectoriales entre magnitudes vectoriales, y productos de matrices con vectores) siendo más eficiente en comparación con las operaciones matriciales asociadas a la formulación Lagrangiana.

De hecho, el orden de complejidad computacional de la formulación recursiva de Newton-Euler es $O(n)$ lo que indica que depende directamente del número de grados de libertad. Si lo comparamos con la formulación Lagrangiana, en esta, el orden de complejidad computacional es $O(n^4)$. Es decir, el número de operaciones a realizar crece con la potencia 4 del número de grados de libertad.

ANÁLISIS DINÁMICO

Para la realización del análisis dinámico, decidimos basarnos en la utilización del ToolKit denominado HEMERO (Herramienta Matlab-simulink para la Enseñanza de la Robótica).

Esta herramienta, puede ser empleada tanto para los manipuladores robóticos como para los robots móviles y está compuesta por funciones MATLAB y por bloques de Simulink que se han definido para facilitar la simulaciones de los modelos cinemáticos y dinámicos.

PARA LA UTILIZACIÓN DE HEMERO

Cada vez que se quiera utilizar una función de HEMERO relacionada con la cinemática se deberán introducir en una matriz los parámetros de Denavit-Hartenberg del manipulador, de acuerdo con la notación de Craig (1986). El modo de introducir dicha información en esa matriz es el siguiente:

- Habrá una fila por cada enlace que tenga el manipulador.
- Cada fila tendrá el siguiente formato: $[\alpha(i-1) \ a(i-1) \ \theta(i) \ d(i) \ \sigma(i)]$ donde:
 - $\alpha(i-1)$, $a(i-1)$, $\theta(i)$, $d(i)$ son los parámetros de D-H según Craig (1986).
 - $\sigma(i)$ indicará el tipo de articulación (será 0 si es de rotación y un número distinto de cero si por el contrario es prismática).

Así pues para un robot con n enlaces, la matriz sería de dimensiones $n \times 5$.

Todos los ángulos deberán ser introducidos en radianes. Las longitudes $a(i-1)$ y $d(i)$ podrán ser expresadas en cualquier unidad y sólo habrá que tener cuidado de recordar que las transformaciones homogéneas y los Jacobianos que se calculen aparecerán en esas mismas unidades.

Quedando entonces la matriz dh en Matlab como la indicada a continuación.

Además, cada vez que se quiera utilizar una función de HEMERO relacionada con la dinámica será necesario introducir en una matriz los parámetros de D-H del manipulador, junto con ciertos parámetros dinámicos. El modo de introducir esta información en dicha matriz (a la que se denominará genéricamente DYN) es el siguiente:

Habrà una fila por cada enlace que tenga el manipulador y cada fila tendrá el siguiente formato:

1	$\alpha(i-1)$	Parámetros de Denavit-Hartenberg
2	$a(i-1)$	
3	$\theta(i)$	
4	$d(i)$	
5	$\sigma(i)$	Tipo de articulación, 0 si es de rotación y 1 si es prismática
6	masa	Masa del enlace i
7	r_x	Centro de masas del enlace respecto al cuadro de referencia de dicho enlace
8	r_y	
9	r_z	
10	I_{xx}	Elementos del tensor de inercia referido al centro de masas del enlace
11	I_{yy}	
12	I_{zz}	
13	I_{xy}	
14	I_{yz}	
15	I_{xz}	
16	J_m	Inercia de la armadura
17	G	Velocidad de la articulación / velocidad del enlace
18	B	Fricción viscosa, referida al motor
19	T_{c+}	Fricción de Coulomb (rotación positiva), referida al motor
20	T_{c-}	Fricción de Coulomb (rotación negativa), referida al motor

Así pues para un robot con n enlaces, la matriz DYN tendría dimensiones $n \times 20$. Todos los ángulos deberán ser introducidos en radianes. El resto de parámetros de la matriz podrán tener las unidades que se deseen, siempre que se sea coherente en el uso de dichas unidades. Es decir que si se introducen las masas en Kg y los centros de masas en metros, al escribir el tensor de inercia se deberá expresar en Kg m^2 .

Para el estudio del brazo robot PUMA 560, hemos tomado los valores para la matriz de parámetros dinámicos del libro de Olleros, quien a su vez, indica que estos valores fueron determinados por Armstrong y otros (1986).

Quedando entonces la matriz dh y de parámetros dinámicos de trabajo sobre Matlab con el ToolKit HEMERO, detalladas a continuación.

MATRIZ DE PARAMETROS DINÁMICOS DE TRABAJO SOBRE MATLAB PUMA560

α	a	θ	d	σ	masa	rx	ry	rz	Ixx	Iyy	Izz	Ixy	Iyz	Ixz	Jm	G	B	Tc+	Tc-
0	0	0	b1	0	0	0	0	0	0	0	0.35	0	0	0	291e-6	-62.61	0	0	0
-90	0	0	d1	0	17.4	0.068	0.006	-0.016	0.13	0.524	0.539	0	0	0	409e-6	107.815	0	0	0
0	l1	0	-d2	0	4.8	0	-0.07	0.014	0.066	0.0125	0.066	0	0	0	299e-6	-53.7063	0	0	0
90	-d3	0	l2	0	0.82	0	0	-0.019	1.8e-3	1.8e-3	1.3e-3	0	0	0	35e-6	76.0364	0	0	0
-90	0	0	0	0	0.34	0	0	0	0.3e-3	0.3e-3	0.4e-3	0	0	0	35e-6	71.923	0	0	0
90	0	0	l3	0	0.09	0	0	0.032	0.15e-3	0.15e-3	0.04e-3	0	0	0	35e-6	76.686	0	0	0

MATRIZ DH DE TRABAJO SOBRE MATLAB PUMA560

α	a	θ	d	σ
0	0	0	b1	0
-90	0	0	d1	0
0	l1	0	-d2	0
90	-d3	0	l2	0
-90	0	0	0	0
90	0	0	l3	0

DIMENSIONES

b1	0.672
d1	0.244
d2	0.094
l1	0.432
l2	0.433
l3	0.056

La función principal para el análisis dinámico será:

`rne(dyn, q, qd, qdd)`

Que calcula el modelo dinámico completo de forma analítica empleando para ello el método de **Newton-Euler recurrente**, tal como se describe en el capítulo 5 del libro de Olleros

Adicionalmente, para la utilización de la misma, es necesario pasarle un vector con la aceleración de la gravedad que sufre el manipulador, así como los valores de las variables articulares y de las velocidades y aceleraciones articulares. La función `rne` devuelve como resultado los pares ejercidos en cada articulación.

