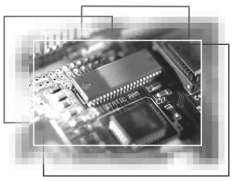


UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES
INGENIERIA ELECTRÓNICA

TRABAJO PRÁCTICO

#1

Cinemática

ROBÓTICA

INTEGRANTES:

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

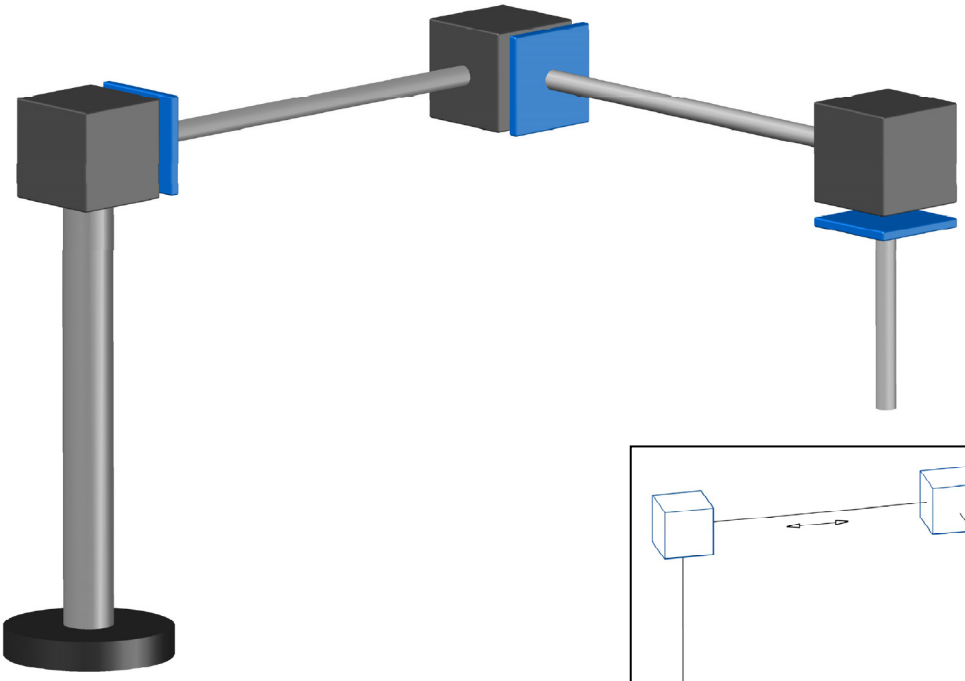
Profesor: Msc. Ing. **GIANNETTA,** Hernán
Jefe de TP: Ing. **GRANZELLA,** Damián
Fecha de Entrega: 12-05-2010
Curso: R6055
Año: 2010

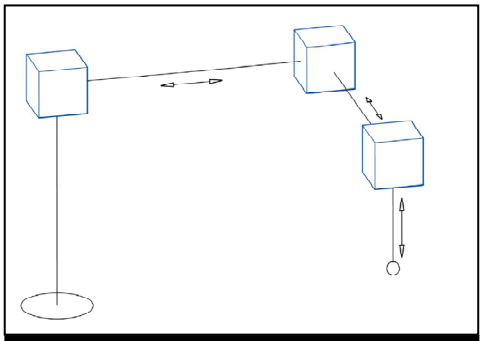
TRABAJO PRÁCTICO N°1
CINEMÁTICA

OBJETIVO

El objetivo del presente trabajo es estudiar las cadenas cinemáticas directa de un robot manipulador industrial de 3 grados de libertad, simular el movimiento mediante el software **Code Warrior** y comprobar los resultados obtenidos mediante el **Matlab**.

