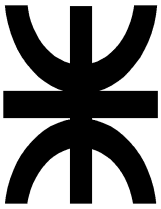


**Universidad Tecnológica Nacional**

**Facultad Regional Buenos Aires**

| | |
|---|---|
| **Materia**: | **Robótica** |
| **Curso:** | R6055 |
| **Profesor**: | M.aS. Ing. Hernan Giannetta |
| **JTP/ Ayudantes**: | Ing. Damian Granzella |

Trabajo Práctico: 1 Fecha realización: 1/07/2011

Título: **Implementación de una matriz cinemática en DSP**


Alumnos:

Ramos, Nahuel Leg: 119175-5

Silber, Fabio Leg: 119211-5


Grupo 2

Fecha entrega: 08/07/2011

Observaciones:

# Objetivos:

Obtener el modelo cinemático de un robot de 4 grados de libertades mediante la obtención de la matriz homogénea y con el uso de un DSP 56800-E simular el movimiento del robot, la cual se verificará con el uso del Matlab.

# Robot a modelar y simular:

# Introducción teórica de la cinemática directa:

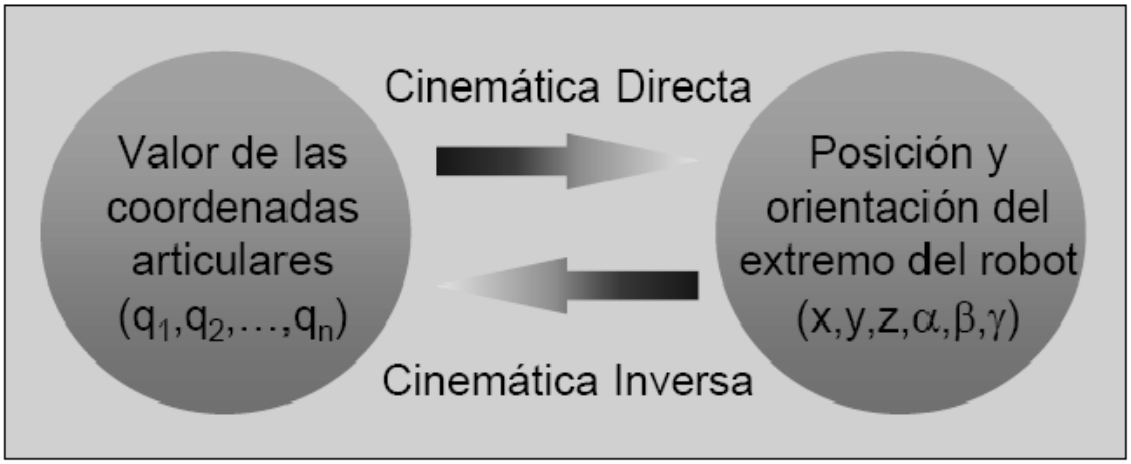
La cinemática es la parte de la mecánica clásica que estudia las leyes del movimiento de los cuerpos sin tener en cuenta las causas que lo producen, limitándose esencialmente, al estudio de la trayectoria en función del tiempo. Cinemática deriva de la palabra griega κινεω (kineo) que significa mover.

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

Estudio del movimiento del robot 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 cinemático 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.




## Modelo Cinemático:

- Método basado en relaciones geométricas (Trigonometria)
  - 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 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.

La resolución del problema cinemático 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 cinemático 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 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, ${}^{0}A_1$ 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 ${}^{0}A_n$ 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:

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

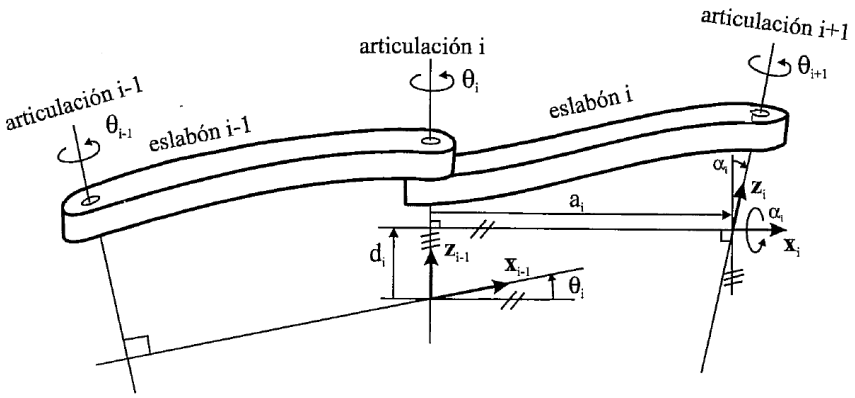
Aunque para describir la relación que existe entre dos elementos contiguos se puede hacer uso de cualquier sistema de referencia ligado a cada elemento.

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 {Si-1} al {Si } mediante estas 4 transformaciones está garantizado solo si los sistemas { Si-1 } al { Si } han sido definidos de acuerdo a unas normas determinadas que se expondrán posteriormente):




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

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

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

Donde $\theta$ i , ai , di, $\alpha$ i son los parámetros D-H del eslabon i. De este modo, basta con identificar los parámetros $\theta$ i , ai , di, $\alpha$ i para obtener las matrices A y relacionar así todos y cada uno los eslabones del robot.

## Algoritmo 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 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 zi con la línea normal común a zi-1 y zi .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 xi en la línea normal común a zi-1 y zi .

**D-H 8.** Situar yi de modo que forme un sistema dextrógiro con xi y zi.

**D-H 9.** Situar el sistema { Sn } en el extremo del robot de modo que zn , coincida con la dirección de zn-1 y xn sea normal a zn-1 y zn.

**D-H 10.** Obtener θi como el ángulo que hay que girar en torno a zi-1 para que xi-1 y xi , queden paralelos.

**D-H 11.** Obtener di, como la distancia, medida a lo largo de zi-1 , que habría que desplazar { Si-1 } para que xi y xi-1 quedasen alineados.

