Universidad Tecnológica Nacional

Facultad Regional Buenos Aires

Materia: Robótica

Curso: 5to. 5 Dto. Electrónica

Profesor: M.aS. Ing. HernanGiannetta

JTP/ Ayudantes: Ing. DamianGranzella

Ing. Musolino, Augusto;

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

Responsable: Ferreiro, Christian Gabriel

Título: **Tesis Final de Robótica sobre Robot M5**

Alumnos:

Ramos, Nahuel Leg: 119175-5

Silber, Fabio Leg: 119211-5

Grupo 2

Fecha entrega: 23/08/2011

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:

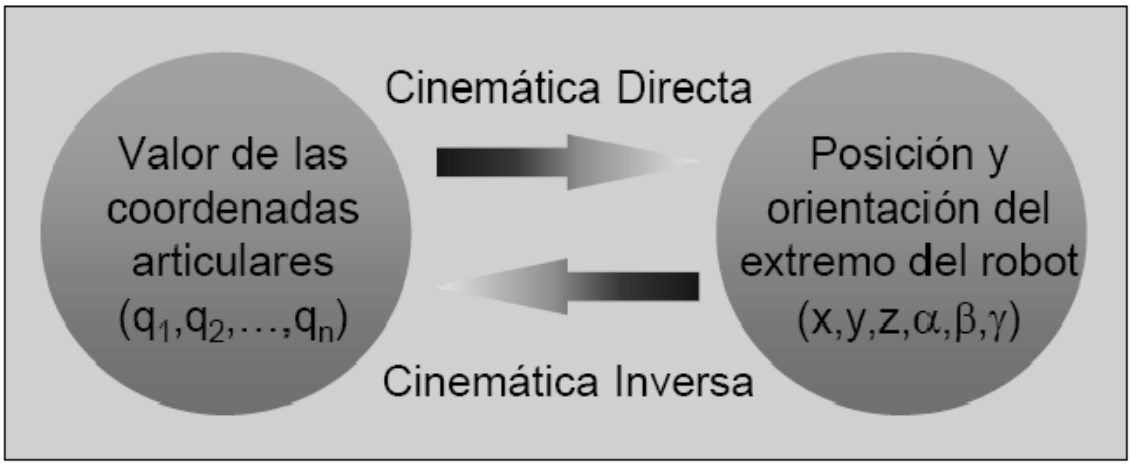
Introducción teórica de la cinemática directa:

La cinemática es la parte de la mecánica clásica que estudia las leyes del movimiento de los cuerpos sin teneren cuenta las causas que lo producen, limitándose esencialmente, al estudio de la trayectoria en función deltiempo. 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 dereferencia.

Estudio del movimiento del robot con respecto a un sistema dereferencia:

- 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 oeslabones unidos entre sí mediante articulaciones, se puede establecer un sistema de referencia fijo situadoen la base del robot y describir la localización de cada uno de los eslabones con respecto a dicho sistema dereferencia.

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 valoresde 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, deforma 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 transformacioneshomogéneas, es posible representar las rotaciones y traslaciones relativas entre los distintos eslabones quecomponen el robot.

Normalmente, la matriz de transformación homogénea que representa la posición y orientación relativaentre 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 conrespecto al sistema de referencia solidario a la base, ${}^{1}A_2$ describe la posición y orientación del segundoeslabón 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 1hasta 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 unrobot de seis grados de libertad, se tiene que la posición y orientación del eslabón final vendrá dada por lamatriz T:

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

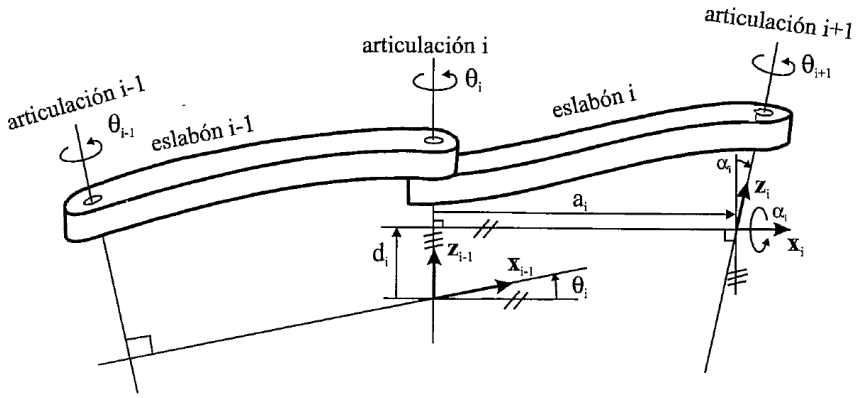
Aunque para describir la relación que existe entre dos elementos contiguos se puede hacer uso de cualquier sistema de referencia ligado a cada elemento.

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 θi ,ai , di, αi son los parámetros D-H del eslabon i. De este modo,basta con identificar los parámetros θi ,ai , di, αi para obtener las matricesA y relacionar así todos y cada unolos 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 (θi ,ai , di, α)dependen únicamente de las características geométricasde cada eslabón y de las articulaciones que le unen con el anterior y siguiente. Enconcreto estos representan:

☐**θi**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 articulacionesgiratorias.

☐**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 enarticulaciones prismáticas.

☐**ai**Es la distancia a lo largo del eje xi que va desde la intersección del eje zi-1 con el eje xihasta 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 .

☐**αi**Es el ángulo de separación del eje zi-1 y el eje zi ,medido en un plano perpendicular aleje 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

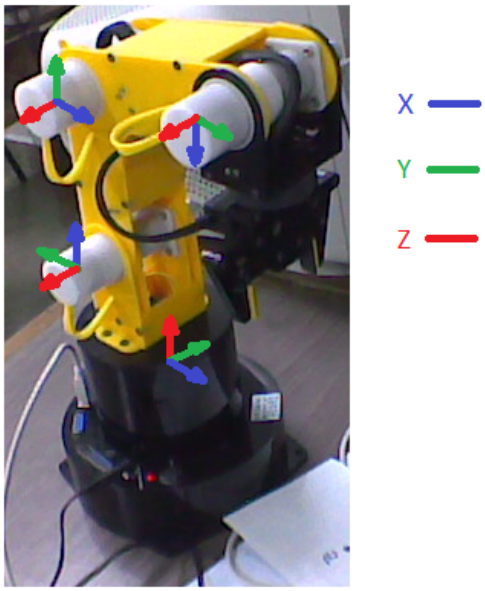


Tabla de movimientos:

	Rotación en Z	Traslación en Z	Traslación en X	Rotación en X
L 1	0	0	l1	90
L 2	90	0	0	0
L 3	0	0	l2	0
L 4	-90	0	l3	0
L 5	90	0	0	0
L	-90	0	l4	0

6				

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

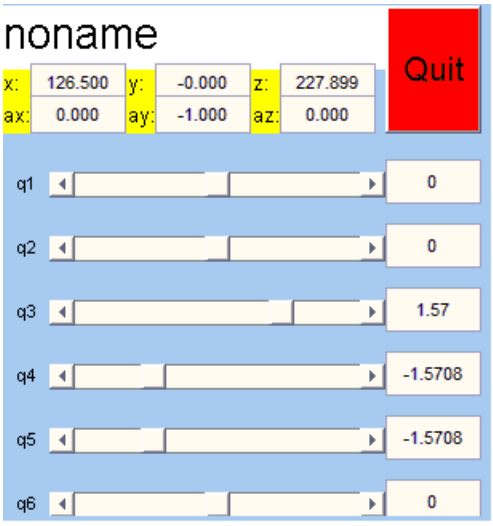
Con el siguiente script se calculan las matrices individuales A desde un eslabón a otro, quedaron 4 matrices debido a que los 2 giros mentiros los insertamos en las demás ya que no representaban muñones reales.

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;
```

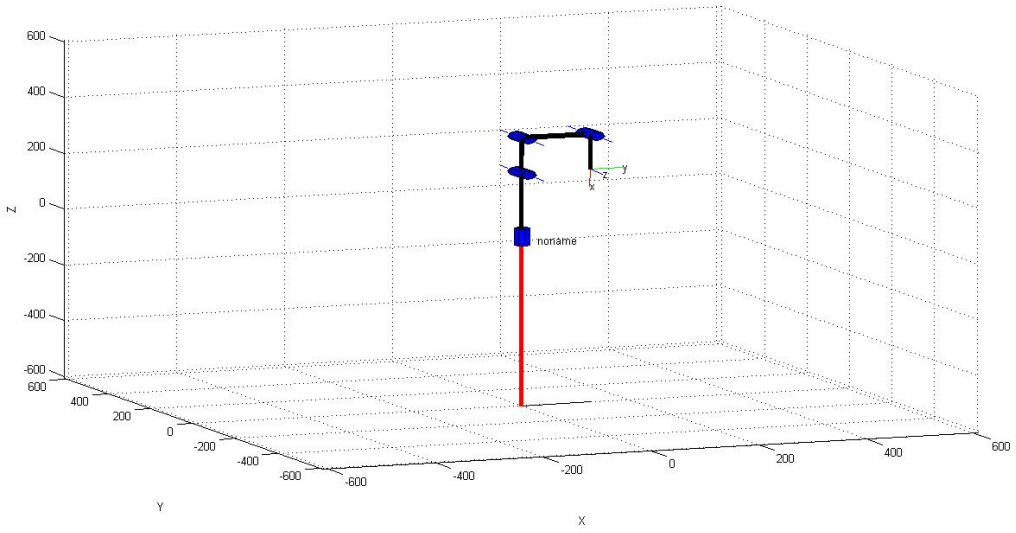
Diseño del robot en Matlab



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

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



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

Robot colocado en el setpoint

d 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 muñóntraslado en zgiro 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   DyHrobo1=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

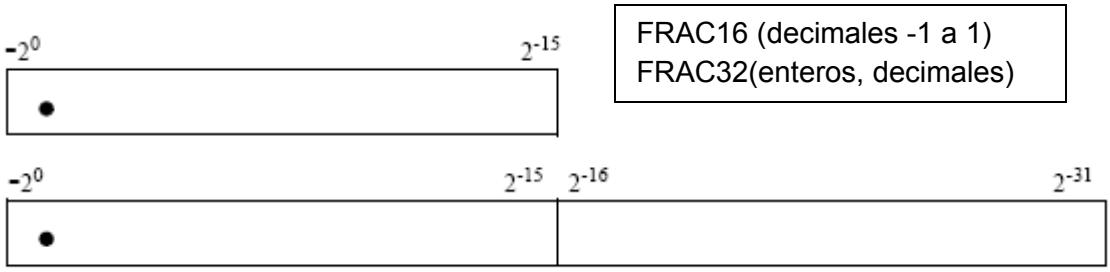
```
clc

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 CodeWarrior, 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 CodeWarrior

```
/* 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
```

```
voident_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_suma_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;

intj,k,i=0;
```

```
*****************************************************************************
void main(void)
{
PE_low_level_init();
                q1=0;
                q2 = 0;
                q3 = 90;
                q4=negate(90);
                q5 = 0;
                q6 = negate(90);
/*********************************************************************
                                TRAYECTORIA 1
/*********************************************************************/
                printf("\n Q2 \n");
                for(q3=90; q3<145; q3+=3)
                        {
                        calcular();
                        }
                printf("\n Q4 \n");
                for(q4=-90; q4>-145; q4-=3)
                        {
                        calcular();
                        }
                printf("\n Q5 \n");
                for(q5=-90; q5>-145; q5-=3)
                        {
                        calcular();
                        }
/*********************************************************************
                                TRAYECTORIA 2
/*********************************************************************/
                q1=0;
                q2 = 0;
                q3 = 90;
                q4=negate(90);
                q5 = 0;
                q6 = negate(90);
                for(q3=90; q3<145; q3+=3)
                        {
```

```
                    calcular();
                    q4=q4+3;
                    calcular();

                    q5=q5+3
                    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]);
        }
voident_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
        ymultipplica 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);
```

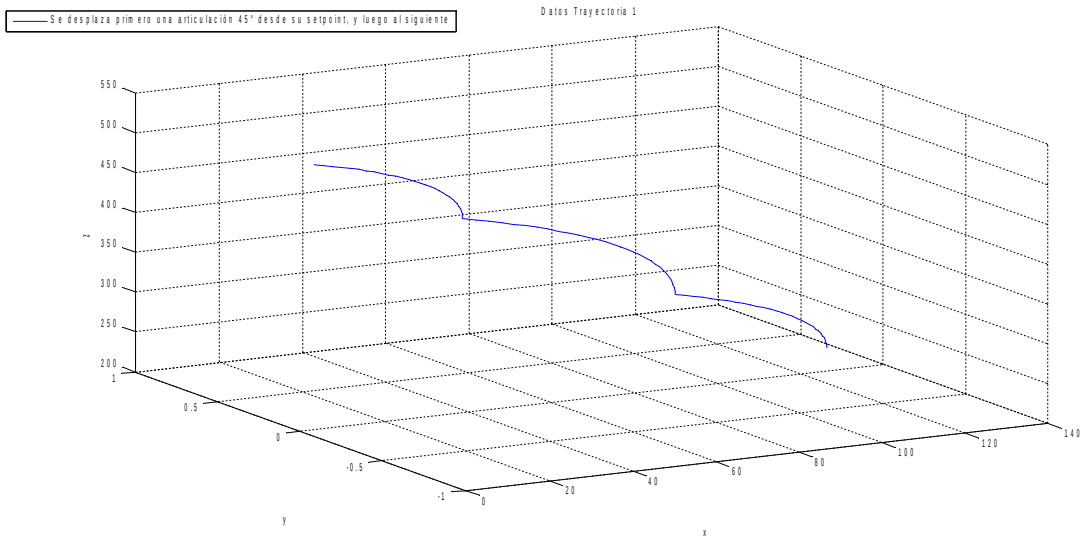
```
                        }
                }
        if(parte_baja<10)
                {
                printf("%d.00%d\n ",parte_alta,parte_baja);
                }
        else
                {
                if (parte_baja<100)
                        printf("%d.0%d\n ",parte_alta,parte_baja);
                else
                        printf("%d.%d\n ",parte_alta,parte_baja);
                } */ No imprimo la 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;
        intin,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
        sivalor_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)
```