Para la gráfica del robot, se utilizó otra función del toolbox:

`plotbot(dh, q)`

Esta función construye una representación gráfica del robot, a partir de los parámetros cinemáticos contenidos en DH y de los valores de las variables articulares (Q) que se le pasen.

SUPOSICIONES DE CÁLCULO

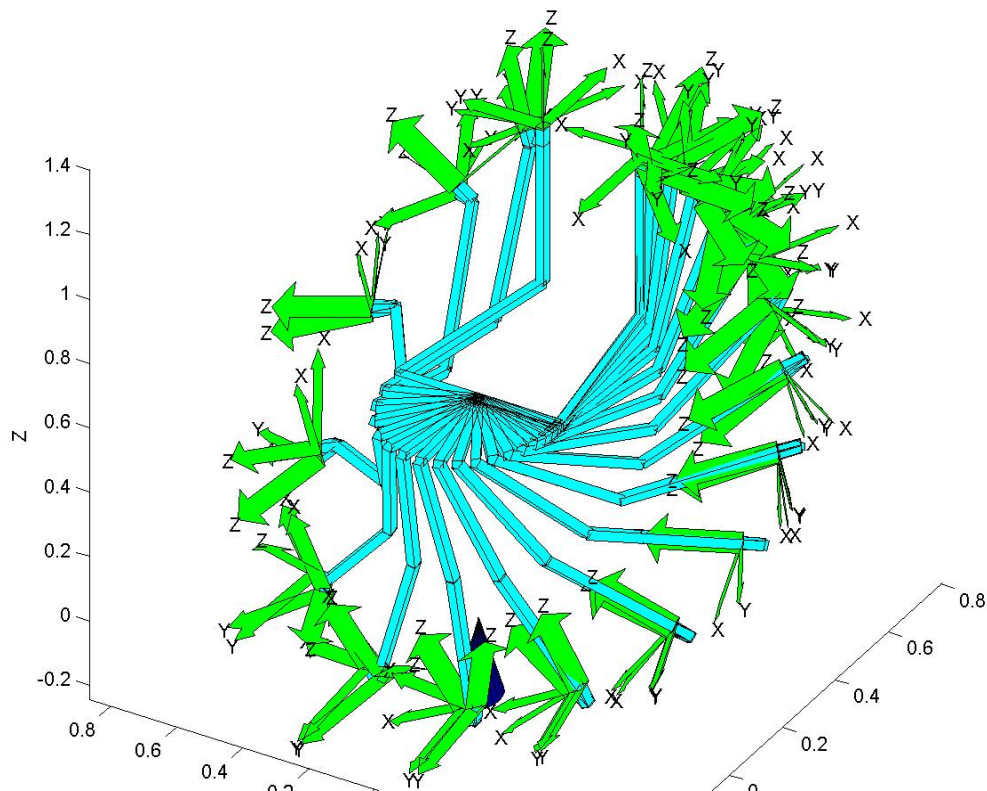
Para comenzar a plantear el análisis, supusimos un movimiento del tipo senoidal aplicado a los ejes 1 a 3 y otro del mismo tipo pero con distinta frecuencia aplicado a los ejes 4 a 6. La idea de esto es poder aplicar un movimiento tal en el que se logren velocidades y aceleraciones cambiantes en todo momento. En si los valores con los que cambia el ángulo fueron elegidos para simular un movimiento tal que reemplace al movimiento de un brazo humano.

Con esto en mente se diseñó una función que comience en un ángulo (q_{ix}) y termine en otro ángulo (q_{fx}), haciéndolo en un tiempo determinado (t_{transx}), y según una función seno.

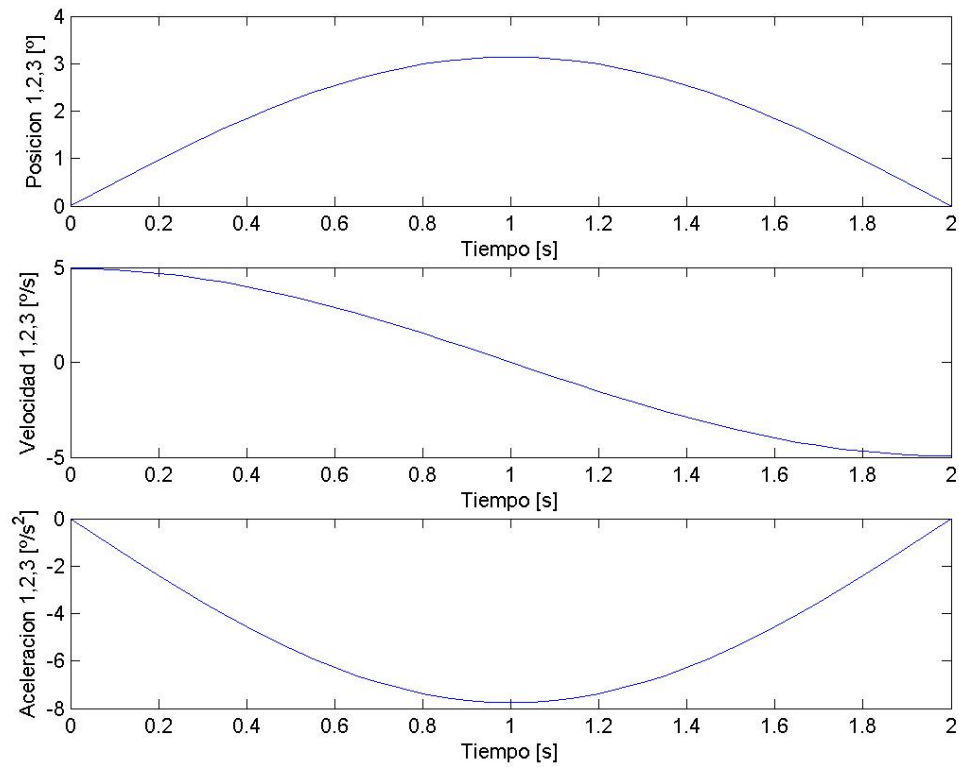
Esta misma función se implementa para todos los motores, pasando a las funciones del toolbox HEMERO los datos de posición, velocidad y aceleración angular. Obteniéndose como resultado el torque de cada motor.

El proceso se repite desde un tiempo inicial (t_1) a un tiempo final (t_2), de a pasos determinados ($pasot$), pudiendo graficarse cada una de los valores intermedios.