**DH 12.** Obtener ai como la distancia medida a lo largo de xi (que ahora coincidiría con xi-1 ) que habría que desplazar el nuevo { Si-1 } para que su origen coincidiese con { Si }.

**DH 13.** Obtener αi como el ángulo que habría que girar entorno a xi, (que ahora coincidiría con xi-1),para que el nuevo { Si-1 } coincidiese totalmente con { Si }.

**DH 14.** Obtener las matrices de transformación i-1Ai definidas anteriormente.

**DH 15.** Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot T = 0A1, 1A2 ..... n-1An .

**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 ( $\theta$ i , ai , di, $\alpha$ )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:

- $\theta$ **i** Es el ángulo que forman los ejes xi-1 y xi , medido en un plano perpendicular al eje zi-1 ,utilizando la regla de la mano derecha. Se trata de un parámetro variable en articulaciones giratorias.

- **di** Es la distancia a lo largo del eje zi-1 desde el origen del sistema de coordenadas ( i − 1) -esimo hasta la intersección del eje zi-1 con el eje xi . Se trata de un parámetro variable en articulaciones prismáticas.

- **ai** Es la distancia a lo largo del eje xi que va desde la intersección del eje zi-1 con el eje xi hasta el origen del sistema i-esimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas, se calcula como la distancia mas corta entre los ejes zi-1 y zi .

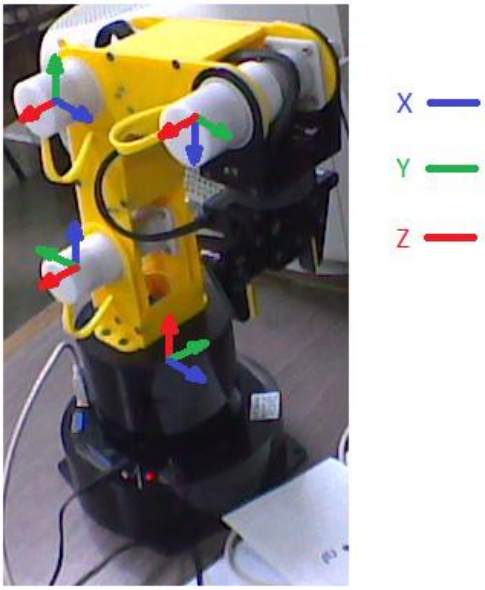
- $\alpha$ **i** Es el ángulo de separación del eje zi-1 y el eje zi ,medido en un plano perpendicular al eje xi ,utilizando la regla de la mano derecha.

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 (xi,yi,zi) 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-1Ai
- Calcular la matriz Tn = 0A1 1A2 ..... n-1An

# Obtención del modelo cinético:

## Sistemas de referencia en los distintos grados de libertad




## Tabla de movimientos:

| | Rotación en Z | Traslación en Z | Traslación en X | Rotación en X |
|---|---|---|---|---|
| L1 | 0 | 0 | l1 | 90 |
| L2 | 90 | 0 | 0 | 0 |
| L3 | 0 | 0 | l2 | 0 |
| L4 | -90 | 0 | l3 | 0 |
| L5 | 90 | 0 | 0 | 0 |
| L6 | -90 | 0 | l4 | 0 |

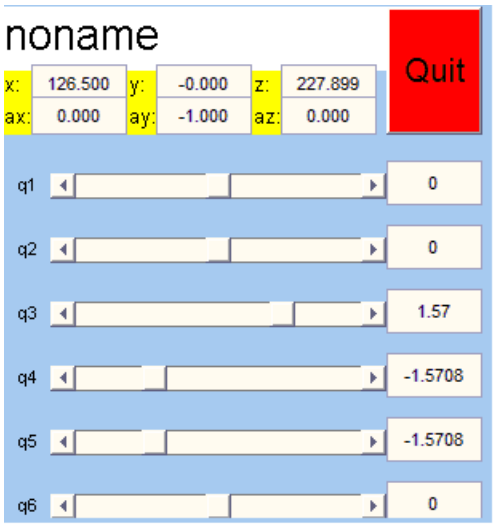
- L2 y L5 son giros necesarios para seguir el método de DyH pero no existen partes físicas que las representen

Con el siguiente script se calculan las matrices individuales A desde un eslabón a otro, quedaron 4 matrices debido a que los 2 giros mentiros los insertamos en las demás ya que no representaban muñones reales.

## Script Matlab 1

```
%*************************************************************
%Valores Iniciales
%*************************************************************
P=pi/2;
q1=0;
q2=0;
q3=P;
q4=-P;
q5=0;
q6=-P;

L1=96+59+66+7;
L2=112.5+14;
L3=112.5+14;
L4=112.5+14;
%*************************************************************

%T(z) A 01
Tz_A0_1=[1 0 0 0;0 1 0 0;0 0 1 L1;0 0 0 1];
%R(x) A01
Rx_A0_1=[1 0 0 0;0 cos(q1+P) -sin(q1+P) 0; 0 sin(q1+P) cos(q1+P)
0;0 0 0 1];

A0_1=Tz_A0_1*Rx_A0_1;

%***************ROTACIÒN MENTIROSA DE Z***************
Rz_A1_2m=[cos(q2+P) -sin(q2+P) 0 0;sin(q2+P) cos(q2+P) 0 0;0 0 1
0;0 0 0 1];

%***************DEL MUÑÓN 1 AL 2**********************
Rz_A1_2=[cos(q3-P) -sin(q3-P) 0 0;sin(q3-P) cos(q3-P) 0 0;0 0 1
0;0 0 0 1];
Tx_A1_2=[1 0 0 L2;0 1 0 0; 0 0 1 0;0 0 0 1];

A1_2=Rz_A1_2*Tx_A1_2;

%***************DEL MUÑÓN 2 AL 3**********************
Rz_A2_3=[cos(q4-P) -sin(q4-P) 0 0;sin(q4-P) cos(q4-P) 0 0;0 0 1
0;0 0 0 1];
Tx_A2_3=[1 0 0 L3;0 1 0 0; 0 0 1 0;0 0 0 1];

A2_3=Rz_A2_3*Tx_A2_3;

%***************DEL MUÑÓN 3 AL 4**********************
Rz_A3_4=[cos(q5-P) -sin(q5-P) 0 0;sin(q5-P) cos(q5-P) 0 0;0 0 1
0;0 0 0 1];
Tx_A3_4=[1 0 0 L4;0 1 0 0; 0 0 1 0;0 0 0 1];

A3_4=Rz_A3_4*Tx_A3_4;
```

