Trabajo Práctico Nº1: Cinemática del Robot

Materia: Robótica

Integrantes:

Febles, Matías
Titolo, Hernán
Gasulla, Juan Manuel

Profesor: Mas. Ing. Giannetta Hernan

JTP: Ing. Granzella Damian

Año: 2012

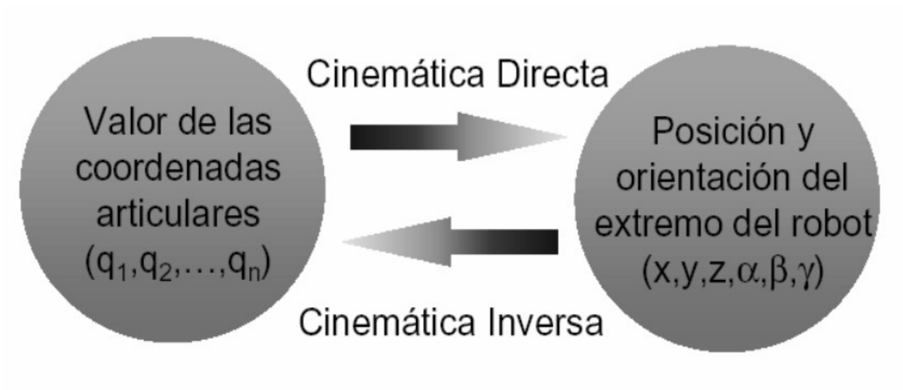
Introducción Teórica

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia. La cinemática se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación de la herramienta del robot con los valores que toman sus coordenadas de sus articulaciones.

Existen dos problemas fundamentales a resolver con respecto a la cinemática del robot:

A) Cinemática Directa. Consiste en determinar la posición y orientación del extremo final del robot con respecto al sistema de la base del robot a partir de conocer los valores de las articulaciones y los parámetros geométricos.

B) Cinemática Inversa. Resuelve la configuración que debe adoptar el robot para una posición y orientación conocidas del extremo.

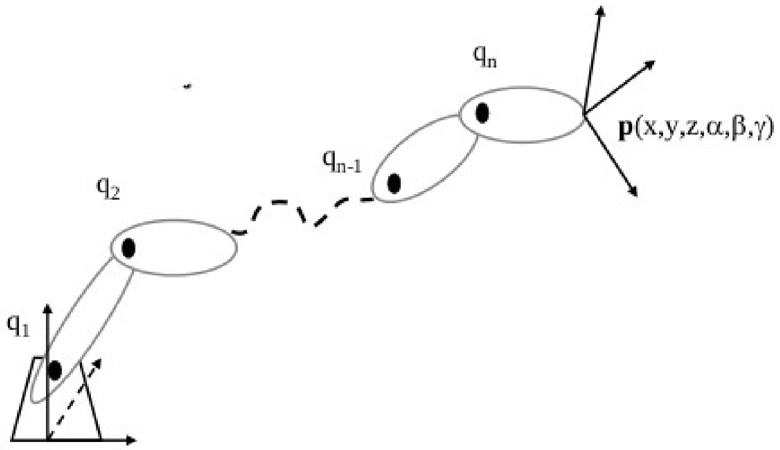


El desarrollo del presente TP se basará en resolver el problema de Cinemática Directa de nuestro manipulador.

El objetivo de la cinemática directa es encontrar una matriz de transformación homogénea **T** que relacione posición y orientación del extremo del robot con respecto a un sistema de referencia fijo y situado, por ejemplo en la base.

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

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 referencias solidario a él y, utilizando las transformaciones homogéneas, es posible representar las rotaciones y traslaciones relativas entre los distintos eslabones que componen al robot.

La matriz que representa la posición y orientación relativa entre los sistemas asociados a dos eslabones consecutivos del robot suele denominarse matriz $^{i-1}\mathbf{A}_i$.

Cuando se consideran todos los grados de libertad mediante cada una de las matrices, la matriz resultante se la denomina matriz de transformación **T**.

Algoritmo de Denavit-Hartenberg

En 1955 **Denavit y Hartenberg** propusieron un método matricial que permite establecer de manera sistemática un sistema de coordenadas. La representación de **Denavit-Hartenberg (D-H)** establece que seleccionándose 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.