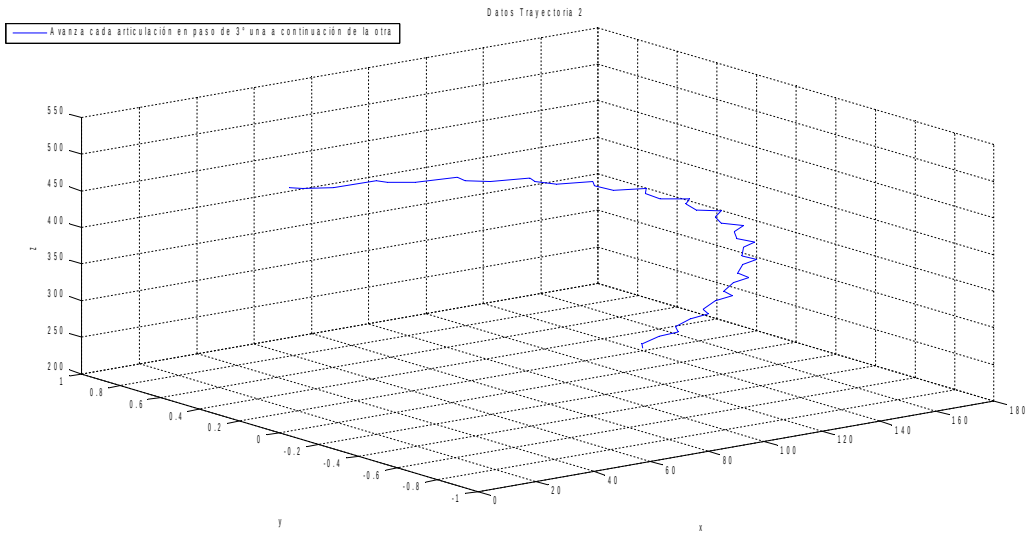
```
                {
                if (in) //in es la parte entera
                        in=0-in; //niego el entero
                else
                        {
                        var_local2=L_negate(var_local2);
                        var_local2=L_add(var_local2,0x10000);
                        /*noseporqué 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 y, no dejándolo en forma de FRAC 16.

TRAYECTORIAS PROPUESTAS

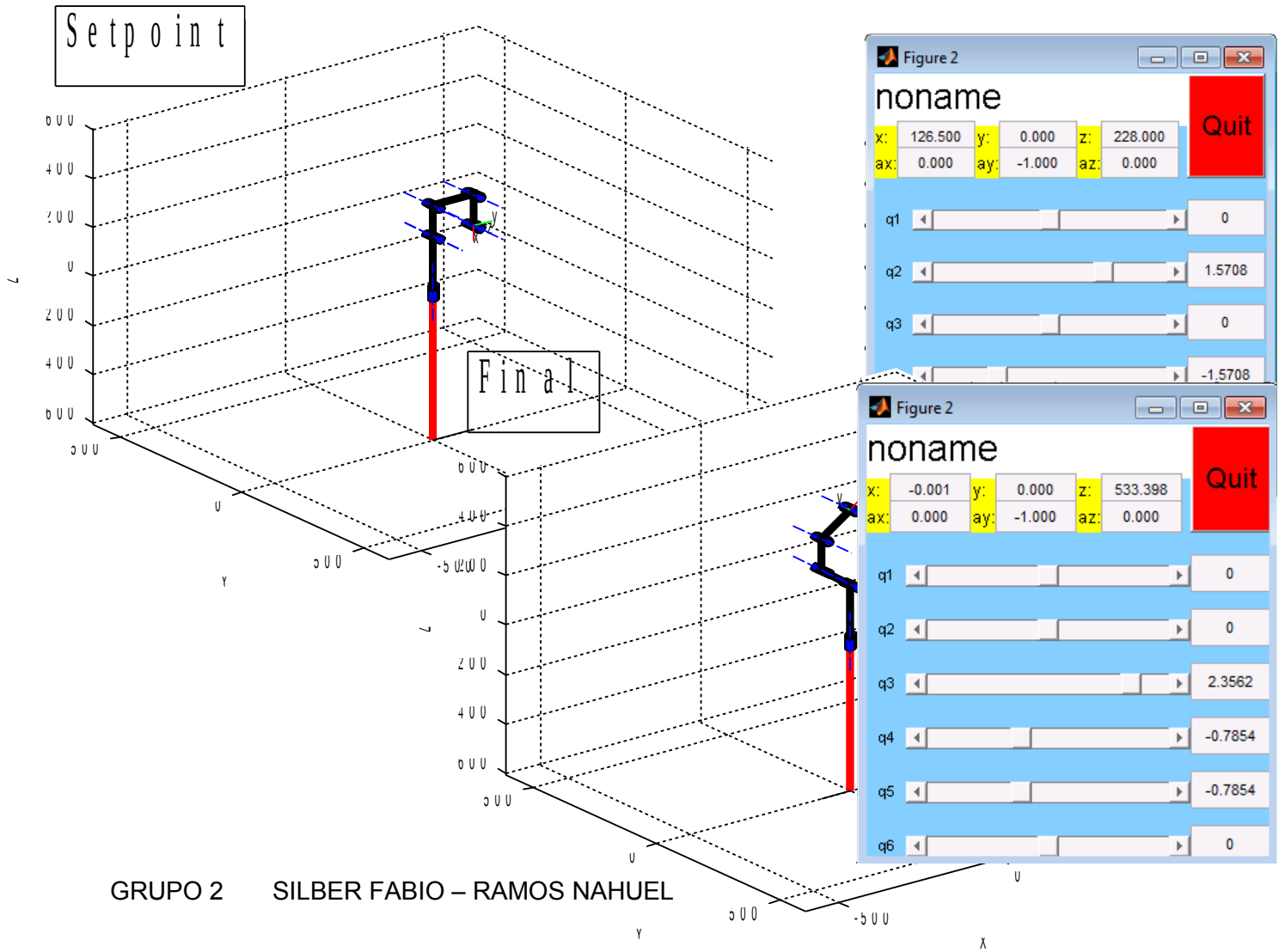


La primera trayectoria se construye moviendo en primer lugar la articulación 2 del robot desde el setpoint inicial hasta 135° (en el gráfico es el primer tramo de la derecha), luego la articulación 3 del robot se desplaza 45° también y por último la cuarta, todas de la misma manera. El robot puede desplazarse de esa manera ya que sus articulaciones lo permiten, de hacerlo en sentido contrario sus eslabones comenzarían a rozarse.



Esta segunda trayectoria cada articulación se desplaza 3° a continuación de la anterior, el punto final es el mismo pero se nota que el recorrido es distinto.

La primera parece ser una trayectoria de menos exigencia para el motor, ya que no tiene constantes picos de variación, como se ve en la segunda.



Los datos comparativos entre Matlab y el DSP los colocamos al final del documento debido a su longitud.

Dinámica del Robot

El análisis dinámico del robot determina la evolución temporal del sistema en relación

con las causas que provocan los cambios de estado físico y/o estado de movimiento. Se necesita conocer los factores que provocan alteraciones o cambios al sistema, para poder cuantificarlos y así poder diseñar ecuaciones de movimiento que describan acordemente al sistema.

Esta relación se obtiene mediante el denominado modelo dinámico, que relaciona matemáticamente:

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

Debido a la complejidad de conocer con exactitud varios de los parámetros necesarios para el cálculo de la dinámica, sumado a que no siempre es posible llegar al conjunto de ecuaciones anteriormente mencionado, el cálculo de éste modelo es de gran dificultad. Agregando que no sólo es necesario conocer los parámetros de los eslabones y motores, sino que se necesita conocer los de cualquier elemento que se desplace junto con el motor, desde los circuitos electrónicos hasta calcular las posibles cargas. Por lo que se deben realizar varios pasos cuidadosamente antes de realizar el estudio

1. Simulación del movimiento del robot.
2. Diseño y evaluación de la estructura mecánica del robot.
3. Dimensionamiento de los actuadores.
4. Diseño y evaluación del control dinámico del robot.

El cálculo dinámico se basa en el planteamiento de ecuaciones del movimiento y su integración. Para problemas extremadamente sencillos se usan las ecuaciones de la mecánica newtoniana directamente auxiliados de las leyes de conservación.

Para el cálculo de la dinámica se deben conocer los siguientes parámetros:

Centro de gravedad

El centro de gravedad de un cuerpo es el punto respecto al cual las fuerzas que la gravedad ejerce sobre los diferentes puntos materiales que constituyen el cuerpo producen un momento resultante nulo.

El centro de gravedad de un cuerpo no corresponde necesariamente a un punto material del cuerpo. Así, el c.g. de una esfera hueca está situado en el centro de la esfera que, obviamente, no pertenece al cuerpo

$$\vec{C}_{CM} = \frac{\int \vec{r} dm}{\int dm} = \frac{\int \vec{r} dm}{M}$$

Ecuación para calcular el punto geométrico del centro de gravedad o centro de masa

Luego se puede aplicar la fórmula de la energía cinética del cuerpo tomando a la velocidad

de desplazamiento en el centro de masas.

Momento Angular

Su importancia se debe a que está relacionada con las simetrías rotacionales de los sistemas físicos.

Bajo ciertas condiciones de simetría rotacional de los sistemas es una magnitud que se mantiene constante con el tiempo a medida que el sistema evoluciona, lo cual da lugar a una ley de conservación conocida como ley de conservación del momento angular. El momento angular para un cuerpo rígido que rota respecto a un eje, es la resistencia que ofrece dicho cuerpo a la variación de la velocidad angular. En el Sistema Internacional de Unidades el momento angularse mide en kg·m²/s.

$$\mathbf{L} = \mathbf{r} \times \mathbf{p} = \mathbf{r} \times m\mathbf{v}$$

Si la simetría es adecuada puede que este valor sea constante lo cual simplifica los cálculos

Momento de inercias

Momento de Inercia es la resistencia que presenta un cuerpo a ser acelerado en rotación

Cuando un cuerpo gira en torno a uno de los ejes principales de inercia, la inercia rotacional puede ser representada como una magnitud escalar llamada momento de inercia. Sin embargo, en el caso más general posible la inercia rotacional debe representarse por medio de un conjunto de momentos de inercia y componentes que forman el llamado tensor de inercia

El momento de inercia refleja la distribución de masa de un cuerpo o de un sistema de partículas en rotación, respecto a un eje de giro. El momento de inercia sólo depende de la geometría del cuerpo y de la posición del eje de giro; pero no depende de las fuerzas que intervienen en el movimiento.

Magnitud escalar

$$I = \int_m r^2 dm = \int_V \rho r^2 \, dV$$

Cálculo de las componentes del tensor de inercias

$$\begin{aligned} I_{xy} = I_{yx} &= \int_M -xy \; dm = \int_V -\rho xy \; d^3\mathbf{r} \\ I_{yz} = I_{zy} &= \int_M -yz \; dm = \int_V -\rho yz \; d^3\mathbf{r} \\ I_{zx} = I_{xz} &= \int_M -zx \; dm = \int_V -\rho zx \; d^3\mathbf{r} \end{aligned}$$

Energía cinética de de rotación (Ω velocidad angular)

$$E_{rot} = \frac{1}{2} \begin{pmatrix} \Omega_x & \Omega_y & \Omega_z \end{pmatrix} \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \begin{pmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{pmatrix} = \frac{1}{2} \sum_j \sum_k I_{jk} \Omega_j \Omega_k$$

Fuerza Centrípeta

Aplicando la ley de Newton a cuerpos en movimiento circular, cuando un objeto puntual

se mueve con una velocidad v en un circulo de radio r , la partícula tiene una

aceleración igual a v2/r , dirigida hacia el centro del circulo, cambie o no la velocidad.

Si la velocidad esta cambiando, entonces hay una componente de la aceleración que es tangencial al círculo y cuyo valor es la variación del cambio de velocidad.

$$\mathbf{a} = -\frac{v^2}{r}\left(\frac{\mathbf{r}}{r}\right) = -\frac{v^2}{r}\hat{\mathbf{u}}_r = -\omega^2\mathbf{r}$$

Fuerza de Coriolis

La fuerza de Coriolis es una fuerza ficticia que aparece cuando un cuerpo está en movimiento con respecto a un sistema en rotación y se describe su movimiento en ese referencial. La fuerza de Coriolis es diferente de la fuerza centrífuga. La fuerza de Coriolis siempre es perpendicular a la dirección del eje de rotación del sistema y a la dirección del movimiento del cuerpo vista desde el sistema en rotación. La fuerza de Coriolis tiene dos componentes:

- una componente tangencial, debido a la componente radial del movimiento del cuerpo, y
- una componente radial, debido a la componente tangencial del movimiento del cuerpo.

El valor de la fuerza de CoriolisFc es:

$$\vec{F}_c = 2m\left(\vec{v} \times \vec{\omega}\right),$$

m es la masa del cuerpo, v es la velocidad del cuerpo en el sistema en rotación,ω es la velocidad angular del sistema en rotación vista desde un sistema inercial.

Todos los parámetros mencionados son necesarios para poder calcular el torque ejercido por cada eslabón

Ecuación del torque

$$\tau = \mathbf{D}\ddot{\mathbf{q}} + \mathbf{H} + \mathbf{C}$$

Para resolver esta ecuación hay varios métodos:

- Lagrange-Euler
- Newton-Euler
- Variables de Estado
- Kane

Para resolver la ecuación del torque nosotros utilizamos las herramientas del toolbooks de Peter Croke. En los instructivos de éste figuran varios métodos para resolver la dinámica directa, utilizan un método llamado RNE (Recursive Newton-Euler).

Antes de poder aplicar los algoritmos necesitamos los siguientes datos de cada eslabón del robot:

- Masa
- Centro de Masa o Gravedad
- Inercias
- Gravedad

Para ello utilizamos un programa de dibujo CAD, llamado T-FLEX el cual ofrece la posibilidad de dibujar elementos en 3D. Además tiene características muy importantes ya que se pueden calcular los parámetros esenciales de cada pieza (centro de masa, inercias, peso, etc), gracias a que se puede especificar también el material con la que están hechas.

El software tiene 2 posibilidades para graficar en 3-D, la primera es dibujar sobre un plano y luego mediante instrucciones expandir el dibujo a uno de 3-D. La otra es directamente hacer un dibujo virtual de un espacio con los 3 planos ortogonales dibujados.

El robot M5 consta de 6 piezas esenciales:

- 1 base
- 1° eslabón
- 3 eslabones siguientes (son distintos al primero)
- 1 gripper

Nosotros no tuvimos en cuenta el gripper para los cálculos del robot por lo que nos quedan 5 piezas, de las cuales dibujar a la perfección es muy difícil y trabajoso. Como ejemplo los eslabones no son una única pieza sino que constan de 4 piezas, 1 por cada lado y son huecos. Al querer dibujar algo similar no era posible calcular los parámetros esenciales

Además debemos mencionar que para realizar las piezas nos basamos en el trabajo de un compañero llamado Daniel Abaca, que nos facilitó sus piezas y nosotros hicimos pequeños cambios, ya que nuestras eslabones son todos sólidas. Esto provoca que la dinámica no sea perfecta, sumado que la base también es un bloque sólido y, en la realidad para conocer la masa del elemento que rota en la base habría que desarmar esa parte del robot.

Piezas hechas en T-FLEX

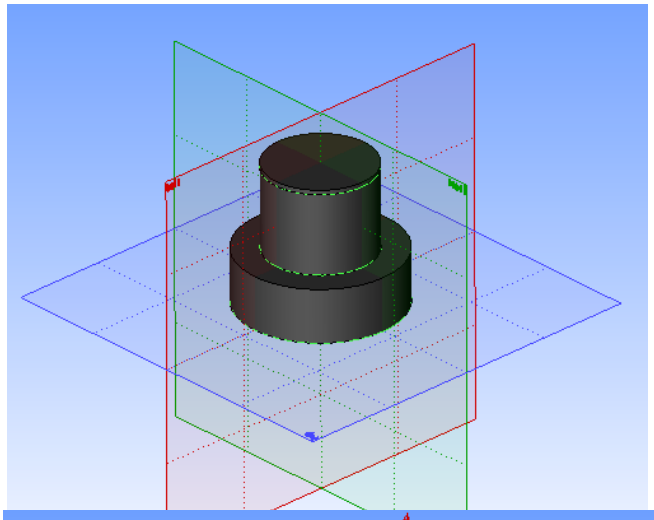


Imagen de la base del robot

Los datos de esta pieza son los siguientes:

Mass=1.22145kg

Xmass=0

Ymass=0

Zmass=105mm

Inerciax= 15390,3

Inerciay= 15390,3

Inerciaz= 2198,61

Inercia-yz=0

Inercia-zx=0

Inercia-xy=0

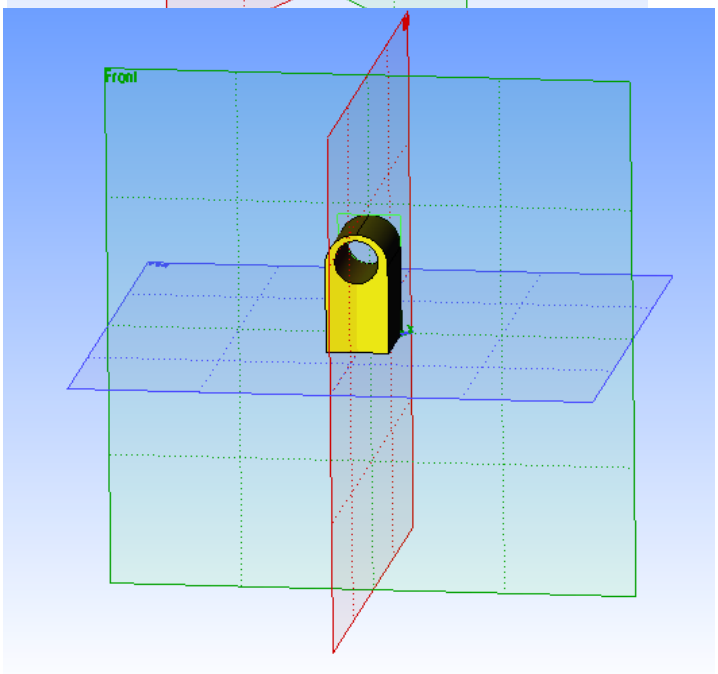


Imagen del primer eslabón del robot

Los datos de esta pieza son los siguientes:

Mass=0,2891kg

Xmass=0

Ymass= 33,75 mm

Zmass= 37,89mm

Inerciax=1026,5

Inerciay=653,73

Inerciaz=505,49

Inercia-yz=0

Inercia-zx=0

Inercia-xy=0

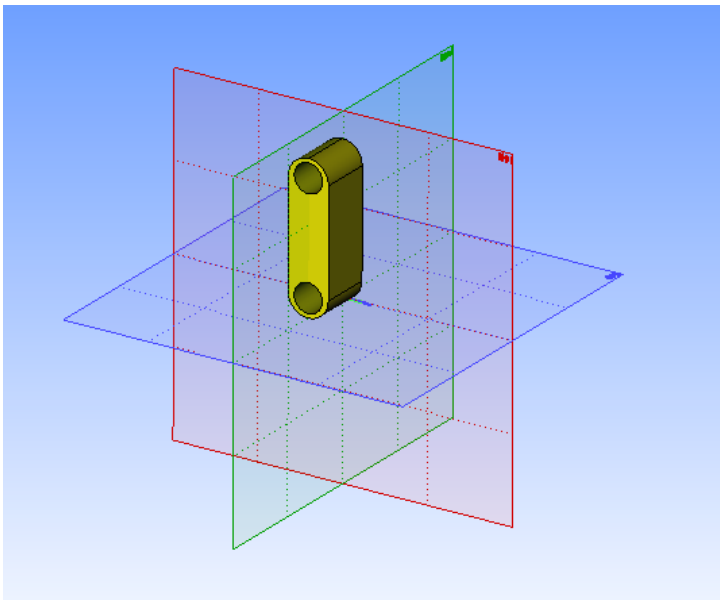


Imagen de los eslabones 2 3 y 4

Los datos de esta pieza son los siguientes:

Mass=0,5022kg

Xmass=-34,5 mm

Ymass= -0,0088 mm

Zmass= 90

Inerciax=5071,27

Inerciay=5724,21

Inerciaz=885,433

Inercia-yz=0

Inercia-zx=0

Inercia-xy=0,3

Script de Matlab