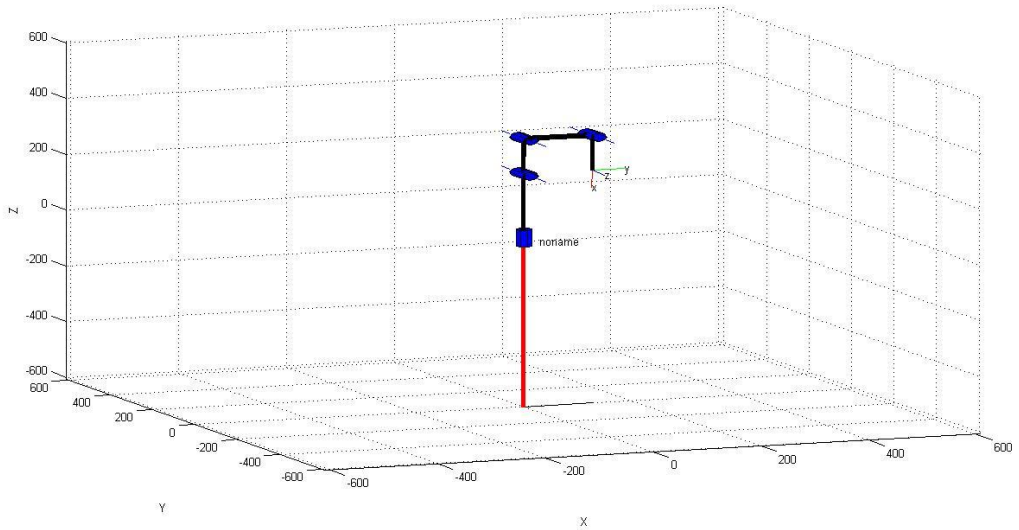
# Diseño del robot en Matlab




Como vamos a simular un robot que se encuentra en la facultad como es el M5, tuvimos la posibilidad de tomar datos del software de simulación del mismo. En ese software realizamos una simulación de una cinemática directa, colocando los ángulos que cada pieza se mueve, realizando el desplazamiento de a una pieza a la vez hasta llegar a un determinado punto.
Como primer paso debimos tomar los datos del setpoint del robot los cuales eran:

Control de articulación del robot

| q1 | q2 | q3 | q4 | q5 |
|---|---|---|---|---|
| 0 | 90 | -90 | -90 | 0 |




Esquema del Robot en el setpoint

Como se ve en la figura del control del robot tenemos un q6 el cual no figura en el setpoint del robot. Eso es debido a que como explicaremos con el script de Matlab tuvimos necesidad de agregar "links inexistentes en la realidad".

## Script de Matlab

```
%*************************************************
%************************TP1**********************
syms q1 q2 q3 q4 q5 q6 q7 q8;

b1=96+59+66+7;%datos hasta el primer muñon
b2=112.5+14;%esta expresado todo en milímetros
b3=112.5+14;
b4=112.5+14;

%LINK R(Z)*T(Z)*T(X)*R(X)

%desde la base al primer muñón traslado en z giro en x
l1=link([pi/2 0 0 b1],'standard');

%rota en z (es un engaño no se toca)
l2=link([0 0 pi/2 0],'standard');

%traslado en x (articulación giro en z)

l3=link([0 b2 0 0],'standard');

%giro en z para correr x y
l4=link([0 b3 -pi/2 0],'standard');

l5=link([0 0 pi/2 0],'standard');

l6=link([0 b4 pi/2 0],'standard');

%define la matriz de   DyH              robo1=robot({l1 l2 l3 l4
l5 l6})

tr=fkine(robo1,[q1 q2 q3 q4 q5 q6])%crea la matriz homogénea

tr=simple(tr)%simplifica la matriz obtenida

drivebot(robo1) %dibuja el robot y el panel de control
```

Como se ve en el script usamos las funciones realizadas por el Peter Croke (las cuales deben cargarse previamente).
Debe prestarse atención que q2 y q6 son eslabones que no existen por lo que no varían son constantes.

Al tener ya el robot diseñado, en el software del M5 tomamos nota de las coordenadas del setpoint las cuales eran

| | x | y | z |
|---|---|---|---|
| Software | 127 | 0 | 208.01 |
| Matlab | 126.5 | 0 | 227.9 |

Se puede advertir que la diferencia en el eje x e y son mínimas pero no lo son para el eje z, pero al probar otros puntos notamos que la diferencia era constante, por lo que llegamos a la conclusión que al no tener en cuenta el griper del robot en el Matlab nos trajo esa distancia no deseada.