Reduciéndose al siguiente patrón de transformaciones que permiten relacionar el sistema de referencia del elemento i con respecto al sistema del elemento i-1:

1 – Rotación alrededor del eje zi-1 un ángulo θi

2 – Traslación a lo largo de Zi-1 una distancia di ; vector di (0,0,di)

3 - Traslación a lo largo de Xi-1 una distancia ai ; vector di (0,0,ai)

4 – Rotación alrededor del eje Xi-1 un ángulo αi

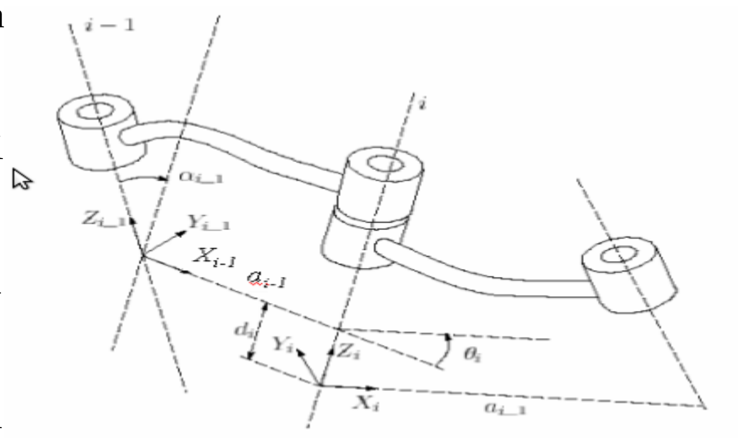
Donde θi - αi – ai – di son los parámetros D-H del eslabón.

- **θi:** Es el ángulo de xi-1 a xi medida sobre zi (utilizando la regla de la mano derecha).

- **di:** Es la distancia de xi-1 a xi medida a lo largo de zi

- **ai:** Es la distancia de zi a zi+1 medida a lo largo de xi

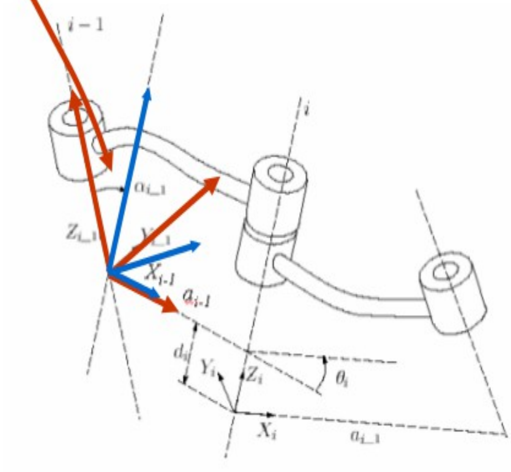
- **αi:** Es el ángulo de zi a zi+1 medida sobre xi (utilizando la regla de la mano derecha).



$$^{i-1}A_i = T(x,\alpha_{i-1})T(a_{i-1},0,0)T(z,\theta_i)T(0,0,d_i)$$

$$^{i-1}A_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ c\alpha_{i-1}s\theta_i & c\alpha_{i-1}c\theta_i & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ s\alpha_{i-1}s\theta_i & s\alpha_{i-1}c\theta_i & c\alpha_{i-1} & d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Para que la matriz $^{i-1}\mathbf{A}_i$ relacione los sistemas coordenados 0_i y O_{i-1} es necesario que los sistemas coordenados se determinen mediante los siguientes pasos:

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 zi sobre el eje de la articulación i + 1.

D-H 5. Situar el origen del sistema de la base {S0} en cualquier punto del eje z0 .Los ejes
x0 e y0 se situaran de modo que formen un sistema dextrógiro con z0.

D-H 6. Para i de 1 a n-1, situar el sistema { Si }(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 {Si}
en el punto de corte. Si fuesen paralelos { Si } 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 { Sn } 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 { Si-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 { Si-1 } para que su origen coincidiese con { Si }.

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 { Si-1 } coincidiese totalmente con { Si }.