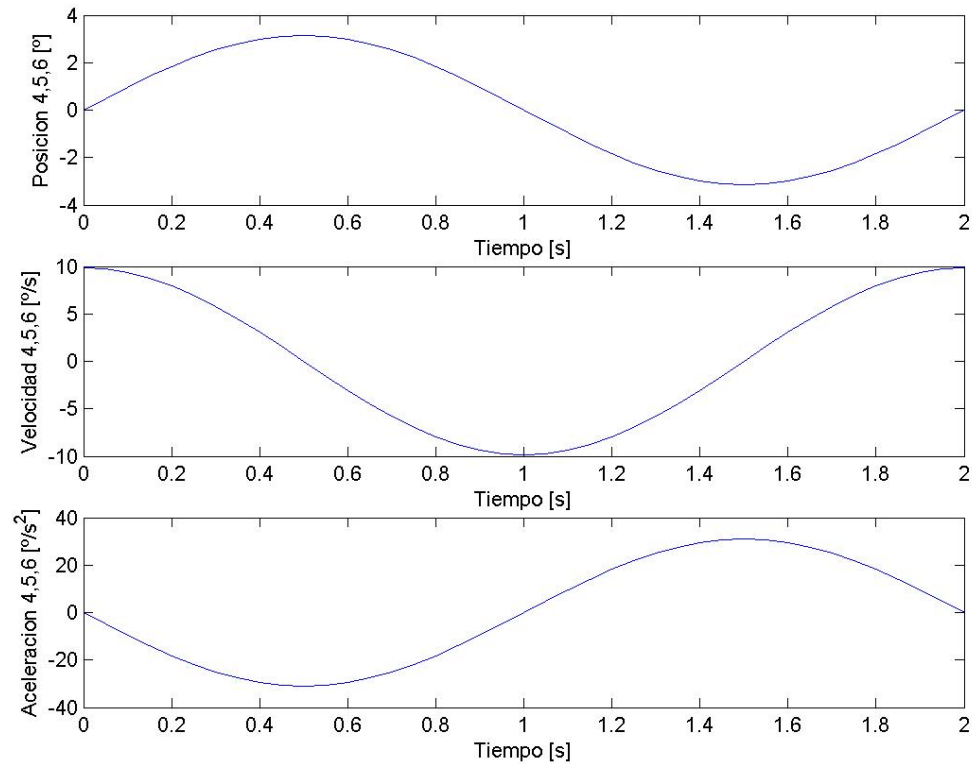
SIMULACION DE MOVIENTO DEL BRAZO



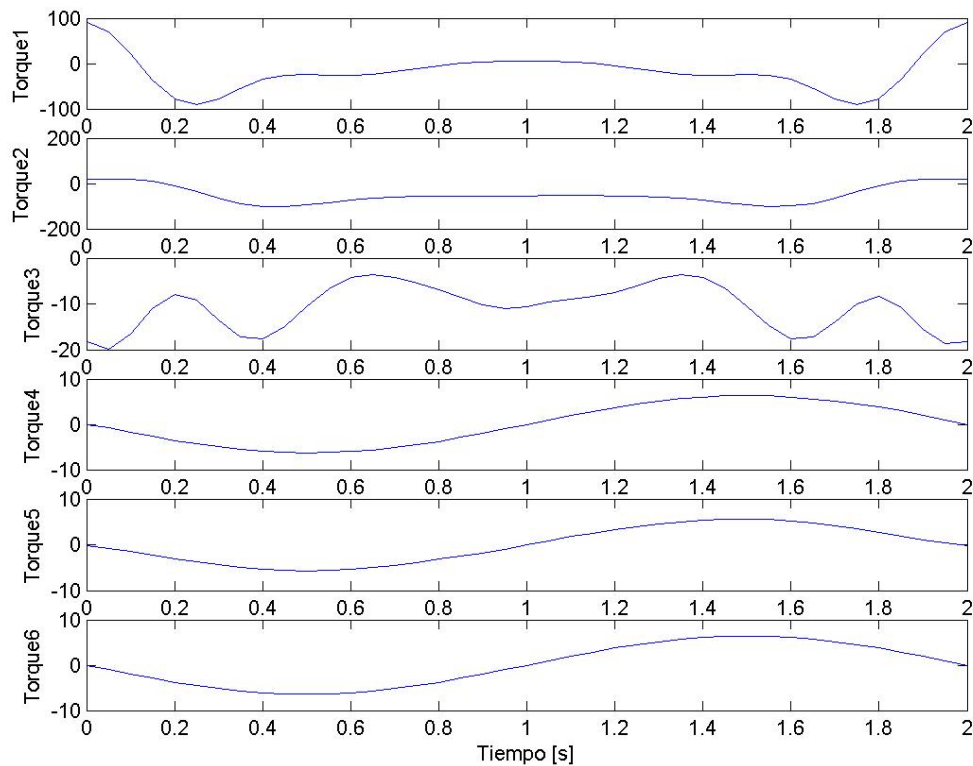
GRÁFICA DE POSICIÓN, VELOCIDAD Y ACELERACION SOBRE MOTOR 1 A 3



GRÁFICA DE POSICIÓN, VELOCIDAD Y ACELERACION SOBRE MOTOR 4 A 6



GRÁFICA DE COMPARACIÓN DE TORQUES



RESULTADOS

Gracias a la utilización del *ToolKit HEMERO* y las herramientas de simulación del Matlab se obtuvieron una graficas finales en las cuales puede observarse con claridad la evolución del torque de los motores en función de la posición, velocidad y aceleración de las articulaciones.

En estas se revela que el máximo torque para el motor 1 y 2 es de aproximadamente 100N.m, para el motor 3 de aproximadamente 20N.m, y para los motores 4 a 6 de 10N.m.

Debe tenerse en cuenta que los torques obtenidos son los finales, o sea, los necesarios en la articulación en si. De manera que con la utilización de unidades de reducción, el torque necesario seria inferior, dependiendo de esta. En estas aplicaciones, donde los requerimientos de velocidad final no son importantes, esto se hace más factible ya que se intercambia velocidad del motor por torque final.

Según algunos sitios el PUMA560 tiene reductores de hasta 110 veces.

COMPILADOR

INTRODUCCIÓN

¿Porque realizar un compilador?