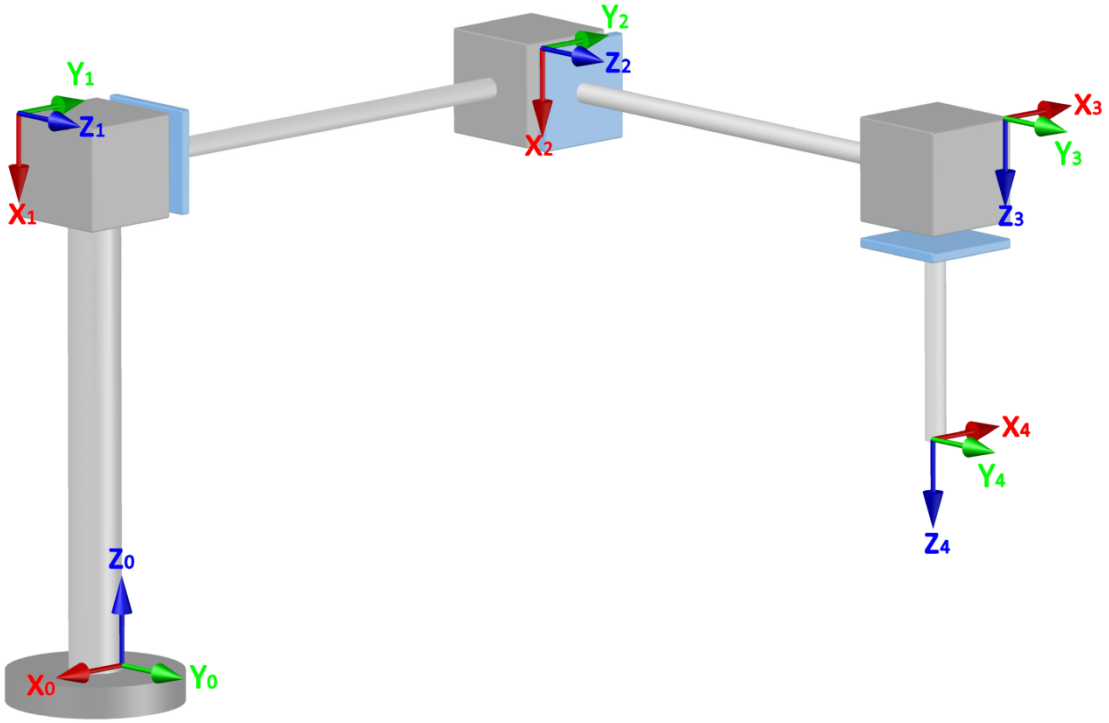
ESQUEMA DEL MANIPULADOR BAJO ESTUDIO





Detalle de Movimientos

Detalle de los sistemas de referencia



INTRODUCIÓN TEÓRICA

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 κ ι ν ε ω (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:

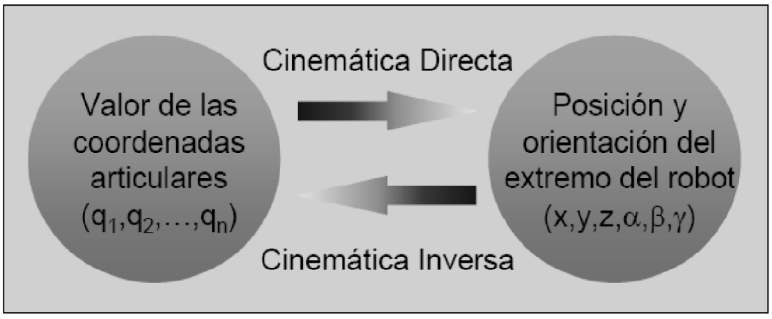
- Relación entre la localización del extremo del robot y los valores de sus articulaciones.
- Descripción analítica del movimiento espacial en función del tiempo

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 algebra 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 si 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:

$$x = f_x(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$y = f_y(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$z = f_z(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$\alpha = f_\alpha(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$\beta = f_\beta(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$\gamma = f_\gamma(q_1, q_2, q_3, q_4, q_5, q_6)$$

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 el 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 ${}^{i-1}Ai$.

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

Del mismo modo, denominando ${}^{0}A_k$ 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 0An 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:**

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

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

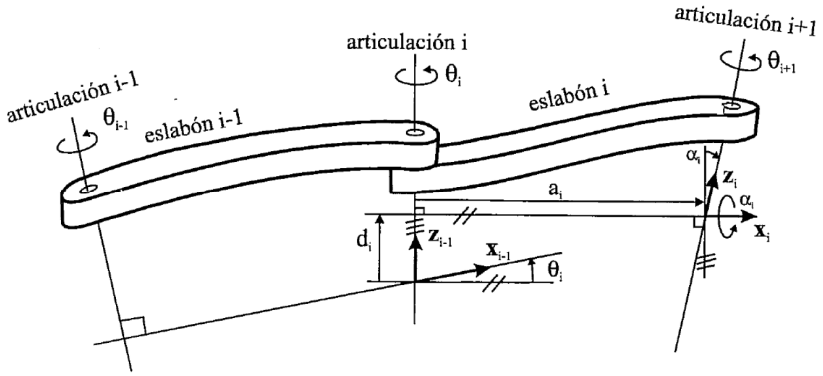
La forma habitual que se suele utilizar en robótica es la representación de *Denavit- Hartenberg (D-H).* Denavit y Hartenberg propusieron en 1955 un método matricial que permite establecer de manera sistemática un sistema de coordenadas { Si } ligado a cada eslabón i de una cadena articulada, pudiéndose 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 { Si } mediante estas 4 transformaciones está garantizado solo si los sistemas { S_{i-1} } al { Si } han sido definidos de acuerdo a unas normas determinadas que se expondrán posteriormente):

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



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

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

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

donde $a_i, d_i, \theta_i, \alpha_i$ son los parámetros D-H del eslabón i. De este modo, basta con identificar los parámetros $a_i, d_i, \theta_i, \alpha_i$ 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 { Si }, 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-Hartemberg 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 $\mathbf{z_i}$ 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 $\mathbf{z_0}$ .Los ejes $\mathbf{x_0}$ e $\mathbf{y_0}$ se situaran de modo que formen un sistema dextrógiro con $\mathbf{z_0}$.

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

D-H 7. Situar $\mathbf{x}_i$ en la línea normal común a $\mathbf{z}_{i-1}$ y $\mathbf{z}_i$.

D-H 8. Situar $\mathbf{y}_i$ de modo que forme un sistema dextrógiro con $\mathbf{x}_i$ y $\mathbf{z}_i$.