```
% Robot5_6 define los parametros del robot en cuestión.
```

```
clearALL;
clc;
symst

% ============================================================
%                          Datos del Robot
% ============================================================

%Datos de las longitudes de los eslabones [m]

b1=(96+59+66+7)/1000;   %Distancia entre el suelo y el primer
motor
b2=(112.5+14)/1000;
b3=(112.5+14)/1000;
b4=(112.5+14)/1000;

% Parametros DH.
% L =LINK([alpha A theta D sigma])

L{1}=link([pi/2 0 0 b1],'standard');    %desde la base al primer
muñón
%tras en z
%giro en x
L{2}=link([0 0 pi/2 0],'standard');     %rota en z (es un engaño
no se toca)
L{3}=link([0 b2 0 0],'standard');       %traslado en x
(articulación giro en z)
L{4}=link([0 b3 -pi/2 0],'standard');   %giro en z para correr x
y
L{5}=link([0 b4 -pi/2 0],'standard');
L{6}=link([0 0 pi/2 0],'standard');

% Masas de los eslabones [kg].

L{1}.m = 1.22145;
L{2}.m = 0;
L{3}.m = 0.2891;
L{4}.m = 0.5;
```

```
L{5}.m = 0.5;
L{6}.m = 0;

% Coordenadas de los centros de masa de cada eslabon [m].

L{1}.r = [0 0 0.105];
L{2}.r = [0 0 0];
L{3}.r = [0 0.03375 0.0379];
L{4}.r = [-0.0345 0 0.09];
L{5}.r = [-0.0345 0 0.09];
L{6}.r = [0 0 0];

% Tensores de inercia (Ixx, Iyy, Izz, Ixy, Iyz, Ixz) [Kg x
(m)^2].

L{1}.I = [0 0 0 0 0 0];
L{2}.I = [0 0 0 0 0 0];
L{3}.I = [0 0 0 0 0 0];
L{4}.I = [0 0 0 0.3 0 0];
L{5}.I = [0 0 0 0.3 0 0];
L{6}.I = [0 0 0 0 0 0];

% Inercia del motor que acciona al elemento.

L{1}.Jm = 0;
L{2}.Jm = 0;
L{3}.Jm = 0;
L{4}.Jm = 0;
L{5}.Jm = 0;
L{6}.Jm = 0;

% Coeficiente de reduccion del actuador del elemento.

L{1}.G = 1;
L{2}.G = 1;
L{3}.G = 1;
L{4}.G = 1;
L{5}.G = 1;
```

```
L{6}.G = 1;

% Rozamiento viscoso del actuador del elemento.

L{1}.B = 0;
L{2}.B = 0;
L{3}.B = 0;
L{4}.B = 0;
L{5}.B = 0;
L{6}.B = 0;

% Rozamiento de Coulomb en la direccion positiva y negativa del
actuador
% del elemento.

L{1}.Tc = [ 0 0 ];
L{2}.Tc = [ 0 0 ];
L{3}.Tc = [ 0 0 ];
L{4}.Tc = [ 0 0 ];
L{5}.Tc = [ 0 0 ];
L{6}.Tc = [ 0 0 ];



% Definicion del robot.
% ROBOT Robot object constructor.
% function r = robot(L, a1, a2, a3).

Robot5_6 = robot(L, 'Polar RD', 'UTN_Robotics', 'robot
cilindrico 2 GLD');
Robot5_6.name = 'Polar RD';
Robot5_6.manuf = 'UTN_Robotics';

% Dibuja torques y fuerzas para una determinada trayectoria
utilizando la
% funcion "rne".