Para la utilización de nuestro robot, así como de cualquier otro dispositivo que necesite reprogramarse con cierta periodicidad, se hacen necesarios un lenguaje de programación y su compilador asociado. Esto es así ya que no sería productivo que el usuario final de nuestro robot (o tecnología) necesite comprender el lenguaje que utilizamos nosotros para la programación del mismo. De esta forma el usuario interactúa con el robot a través de un lenguaje de un nivel superior, mucho mas intuitivo y sencillo. El compilador es el puente entre el lenguaje del usuario y de el de programación del robot.

¿Qué software disponible existe y cual se utilizó?

Software disponibles hay muchos, los más conocidos son FLEX en conjunto con BISON, y algo un poco más nuevo, ANTLR. Nosotros basamos nuestro compilador en FLEX+BISON ya que su programación es en c y estamos más acostumbrados a su utilización.

¿Cuál es la idea?

Lo que se busca es que, al alimentar con texto plano al FLEX, este reconozca combinaciones de caracteres predefinidas (llamados “tokens”), ejecute una acción y genere un archivo que se utilizara para que BISON siga con el trabajo. BISON, a continuación, realiza un análisis de estos “tokens” para detectar estructuras predefinidas y ejecutar una acción en base a esto.

Se generan archivos en ‘c’ que luego se pueden compilar con, por ejemplo, el gcc.

¿Cómo funciona?

Las herramientas FLEX y BISON permiten realizar análisis léxico y sintáctico, respectivamente, sobre una entrada de texto plano. Mediante ambos análisis se pueden realizar sondeos para determinar el reconocimiento de sentencias independientes del contexto, lo cual constituye la base para el procesamiento sintáctico de los lenguajes de programación que ustedes ya conocen.

Cada una por separado:

FLEX permite generar plantillas de análisis léxico basadas en expresiones regulares. Para cada expresión regular se puede definir una sentencia ejecutable. Cuando se detecta un patrón que calza con una expresión regular, se ejecuta la acción correspondiente definida.

BISON permite generar análisis sintáctico a través de la definición de una gramática independiente del contexto. Para cada producción de la gramática es posible definir una regla de procesamiento que permite realizar acciones semánticas en fase de parsing.

No es la idea explicar más que esto, ya que existen cantidad de manuales en la red.

¿Qué esperamos obtener de nuestro compilador?

La idea es, en esta primera etapa, entrarle por línea de comandos con algunas instrucciones básicas para el posicionamiento del brazo.

Lo mas práctico sería que nuestro compilador tome la instrucción `movr(xi yi zi xf yf zf t1)`, la descomponga por “tokens” y obtenga los puntos interpolados entre los dos puntos de inicio y fin. Más luego realice la cinemática inversa según la configuración de nuestro robot, y obtener los q_1 , q_2 , q_3 para cada punto interpolado. Esto, finalmente, canalizarlo a una puerta serie para enviarlo como parámetros de posición para el FPGA.

Es así que creamos un archivo para el FLEX donde definíamos lo que nosotros necesitábamos:

<code>digito [0-9]</code>	Definimos lo que era un dígito
<code>-?{digito}+</code>	un numero entero
<code>-?{digito}+\".\"{digito}*</code>	un número real
<code>movr</code>	nuestra función movr
<code>pause</code>	función pausa
<code>ayuda</code>	función ayuda
<code>exit</code>	función salir
<code>"(</code>	Definimos lo que era un abrir paréntesis
<code>)"</code>	Definimos lo que era un cerrar paréntesis
<code>;"</code>	Definimos lo que era un punto y coma.

Y un archivo para el BISON donde indicamos como interrelacionar estas definiciones, y si encontraba alguna correspondiente a nuestras funciones de lenguaje de programación del robot realice cierta acción. En nuestro caso, que defina la trayectoria, la divida en una cantidad determinada de avances y para cada una de estas posiciones realice la cinemática inversa.

El resultado será impreso en pantalla, con los valores de “movimiento numero”, “angulos” y “tiempo”.

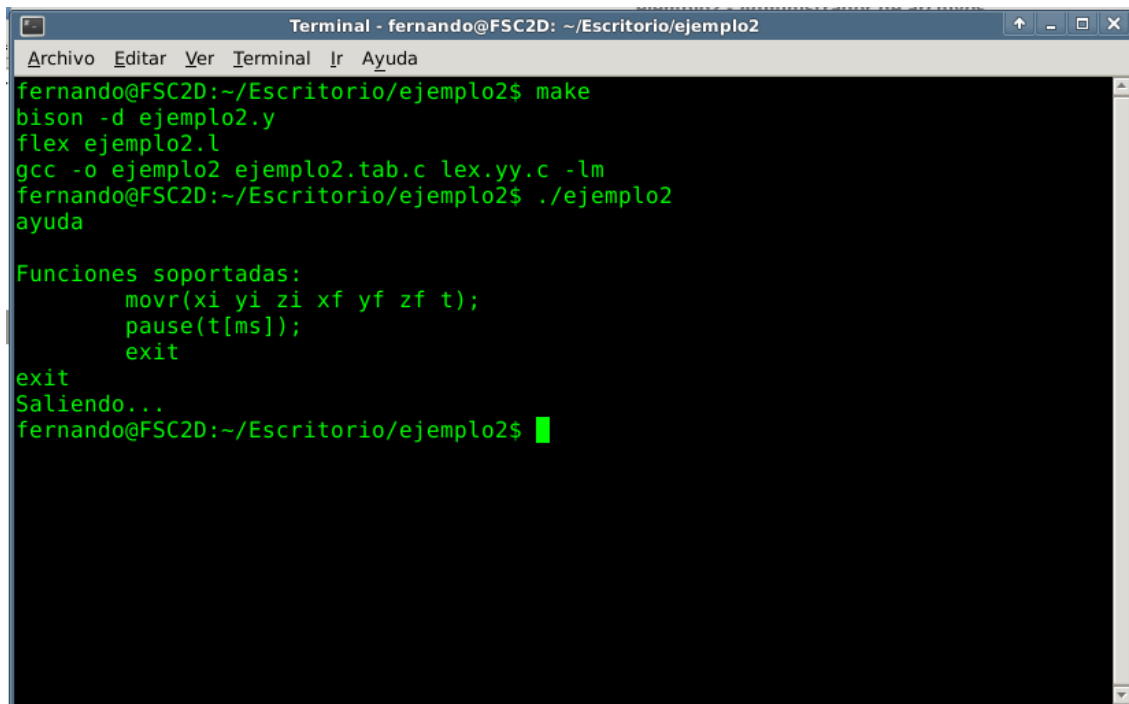
Se adjunta programa `ejemplo2.l`, `ejemplo2.y` y un `make`.