Luego tomando el resulta de `tr=simple(tr)` simplificamos el mismo para llegar a expresiones individuales de cada componente de la matriz homogénea total (script2 ). Esta matriz quedó en función de todos los ángulos y las distancias, pero sólo basta con reemplazar cualquier ángulo para poder obtener la posición del robot

## Script 2 de Matlab

```
clc

syms q1 q2 q3 q4 q5 q6;
%Posición inicial del robot (simulado con drivebot)
%x=126.6 y=0 z=228.1
%q1=0;q2=pi/2;q3=0;q4=-pi/2;q5=-pi/2;q6=0;


q1=0;
q2=0;
q3=pi/2;
q4=-pi/2;
q5=0;
q6=-pi/2;

vec=[0 0.26 0.26 0.26];

for i=1:4

q3=q3+vec(1,i);
Q3=q3*180/pi

%fila 1 de la matriz 4x4
```

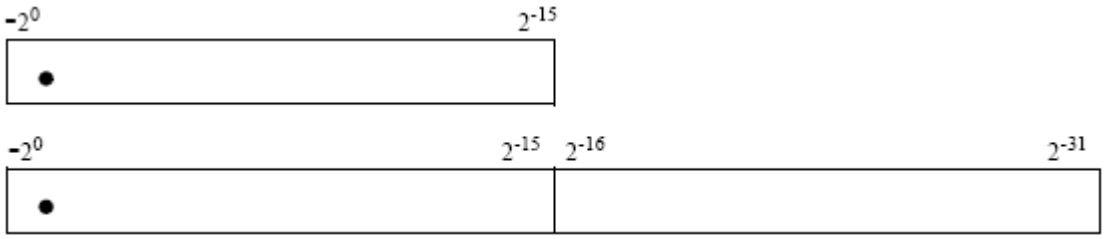
```
f1=[(0.5*cos(q1 - q2 - q3 - q4 - q5 - q6)) + (0.5*cos(q1 + q2 +
q3 + q4 ...
+ q5 + q6)), (0.5*sin(q1 - q2 - q3 - q4 - q5 - q6)) -
(0.5*sin(q1 + q2 +...
q3 + q4 + q5 + q6)),sin(q1), (63.25*cos(q1 + q2 + q3))+
(63.25*cos(q1 - ...
q2 - q3 - q4 - q5 - q6)) + (63.25*cos(q1 + q2 + q3 + q4 + q5 +
q6))+ ...
(63.25*cos(q1 + q2 + q3 + q4))+ (63.25*cos(q1 - q2 - q3 - q4)) +
...
(63.25*cos(q1 - q2 - q3))];

%fila 2 de la matriz 3x3
f2=[ 0.5*sin(q1 - q2 - q3 - q4 - q5 - q6) + 0.5*sin(q1 + q2 + q3
+ q4 + ...
q5 + q6),0.5*cos(q1 + q2 + q3 + q4 + q5 + q6) - 0.5*cos(q1 - q2
- q3 - ...
q4 - q5 - q6),-cos(q1),(63.25*sin(q1 + q2 + q3)) + (63.25*sin(q1
- q2 -...
q3 - q4 - q5 - q6)) + 63.25*sin(q1 + q2 + q3 + q4 + q5 +
q6)+(63.25*...
sin(q1 + q2 + q3 + q4))+(63.25*sin(q1 - q2 - q3 - q4)) + ...
(63.25*sin(q1 - q2 - q3))];


%fila 3 de la matriz 3x3
f3=[sin(q2 + q3 + q4 + q5 + q6),cos(q2 + q3 + q4 + q5 + q6),
6.1232e-017...
,(253*sin(q2 + q3 + q4 + q5 + q6))/2 + (253*sin(q2 + q3 + q4))/2
+ ...
(253*sin(q2 + q3))/2 + 228];

%fila 4 de la matriz 4x4
f4=[0,0,0,1];

MH=[f1;f2;f3;f4];

pos_inic=[0;0;0;1];

cordenadas=MH*pos_inic;
cordenadas(1:3,1)
end
```

# Programación DSP

Obtenidas las expresiones de las componentes, notamos que son muchos cálculos para realizar y deben ser hechos a gran velocidad, para poder lograr que el robot se desplace a una velocidad aceptable. Para ello usamos un DSP, el cual programado con funciones propias del

mismo puede lograr velocidades muy grandes de cálculos.

Este dispositivo  trabaja con variables de punto fijo no flotante, llamadas FRAC 16 0 FRAC32, las primeras son números de 16bits que pueden representar decimales entre -1 y 1. Mientras que los segundos pueden representar enteros (16bits) y decimales (16 bits menos significativos). El programa realizado toma cada expresión de la matriz homogénea total y la calcula usando funciones específicas del dispositivo, que trabajan con el tipo de variables antes mencionada.




El programa utilizado para escribir y correr el código es el Code Warrior, el cual tiene una versión estudiantil para bajar. Para poder correr el programa se necesitan debemos agregar 3 librerías al programa como son:

**TFR1:DSP_Func_TFR**

la cual contiene funciones trigonométricas básicas para tanto variables de 16 y 32 bit

**MFR1:DSP_Func_MFR**

contiene funciones matemáticas básicas

**MEM:DSP_MEM**

contiene funciones específicas para la memoria, como ser pedir memoria

El programa define muchas variables especialmente trigonométricas como:
p1  =  q1 + q2 + q3 + q4 + q5 + q6
p4  =  q1 + q2 + q3
Debido a que en el cálculo de las matrices muchas de esas sumas se usan más de una vez, ya que la matriz simplificada tiene similares características entre sus filas.

Luego se realizan los cálculos trigonométricos de las mismas, y después se los divide por 4 (shr) para lograr que las sumas no den más de 1.