D-H 9. Situar el sistema { Sn } en el extremo del robot de modo que $\mathbf{z}_n$, coincida con la dirección de $\mathbf{z}_{n-1}$ y $\mathbf{x}_n$ sea normal a $\mathbf{z}_{n-1}$ y $\mathbf{z}_n$.

D-H 10. Obtener θ_i como el ángulo que hay que girar en torno a $\mathbf{z}_{i-1}$ para que $\mathbf{x}_{i-1}$ y $\mathbf{x}_i$, queden paralelos.

D-H 11. Obtener di, como la distancia, medida a lo largo de $\mathbf{z}_{i-1}$, que habría que desplazar { S_{i-1} } para que $\mathbf{x}_i$ y $\mathbf{x}_{i-1}$ quedasen alineados.

DH 12. Obtener a_i como la distancia medida a lo largo de $\mathbf{x}_i$ (que ahora coincidiría con $\mathbf{x}_{i-1}$) 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 $\mathbf{x}_i$, (que ahora coincidiría con $\mathbf{x}_{i-1}$),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 $\mathbf{T} = {}^{0}A_1, {}^{1}A_2 \ldots\ldots {}^{n-1}A_n$

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:

- $\boldsymbol{\theta_i}$ Es el angulo que forman los ejes $\mathbf{x}_{i-1}$ y $\mathbf{x}_i$, medido en un plano perpendicular al eje $\mathbf{z}_{i-1}$,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 $\mathbf{z}_{i-1}$ desde el origen del sistema de coordenadas (i−1) -ésimo hasta la intersección del eje $\mathbf{z}_{i-1}$ con el eje $\mathbf{x}_i$. Se trata de un parámetro variable en articulaciones prismáticas.

- a_i Es la distancia a lo largo del eje $\mathbf{x}_i$ que va desde la intersección del eje $\mathbf{z}_{i-1}$ con el eje $\mathbf{x}_i$ hasta el origen del sistema i-esimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas, se calcula como la distancia más corta entre los ejes $\mathbf{z}_{i-1}$ y $\mathbf{z}_i$.

- d_i Es el ángulo de separacion del eje $\mathbf{z}_{i-1}$ y el eje $\mathbf{z}_i$, medido en un plano perpendicular al eje $\mathbf{x}_i$, 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,....,n (n= numero 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.
- Calcularlas matrices ${}^{i-1}A_i$
- Calcular la matriz $\mathbf{T}_n = {}^{0}A_1 \, {}^{1}A_2 \ldots\ldots {}^{n-1}A_n$

TABLA DE MOVIMIENTOS (según el algoritmo de *Denavit-Hartenberg*)

	ROTACION EN Z	TRASLACION EN Z	TRASLACION EN X	ROTACION EN X
1	90	l1	0	-90
2	90	d1	0	90
3	90	d2	0	90
4	0	d3	0	0

Calculo de las matrices

Articulación #1

	ROTACION EN Z	TRASLACION EN Z	TRASLACION EN X	ROTACION EN X
1	90	l1	0	-90

$$r_z = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(90°) & -\sin(90°) & 0 & 0 \\ \sin(90°) & \cos(90°) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$r_x = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-90°) & -\sin(-90°) & 0 \\ 0 & \sin(-90°) & \cos(-90°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$t_z = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$${}^0A_1 = r_z . r_x t_z = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{-1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{-1} & \mathbf{0} & \boldsymbol{l_1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Articulación #2

	ROTACION EN Z	TRASLACION EN Z	TRASLACION EN X	ROTACION EN X
2	90	d1	0	90

$$r_z = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(90°) & -\sin(90°) & 0 & 0 \\ \sin(90°) & \cos(90°) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$r_x = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(90°) & -\sin(90°) & 0 \\ 0 & \sin(90°) & \cos(90°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$t_z = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$^{1}A_2 = r_z.r_x t_z = \begin{bmatrix}0 & -1 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & d_1\\ 0 & 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0}\\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \boldsymbol{d_1}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1}\end{bmatrix}$$

Articulación #3

	ROTACION EN Z	TRASLACION EN Z	TRASLACION EN X	ROTACION EN X
3	90	d2	0	90

$$r_z = \begin{bmatrix}x'\\ y'\\ z'\\ 1\end{bmatrix} = \begin{bmatrix}C\theta & -S\theta & 0 & 0\\ S\theta & C\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix} = \begin{bmatrix}\cos(90°) & -\sin(90°) & 0 & 0\\ \sin(90°) & \cos(90°) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix} = \begin{bmatrix}0 & -1 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix}$$

$$r_x = \begin{bmatrix}x'\\ y'\\ z'\\ 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & C\theta & -S\theta & 0\\ 0 & S\theta & C\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos(90°) & -\sin(90°) & 0\\ 0 & \sin(90°) & \cos(90°) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix}$$

$$t_z = \begin{bmatrix}x'\\ y'\\ z'\\ 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & t_x\\ 0 & 1 & 0 & t_y\\ 0 & 0 & 1 & t_z\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & d_2\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix}$$

$$^{2}A_3 = r_z.r_x t_z = \begin{bmatrix}0 & -1 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & d_2\\ 0 & 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0}\\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \boldsymbol{d_2}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1}\end{bmatrix}$$