% Se define la gravedad con respecto al sistema {S0}.

grav = [ 0 0 -9.8];
```

```
% ================================================================
%                Determina la trayectoria
% ================================================================

% Definicionsimbolica de la posicion.

for i=1:6
sq(i,1)= sin(t) * t + pi/2;
end

% Obtención simbólica de la velocidad y aceleración.

for i=1:6
sqd(i,1) = diff(sq(i,1),'t');
sqdd(i,1) = diff(sqd(i,1), 't');
end


% ================================================================
% Evaluacionnumerica de posicion, velocidad, aceleracion y XY.
% ================================================================

fortk = 1:1:50

t = (tk - 1)/10;

for i=1:6

q(tk,i) = eval(sq(i,1));
qd(tk,i) = eval(sqd(i,1));
qdd(tk,i) = eval(sqdd(i,1));
XY(tk,i) = q(tk,i) * cos(q(tk,i));

end

end
```

```
% ================================================================
%                          Dibuja la trayectoria.
% ================================================================

figure(7);

subplot(3,1,1);
plot(q);
title('Posicion');
grid;
subplot(3,1,2);
plot(qd);
title('Velocidad');
subplot(3,1,3);
grid;
plot(qdd);
title('Aceleración');
grid;



% ================================================================
%              Calcula matriz gravedad, coriolis, torque
% ================================================================

Tau = rne(Robot5_6, q, qd, qdd, grav);
TauG = GRAVLOAD(Robot5_6, q, grav);
TauC = CORIOLIS(Robot5_6, q, qd);
TauI = ITORQUE(Robot5_6, q, qdd);

% ================================================================
                 Gráficagravedad, coriolis, torque
%================================================================
for i=1:2:5
for i=1:2:5
figure(i)
subplot(4,1,1);
plot(Tau(:,i));
title('Torque');
```

```
subplot(4,1,2);
plot(TauI(:,i));
title('Inercia');
subplot(4,1,3);
plot(TauC(:,i));
title('Coriolis');
subplot(4,1,4);
plot(TauG(:,i));
title('PAR Gravedad');
end
```

Gráficos Temporales

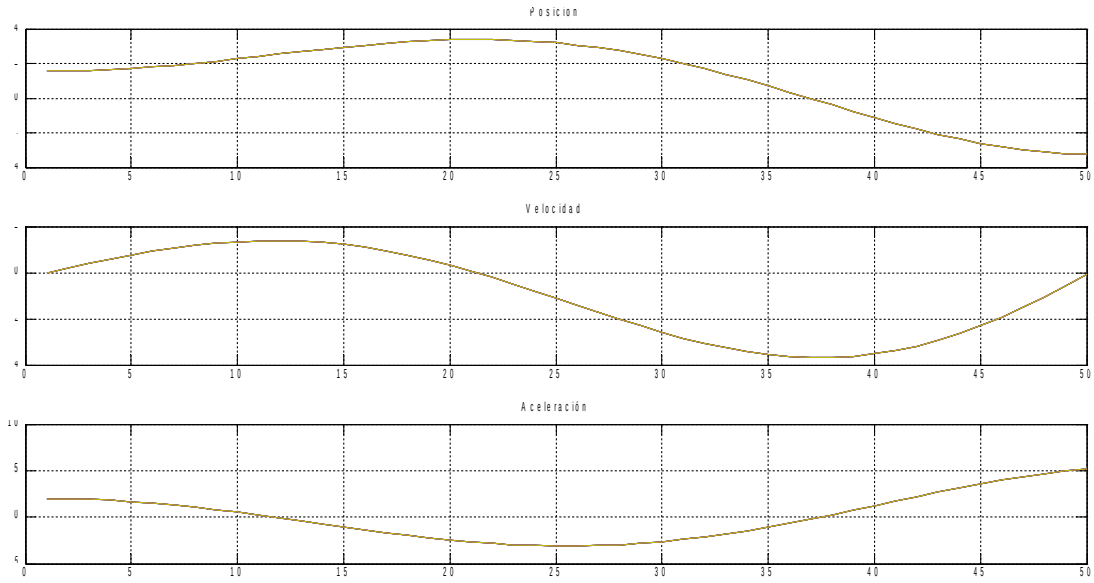
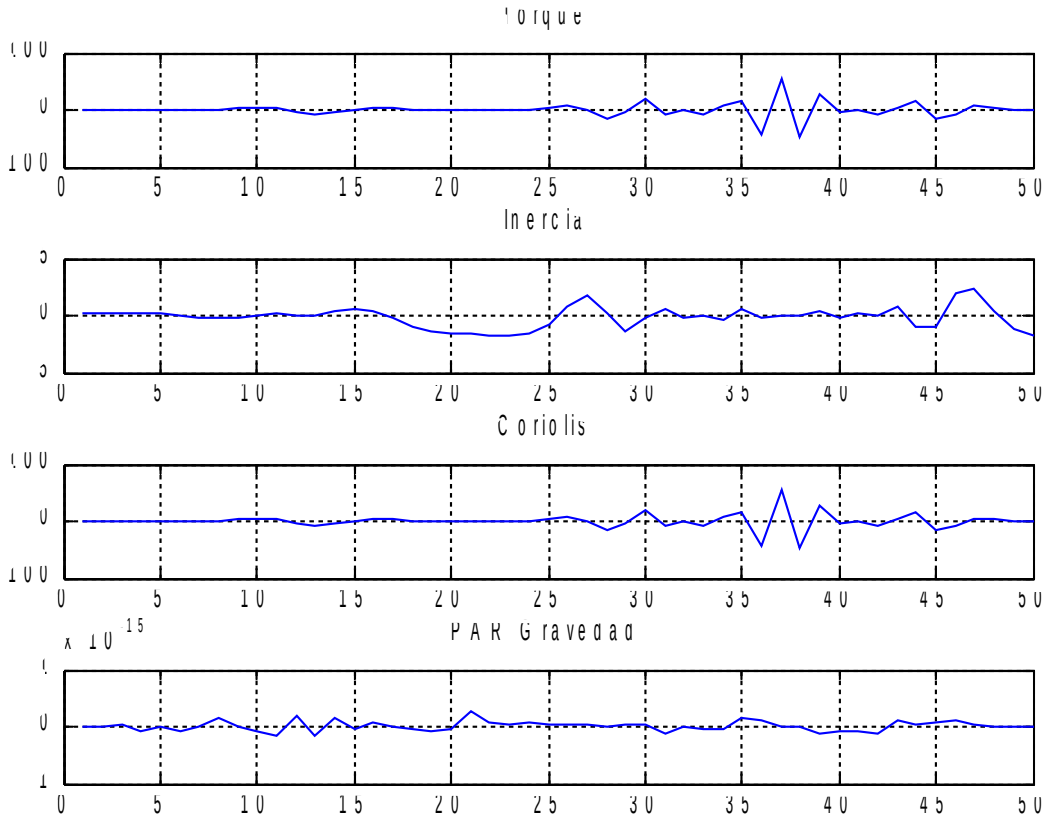
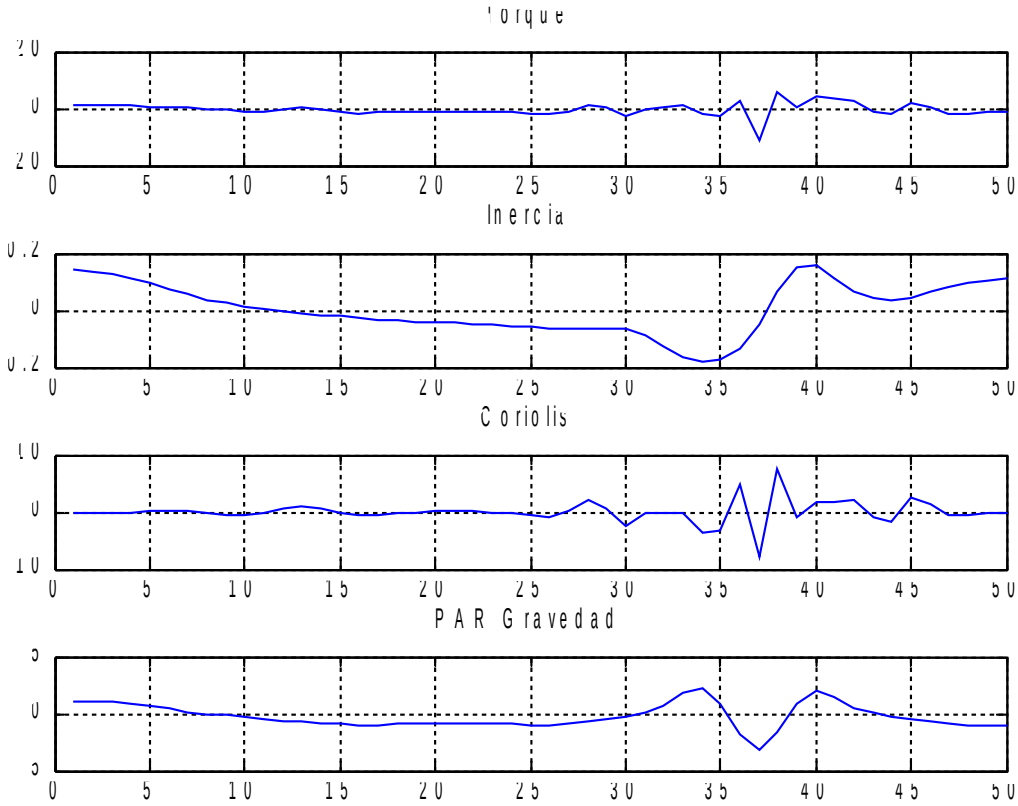


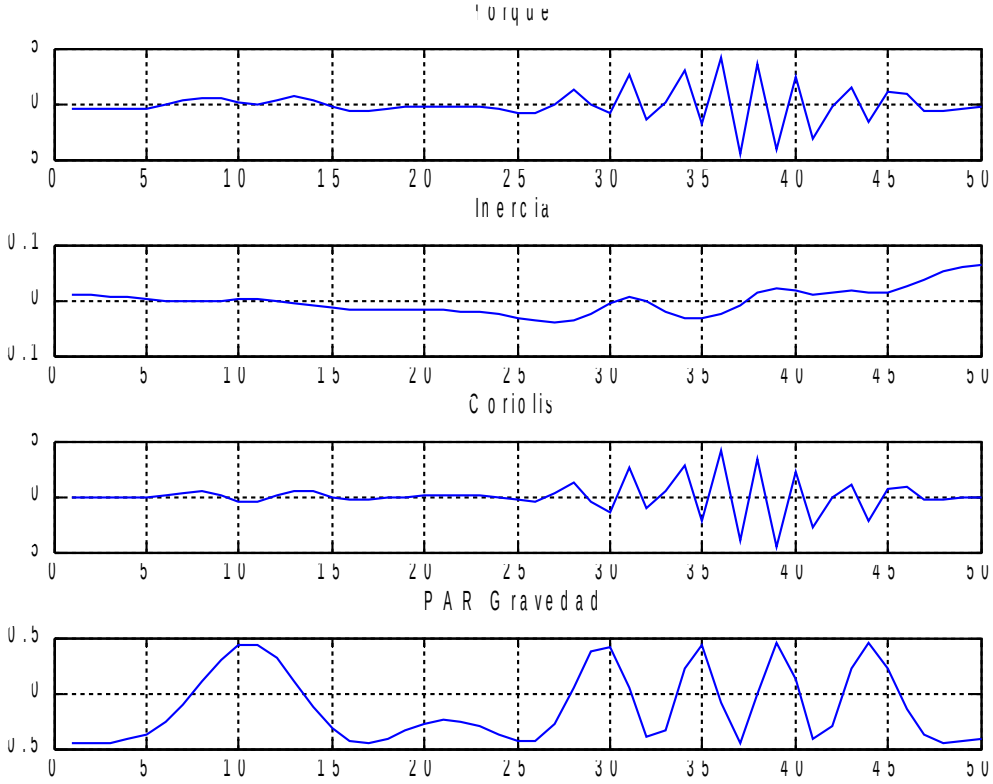
Gráfico de posición, velocidad y aceleración. Todas los eslabones suponemos la misma trayectoria



Gráficos temporales de la base del robot



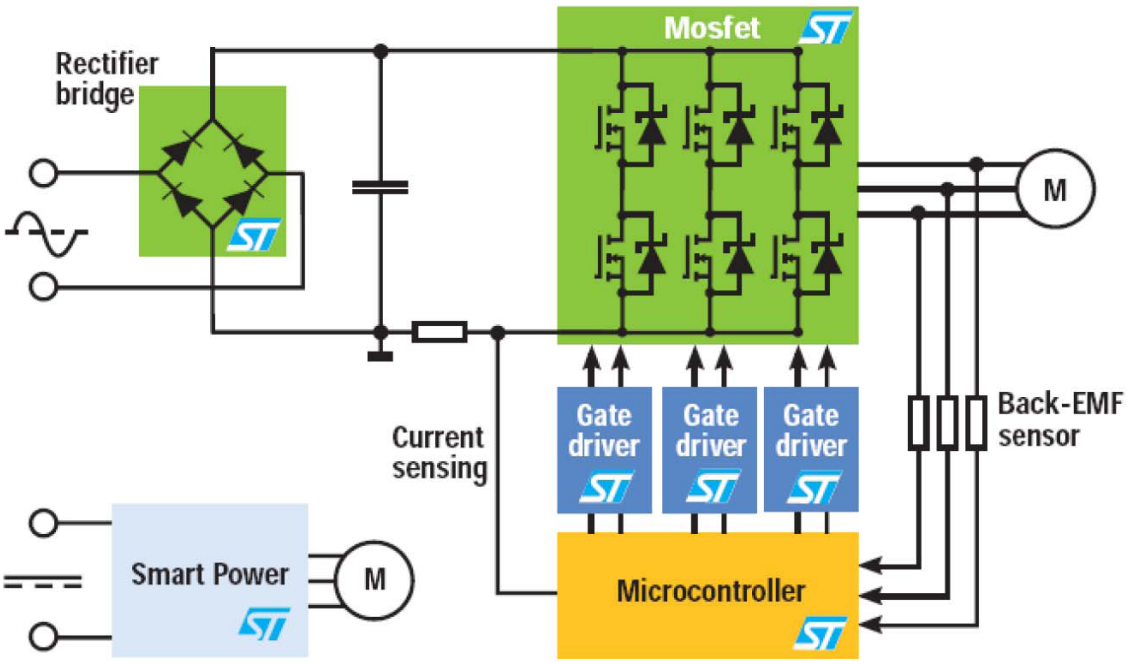
Gráficos temporales del primer eslabón



Gráficos temporales de los demás eslabones

Implementación de VHDL para el control del motor de CC

Circuito de Control:



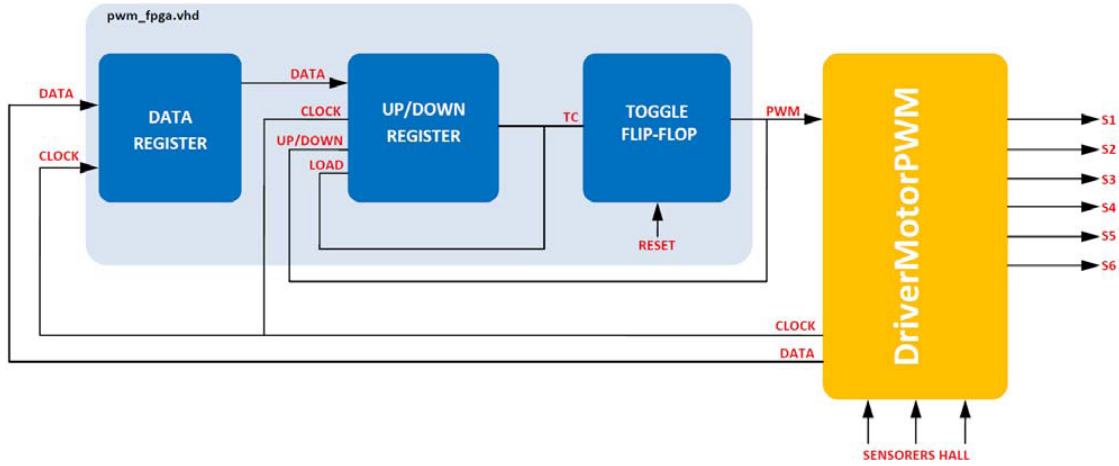


Tabla de estados:

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

Código en VHDL para la implementación del driver:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

entitydriver_motor is
    Port (
porcentaje_pwm : in std_logic_vector (7 downto 0);   -- 256 niveles de pwm
sensores_hall: in std_logic_vector (2 downto 0);   -- 3 sensores Hall
reset : in std_logic;
sentido_de_giro: in std_logic;
                        S1: out std_logic;  -- 6 transistores
                        S2: out std_logic;
                        S3: out std_logic;
                        S4: out std_logic;
                        S5: out std_logic;
                        S6: out std_logic
        );
enddriver_motor;

ARCHITECTURE sintesis OF driver_motor IS
  SIGNAL cuenta              : unsigned(7 DOWNTO 0);
  SIGNAL cuenta_final     : std_logic;
  SIGNAL suma_resta      : std_logic;
  SIGNAL salida_pwm     : std_logic;

  SIGNAL    clock_cont  :std_logic;   -- Variables/SeñalesReloj
```

```
constant  periodo_clk : TIME :=390ns;  -- Tclk*contador = 390nS * 2^(8)= 100uS =>
f=10kHz


BEGIN

   PROCESS            -- Genero el clock interno

   BEGIN

clock_cont<= '1';

wait for periodo_clk/2;

clock_cont<= '0';

wait for periodo_clk/2;

   END PROCESS;


   PROCESS (clock_cont,reset,salida_pwm,cuenta_final)  -- Contadorbinario de 8bits
con up/down

   BEGIN

     IF (reset='1') OR (cuenta_final='0') THEN

cuenta<=unsigned(porcentaje_pwm);

cuenta_final<='1';

ELSIF  (clock_cont'EVENT) AND (clock_cont = '1') THEN

        IF (salida_pwm='0') THEN

          IF (cuenta = "11111111") THEN

cuenta<=unsigned(porcentaje_pwm);

cuenta_final<='0';

          ELSE

cuenta_final<='1';

cuenta<=cuenta+1;

END IF;

        ELSE

          IF (cuenta = "00000000") THEN
```

```
cuenta<=unsigned(porcentaje_pwm);
cuenta_final<='0';
          ELSE
cuenta_final<='1';
cuenta<=cuenta-1;
END IF;
END IF;
     END IF;
   END PROCESS;

   PROCESS (cuenta_final,reset) -- Flip-flop Toggle
   BEGIN
     IF (reset='1') THEN
salida_pwm<='0';
ELSIF  (cuenta_final'EVENT) AND (cuenta_final = '0') THEN
salida_pwm<=notsalida_pwm;
     END IF;
   END PROCESS;

   PROCESS (sensores_hall,sentido_de_giro,salida_pwm) -- Manejo de las salida
BEGIN
     IF (sentido_de_giro='0') THEN
       CASE sensores_hall IS
         WHEN "000" =>
           S1<='1'; S2<='0'; S3<='0'; S4<=salida_pwm; S5<='0'; S6<='0';
         WHEN "001" =>
           S1<='1'; S2<='0'; S3<='0'; S4<='0'; S5<='0'; S6<=salida_pwm;
         WHEN "010" =>
           S1<='0'; S2<='0'; S3<='1'; S4<='0'; S5<='0'; S6<=salida_pwm;
```

```
        WHEN "011" =>
          S1<='0'; S2<=salida_pwm; S3<='1'; S4<='0'; S5<='0'; S6<='0';
        WHEN "100" =>
          S1<='0'; S2<=salida_pwm; S3<='0'; S4<='0'; S5<='1'; S6<='0';
        WHEN "101" =>
          S1<='0'; S2<=salida_pwm; S3<='0'; S4<='0'; S5<='1'; S6<='0';
        WHEN OTHERS =>
          S1<='0'; S2<='0'; S3<='0'; S4<='0'; S5<='0'; S6<='0';
        END CASE;
      ELSE
       CASE sensores_hall IS
        WHEN "101" =>
          S1<='1'; S2<='0'; S3<='0'; S4<=salida_pwm; S5<='0'; S6<='0';
        WHEN "100" =>
          S1<='1'; S2<='0'; S3<='0'; S4<='0'; S5<='0'; S6<=salida_pwm;
        WHEN "011" =>
          S1<='0'; S2<='0'; S3<='1'; S4<='0'; S5<='0'; S6<=salida_pwm;
        WHEN "010" =>
          S1<='0'; S2<=salida_pwm; S3<='1'; S4<='0'; S5<='0'; S6<='0';
        WHEN "001" =>
          S1<='0'; S2<=salida_pwm; S3<='0'; S4<='0'; S5<='1'; S6<='0';
        WHEN "000" =>
          S1<='0'; S2<=salida_pwm; S3<='0'; S4<='0'; S5<='1'; S6<='0';
        WHEN OTHERS =>
          S1<='0'; S2<='0'; S3<='0'; S4<='0'; S5<='0'; S6<='0';
        END CASE;
      END IF;
    END PROCESS
END sintesis;
```