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

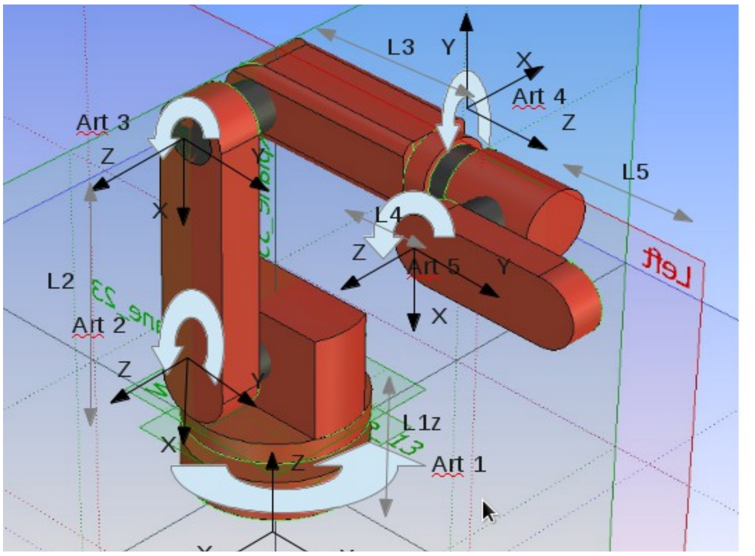
DH 15. Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot $T = {}^0A_1, {}^1A_2 \ldots {}^{n-1}A_n$

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.

Modelo Cinemático de nuestro Robot:

El manipulador que analizaremos es el que tenemos debajo.
Como primer medida debemos hallar la matriz de transformación T de toda la cadena cinemática, aplicando el método matricial de Denavit Hartenberg (D.H.)

Por lo tanto como primera medida lo que haremos es esquematizar el robot indicando las articulaciones y los sistemas de referencia convenientes.

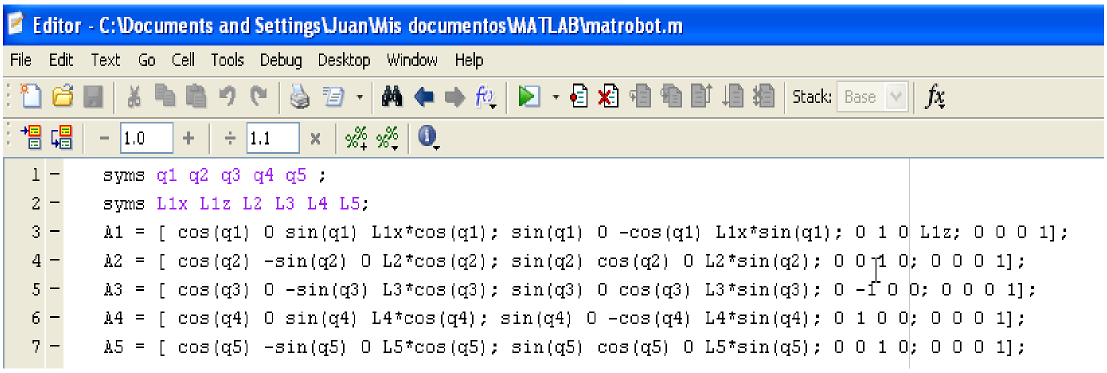


Entonces ahora procedemos a determinar los parámetros θ, d, a, α

	θ	d	a	α
Articulación 1	q1	L1z	0	90
Articulación 2	q2	0	L2	0
Articulación 3	q3	0	L3	90
Articulación 4	q4	0	L4	-90
Articulación 5	q5	0	L5	0

Con esta información procedimos a realizar un script en matlab para poder determinar la matriz de transformación homogénea T.

En primera medida definimos las matrices de transformación entre articulación y articulación.



Luego las multiplico para hallar la matriz de transformación homogénea T

$[T]=[{}^{0}A_{5}]=[{}^{0}A_{1}]*[{}^{1}A_{2}]*[{}^{2}A_{3}]*[{}^{3}A_{4}]*[{}^{4}A_{5}]$

Obtenida la Matriz T realizamos la siguiente operación:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [T] * \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}$$

Y hacemos u=0, v=0, w=0 (en el origen) tendremos las coordenadas (x, y, z) en función de las
coordenadas articulares q1, q2, q3, q4 y las dimensiones L1x, L2, L3, L4 L5 de nuestro robot.