Articulación #4

	ROTACION EN Z	TRASLACION EN Z	TRASLACION EN X	ROTACION EN X
3	0	D3	0	0

$$t_z = \begin{bmatrix}x'\\ y'\\ z'\\ 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & t_x\\ 0 & 1 & 0 & t_y\\ 0 & 0 & 1 & t_z\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & d_3\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix}$$

$$^{3}A_4 = t_z = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & d_3\\ 0 & 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0}\\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \boldsymbol{d_3}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1}\end{bmatrix}$$

Obtención de la Matriz Homogénea

$$T = {}^{0}A_1.\ {}^{1}A_2.\ {}^{2}A_3.\ {}^{3}A_4 = \begin{bmatrix}0 & 0 & -1 & 0\\ 1 & 0 & 0 & 0\\ 0 & -1 & 0 & l_1\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}0 & 0 & 1 & 0\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & d_1\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}0 & 0 & 1 & 0\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & d_2\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}0 & 0 & 1 & 0\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & d_3\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$\boldsymbol{T} = \begin{bmatrix}\mathbf{-1} & \mathbf{0} & \mathbf{0} & \boldsymbol{-d_1}\\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \boldsymbol{d_2}\\ \mathbf{0} & \mathbf{0} & \mathbf{-1} & \boldsymbol{-d_3 + l_1}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1}\end{bmatrix}.\begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{v}\\ \boldsymbol{w}\\ \mathbf{1}\end{bmatrix}$$

$$\boldsymbol{x = -d_1}$$
$$\boldsymbol{y = d_2}$$
$$\boldsymbol{x = -d_3 + l_1}$$

SIMULACION EN CODE WARRIOR

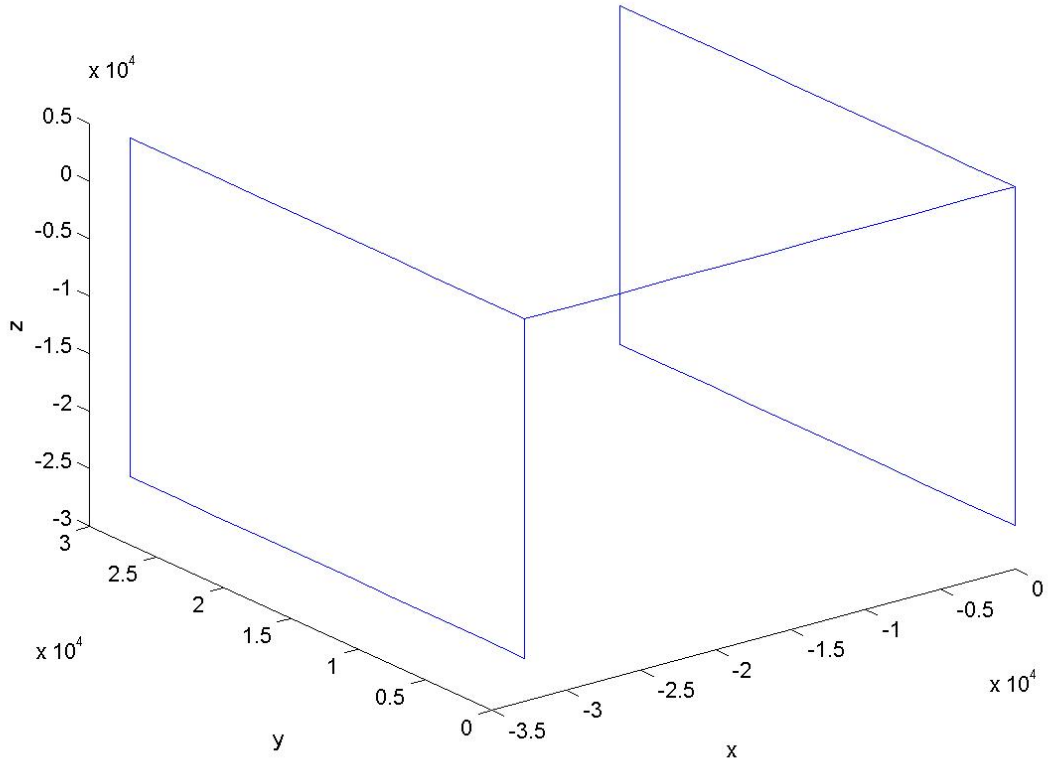
Código del DSP– Espacio de trabajo límite