# Programa DSP en Code Warrior

```
/* Librerias */
#include "Cpu.h"
#include "TFR1.h"
#include "MFR1.h"
#include "MEM1.h"

#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include "stdio.h"
//*********************************************************************
//         DEFINE
//*********************************************************************

//1 rad = 57.3° si 180 es 32767 entonces 1rad = 10430  (1/2)rad=5215
#define MXRAD 361 //100
#define MAXIMO 32767
#define PULSE2RAD 32767/MXRAD  // 32767/100 impulsos // #define PULSE2RAD 450
#define rad 57
#define valor0_5 0x4000
#define valor_1000 0x3E80000
#define valor_100 0x640000
#define norm_a_4        2          //cuando sumo senos y cosenos es resultado puede ser
                                   //mayor a 1 por lo tanto normalizo a 4 para no excederme
```

```
#define desnorm_a_4              4

#define valor001 328

#define div_por_2        1        //divido por 2

#define valor_4000  0xFA00000

#define valor_6325 0x3f2000 //constante 63.25

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

                        Definicion de funciones

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

void calcular (void);                        //FUNCION PRINCIPAL


void ent_dec (Frac32 var_imp); //IMPRIME LOS DATOS DE VARIABLES DE 32BITS

//PARTE ALTA en forma enteros PARTE BAJA en forma .decimal


void imprimir_32 (Word32 valor_a_impr );

//IMPRIME LOS DATOS DE VARIABLES DE 32BITS PARTE ALTA en forma enteros

//PARTE BAJA en forma FRAC16


void desnor_mayor1 (Word16 valor_a_desnor,int factor);

//DESNORMALIZA UN NUM (LIM DE FACTOR ES 32) LO COPIA EN uso_gen_32[0]


void mult_32x16 (Word32 valor_32,Word16 valor_16);


void mult_32x32 (Word32 num1,Word32 num2);

void sumar_32 (Word32 num1,Word32 num2);

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

//        GLOBAL VARIABLE

//***********************************************************************
```

```
Word32 uso_gen_32[2],q1_32;

Word32 pi=316,p1_32,p2_32,p3_32,p4_32,p5_32,p6_32,p7_32,p10_32,p9_32;

Word32          var_6325=0x3F2000,var_1265;

Word32 Homogenea[4][4];

Word16 q1_l,shifteo=1,norm_2,uso_gen;

Word16 p1,p2,p3,p4,p5,p6,p7,sen_total_resta,p8,p9,p10;

Frac16  q1, q2, q3,q4,q5,q6,q1_;

Frac16  cosq1, senq1, cosq2, senq2, L1, c;

Frac16
        cos_total_suma,cos_total_resta,cos_suma3,cos_resta3,cos_suma4,cos_resta4,cos_su
ma_sinq1;

Frac16  sen_total_suma,sen_suma3,sen_resta3,sen_suma4,sen_resta4,sen_suma_sinq1;

Frac16  sen_suma3_A,sen_suma2_A;

Frac16 paso_min=PULSE2RAD,MAX=MAXIMO;


int j,k,i=0;
```