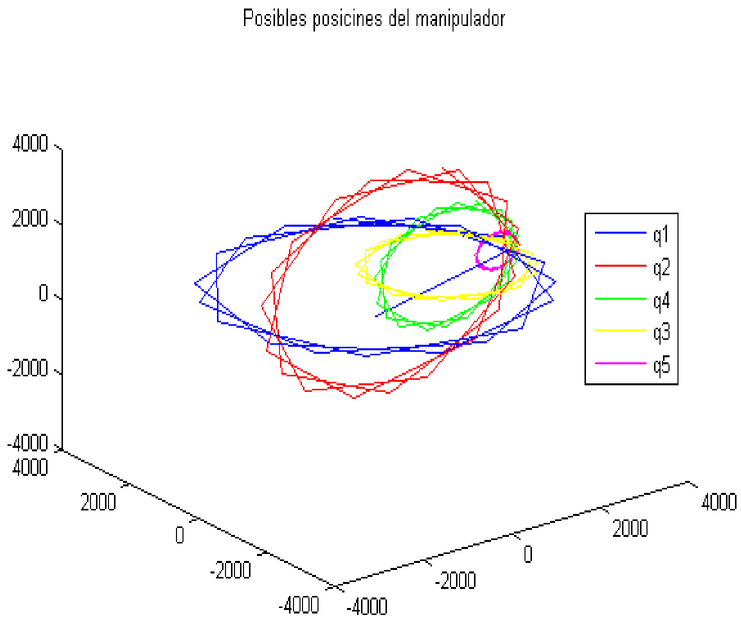
Obteniendo en matlab los siguientes resultados:

x =
cos(q1)*(L1x + L3*cos(q2 + q3) + L2*cos(q2)) - L4*sin(q1)*sin(q4) - L5*cos(q5)*(sin(q1)*sin(q4) - cos(q2 + q3)*cos(q1)*cos(q4)) - L5*sin(q2 + q3)*cos(q1)*sin(q5) + L4*cos(q2 + q3)*cos(q1)*cos(q4)

y = sin(q1)*(L1x + L3*cos(q2 + q3) + L2*cos(q2)) + L4*cos(q1)*sin(q4) + L5*cos(q5)*(cos(q1)*sin(q4) + cos(q2 + q3)*cos(q4)*sin(q1)) + L4*cos(q2 + q3)*cos(q4)*sin(q1) - L5*sin(q2 + q3)*sin(q1)*sin(q5)

z = L1z + L3*sin(q2 + q3) + L2*sin(q2) + L4*sin(q2 + q3)*cos(q4) + L5*cos(q2 + q3)*sin(q5) + L5*sin(q2 + q3)*cos(q4)*cos(q5)