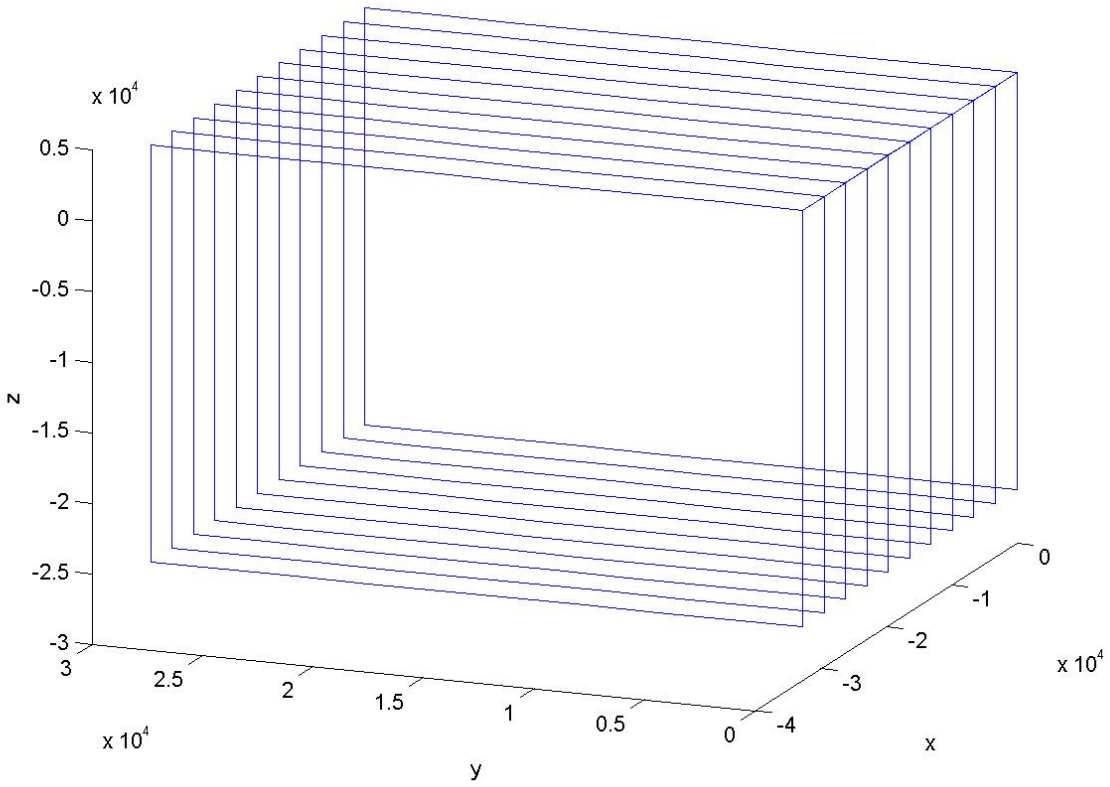
```
/** ###################################################################
**     Filename  : tp1.C
**     Project   : tp1
**     Processor : 56F8367
**     Version   : Driver 01.14
**     Compiler  : Metrowerks DSP C Compiler
**     Date/Time : 25/04/2010, 10:57 p.m.
**     Abstract  :
**         Main module.
**         This module contains user's application code.
**     Settings  :
**     Contents  :
**         No public methods
**
** ###################################################################*/
/* MODULE Tp1 */


/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "DFR1.h"
#include "MEM1.h"
#include "TFR1.h"
#include "MFR1.h"
#include "CCNT1.h"
#include "Inhr1.h"
/* Including shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

#include "stdio.h"

#define MXLONG 10
#define D1INICIAL 0
#define D2INICIAL 0
#define D3INICIAL 0
#define LARGOL1 1
#define PASONORMALIZADO 32767/(2*MXLONG)
Frac16 c,i;
Frac16 pasonormalizado=PASONORMALIZADO;


void main(void)
{
  /* Write your local variable definition here */


Word16 d1_16,d2_16,d3_16,d1,d2,d3,x_16,y_16,z_16,l_16;
Word32 d1_32,d2_32,d3_32,l_32;

l_32=L_mult(pasonormalizado,LARGOL1);
l_16=extract_l(l_32);


  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */


                  /* Calibrate the cycle counter                            */
                  //CCNT1_cycleCountCalibrate();

          for(;;)
          {
                  //c=0;


                  d1=D1INICIAL;
                  d2=D2INICIAL;
                  d3=D3INICIAL;

                  d1_32=(L_mult(pasonormalizado,d1));
                  d1_16=extract_l(d1_32);
                  x_16=-d1_16;

                  d2_32=(L_mult(pasonormalizado,d2));
                  d2_16=extract_l(d2_32);
                  y_16=d2_16;
```

```
//       Incremento en z hasta el maximo, incremento en y, decremento en z, decremento en y,
//       Conclusion, dibujo un cuadrado de min a max

                  for(d3=D3INICIAL;d3<MXLONG;d3++)
                  {
                           d3_32=(L_mult(pasonormalizado,d3));
                           d3_16=extract_l(d3_32);
                           z_16=add(-d3_16,l_16);
                           printf ("%d \t %d \t %d \n",x_16,y_16,z_16);
                  }
                  d3--;

                  for(d2=D2INICIAL;d2<MXLONG;d2++)
                  {
                           d2_32=(L_mult(pasonormalizado,d2));
                           d2_16=extract_l(d2_32);
                           y_16=d2_16;
                           printf ("%d \t %d \t %d \n",x_16,y_16,z_16);
                  }
                  d2--;

                  for(d3;d3>=D3INICIAL;d3--)
                  {
                           d3_32=(L_mult(pasonormalizado,d3));
                           d3_16=extract_l(d3_32);
                           z_16=add(-d3_16,l_16);
                           printf ("%d \t %d \t %d \n",x_16,y_16,z_16);
                  }

                  for(d2;d2>=D2INICIAL;--d2)
                  {
                           d2_32=(L_mult(pasonormalizado,d2));
                           d2_16=extract_l(d2_32);
                           y_16=d2_16;
                           printf ("%d \t %d \t %d \n",x_16,y_16,z_16);
                  }

//Incremento en x hasta el max

                  for(d1=D1INICIAL;d1<=MXLONG;++d1)
                  {
                           d1_32=(L_mult(pasonormalizado,d1));
                           d1_16=extract_l(d1_32);
                           x_16=-d1_16;
                           printf ("%d \t %d \t %d \n",x_16,y_16,z_16);
                  }

//Repito el cuadrado de min a max

                  for(d3=D3INICIAL;d3<MXLONG;d3++)
                  {
                           d3_32=(L_mult(pasonormalizado,d3));
                           d3_16=extract_l(d3_32);
                           z_16=add(-d3_16,l_16);
                           printf ("%d \t %d \t %d \n",x_16,y_16,z_16);
                  }
                  d3--;

                  for(d2=D2INICIAL;d2<MXLONG;d2++)
                  {
                           d2_32=(L_mult(pasonormalizado,d2));
                           d2_16=extract_l(d2_32);
                           y_16=d2_16;
                           printf ("%d \t %d \t %d \n",x_16,y_16,z_16);
                  }
                  d2--;

                  for(d3;d3>=D3INICIAL;d3--)
                  {
                           d3_32=(L_mult(pasonormalizado,d3));
                           d3_16=extract_l(d3_32);
                           z_16=add(-d3_16,l_16);
                           printf ("%d \t %d \t %d \n",x_16,y_16,z_16);
                  }

                  for(d2;d2>=D2INICIAL;--d2)
                  {
                           d2_32=(L_mult(pasonormalizado,d2));
                           d2_16=extract_l(d2_32);
                           y_16=d2_16;
                           printf ("%d \t %d \t %d \n",x_16,y_16,z_16);
                  }

          }
}

/* END Tp1 */
```