---

Programa Principal

```
**************************************************************************
***

void main(void)

{

  PE_low_level_init();

                q1=0;

                q2 = 0;

                q3 = 90;

                q4=negate(90);

                q5 = 0;

                q6 = negate(90);
```

```
                printf("\n Q3 \n");
                for(q3=90; q3<145; q3+=15)
                        {
                        calcular();
                        }
                printf("\n Q4 \n");
                for(q4=-90; q4>-145; q4-=15)
                        {
                        calcular();
                        }
                printf("\n Q5 \n");
                for(q5=-90; q5>-145; q5-=15)
                        {
                        calcular();
                        }
}
/*************************************************************************
                Los q' guardan el valor de los ángulos (no en radianes)
*************************************************************************/

 void calcular ()

 {
 Frac16 var1,var2,var3,var4,var5,var_05;
 /*************************************************************************
                        Los p' calculan operaciones de suma y restas de los q'
                        y, después lo multiplican por el paso_min (que es el valor
                        de 1 grado para el DSP)
```

```
/*****************************************************************************/

                p1=sub(q1,add(q2,add(q3,add(q4,add(q5,q6))))          ;//q1 - q2 - q3 - q4 - q5
- q6;

                p1_32=L_mult(p1,paso_min);

                p1=extract_l(p1_32);


                p2=add(q1,add(q2,add(q3,add(q4,add(q5,q6))))          ;//q1 + q2 + q3 + q4 +
q5 + q6;

                p2_32=L_mult(p2,paso_min);

                p2=extract_l(p2_32);


                p3=add(q1,add(q2,q3))             ;//q1 + q2 + q3;

                p3_32=L_mult(p3,paso_min);

                p3=extract_l(p3_32);


                p4=add(q1,add(q2,add(q3,q4)));//q1 + q2 + q3 + q4;

                p4_32=L_mult(p4,paso_min);

                p4=extract_l(p4_32);


                p5=sub(q1,add(q2,add(q3,q4))) ;//q1 - q2 - q3 - q4;

                p5_32=L_mult(p5,paso_min);

                p5=extract_l(p5_32);


                p6=sub(q1,add(q2,q3)) ;//q1 - q2 - q3;

                p6_32=L_mult(p6,paso_min);

                p6=extract_l(p6_32);
```

```
p7=add(q2,add(q3,add(q4,add(q5,q6))))          ;//q2 + q3 + q4 + q5 + q6 + q7
p7_32=L_mult(p7,paso_min);
p7=extract_l(p7_32);

p9=add(q2,q3)  ;
p9_32=L_mult(p9,paso_min);
p9=extract_l(p9_32);

p10=add(q2,add(q3,q4));
p10_32=L_mult(p10,paso_min);
p10=extract_l(p10_32);
```

```
/*************************************************************************
                        Cálculo de cosenos
*************************************************************************/

q1_32=L_mult(q1,paso_min);
q1_=extract_l(q1_32);
cosq1 = tfr16CosPIx (q1_);
cos_total_suma = tfr16CosPIx (p2);
cos_total_resta = tfr16CosPIx (p1);
cos_suma3=tfr16CosPIx (p3);
cos_resta3=tfr16CosPIx (p6);
cos_suma4=tfr16CosPIx (p4);
cos_resta4=tfr16CosPIx (p5);
cos_suma_sinq1=tfr16CosPIx(p7);
```

```
/*************************************************************************
                              Cálculo de senos
/*************************************************************************/


                senq1 = tfr16SinPIx (q1_);
                sen_total_suma = tfr16SinPIx (p2);
                sen_total_resta = tfr16SinPIx (p1);
                sen_suma3=tfr16SinPIx (p3);
                sen_resta3=tfr16SinPIx (p6);
                sen_suma4=tfr16SinPIx (p4);
                sen_resta4=tfr16SinPIx (p5);
                sen_suma_sinq1=tfr16SinPIx(p7);
                sen_suma3_A=tfr16SinPIx(p10) ;//seno (q2 + q3 + q4)
                sen_suma2_A=tfr16SinPIx(p9)   ;//seno (q2 + q3)



/*************************************************************************
                Normalizo nuevamente para poder representar valores de las sumas
                de cosenos y de senos
 /*************************************************************************/
               shifteo=norm_a_4;
                cosq1=shr_r(cosq1,shifteo);
                cos_total_suma=shr_r(cos_total_suma,shifteo);
                cos_total_resta=shr_r(cos_total_resta,shifteo);
                cos_suma3=shr_r(cos_suma3,shifteo);
                cos_resta3=shr_r(cos_resta3,shifteo);
                cos_suma4=shr_r(cos_suma4,shifteo);
```

```
                cos_resta4=shr_r(cos_resta4,shifteo);

                cos_suma_sinq1=shr_r(cos_suma_sinq1,shifteo);


                senq1=shr(senq1,shifteo);

                sen_total_suma=shr(sen_total_suma,shifteo);

                sen_total_resta=shr(sen_total_resta,shifteo);

                sen_suma3=shr_r(sen_suma3,shifteo);

                sen_resta3=shr_r(sen_resta3,shifteo);

                sen_suma4=shr_r(sen_suma4,shifteo);

                sen_resta4=shr_r(sen_resta4,shifteo);

                sen_suma_sinq1=shr_r(sen_suma_sinq1,shifteo);

                sen_suma3_A=shr_r(sen_suma3_A,shifteo);

                sen_suma2_A=shr_r(sen_suma2_A,shifteo);
/*************************************************************************
                                Cálculo de Matriz
/*************************************************************************/
        /* FILA 1*/

                uso_gen = shr(add(cos_total_suma,cos_total_resta),div_por_2);

                //como divido por 2 shifteo para

                desnor_mayor1(uso_gen,desnorm_a_4);//devuelve el numero en
uso_gen_32[0]

                Homogenea[0][0] = uso_gen_32[0];




                uso_gen = shr(sub(sen_total_resta,sen_total_suma),div_por_2);          //como
divido por 2 shifteo para

                desnor_mayor1(uso_gen,desnorm_a_4);
```

```
                Homogenea[0][1] = uso_gen_32[0];                                    //la
derecha

                Homogenea[0][2] = L_deposit_l(senq1);

                        /* Calculo de homogenea[0][3] */

                var1=add(cos_total_suma,cos_total_resta);
                var2=add(cos_suma3,cos_resta3);
                var3=add(cos_suma4,cos_resta4);
                var4=add(var1,var2);
                var5=add(var4,var3);

                desnor_mayor1(var5,4); //devuelve el numero en uso_gen_32[0]
                mult_32x32(uso_gen_32[0],var_6325);
                Homogenea[0][3]=uso_gen_32[0];


/* FILA 2*/

                uso_gen = shr(add(sen_total_suma,sen_total_resta),div_por_2);
                desnor_mayor1(uso_gen,desnorm_a_4);//devuelve el numero en
uso_gen_32[0]
                Homogenea[1][0] = uso_gen_32[0];


                uso_gen = shr(sub(cos_total_suma,cos_total_resta),div_por_2);
```

```
                desnor_mayor1(uso_gen,desnorm_a_4);//devuelve el numero en
uso_gen_32[0]

                Homogenea[1][1] = uso_gen_32[0];


                uso_gen_32[0]=L_deposit_l(0); //pongo todo a cero, porque la parte alta

                                                                        //tambien la
llena con ceros

                uso_gen=negate(cosq1);

                desnor_mayor1(uso_gen,desnorm_a_4);//devuelve el numero en
uso_gen_32[0]

                Homogenea[1][2] = uso_gen_32[0];


                var1=add(sen_total_suma,sen_total_resta);

                var2=add(sen_suma3,sen_resta3);

                var3=add(sen_suma4,sen_resta4);

                var4=add(var1,var2);

                var5=add(var4,var3);


                desnor_mayor1(var5,4); //devuelve el numero en uso_gen_32[0]

                mult_32x32(uso_gen_32[0],var_6325);//devuelve el numero en
uso_gen_32[0]

                Homogenea[1][3]=uso_gen_32[0];


/* FILA 3*/


                uso_gen=sen_suma_sinq1;

                desnor_mayor1(uso_gen,desnorm_a_4);//devuelve el numero en
uso_gen_32[0]

                Homogenea[2][0] = uso_gen_32[0];
```

```
                uso_gen=cos_suma_sinq1;

                desnor_mayor1(uso_gen,desnorm_a_4);//devuelve el numero en
uso_gen_32[0]

                Homogenea[2][1] = uso_gen_32[0];


                Homogenea[2][2] = 0    ;          //es 6.1232e-017 pero como es muy chico para