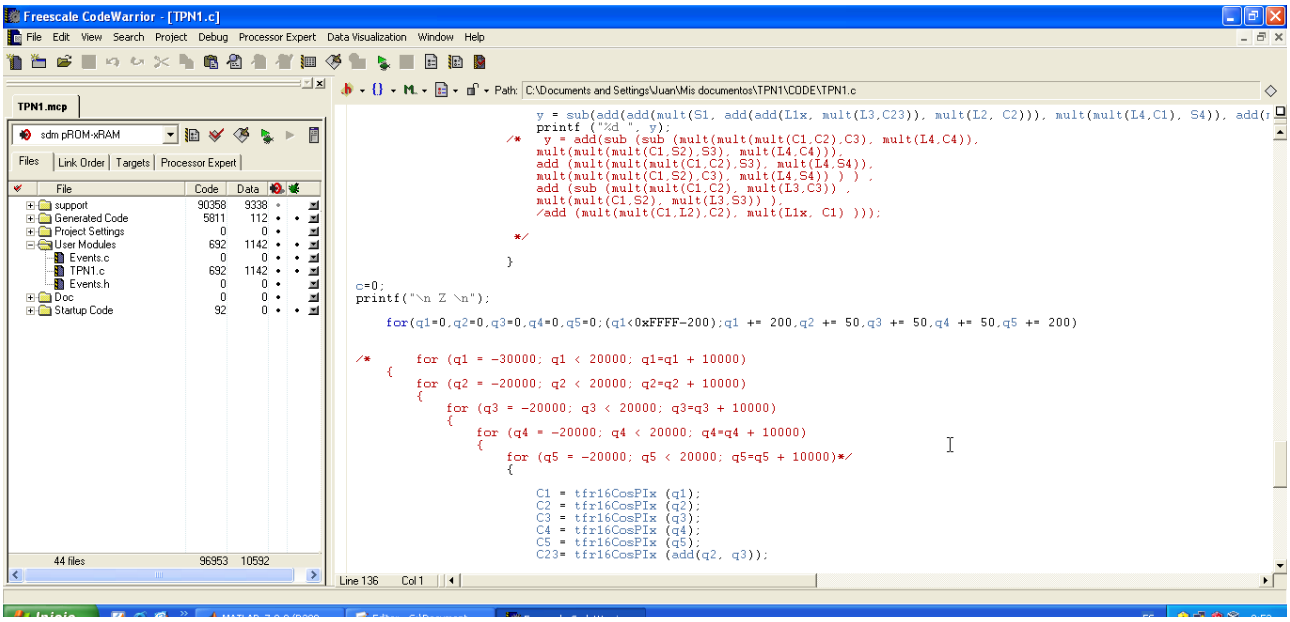
A partir de estos resultados realicé en **matlab** con la función **plot3d** una gráfica que muestra los distintos valores x, y, z evaluando solo q1, q2, q3, q4 y q5 por separado.



Implementación del modelo cinematico directo del Robot M5 a través del CodeWarrior.

Obtenido el modelo cinemático anterior, ahora procederemos a implementarlo en el elemento que luego controlará a nuestro robot. Como observamos, la tarea de controlar un manipulador requiere de un gran procesamiento matemático y además que ese procesamiento sea muy veloz, es por ello que generalmente se utilizan DSP para tal tarea.
En nuestro ejemplo utilizamos el procesador **56F8367** de la familia 56800E de **Freescale**.
Para programarlo utilizamos el entorno de desarrollo que otorga el fabricante que es el **CodeWarrior.**

A continuación se muestra una imagen de cómo se ve el entorno de desarrollo del **CodeWarrior.**
Debajo de la imagen se presenta el código en C que implementamos en el software.




```
/** ###################################################################
**     Filename  : TPN1.C
**     Project   : TPN1
**     Processor : 56F8367
**     Version   : Driver 01.12
**     Compiler  : Metrowerks DSP C Compiler
**     Date/Time : 10/05/2009, 19.19
**     Abstract  :
**         Main module.
**         Here is to be placed user's code.
**     Settings  :
**     Contents  :
**         No public methods
**
**     (c) Copyright UNIS, spol. s r.o. 1997-2006
**     UNIS, spol. s r.o.
**     Jundrovska 33
**     624 00 Brno
**     Czech Republic
**     http      : www.processorexpert.com
**     mail      : info@processorexpert.com
** ###################################################################*/
/* MODULE TPN1 */
```