Código en VHDL para la simulación del driver:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

entitydriver_motor_simulado is
    Port (
--porcentaje_pwm : in std_logic_vector (7 downto 0);   -- 256 niveles de pwm
--sensores_hall: in std_logic_vector (2 downto 0);   -- 3 sensores Hall
                    -- reset : in std_logic;
                            S1: out std_logic;  -- 6 transistores
                            S2: out std_logic;
                            S3: out std_logic;
                            S4: out std_logic;
                            S5: out std_logic;
                            S6: out std_logic
        );
enddriver_motor_simulado;


ARCHITECTURE sintesis OF driver_motor_simulado IS

  SIGNAL reset                  : std_logic:='0';
SIGNAL porcentaje_pwm :std_logic_vector (7 downto 0);
  SIGNAL cuenta                 : unsigned(7 DOWNTO 0);
  SIGNAL cuenta_final        : std_logic;
```

```
SIGNAL suma_resta          : std_logic;
SIGNAL salida_pwm          : std_logic;
SIGNAL sensores_hall       : std_logic_vector (2 downto 0);
SIGNAL sentido_de_giro :std_logic:='0';

SIGNAL    clock_cont  :std_logic;   -- Variables/SeñalesReloj
constant  periodo_clk : TIME :=390ns;  -- Tclk*contador = 390nS * 2^(8)= 100uS =>
f=10kHz

constant  periodo_por_sensor_hall: TIME :=200us;

BEGIN

PROCESS           -- Reset inicial y luego un cambio de giro después de 5mSeg
BEGIN
wait for 5ns;
reset<='1';
wait for 20ns;
reset<='0';
waitfor 5ms;       -- Espera 3mSeg. para realizar un cambio de sentido de giro en el
motor
sentido_de_giro<='1';
waitfor 5ms;    -- Espera 10mSeg. para realizar otro reset
END PROCESS;

   PROCESS           -- Genero el clock interno
   BEGIN
clock_cont<= '1';
wait for periodo_clk/2;
```

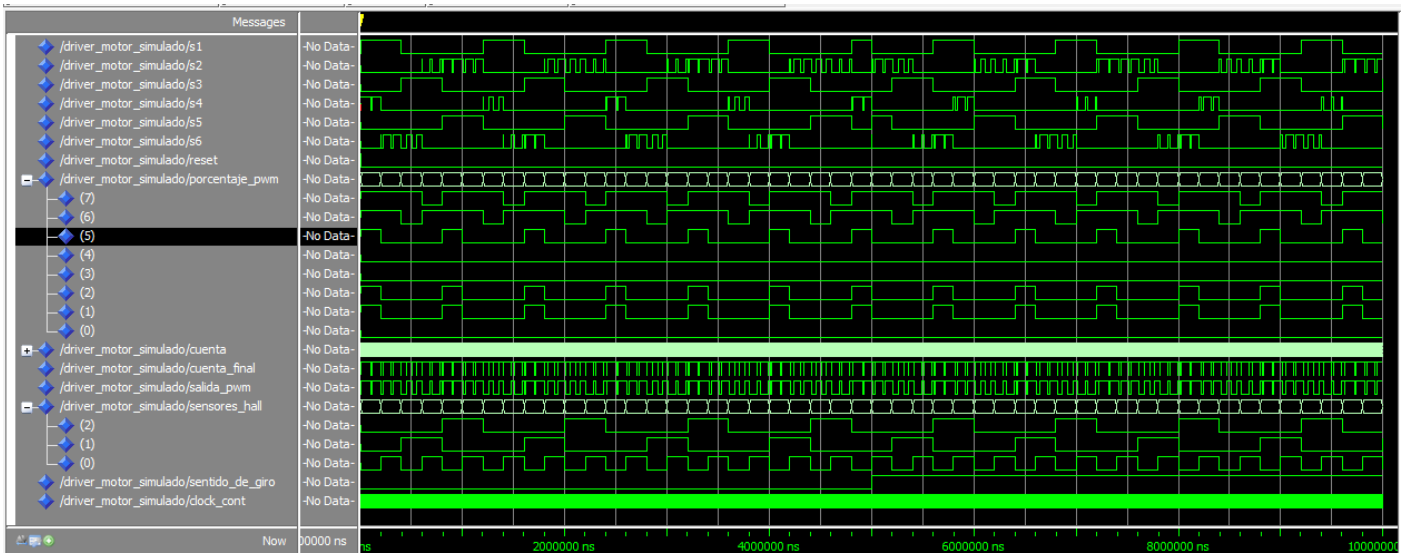
```
clock_cont<= '0';
wait for periodo_clk/2;
END PROCESS;

PROCESS            -- Variación del porcentaje de Pwm en el tiempo
   BEGIN
porcentaje_pwm<="11100110"; -- Duty al 90%
waitforperiodo_por_sensor_hall;
porcentaje_pwm<="11000000"; -- Duty al 75%
waitforperiodo_por_sensor_hall;
porcentaje_pwm<="10000000"; -- Duty al 50%
waitforperiodo_por_sensor_hall;
porcentaje_pwm<="01000000"; -- Duty al 25%
waitforperiodo_por_sensor_hall;
porcentaje_pwm<="00011001"; -- Duty al 10%
END PROCESS;

   PROCESS (clock_cont,reset,salida_pwm,cuenta_final)  -- Contadorbinario de 8bits
con up/down
   BEGIN
     IF (reset='1') OR (cuenta_final='0') THEN
cuenta<=unsigned(porcentaje_pwm);
cuenta_final<='1';
ELSIF  (clock_cont'EVENT) AND (clock_cont = '1') THEN
        IF (salida_pwm='0') THEN
          IF (cuenta = "11111111") THEN
cuenta<=unsigned(porcentaje_pwm);
cuenta_final<='0';
          ELSE
```

```
cuenta_final<='1';
cuenta<=cuenta+1;
END IF;
        ELSE
          IF (cuenta = "00000000") THEN
cuenta<=unsigned(porcentaje_pwm);
cuenta_final<='0';
          ELSE
cuenta_final<='1';
cuenta<=cuenta-1;
END IF;
END IF;
     END IF;
   END PROCESS;