Para compilarlo es necesario tener instalado el Flex, Bison y gcc y sobre linux, en consola: “make”, para limpiar los archivos creados, “make clean”

Para ejecutarlo sobre linux, en la consola: “./ejemplo2”

Con el comando ayuda pueden obtenerse las funciones soportadas.

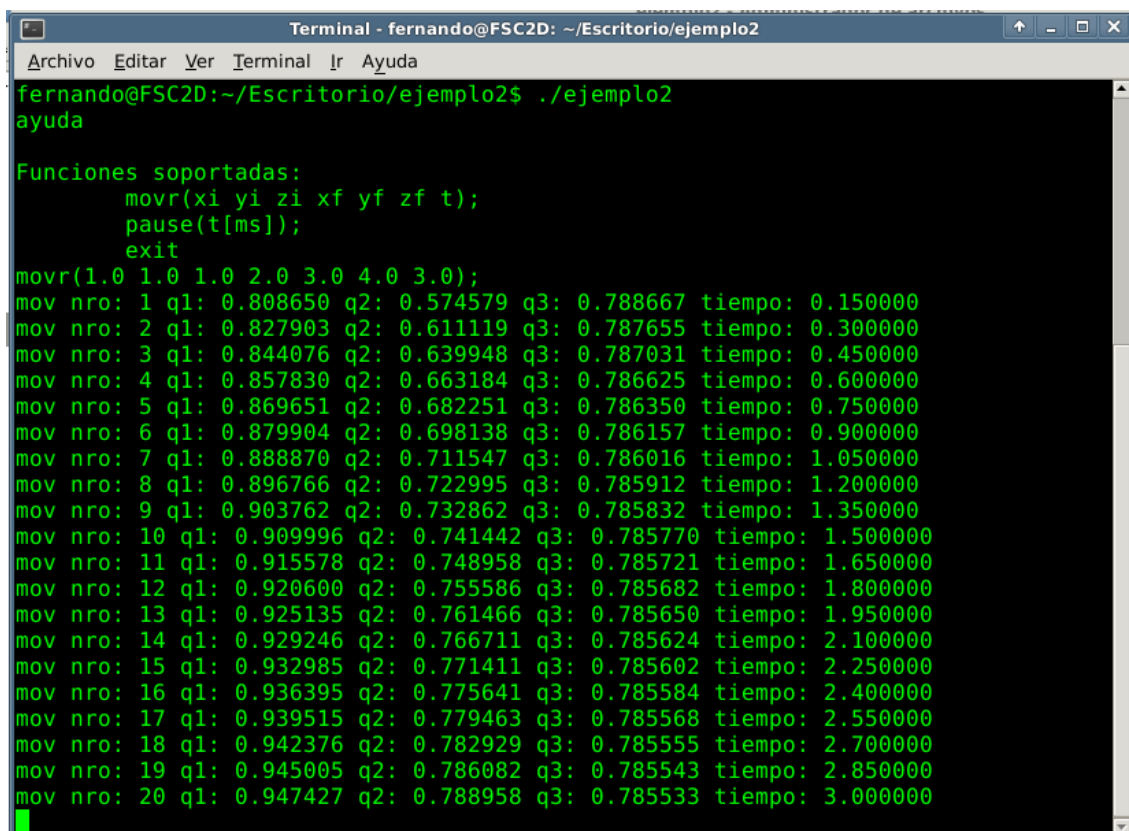
CAPTURAS DE PANTALLA



```

Terminal - fernando@FSC2D: ~/Escritorio/ejemplo2
Archivo Editar Ver Terminal Ir Ayuda
fernando@FSC2D:~/Escritorio/ejemplo2$ make
bison -d ejemplo2.y
flex ejemplo2.l
gcc -o ejemplo2 ejemplo2.tab.c lex.yy.c -lm
fernando@FSC2D:~/Escritorio/ejemplo2$ ./ejemplo2
ayuda

Funciones soportadas:
    movr(xi yi zi xf yf zf t);
    pause(t[ms]);
    exit
exit
Saliendo...
fernando@FSC2D:~/Escritorio/ejemplo2$ █
  
```



```

Terminal - fernando@FSC2D: ~/Escritorio/ejemplo2
Archivo Editar Ver Terminal Ir Ayuda
fernando@FSC2D:~/Escritorio/ejemplo2$ ./ejemplo2
ayuda

Funciones soportadas:
    movr(xi yi zi xf yf zf t);
    pause(t[ms]);
    exit
movr(1.0 1.0 1.0 2.0 3.0 4.0 3.0);
mov nro: 1 q1: 0.808650 q2: 0.574579 q3: 0.788667 tiempo: 0.150000
mov nro: 2 q1: 0.827903 q2: 0.611119 q3: 0.787655 tiempo: 0.300000
mov nro: 3 q1: 0.844076 q2: 0.639948 q3: 0.787031 tiempo: 0.450000
mov nro: 4 q1: 0.857830 q2: 0.663184 q3: 0.786625 tiempo: 0.600000
mov nro: 5 q1: 0.869651 q2: 0.682251 q3: 0.786350 tiempo: 0.750000
mov nro: 6 q1: 0.879904 q2: 0.698138 q3: 0.786157 tiempo: 0.900000
mov nro: 7 q1: 0.888870 q2: 0.711547 q3: 0.786016 tiempo: 1.050000
mov nro: 8 q1: 0.896766 q2: 0.722995 q3: 0.785912 tiempo: 1.200000
mov nro: 9 q1: 0.903762 q2: 0.732862 q3: 0.785832 tiempo: 1.350000
mov nro: 10 q1: 0.909996 q2: 0.741442 q3: 0.785770 tiempo: 1.500000
mov nro: 11 q1: 0.915578 q2: 0.748958 q3: 0.785721 tiempo: 1.650000
mov nro: 12 q1: 0.920600 q2: 0.755586 q3: 0.785682 tiempo: 1.800000
mov nro: 13 q1: 0.925135 q2: 0.761466 q3: 0.785650 tiempo: 1.950000
mov nro: 14 q1: 0.929246 q2: 0.766711 q3: 0.785624 tiempo: 2.100000
mov nro: 15 q1: 0.932985 q2: 0.771411 q3: 0.785602 tiempo: 2.250000
mov nro: 16 q1: 0.936395 q2: 0.775641 q3: 0.785584 tiempo: 2.400000
mov nro: 17 q1: 0.939515 q2: 0.779463 q3: 0.785568 tiempo: 2.550000
mov nro: 18 q1: 0.942376 q2: 0.782929 q3: 0.785555 tiempo: 2.700000
mov nro: 19 q1: 0.945005 q2: 0.786082 q3: 0.785543 tiempo: 2.850000
mov nro: 20 q1: 0.947427 q2: 0.788958 q3: 0.785533 tiempo: 3.000000
█
  