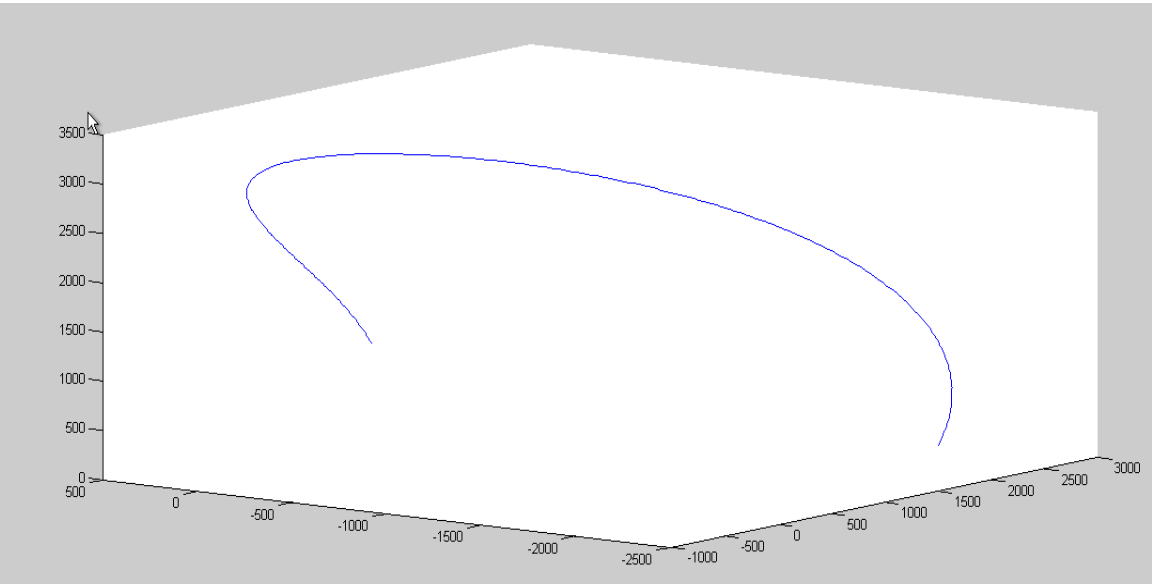
```
/* Including used modules for compiling procedure */

#include "Cpu.h"
#include "Events.h"

#include "TFR1.h"

#include "MFR1.h"

#include "MEM1.h"

/* Include 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

//*******************************************************************


#define MXRAD 361 //100


#define PULSE2RAD 32767/MXRAD  // 32767/100 impulsos // #define PULSE2RAD 450

#define a1 10

#define a2 4

Word16 c16[MXRAD];

Word32 c32[MXRAD];


Frac16 Homogenea[4][4];


Frac16 C1, S1, C2, S2, C3, S3, C4, S4, C5, S5, C6, S6, C23, S23, c, i, q1, q2, q3, q4, q5, L1x, L1z, L2, L3, L4, L5;

Frac16 x, y, z;

int j,k;


void main(void)

{

  for(;;)

  {

                    c=0;
                    q1 = 0;
                    q2 = 0;
                    q3 = 0;
```

```
                q4 = 0;
                q5 = 0;
                L1x = 320;
                L1z = 780;
                L2 = 1280;
                L3 = 0;
                L4 = 1142;
                L5 = 500;


printf("\n X \n");



for(q1=0,q2=0,q3=0,q4=0,q5=0;(q1<0xFFFF-200);q1 += 200,q2 += 50, q3 += 50,q4 += 50,q5 += 200)

        {

        C1 = tfr16CosPIx (q1);
        C2 = tfr16CosPIx (q2);
        C3 = tfr16CosPIx (q3);
        C4 = tfr16CosPIx (q4);
        C5 = tfr16CosPIx (q5);
        C23= tfr16CosPIx (add(q2, q3));
        S1 = tfr16SinPIx (q1);
        S2 = tfr16SinPIx (q2);
        S3 = tfr16SinPIx (q3);
        S4 = tfr16SinPIx (q4);
        S5 = tfr16SinPIx (q5);
        S23 = tfr16SinPIx (add(q2, q3));


        x=sub(add(mult(C1,add(add(L1x,mult(L3,C23)),mult(L2,C2))),mult(mult(L4,C23),mult(C1,C4))),
        add(add(mult(mult(L4,S1), S4), mult(L5, mult(C5, sub(mult(S1, S4), mult(mult(C23, C1), C4))))), mult(mult(L5,
        S23), mult(C1, S5))));
        printf ("%d ", x);


        }

c=0;
q1 = 0;
q2 = 0;
q3 = 0;
q4 = 0;
q5 = 0;

L1x = 320;
L1z = 780;
L2 = 1280;
L3 = 1142;
L4 = 200;
L5 = 0;
printf("\n Y \n");

for(q1=0,q2=0,q3=0,q4=0,q5=0;(q1<0xFFFF-200);q1 += 200,q2 += 50,q3 += 50,q4 += 50,q5 += 200)

        {

        C1 = tfr16CosPIx (q1);
        C2 = tfr16CosPIx (q2);
        C3 = tfr16CosPIx (q3);
        C4 = tfr16CosPIx (q4);
        C5 = tfr16CosPIx (q5);
        C23= tfr16CosPIx (add(q2, q3));

        S1 = tfr16SinPIx (q1);
        S2 = tfr16SinPIx (q2);
        S3 = tfr16SinPIx (q3);
        S4 = tfr16SinPIx (q4);
        S5 = tfr16SinPIx (q5);
        S23 = tfr16SinPIx (add(q2, q3));

        y = sub(add(add(mult(S1, add(add(L1x, mult(L3,C23)), mult(L2, C2))), mult(mult(L4,C1), S4)), add(mult(L5,
        mult(C5, add(mult(C1, S4), mult(mult(C23, C4), S4)))), mult(mult(L4, C23), mult(C4, S1)))), mult(mult(L5, S23),
```

```
        mult(S1, S4)));
        printf ("%d ", y);

        }
c=0;
printf("\n Z \n");

for(q1=0,q2=0,q3=0,q4=0,q5=0;(q1<0xFFFF-200);q1 += 200,q2 += 50,q3 += 50,q4 += 50,q5 += 200)
        {

        C1 = tfr16CosPIx (q1);
        C2 = tfr16CosPIx (q2);
        C3 = tfr16CosPIx (q3);
        C4 = tfr16CosPIx (q4);
        C5 = tfr16CosPIx (q5);
        C23= tfr16CosPIx (add(q2, q3));
        S1 = tfr16SinPIx (q1);
        S2 = tfr16SinPIx (q2);
        S3 = tfr16SinPIx (q3);
        S4 = tfr16SinPIx (q4);
        S5 = tfr16SinPIx (q5);
        S23 = tfr16SinPIx (add(q2, q3));

        z = add(add (add (L1z, mult(L3,S23)), mult(L2,S2)), add(add(mult(mult(L4, S23), C4),
        mult(mult(L5, C23), S5)),mult(mult(L5, S23), mult(C4, C5))));
        printf ("%d ", z);

        }
 }

}
```