   PROCESS (cuenta_final,reset) -- Flip-flop Toggle
   BEGIN
     IF (reset='1') THEN
salida_pwm<='0';
ELSIF  (cuenta_final'EVENT) AND (cuenta_final = '0') THEN
salida_pwm<=notsalida_pwm;
     END IF;
   END PROCESS;

   PROCESS (sensores_hall,sentido_de_giro,salida_pwm) -- Manejo de las salida
BEGIN
     IF (sentido_de_giro='0') THEN
       CASE sensores_hall IS
```

```
    WHEN "000" =>
      S1<='1'; S2<='0'; S3<='0'; S4<=salida_pwm; S5<='0'; S6<='0';
    WHEN "001" =>
      S1<='1'; S2<='0'; S3<='0'; S4<='0'; S5<='0'; S6<=salida_pwm;
    WHEN "010" =>
      S1<='0'; S2<='0'; S3<='1'; S4<='0'; S5<='0'; S6<=salida_pwm;
    WHEN "011" =>
      S1<='0'; S2<=salida_pwm; S3<='1'; S4<='0'; S5<='0'; S6<='0';
    WHEN "100" =>
      S1<='0'; S2<=salida_pwm; S3<='0'; S4<='0'; S5<='1'; S6<='0';
    WHEN "101" =>
      S1<='0'; S2<=salida_pwm; S3<='0'; S4<='0'; S5<='1'; S6<='0';
    WHEN OTHERS =>
      S1<='0'; S2<='0'; S3<='0'; S4<='0'; S5<='0'; S6<='0';
    END CASE;
  ELSE
    CASE sensores_hall IS
    WHEN "101" =>
      S1<='1'; S2<='0'; S3<='0'; S4<=salida_pwm; S5<='0'; S6<='0';
    WHEN "100" =>
      S1<='1'; S2<='0'; S3<='0'; S4<='0'; S5<='0'; S6<=salida_pwm;
    WHEN "011" =>
      S1<='0'; S2<='0'; S3<='1'; S4<='0'; S5<='0'; S6<=salida_pwm;
    WHEN "010" =>
      S1<='0'; S2<=salida_pwm; S3<='1'; S4<='0'; S5<='0'; S6<='0';
    WHEN "001" =>
      S1<='0'; S2<=salida_pwm; S3<='0'; S4<='0'; S5<='1'; S6<='0';
    WHEN "000" =>
      S1<='0'; S2<=salida_pwm; S3<='0'; S4<='0'; S5<='1'; S6<='0';
```

```
        WHEN OTHERS =>
          S1<='0'; S2<='0'; S3<='0'; S4<='0'; S5<='0'; S6<='0';
        END CASE;
      END IF;
    END PROCESS

PROCESS      -- Simulacion de los sensores hall
   BEGIN
sensores_hall<="000";
waitforperiodo_por_sensor_hall;
sensores_hall<="001";
waitforperiodo_por_sensor_hall;
sensores_hall<="010";
waitforperiodo_por_sensor_hall;
sensores_hall<="011";
waitforperiodo_por_sensor_hall;
sensores_hall<="100";
waitforperiodo_por_sensor_hall;
sensores_hall<="101";
waitforperiodo_por_sensor_hall;
   END PROCESS;

END sintesis;
```

Resultado de la simulación en Modelsim:



Compilador M5

A la hora de utilizar el robot, es necesario programarlo para poder operarlo. Para facilitar su uso, se implemento un lenguaje de alto nivel que permita al operador tener una curva de aprendizaje mucho menor comparado con el que se obtendría con otros lenguajes de programación más complejos como C, C++, Java, .Net, etc que requieren de un mayor tiempo de aprendizaje. El lenguaje implementado, busca ofrecer simplicidad al operario, por lo cual las instrucciones implementadas son de fácil entendimiento y a la vez permiten un cierto grado de libertad.

Limitaciones

Sin embargo, dicho lenguaje carece de instrucciones importantes para que el operario tenga un control total, como por ejemplo instrucciones condicionales y repetitivas, las cuales suelen ser comunes en la mayoría de los lenguajes. Por tal motivo, se puede decir que el lenguaje M5 todavía se encuentra incompleto y en fase de desarrollo.

Por otro lado, las instrucciones básicas de movimiento están pensadas para su uso en cinemática directa (a partir de los distintos valores dados a las articulaciones, se calcula el punto final), eso se debe a que el análisis realizado durante toda esta tesis se baso en ese tipo de cinemática. Esto trae como consecuencia, la carencia de utilidad de dichas instrucciones

para el operario, debido a que su interés radica en poder realizar movimientos en base a la cinemática indirecta (a partir del punto final, calcular los distintos valores que deberán tomar las articulaciones).

ANTLR

Para el desarrollo del lenguaje M5, se utilizó el programa ANTLR, el cual nos ofrece varias herramientas a la hora de crear e implementar instrucciones en nuestro lenguaje, como así también nos brinda diversas verificaciones que permiten comprobar el correcto funcionamiento del lenguaje desarrollado.

El ANTRL es libre y está hecho en lenguaje Java, pude ser descargado de su página oficial (http://www.antlr.org/). Es capaz de generar un analizador léxico, sintáctico o semántico en varios lenguajes (java, C++ y C# en su versión 2.7.2) a partir de unos ficheros escritos en un lenguaje propio. Dicho lenguaje es básicamente una serie de reglas EBNF y un conjunto de construcciones auxiliares.

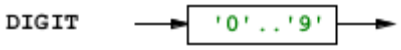
```
grammar SimpleCalc;

tokens {
            PLUS     = '+' ;
            MINUS    = '-' ;
            MULT     = '*' ;
            DIV      = '/' ;
}

@members {
   public static void main(String[] args) throws Exception {
      SimpleCalcLexer lex = new SimpleCalcLexer(new ANTLRFileStream(args[0]));
            CommonTokenStream tokens = new CommonTokenStream(lex);

      SimpleCalcParser parser = new SimpleCalcParser(tokens);

      try {
         parser.expr();
      } catch (RecognitionException e)  {
         e.printStackTrace();
      }
   }
}

/*------------------------------------------------------------------
 * PARSER RULES
 *------------------------------------------------------------------*/

expr       : (asig+) ;

asig       : ID '=' op {System.out.println($ID.text +"="+ $op.value); variables.put($ID.text, new
Integer($op.value));};

op         returns [int value]
   :  e=factor {$value = $e.value;}
      (   PLUS e=factor {$value += $e.value;}
      |  MINUS e=factor {$value -= $e.value;}
      |  MULT e=factor {$value = $e.value*$value;}
            |  DIV e=factor {$value = $value/$e.value;}
      )*
   ;


factor     returns [int value]
   :  NUMBER {$value = Integer.parseInt($NUMBER.text);}
   |  ID
            {
      int v = (Integer)variables.get($ID.text);
      if ( v!=null ) $value = v.doubleValue();
      else System.err.println("Variable no definida: "+$ID.text);
```

```
/*------------------------------------------------------------------
 * LEXER RULES
 *------------------------------------------------------------------*/
ID  :   ('a'..'z'|'A'..'Z')+ ;

NUMBER              : (DIGIT)+ ;

WHITESPACE : ( '\t' | ' ' | '\r' | '\n'| '\u000C' )+      { $channel = HIDDEN; } ;
```

Declaraciones realizadas:

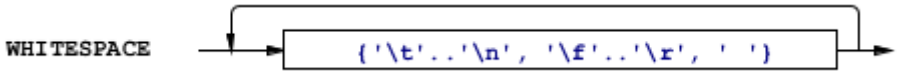
- Los “tokens” declarados son la suma (+), la resta (-), la multiplicación (*) y la división (/); esto implica que cuando en nuestro código usemos dichos caracteres, el compilador sabrá que habrá una cierta acción asociado a ese carácter.

- DIGIT, el compilador reconocerá un dígito cuando este se componga de números del 0 al 9. Diagrama de interpretación de un dígito:



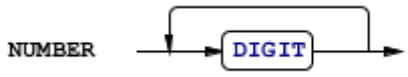
Si en nuestro código ingresamos “5”, el compilador lo reconocerá como un dígito.

- WHITESPACE, el compilador reconocerá un espacio en blanco cuando se usen los caracteres especiales como “Space”, “Tab”, etc. Diagrama de interpretación de un espacio en blanco:



Se puede ver una especie de realimentación en el diagrama, eso significa que el espacio en blanco puede repetirse más de una vez, con lo cual si tendríamos dos “Space”, el compilador reconocería a los dos caracteres como espacio en blanco.

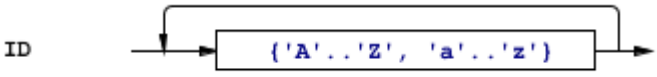
- NUMBER, el compilador reconocerá un número cuando el mismo este compuesto de uno o más dígitos. Diagrama de interpretación de un número:



Nuevamente se puede ver la realimentación, un número será un conjunto “n” de dígitos.

Si en nuestro código ingresamos “563”, el compilador reconocerá a 563 como un único número.

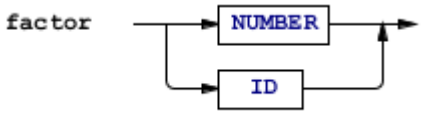
- ID, el compilador reconocerá una variable al momento de usar una letra (mayúscula o minúscula). Diagrama de interpretación de una variable:



La realimentación permitirá reconocer variables compuestas con más de un carácter.

Si en nuestro código ingresamos “var”, el compilador reconocerá var como una variable.

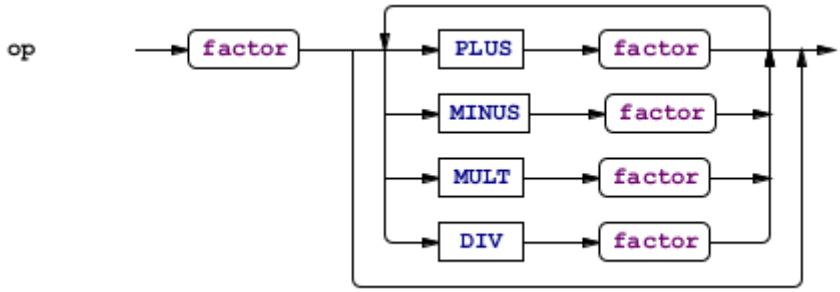
- Factor, declaración en la que ya se puede observar código en java. El código en Java se encuentra entre las llaves “{”, ”}”. La acción del compilador será la de devolver el valor entero (int) de una variable si se encuentra definida o la de devolver el valor de un número. Para lo cual, se encarga de convertir el texto de nuestro lenguaje a código en Java. Diagrama de interpretación de un factor:



Se pueden ver dos ramas, o se trata de un número o de una variable. En función de que rama sea, se ejecuta un código Java distinto.

Si en nuestro código ingresamos “12”, el compilador interpretara 12 como un número y devolverá su valor en código Java.

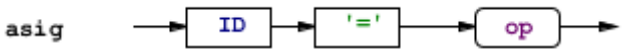
- OP, declaración que nos permite realizar las operaciones básicas (suma, resta, multiplicación y división) y devuelve el resultado entero (int) de la operación. Diagrama de interpretación de una operación:



Para realizar la operación, como mínimo se requieren dos factores (recordemos que los factores podían ser números o variables asignadas). Sin embargo también está previsto el caso en donde la operación sea un solo factor, es decir no hay operación alguna.

Si en nuestro código ingresamos “a+12-2”, el compilador interpretara como “a” el primer factor, “+” como PLUS, “12” como el segundo factor, y al haber un “-” que será reconocido como MINUS, habrá una repetición, tomando a “2” como el tercer factor. OP devolverá el resultado de dicha operación.

- ASIG, declaración que nos permite asignar un valor a una variable. En dicha asignación, puede haber declarado una operación. Diagrama de interpretación de una asignación:

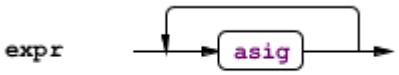


La asignación será reconocida cuando la instrucción empiece con una variable, seguida de un carácter igual "=" y la posible operación (recordemos que op no necesariamente debe ser una operación, puede ser un solo factor).

Si en nuestro código ingresamos "a=12+3", el compilador interpretará una asignación de "a" igual a 15.

El código Java declarado en dicha asignación, es simplemente la de imprimir el resultado final de dicha asignación y la de declarar la variable en el propio código Java.

- EXPR, reconoce una expresión cuando hay una asignación. Diagrama de interpretación de una expresión:



Definiciones agregadas y modificadas:

```
grammar M5;
tokens {
            PLUS     = '+' ;
            MINUS    = '-' ;
            MULT     = '*' ;
            DIV      = '/' ;
}

@header {
import java.util.HashMap;
import java.lang.Math;                    // Agrego las librerias Math de java
}

@members {

            HashMap variables = new HashMap();

   public static void main(String[] args) throws Exception {
      SimpleCalcLexer lex = new SimpleCalcLexer(new ANTLRFileStream(args[0]));
            CommonTokenStream tokens = new CommonTokenStream(lex);

      SimpleCalcParser parser = new SimpleCalcParser(tokens);

      try {
         parser.expr();
      } catch (RecognitionException e)  {
         e.printStackTrace();
      }
   }
}