```


COMPILADOR: EJEMPLO2.L (FLEX)

```
%{
#include <stdio.h>
#include "ejemplo2.tab.h"
%}

digito [0-9]

%%

-?{digito}+      {
                    yyval.entero = atoi(yytext);
                    return (NUM_ENTERO);
                }

-?{digito}+"."{digito}* {
                    yyval.flotante = atof(yytext);
                    return (NUM_FLOT);
                }

movr              {
                    return (FUNC_MOVR);
                }

pause             {
                    return (FUNC_PAUSE);
                }

ayuda             {
                    return (SIG_AYUDA);
                }

exit              {
                    return (SIG_EXIT);
                }

"("              {
                    return (APAR);
                }

")"              {
                    return (CPAR);
                }

";"              {
                    return (SEMICOLON);
                }

"\n"              {
                    return (yytext[0]);
                }

[ \t]+

.                ;
%%
```

COMPILADOR: EJEMPLO2.Y (BISON)

```
%{
#include <math.h>
#include <stdio.h>

int i, ts, pASOS=20;

float xi, yi, zi, xf, yf, zf, t, te;

float q1, q2, q3, cosq3;

float l2=0.432;
float l3=0.433;

%}

/* Declaraciones de BISON */

%union
{
    int entero;
    float flotante;
}

%token SIG_EXIT SIG_AYUDA
%token FUNC_MOVR FUNC_PAUSE
%token APAR CPAR SEMICOLON

%token <entero>NUM_ENTERO
%token <flotante>NUM_FLOT

/* Gramática */
%%

COMMANDS:
    | COMMANDS COMMAND
;

COMMAND:
    '\n'
    | MOVR
    | PAUSE
    | EXIT
    | AYUDA
;

MOVR:
    FUNC_MOVR APAR NUM_FLOT NUM_FLOT NUM_FLOT NUM_FLOT NUM_FLOT
    NUM_FLOT NUM_FLOT NUM_FLOT CPAR SEMICOLON '\n'
    {
        xi=$3;
        yi=$4;
        zi=$5;
        xf=$6;
        yf=$7;
        zf=$8;
        te=$9;

        t=0;
        i=0;

        while(t<te)
        {
            t=t+(te/pASOS);
```

```

        i++;
        xi=xi+((xf-xi)/pASOS);
        yi=yi+((yf-yi)/pASOS);
        zi=zi+((zf-zi)/pASOS);

        q1=atan(yi/xi);
        cosq3=(pow(xi,2)+pow(yi,2)+pow(zi,2)-pow(l2,2)-
pow(l3,2))/(2*l2*l3);
        q3=atan(sqrt(1+pow(cosq3,2))/cosq3);
        q2=atan(zi/sqrt(pow(xi,2)+pow(yi,2)))-
atan((l3*sin(q3))/(l2+l3*cosq3));

        printf("mov nro: %d q1: %f q2: %f q3: %f tiempo:
%f\n", i, q1, q2, q3, t);

    }
}

;

PAUSE:      FUNC_PAUSE NUM_ENTERO SEMICOLON '\n'
{
    ts=$2;
    sleep(ts);
    printf("\n\t (%d000)Sleeping...\n", $2);
}

;

AYUDA:      SIG_AYUDA
{
    printf("\nFunciones soportadas:\n\tmovr(xi yi zi xf yf zf
t);\n\tpause(t[ms]);\n\texit\n");
}

EXIT:      SIG_EXIT
{
    printf("Saliendo...\n");
    return 1;
}

;
%%

int main() {
    yyparse();
}

yyerror (char *s)
{
    printf ("%s\n", s);
}

int yywrap()
{
    return 1;
}

```

IMPLEMENTACIÓN EN CODIGO VHDL

En este apartado daremos un ejemplo de cómo se implementaría el control de los motores de cada una de las articulaciones, mediante la utilización de FPGA.