RESULTADO DE LA SIMULACION

Espacio de trabajo límite



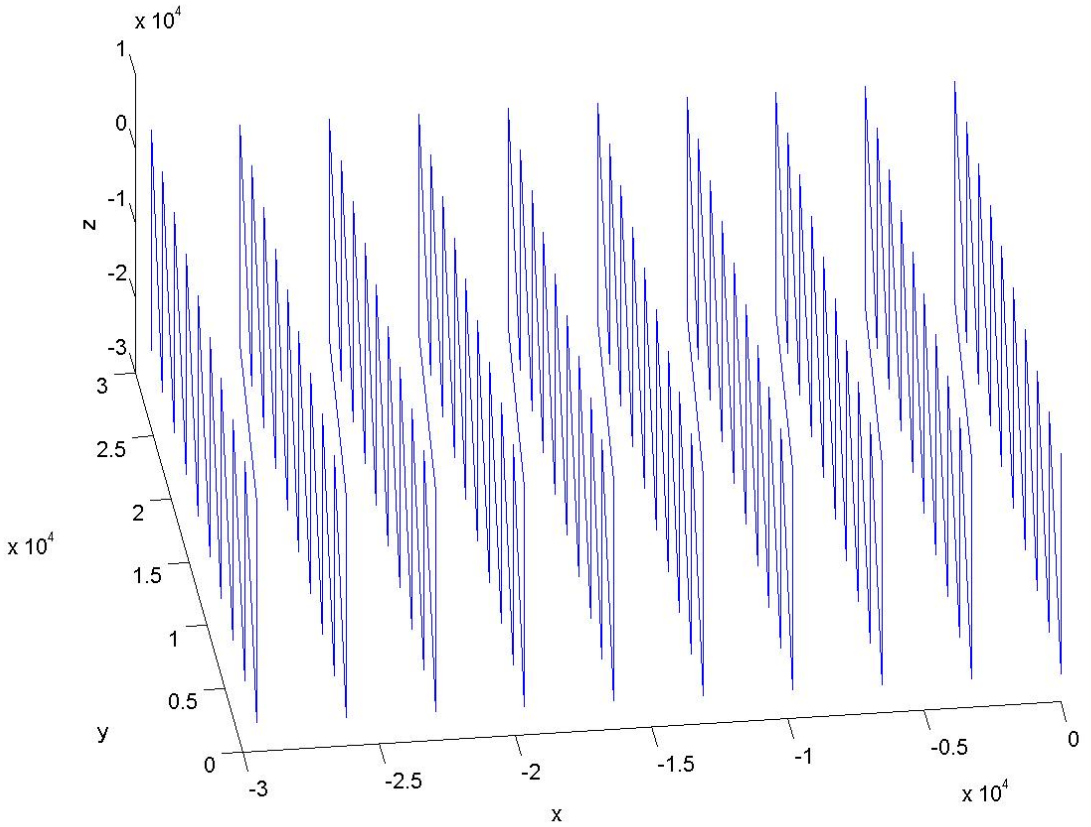


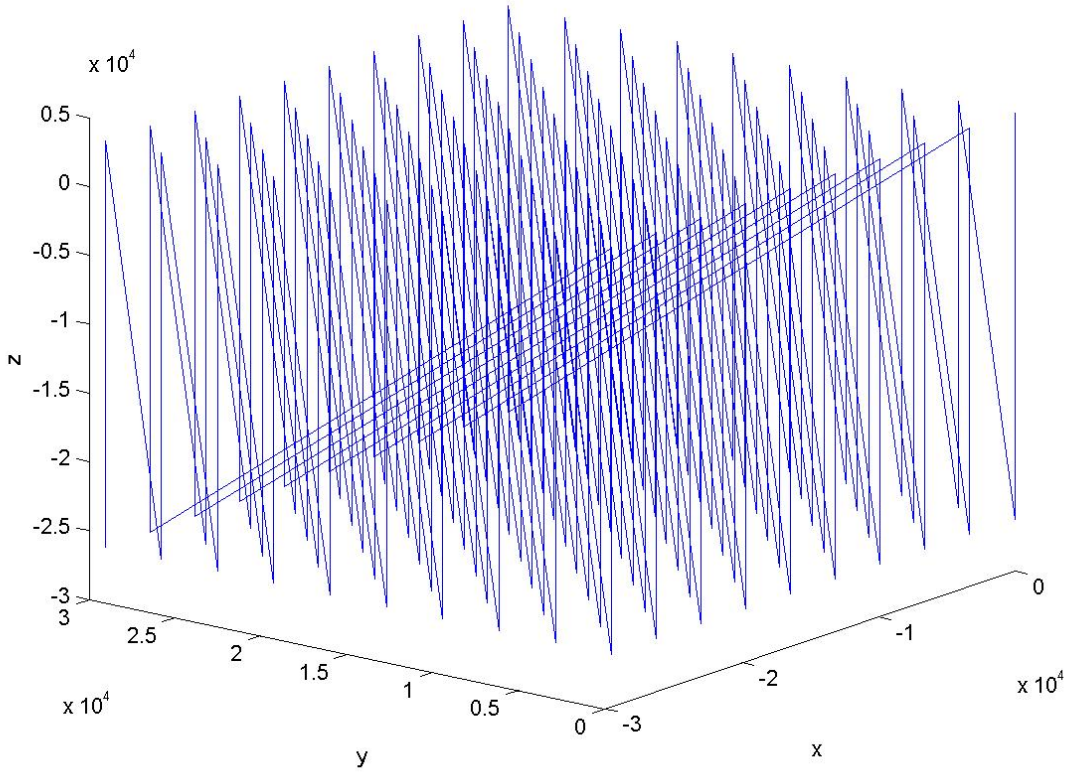
Código del DSP - SIMULACIÓN DE UNA TRAYECTORIA LINEAL CONTINUA PARA CADA EJE

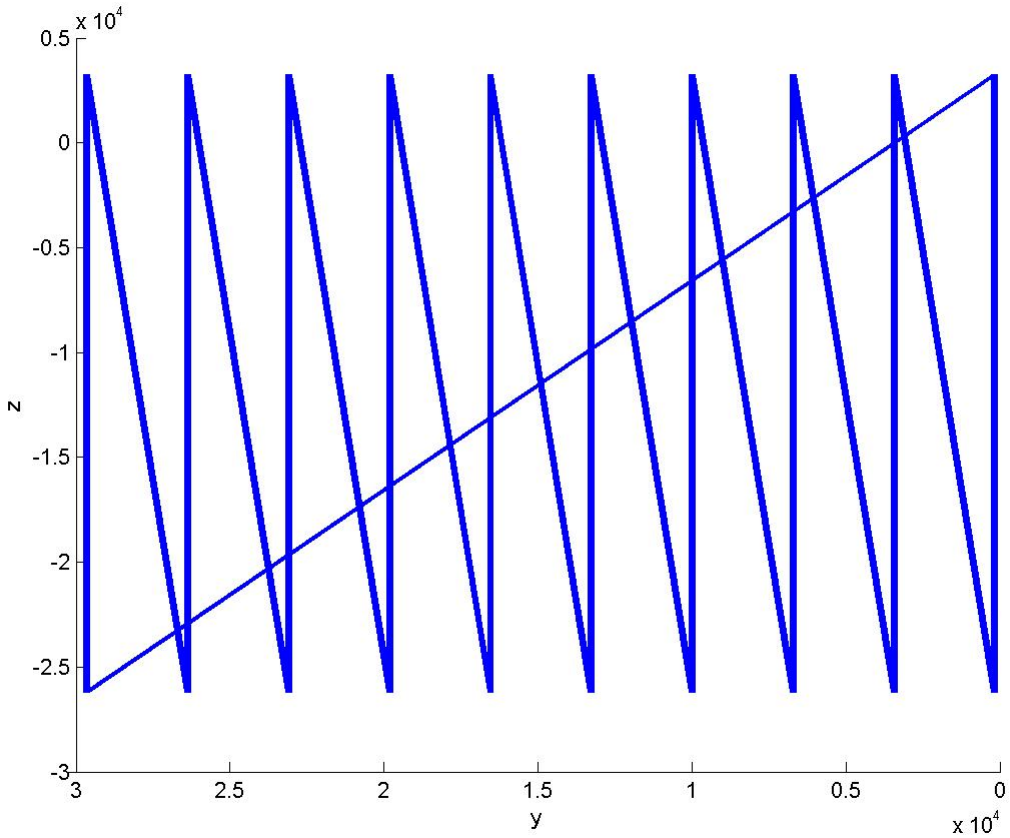
```
/** ###################################################################
**     Filename  : tp1.C
**     Project   : tp1
**     Processor : 56F8367
**     Version   : Driver 01.14
**     Compiler  : Metrowerks DSP C Compiler
**     Date/Time : 25/04/2010, 10:57 p.m.
**     Abstract  :
**         Main module.
**         This module contains user's application code.
**     Settings  :
**     Contents  :
**         No public methods
** ###################################################################*/
/* MODULE tp1 */

/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "DFR1.h"
#include "MEM1.h"
#include "TFR1.h"
#include "MFR1.h"
#include "CCNT1.h"
#include "Inhr1.h"
/* Including shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

#include "stdio.h"

#define MXLONG 2
#define D1INICIAL 0
#define D2INICIAL 0
#define D3INICIAL 0
#define LARGOL1 1
#define PASONORMALIZADO 32767/(2*MXLONG)
Frac16 c,i;
Frac16 pasonormalizado=PASONORMALIZADO;
void main(void)
{
  /* Write your local variable definition here */

Word16 d1_16,d2_16,d3_16,d1,d2,d3,x_16,y_16,z_16,l_16;
Word32 d1_32,d2_32,d3_32,l_32;

l_32=L_mult(pasonormalizado,LARGOL1);
l_16=extract_l(l_32);

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */

        for(;;)
        {
        for(d1=D1INICIAL;d1<MXLONG;d1++)
                {
                        d1_32=(L_mult(pasonormalizado,d1));
                        d1_16=extract_l(d1_32);
                        x_16=-d1_16;

                        for(d2=D2INICIAL;d2<MXLONG;d2++)
                        {
                                d2_32=(L_mult(pasonormalizado,d2));
                                d2_16=extract_l(d2_32);
                                y_16=d2_16;

                                for(d3=D3INICIAL;d3<MXLONG;d3++)
                                {
                                        d3_32=(L_mult(pasonormalizado,d3));
                                        d3_16=extract_l(d3_32);
                                        z_16=add(-d3_16,l_16);

                                        printf ("%d \t %d \t %d \n",x_16,y_16,z_16);

                                }
                        }
                }
         }
}

/* END tp1 */
```