                                                                  //un numero de 16bits pongo 0



                var2=add(sen_suma_sinq1,add(sen_suma3_A,sen_suma2_A));


                uso_gen_32[1]=0xFD0000;       //numero 253 en la parte alta

                desnor_mayor1(uso_gen,desnorm_a_4);//devuelve el numero en
uso_gen_32[0]

                mult_32x32(uso_gen_32[0],uso_gen_32[1]);//devuelve el numero en
uso_gen_32[0]

                uso_gen_32[1]=0xE40000;       //numero 228 en la parte alta

                sumar_32(uso_gen_32[1],uso_gen_32[0]);//devuelve el numero en
uso_gen_32[0]

                Homogenea[2][3]=uso_gen_32[0];



/* FILA 4*/


                Homogenea[3][0] = L_deposit_h(0);

                Homogenea[3][1] = L_deposit_h(0);

                Homogenea[3][2] = L_deposit_h(0);

                Homogenea[3][3] = L_deposit_h(1);
```

```
/*************************************************************************
                                Impresion de Coordenadas
/*************************************************************************/
                printf("\nCOORDENADAS\n");
                for(i=0; i<3 ;i++)
                        {
                        ent_dec(Homogenea[i][3]);
                        }
                printf("\n");



}

/*La funcion tiene el límite de 326.xxxx*/

void mult_32x16 (Word32 valor_32, Word16 valor_16)
        {
        Word16 ent,dec,general,general2;
        Word32 var_local;

        /*ejemplo valor_32=63.0x2000 (63.25) valor_16=0.25=0x2000*/

        ent=extract_h(valor_32);//63
        dec=extract_l(valor_32);//0x2000
        dec=extract_h(L_mult_ls(valor_100,dec));        //(100x0.25) lo guarda en la parte

        // alta y lo guardo en dec
```

```
        //var_local=L_deposit_l(ent);
        var_local=L_shr(L_mult(ent,100),div_por_2);     //63x100=6300
                                                                    //por el doble
signo nos da el doble
        general=extract_l(var_local);
        general=add(general,dec);// 6300+25

        general=mult(general,valor_16); //multiplica en formato Frac16
        // 6325 x 0x2000 (es como dividir por 0.25) general=1581
        //printf(" %d\n",general);
        general2=mult(general,valor001);                //general2 = 15
        ent=general2;
        var_local=L_shr(L_mult(ent,100),div_por_2);     //15x100=1500
                                                                    //por el doble
signo nos da el doble
        general2=extract_l(var_local);          //general2=1500
        dec=sub(general,general2);                      //dec=1581-1500


        uso_gen_32[0]=L_deposit_h(ent);
        //var_local=L_shr(L_mult(dec,valor001),div_por_2);esta línea es la de abajo
        uso_gen_32[0]=L_add(uso_gen_32[0],L_shr(L_mult(dec,valor001),div_por_2));
        //ent_dec(uso_gen_32[0]);
        }


/*La función multiplica 2 numeros de 32 bits dividiendo la multiplicacion en 4
```

```
ent1*ent2

ent1*dec2

ent2*dec1

dec1*dec2

y suma cada resultado con el anterior

*/

        void mult_32x32 (Word32 num1,Word32 num2)

        {

        Word16 ent1,ent2,dec1,dec2,general;

        ent1=extract_h(num1);

        ent2=extract_h(num2);

        dec1=extract_l(num1);

        dec2=extract_l(num2);


        // ent1*dec2 = A //

        uso_gen_32[1]=L_mult(ent1,ent2);       //multiplica comunmente ambos numero y los

                                                                                    //guarda en la
P baja

        general=extract_l(uso_gen_32[1]);      //Extrae la parte baja

        general=shr(general,div_por_2);        //la multiplicacion  por el doble signo

                                                                                    //da el
doble

        uso_gen_32[1]=L_deposit_h(general);  //copia general en la parte alta


        // ent1*dec2 = B //

        uso_gen_32[0]=L_deposit_h(ent1);

        mult_32x16(uso_gen_32[0],dec2);                 //Lo guarda en uso_gen_32[0]

        // A + B //
```

```
        sumar_32(uso_gen_32[1],uso_gen_32[0]);//Lo guarda en uso_gen_32[0]
        uso_gen_32[1]=uso_gen_32[0]; //guarda la suma

        // ent2*dec1 = C //
        uso_gen_32[0]=L_deposit_h(ent2);
        mult_32x16(uso_gen_32[0],dec1);                 //Lo guarda en uso_gen_32[0]
        // A + B + C //
        sumar_32(uso_gen_32[1],uso_gen_32[0]);//Lo guarda en uso_gen_32[0]
        uso_gen_32[1]=uso_gen_32[0];

        // dec1*dec2 = D//
        general=mult(dec1,dec2);  //multiplica en modo FRAC16
        uso_gen_32[0]=L_deposit_l(general);    //copia general en la parte alta

        // A + B + C + D //
        sumar_32(uso_gen_32[1],uso_gen_32[0]);//Lo guarda en uso_gen_32[0]
        }

/*La función suma dos numeros de 32bits teniendo en cuenta que si la parte
decimal suma mas de 1,   suma 1 la parte entera*/

void sumar_32 (Word32 num1,Word32 num2)
        {
        Word16 ent1,ent2,dec1,dec2,general;
        ent1=extract_h(num1);
        ent2=extract_h(num2);
        dec1=extract_l(num1);
```

```
        dec2=extract_l(num2);
        /*Ejemplo dec1=0.35=11468 dec2=0.8=26213*/

        dec1=shr(dec1,div_por_2);       //0.1525=5734
        dec2=shr(dec2,div_por_2);       //0.4=13106
        general=add(dec1,dec2);         //18840
        if(general>16383) //0.5
                {
                dec1=shl(sub(general,16383),div_por_2); //2457x2
                ent1=add(ent1,1); //como se paso le sumo 1 a entero
                }
        else
                {
                dec1=shl(general,div_por_2);
                }
        ent1=add(ent1,ent2)     ;
        uso_gen_32[1]=L_deposit_l(dec1);
        uso_gen_32[0]=L_deposit_h(ent1);
        uso_gen_32[0]=L_add(uso_gen_32[0],uso_gen_32[1]);
        }
void ent_dec (Frac32 var_imp)  //IMPRIME LOS DATOS DE VARIABLES DE 32BITS

        {
        Frac32 var_local=0;
        Frac16 parte_alta,parte_baja;

        parte_alta=extract_h(var_imp);
```

```
parte_baja=extract_l(var_imp);
/*****************************************
La parte alta la imprime directamente
la parte baja la multiplica por 1000
(pone 1000 en la parte alta de otra variable
y multipplica por una variable de 16 que contiene
la parte baja de la varible a imprimir)
y luego la imprime
*****************************************/
//var_local=L_add(valor_1000,valor_1000);
var_local=valor_1000;
var_local=L_mult_ls(var_local,parte_baja);        //para sacar los decimales
parte_baja=extract_h(var_local);

if (parte_baja<0 )
        {
        if(!parte_alta)
                /* como no puedo poner -0 escribo - y luego el numero */
                {
                parte_baja=negate(parte_baja);//para que siga la regla de los if de
abajo
                printf("-");

                }
        else
                {
                parte_baja=negate(parte_baja);
                }
```

```
		}
	if(parte_baja<10)
		{
		printf("%d.00%d\n ",parte_alta,parte_baja);
		}
	else
		{
		if (parte_baja<100)
			printf("%d.0%d\n ",parte_alta,parte_baja);
		else
			printf("%d.%d\n ",parte_alta,parte_baja);
		}
}
/*Imprime el número de 32 bit
PARTE ALTA en forma de entero
PARTE BAJA en forma FRAC16
*/
void imprimir_32 (Word32 valor_a_impr )
	{
	uso_gen=extract_h(valor_a_impr);
	printf("parte alta %d\n",uso_gen);
	uso_gen=extract_l(valor_a_impr);

	printf("parte baja %d\n",uso_gen);
	}
void desnor_mayor1 (Word16 valor_a_desnor,int factor)
	{
```

```
Frac32 var_local=valor_1000,var_local2;
Word16 general,parte_alta,parte_baja;
int in,jn,neg=0;

/*****************************************************
Este for incrementa en 1000 el valor a multiplicar, porque
valor_a_desnor es -1 a 1 al multiplicar por factor X 1000
EJ
si valor_a_desnor=0.752 no va a dar 752 X factor
si factor=4 752x4=3008 que significa que es 3.008
*****************************************************/
if(valor_a_desnor<0)
        {
        //printf("\nmenor a cero\n");
        neg=1;
        valor_a_desnor=negate(valor_a_desnor);
        }
for(in=factor-1;in>0;in--)
        {
        var_local=L_add(var_local,valor_1000);
        }

var_local=L_mult_ls(var_local,valor_a_desnor);
general=extract_h(var_local);
/*****************************************************
Este for busca cuantos miles tengo que restarle para
quedarme con lo que sería los decimales
```