ESQUEMA DEL CONTROL

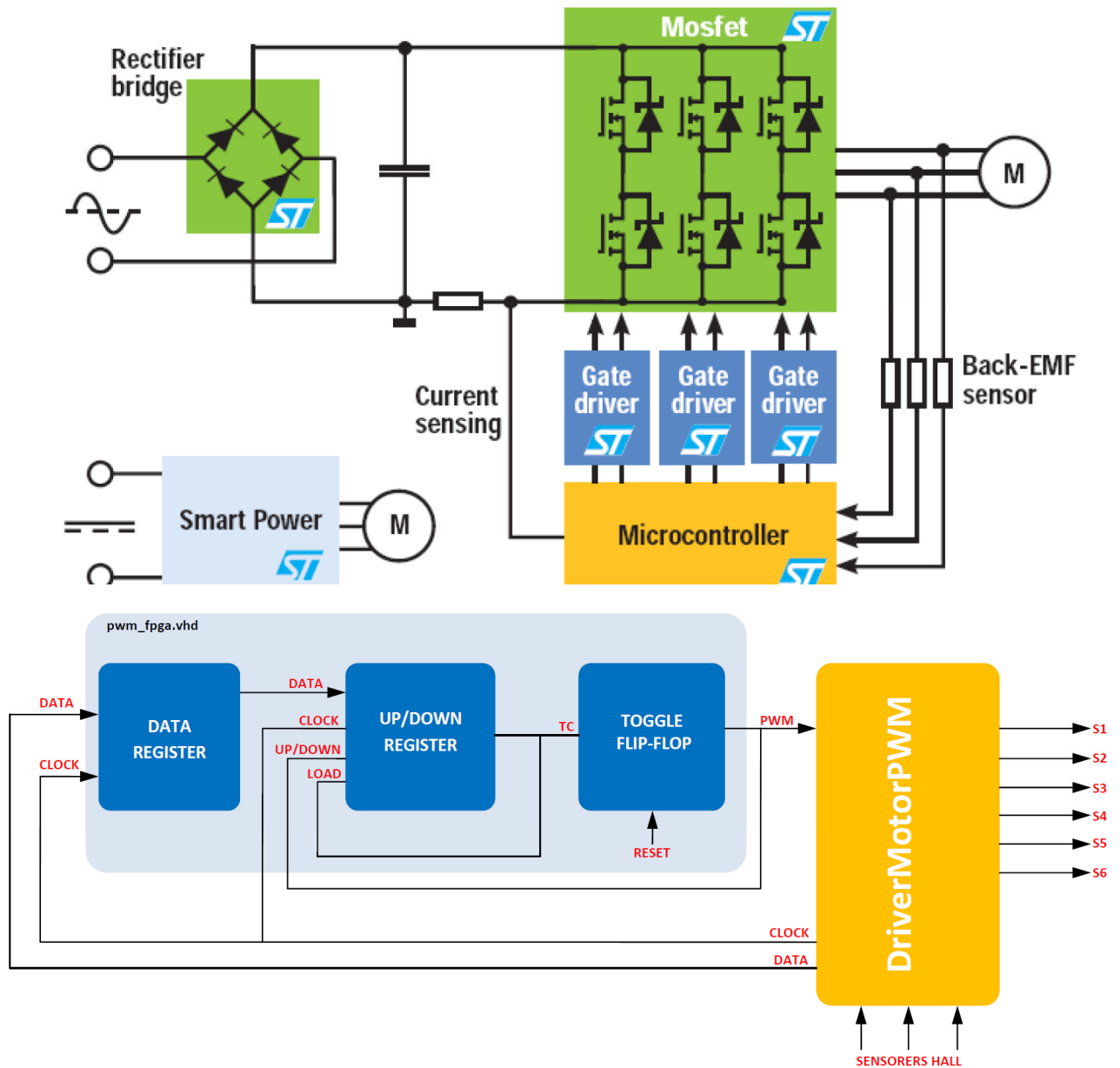


TABLA DE ESTADOS

	S1	S2	S3	S4	S5	S6
ESTADO 0	1	0	0	PWM	0	0
ESTADO 1	1	0	0	0	0	PWM
ESTADO 2	0		1	0	0	PWM
ESTADO 3	0	PWM	1	0	0	0
ESTADO 4	0	PWM	0	0	1	0
ESTADO 5	0		0	PWM	1	0

CÓDIGO EN VHDL DE DRIVER MOTOR PWM

```

library IEEE;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all ;
USE    work.user_pkg.all;

ENTITY DriverMotorPWM 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
    );

END DriverMotorPWM;

ARCHITECTURE arch_DriverMotorPWM OF DriverMotorPWM IS

--SEÑALES INTERNAS
SIGNAL sig_clock                : std_logic;
SIGNAL sig_reset                : std_logic;
SIGNAL sig_data_value          : std_logic_vector(7 downto 0);

--SIGNAL sig_pwm                : std_logic;
shared variable ENDSIM: boolean:=false;
constant clk_period:TIME:=100ns;

SIGNAL reg_out                  : std_logic_vector(7 downto 0);
SIGNAL cnt_out_int              : std_logic_vector(7 downto 0);
SIGNAL pwm_int, rco_int, clock_div10 : STD_LOGIC;
SIGNAL sel                      : integer range 0 to 5;
SIGNAL cont_sel                 : std_logic_vector(3 downto 0);

BEGIN

--GENERACION DE LA SEÑAL DE CLOCK
clk_gen: process
    BEGIN
        If ENDSIM = FALSE THEN
            sig_clock <= '1';
            wait for clk_period/2;
            sig_clock <= '0';
            wait for clk_period/2;
        else
            wait;
        end if;
    end process;

--GENERACION DE 3 VELOCIDADES DIFERENTES PARA EL CONTROL PWM
stimulus_process: PROCESS
    BEGIN
        sig_reset <= '1';
        wait for 100 ns;
        sig_reset <= '0';
        sig_data_value <= "11000000";
        wait for 100 us;
        sig_data_value <= "10000000";
        wait for 100 us;
        sig_data_value <= "01000000";
        wait for 100 us;
        sig_data_value <= "01000000";
        wait for 100 us;
        sig_data_value <= "11000000";
        wait for 100 us;
        sig_data_value <= "10000000";
        wait for 100 us;
        sig_data_value <= "01000000";
        wait for 100 us;
        sig_data_value <= "11000000";
    
```

```

        wait for 100 us;
    wait;

END PROCESS stimulus_process;

-- 8 BIT DATA REGISTER TO STORE THE MARKING VALUES .
-- THE MARKING VALUES WILL DETERMINE THE DUTY CYCLE OF PWM OUTPUT

PROCESS(sig_clock,reg_out,sig_reset)
    BEGIN
        IF (sig_reset = '1') THEN
            reg_out <="00000000";
        ELSIF (rising_edge(sig_clock)) THEN
            reg_out <= sig_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_package library

PROCESS (sig_clock, cnt_out_int, rco_int, reg_out)

    BEGIN
        IF (rco_int = '1') THEN
            cnt_out_int <= reg_out;
        ELSIF rising_edge(sig_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, sig_clock, sig_reset)
    BEGIN
        IF (sig_reset = '1') THEN
            rco_int <='1';
        ELSIF rising_edge(sig_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 (sig_clock, rco_int, sig_reset)
    BEGIN
        IF (sig_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;