RESULTADO DE LA SIMULACION

Simulación de una trayectoria lineal continua para cada eje







CONCLUSIONES FINALES

Para la grafica del espacio de trabajo limite, se realizo un código en el CodeWarrior, para recorrer un cuadrado máximo sobre el plano x=0, luego nos movimos en x hasta su límite y volvimos a repetir la operación del cuadrado, dando como resultado un cubo delimitador del volumen máximo de trabajo.

Podemos ver en las gráficas el recorrido del extremo del brazo si simulamos el desplazamiento de los motores. Para el gráfico último, utilizamos tres lazos "for" anidados, que incrementaban la variable correspondiente a d3 ('z') al máximo, luego se incrementaba en un paso la variable correspondiente a d2 ('y') y para esta se incrementaba la variable correspondiente a d3 nuevamente hasta el máximo. Siguiendo esta técnica se “barren” todos las posibles posiciones de los motores (de movimiento lineal en nuestro caso), lo que en conjunto con la solución a la matriz homogénea, nos da los valores de la posición ('x', 'y', 'z') del extremo del brazo. Esto fue implementado en el CodeWarrior, para simular el comportamiento de un DSP. Paso siguiente, se tomaron los valores arrojados por el mismo y se importaron en Matlab, donde con el comando plot3 se graficaron.

En nuestro ejercicio, no existen zonas prohibidas. Es decir, cualquiera sea los desplazamientos de los motores, habrá una posición final del extremo del robot, que será alcanzada sin inconvenientes.
Además, si se tiene en cuenta que los desplazamientos del motor son siempre positivos, puede verse, tanto en la matriz como en el gráfico, que las variaciones de 'x' y 'z' son negativas. Nuevamente, esto se debe a como nosotros adoptamos los ejes coordenados y los desplazamientos para cada eslabón.
También puede observarse que al tenerse en cuenta la altura L1, desde el apoyo del robot sobre la mesa al comienzo del primer eslabón, este se refleja en el resultado de la simulación, como un "levantamiento" del espacio de trabajo en esta misma altura, en el eje correspondiente ('z'). Suele ocurrir que, al realizar todo el procedimiento de cálculo de la matriz homogénea, se pierde noción de cada uno de sus partes, pero al graficar los resultados, estos pueden verse simplemente.

Otra singularidad en nuestra simulación es la forma del gráfico, donde se observan triángulos "puntiagudos". Esto es así ya que el Matlab une con una línea el paso de una coordenada a la siguiente. Además, es tan notorio, debido a la escasa cantidad de pasos elegidos en la simulación. Si esta cantidad fuese mucho mayor, la simulación tardaría demasiado tiempo en arrojar resultados, y se vería una "mancha" con forma de cubo. Esto obviamente no ocurre en el DSP, donde esta tarea demora fracciones de segundo.
A modo de velocidad en la obtención de resultados y buena visualización de los mismos se eligió este valor de cantidad pasos.