EJ

nos va a dar uso_den=8

```
/*****************************************************/


for(in=0,jn=1;in<32&jn==1;in++)
        {
        if(general>1000)
                {
                general=sub(general,1000);
                }
        else
                {
                jn=0;
                }
        }
in--;


var_local2=L_mult(general,32);//33 es 1x10-3
var_local2=L_shr(var_local2,div_por_2);
                                //cuando multiplico me da el doble por el doble signo
if (neg)
        {
        if (in) //in es la parte entera
                in=0-in; //niego el entero
        else
                {
                var_local2=L_negate(var_local2);
```

```
                    var_local2=L_add(var_local2,0x10000);
                    /*no se porqué cuando negaba la varible en la parte alta
                    se colocaba un -1, entonces le  sumo 1*/

                    }
                }

        var_local=L_deposit_h(in);
        var_local2=L_add(var_local2,var_local);
        uso_gen_32[0]=var_local2;
}
```

Muchas de las funciones que contiene el programa son para poder expresar los datos en un formato entero.decimal  y, no dejándolo en forma de FRAC 16.
Así el resultado expresado por el simulador tiene la forma :

COORDENADAS

130.817

0.000

232.200

Hicimos variar las 3 variables principales (no hicimos girar la base) en 45° cada una, pero en forma secuencial.

Los datos los presentamos en una tabla para que sea más sencillo de ver

| q2 | q4 | q5 | X | Y | Z |
|---|---|---|---|---|---|
| 90 | -90 | -90 | 126,5 | 0 | 228 |
| 105 | -90 | -90 | 122,2483 | 0 | 260,5207 |
| 130 | -90 | -90 | 109,7791 | 0 | 290,8553 |
| 145 | -90 | -90 | 89,9306 | 0 | 316,9648 |
| 145 | -75 | -90 | 84,1662 | 0 | 362,9878 |
| 145 | -60 | -90 | 66,7639 | 0 | 405,9821 |
| 145 | -45 | -90 | 38,8935 | 0 | 443,0575 |
| 145 | -45 | -75 | 34,9932 | 0 | 475,6222 |
| 145 | -45 | -60 | 22,8522 | 0 | 506,0897 |
| 145 | -45 | -45 | 3,2867 | 0 | 532,412 |

## Conclusiones

El sistema evaluado con las herramientas de Matlab, nos dio muy similar salvo por el pequeño offset que es constante. Eso nos ayudó mucho para poder corroborar los valores con el Code Warrior. Nos dimos cuenta que el modelo de DyH es una herramienta sencilla para determinar la cinemática del robot, pero tuvimos que agregar 2 eslabones ficticios para poder adaptarlo a la configuración del M5.

La programación del DSP nos da una gran velocidad de resolución, lo que nos permite calcular el próximo punto de la trayectoria con exactitud. Así para luego poder aplicar el cálculo dinámico para conocer los torques necesarios y poder regular el PWM.