/*------------------------------------------------------------------
 * PARSER RULES
 *------------------------------------------------------------------*/
expr        : (movr|movd|asig|delay|imprime)+;
```

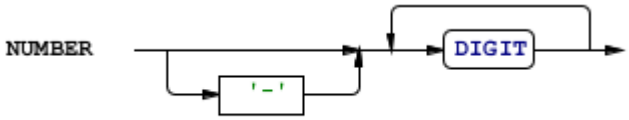
```
asig        : ID '=' op {System.out.println($ID.text +"="+ $op.value); variables.put($ID.text, new Double($op.value));};

op          returns [double value]
   :   e=factor {$value = $e.value;}
       (   PLUS e=factor {$value += $e.value;}
       |   MINUS e=factor {$value -= $e.value;}
       |   MULT e=factor {$value = $e.value*$value;}
       |   DIV e=factor {$value = $value/$e.value;}
       )*;


movr        returns [double q1,double q2,double q3,double q4,double q5,double q6]
            : 'movr '
            e=factor {$q1 = $e.value;}
            ','
            e=factor {$q2 = $e.value;}
            ','
            e=factor {$q3 = $e.value;}
            ','
            e=factor {$q4 = $e.value;}
            ','
            e=factor {$q5 = $e.value;}
            ','
            e=factor {$q6 = $e.value;}
            {
            double q1 = $q1; double q2 = $q2; double q3 = $q3; double q4 = $q4; double q5 = $q5; double q6 = $q6;

            double fila1=(63.25*Math.cos(q1 + q2 + q3))+ (63.25*Math.cos(q1 -q2 - q3 - q4 - q5 - q6)) +
(63.25*Math.cos(q1 + q2 + q3 + q4 + q5 + q6))+(63.25*Math.cos(q1 + q2 + q3 + q4))+ (63.25*Math.cos(q1 - q2 - q3 -
q4)) + (63.25*Math.cos(q1 - q2 - q3)); //Tomo que mi pt de referencia es [0,0,0] => Solo tomo la columna 4

            double fila2=(63.25*Math.sin(q1 + q2 + q3)) + (63.25*Math.sin(q1 - q2 -q3 - q4 - q5 - q6)) +
63.25*Math.sin(q1 + q2 + q3 + q4 + q5 + q6)+(63.25*Math.sin(q1 + q2 + q3 + q4))+(63.25*Math.sin(q1 - q2 - q3 -
q4)) +(63.25*Math.sin(q1 - q2 - q3));

            double fila3=(253*Math.sin(q2 + q3 + q4 + q5 + q6))/2 + (253*Math.sin(q2 + q3 + q4))/2
+(253*Math.sin(q2 + q3))/2 + 228;

            System.out.println("["+fila1+";"+fila2+";"+fila3+"]");
            };

movd        returns [double q1,double q2,double q3,double q4,double q5,double q6]
            : 'movd '
            e=factor {$q1 = $e.value;}
            ','
            e=factor {$q2 = $e.value;}
            ','
            e=factor {$q3 = $e.value;}
            ','
            e=factor {$q4 = $e.value;}
            ','
            e=factor {$q5 = $e.value;}
            ','
            e=factor {$q6 = $e.value;}
            {
            double q1 = Math.toRadians($q1);
            double q2 = Math.toRadians($q2);
```

```
            double q4 = Math.toRadians($q4);
            double q5 = Math.toRadians($q5);
            double q6 = Math.toRadians($q6);

            double fila1=(63.25*Math.cos(q1 + q2 + q3))+ (63.25*Math.cos(q1 -q2 - q3 - q4 - q5 - q6)) +
(63.25*Math.cos(q1 + q2 + q3 + q4 + q5 + q6))+(63.25*Math.cos(q1 + q2 + q3 + q4))+ (63.25*Math.cos(q1 - q2 - q3 -
q4)) + (63.25*Math.cos(q1 - q2 - q3)); //Tomo que mi pt de referencia es [0,0,0] => Solo tomo la columna 4

            double fila2=(63.25*Math.sin(q1 + q2 + q3)) + (63.25*Math.sin(q1 - q2 -q3 - q4 - q5 - q6)) +
63.25*Math.sin(q1 + q2 + q3 + q4 + q5 + q6)+(63.25*Math.sin(q1 + q2 + q3 + q4))+(63.25*Math.sin(q1 - q2 - q3 -
q4)) +(63.25*Math.sin(q1 - q2 - q3));

            double fila3=(253*Math.sin(q2 + q3 + q4 + q5 + q6))/2 + (253*Math.sin(q2 + q3 + q4))/2
+(253*Math.sin(q2 + q3))/2 + 228;

            System.out.println("["+fila1+";"+fila2+";"+fila3+"]");
            };

delay       returns [double tiempo]
            :'delayms '
            e=factor {$tiempo = $e.value;}
            {
            long t0,t1;
            t0=System.currentTimeMillis();
            do{
                        t1=System.currentTimeMillis();
                }while (t1-t0<$tiempo);
            };

imprime returns [String texto]
            :'imprime '
            e=string {$texto = $e.cadena;}
            {
            System.out.println($texto.substring(1, $texto.length()-1));
            };


factor      returns [double value]
    :   NUMBER {$value = Integer.parseInt($NUMBER.text);}
    |   FLOTANTE {$value =Double.parseDouble($FLOTANTE.text);}
    |   ID
            {
        Double v = (Double)variables.get($ID.text);
        if ( v!=null ) $value = v.doubleValue();
        else System.err.println("Variable no definida: "+$ID.text);
        };


string returns [String cadena]
            : CADENA
            {
            cadena=$CADENA.text;
            };

/*------------------------------------------------------------------
 * LEXER RULES
 *------------------------------------------------------------------*/
```

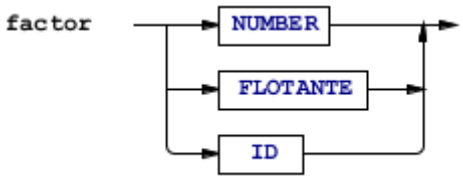
```
ID  :   ('a'..'z'|'A'..'Z')+ ;
CADENA              : '"'('a'..'z'|'A'..'Z'|WHITESPACE|'0'..'9'|'.'|':'|';'|'['|']'|'-')+'"';

NUMBER              : (|'-')(DIGIT)+;
FLOTANTE :(|'-') DIGIT+ '.' DIGIT+ ;

WHITESPACE : ( '\t' | ' ' | '\r' | '\n'| '\u000C' )+      { $channel = HIDDEN; } ;