```

```
--GENERACION DE UNA SEÑAL DE FRECUENCIA 10 VECES MENOR AL CLOCK
Gen_clk10: PROCESS (sig_reset, rco_int) IS
VARIABLE cuenta: INTEGER range 0 to 20;

BEGIN
    IF (sig_reset = '1') THEN
        cuenta:= 0;
        clock_div10 <= '1';
    ELSE
        IF rising_edge(rco_int) THEN
            IF (cuenta = 20) THEN
                cuenta:= 0;
                clock_div10 <= '1';
            ELSE
                cuenta:= cuenta + 1;
                clock_div10 <= '0';
            END IF;
        END IF;
    END IF;

END PROCESS;

GenVarState: PROCESS (sig_reset, clock_div10) IS
VARIABLE cuenta: INTEGER range 0 to 5;
BEGIN
    IF (sig_reset = '1') THEN
        cuenta:= 0;
    ELSE
        IF rising_edge (clock_div10) THEN
            IF (cuenta = 5) THEN
                cuenta:= 0;
            ELSE
                cuenta:= cuenta + 1;
            END IF;
        END IF;
    END IF;

    sel <= cuenta;

END PROCESS;

--CONTROL DE LOS DEL SERVOMOTOR
Cntrl_PWM_puenteH: PROCESS (sig_reset, sel, pwm_int) IS
BEGIN
    IF (sig_reset = '1') THEN
        S1<= '0'; S2<= '0'; S3<= '0'; S4<= '0'; S5<= '0'; S6<= '0';
    ELSE
        CASE sel IS
            WHEN 0 => --ESTADO #1
                S1<= '1';
                S2<= '0';
                S3<= '0';
                S4<= pwm_int;
                S5<= '0';
                S6<= '0';

            WHEN 1 => --ESTADO #2
                S1<= '1';
                S2<= '0';
                S3<= '0';
                S4<= '0';
                S5<= '0';
                S6<= pwm_int;

            WHEN 2 => --ESTADO #3
                S1<= '0';
                S2<= '0';
                S3<= '1';
                S4<= '0';
```

```

        S5<= '0';
        S6<= pwm_int;

    WHEN 3 =>                                --ESTADO #4
        S1<= '0';
        S2<= pwm_int;
        S3<= '1';
        S4<= '0';
        S5<= '0';
        S6<= '0';

    WHEN 4 =>                                --ESTADO #5
        S1<= '0';
        S2<= pwm_int;
        S3<= '0';
        S4<= '0';
        S5<= '1';
        S6<= '0';

    WHEN 5 =>                                --ESTADO #6
        S1<= '0';
        S2<= '0';
        S3<= '0';
        S4<= pwm_int;
        S5<= '1';
        S6<= '0';

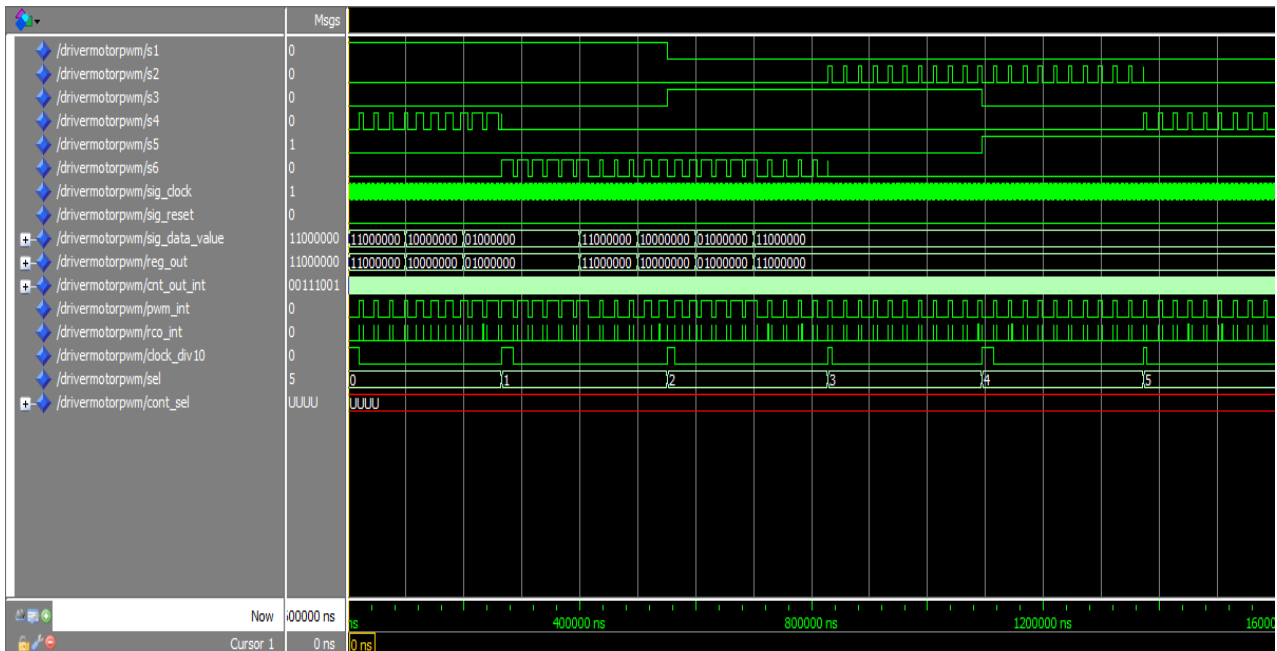
    WHEN OTHERS =>
        S1<= '0'; S2<= '0'; S3<= '0'; S4<= '0'; S5<= '0';
        S6<= '0';

    END CASE;
END IF;

END PROCESS;
END ARCHITECTURE arch_DriverMotorPWM;

```

RESULTADO DE LA SIMULACIÓN EN MODELSIM



CONCLUSIONES FINALES

El trabajo práctico nos ha mostrado que hay un mundo detrás de cada parte del análisis de un robot. Y esto no se aplica solamente a robots de 6 grados de libertad, sino a todos.

Tratamos de enfocarnos en las partes más importantes del análisis, y de ahondar más en las que podrían ser más útiles en el futuro de este análisis. Esto nos da a entender que este es un inicio, un puntapié para un desarrollo mejor y más completo.

Si bien pensamos que sería más sencillo, hubo varias partes que demoraron mucho más de lo planeado. Una de ellas fue el desarrollo del compilador. Sin conocer ninguno de los programas disponibles, en un principio, le apostamos a la herramienta nueva, ANTLR. Nos fue sencillo empezar a utilizarlo. Si bien parece ser más cómodo, nos fue complicado lograr hacer funcionar un ejemplo que nos sirva. Atribuimos gran parte de este destiempo a no encontrar ejemplos que no fuesen en java, lenguaje poco utilizado por nosotros.

Si bien la idea del diseño del compilador es la de entender que es un puente entre lenguajes y que una vez adquiridos estos conocimientos podríamos realizar cualquier tipo de compilador, no solo para robots, sino para Relés programables, PLC o microcontroladores, sabemos que el nuestro puede mejorarse mucho. Esta ha sido nuestra intención, pero los contratiempos nos jugaron en contra.

Otro punto en el que quisimos adentrarnos fue en la simulación, para ello íbamos a utilizar el software RoboWorks, en el cual habíamos diseñado un modelo del PUMA, y pensábamos integrarlo con la ayuda del RoboTalk con el código del compilador. Lamentablemente, tampoco pudimos adentrarnos demasiado en este punto.

En cuanto a la parte técnica, hemos mostrado que es posible el análisis de un robot de esta complejidad y porque no también, el diseño de alguno de sus componentes. Las herramientas informáticas a disposición de estas tareas son muchas y muy poderosas.