Como podemos observar en el código utilizamos la función printf, que gracias a las herramientas de DEBUG del software podemos ver la trayectoria que realiza nuestro robot según se modifican los valores de q1, q2, q3, q4 y q5.

A continuación le pasamos esos parámetros al Matlab que con la función de plot3d los grafica.



Simulación en Matlab a través del Robotics Toolbox

Para llevar a cabo la siguiente simulación es necesario disponer de la herramienta llamada Robotics Toolbox for Matlab (release 8) que se debe agregar a nuestro programa Matlab; este paquete puede descargarse de la página www.petercorke.com.

A continuación se muestra el script para la simulación con el toolbox.
Para tal fin se muestra nuevamente la tabla que habíamos con realizado gracias al algoritmo de Denavit Hartenberg.

	θ	d	a	α
Articulación 1	q1	L1x	0	90
Articulación 2	q2	0	L2	0
Articulación 3	q3	0	L3	90
Articulación 4	q4	0	L4	-90
Articulación 5	q5	0	L5	0

```
art1 = link([90, L1x, 0, L1z]);
art2 = link([0, L2, 0, 0]);
art3 = link([-90, L3, 0, 0]);
art4 = link([90, L4, 0, 0]);
art5 = link([0, L5, 0, 0]);


mi_robot = robot({art1, art2, art3, art4, art5});

subplot(1,2,2)
plot (mi_robot,[0 0 0 0 0])
drivebot(mi_robot);

```

Se obtiene la siguientes simulaciones:

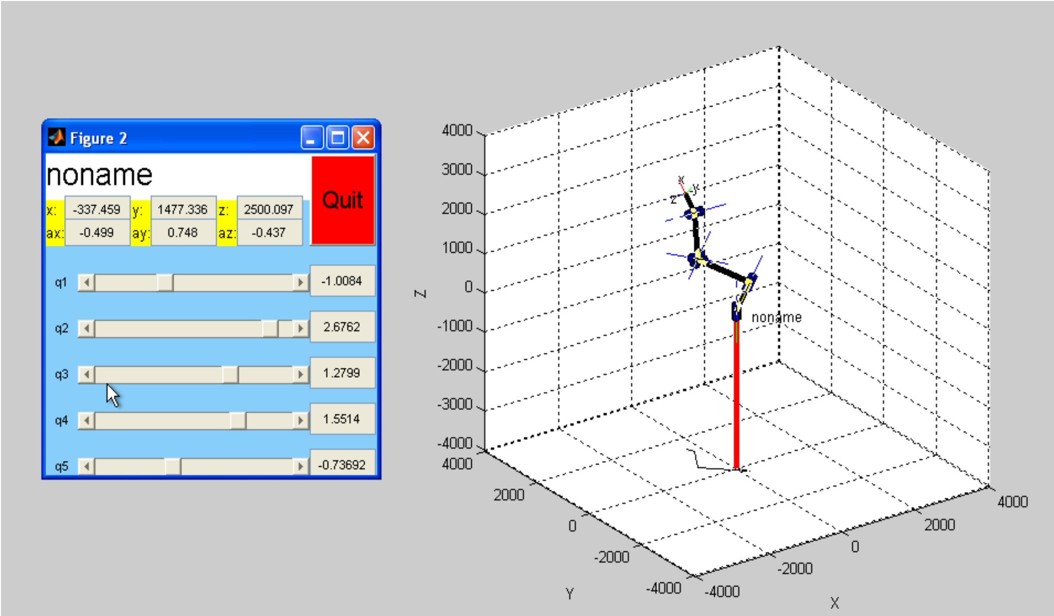
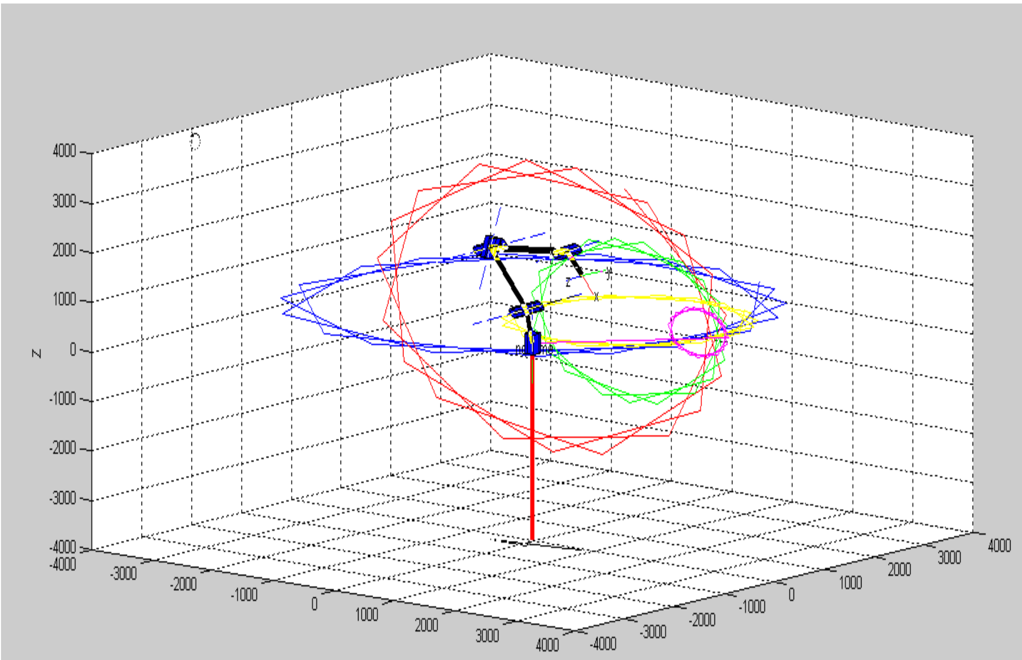


Figure 2
noname
Quit
x: -337.459
y: 1477.336
z: 2500.097
ax: -0.499
ay: 0.748
az: -0.437
q1 -1.0084
q2 2.6762
q3 1.2799
q4 1.5514
q5 -0.73692
noname
Z
Y
X


Z