fragment DIGIT      : '0'..'9' ;
```

Declaraciones modificadas:

- NUMBER, se agregó la posibilidad de reconocer números negativos. Diagrama final:

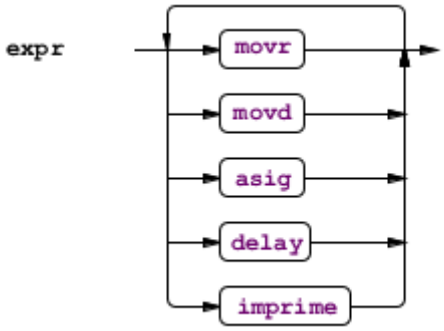


- FACTOR, se agregó la posibilidad de reconocer números flotantes y devolver números del tipo Double. Diagrama final:



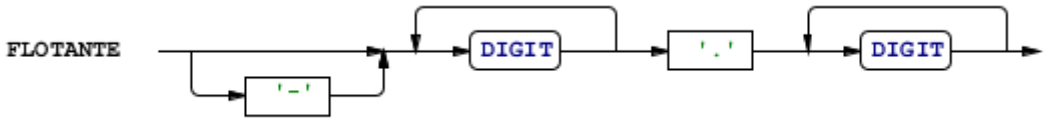
- OP, ahora devuelve números del tipo Double.
- ASIG, ahora asigna variables del tipo Double.

- EXPR, se agregaron las nuevas instrucciones incorporadas. Diagrama final:

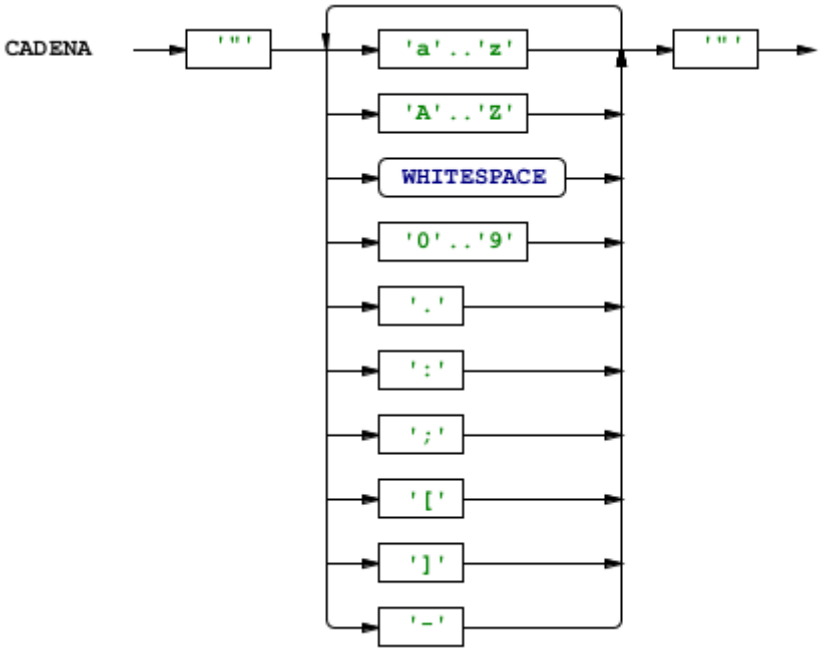


Declaraciones agregadas:

- FLOTANTES, declaración que nos permite reconocer números flotantes a diferencia de NUMBER que solo permitía reconocer números enteros. Diagrama de interpretación de un número flotante:



- CADENA, declaración que nos permite reconocer un String para luego ser utilizado en la instrucción imprimir. Diagrama de interpretación de una cadena:



Para ser reconocido como un String, el texto deberá encontrarse dentro de las comillas (“,”).

Si bien no se encuentran todos los caracteres definidos (ej. los paréntesis), con las definiciones realizadas, alcanzaron para realización de la tesis.

- STRING, simplemente devuelve la cadena reconocida en formato String al lenguaje Java. Diagrama de interpretación de un String:

- IMPRIME, declaración que nos permite reconocer la instrucción “imprime ”, para luego imprimir en pantalla el String ingresado. Diagrama de interpretación de imprime:

- DELAY, declaración que nos permite reconocer la instrucción “delayms”, la cual funciona como espera durante un cierto tiempo en mSeg. El código en Java, implementa un while que va comparando con el tiempo inicial y el tiempo final pedido en mSeg. Diagrama de interpretación de delayms:

Si bien el código implementado en Java funciona, sería conveniente evitar este tipo de instrucciones de espera usando lazos, debido al alto consumo que implica al sistema tener el uP en estado de 100% activo. Una forma de evitar esto sería usando instrucciones “sleep” y dejar dormido el proceso, sin embargo a la hora de implementar dicho código en Java, el ANTLR no permitía su uso (thread.sleep(long …)).

- MOVR, declaración que nos permite reconocer la instrucción “movr”, la cual será la utilizada para mover el robot M5 usando análisis de cinemática directa (para la tesis, simplemente se devuelve en pantalla el punto final). Dicha instrucción requiere de seis argumentos, los cuales representan las variables de cinemática directa ya vistas en el TPN1 (q1,….,q6). Los argumentos para esta instrucción deberán estar en ángulos radianes. Diagrama de interpretación de movr:

El código en Java empleado, necesito de las librerías matemáticas y de implementación parcial de la matriz homogénea obtenida en el TPN1.

- MOVD, declaración que nos permite reconocer la instrucción “movd”, la cual funciona igual que “movr”, a diferencia que la primera requiere ángulos degree. Los argumentos para esta instrucción deberán estar en ángulos radianes. Diagrama de interpretación de movd:

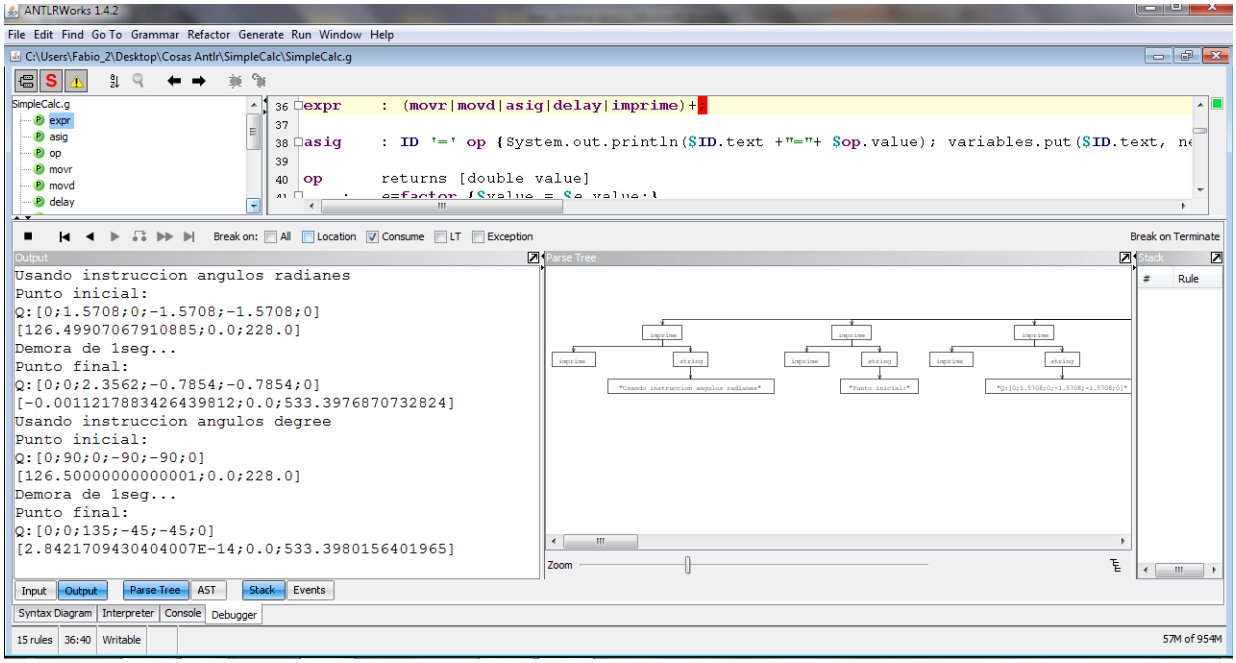
El código en Java empleado, necesito de las librerías matemáticas y de implementación parcial de la matriz homogénea obtenida en el TPN1.

Implementación del lenguaje M5

Código implementado:

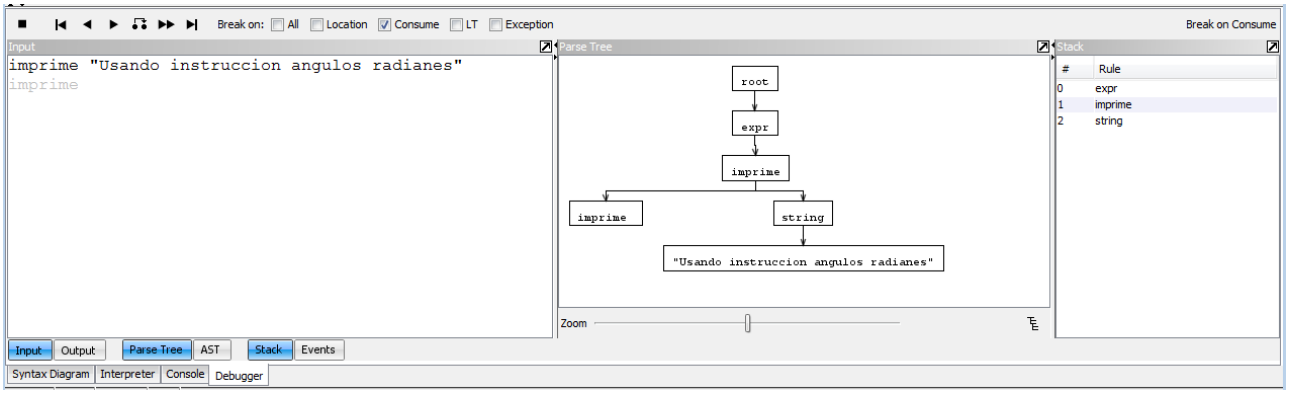
```
imprime "Usando instruccion angulos radianes"
imprime "Punto inicial:"
imprime "Q:[0;1.5708;0;-1.5708;-1.5708;0]"
movr 0,1.5708,0,-1.5708,-1.5708,0
imprime "Demora de 1seg..."
delayms 1000
imprime "Punto final:"
imprime "Q:[0;0;2.3562;-0.7854;-0.7854;0]"
movr 0,0,2.3562,-0.7854,-0.7854,0
imprime "Usando instruccion angulos degree"
imprime "Punto inicial:"
imprime "Q:[0;90;0;-90;-90;0]"
movd 0,90,0,-90,-90,0
imprime "Demora de 1seg..."
delayms 1000
imprime "Punto final:"
imprime "Q:[0;0;135;-45;-45;0]"
movd 0,0,135,-45,-45,0
```

Resultado obtenido:

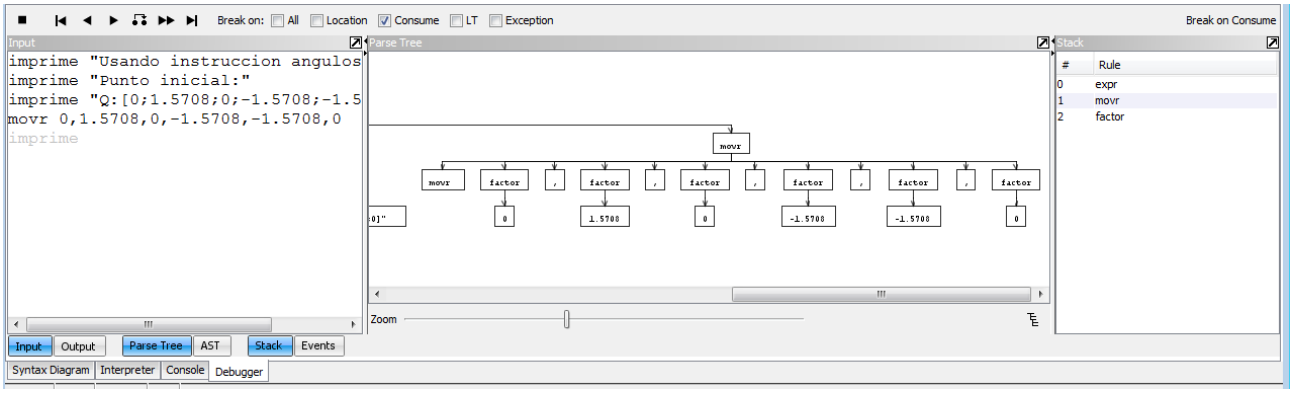


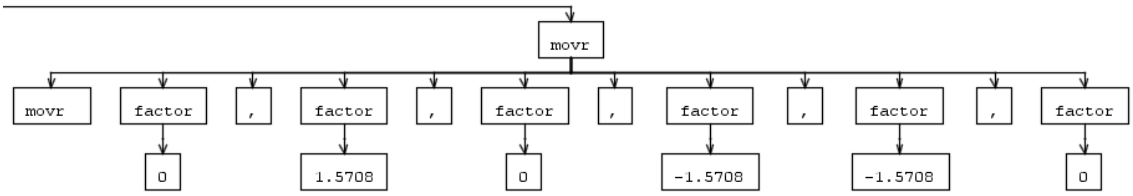
Interpretación del compilador:

- Imprime

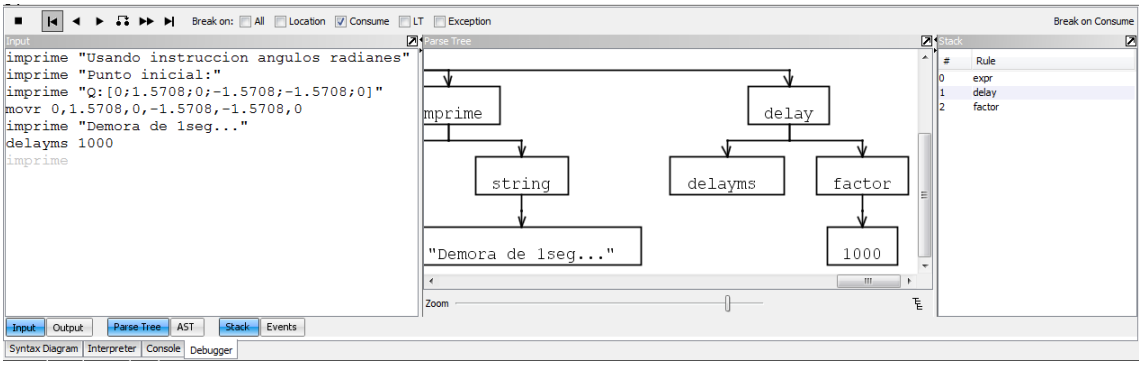


- MOVR

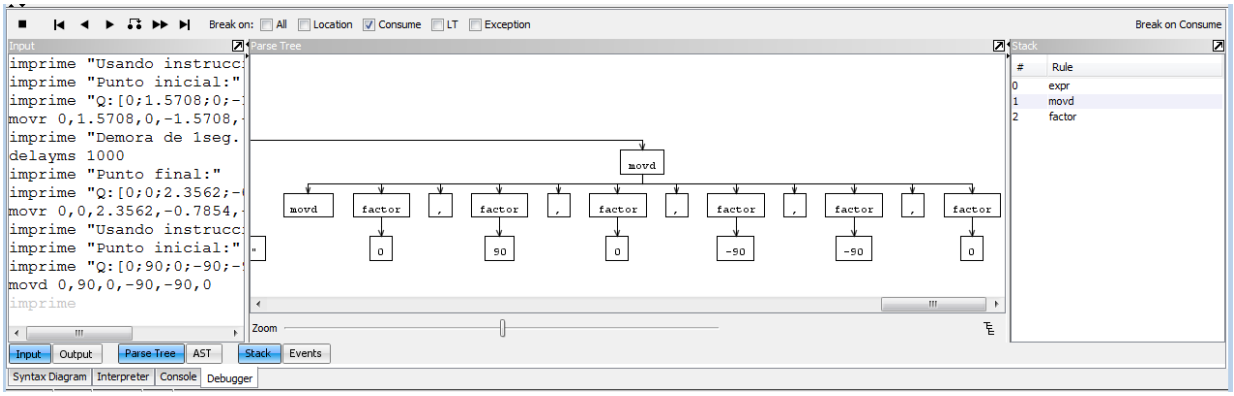


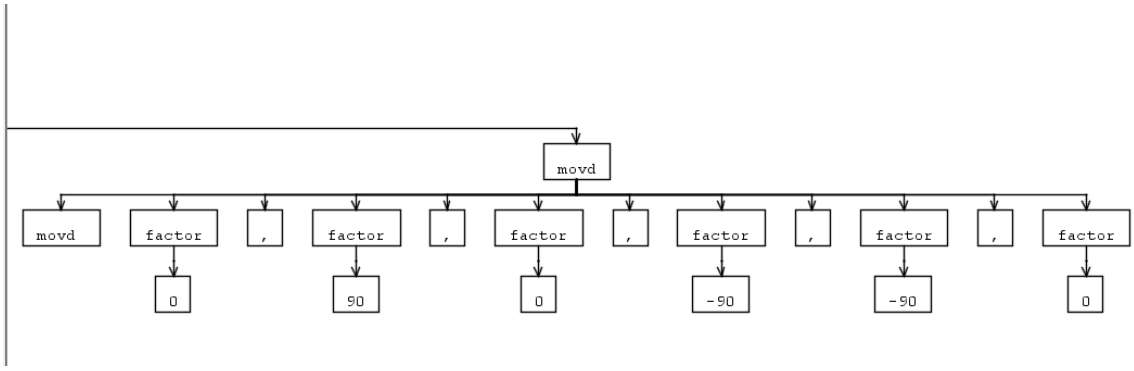


- DELAYMS



- MOVD





En el código implementado se puede observar que la transformación del punto [0;0;0] que es la base del robot, concuerda con los puntos obtenidos en el TPN 1.

Conclusiones

En primer parte del trabajo basado en la cinemática directa se utilizaron varias herramientas para la verificación y desarrollo. En el comienzo se realizó un movimiento con el software propietario del robot, de allí se tomaron los puntos necesarios para arribar a un modelo virtual en el Matlab. Contrastando con los valores del Matlab con el software propietario se notaba una diferencia de 20 mm, los cuales son debido al gripper que no tuvimos en cuenta. Además con el simulador matemático haciendo uso de las herramientas de Peter Croke llegamos a la expresión de la matriz homogénea, que utilizamos luego para realizar y verificar el modelo en el DSP. Comprobando así que la programación del mismo fue correcta.

La segunda parte es el análisis dinámico, el diseño de las piezas del robot se desarrolló con dificultades debido a la imposibilidad de representar las piezas igual que en la realidad, lo cual nos dio datos erróneos de la masa y de los momentos de inercias. Esos datos luego fueron usados en el cálculo de los torques, par coriolis, y gravedad. Observando los gráficos se puede apreciar que los mismos dieron con picos abruptos casi imposibles de representar en la realidad.

La implementación de FPGA nos brinda la posibilidad de crear tantas salidas de PWM como motores tengamos, lo cual termina siendo una solución a futuro en caso de tener que manejar otros motores. Se pudo observar mediante las salidas obtenidas en la simulación, que la velocidad de trabajos de este tipo de dispositivos suelen ser muy altas (tiempos del orden del nSeg) y los cuales permiten trabajar en forma paralela, virtudes muy importantes si se lo compara con un microcontrolador.

El desarrollo de un compilador propio para el Robot le da la posibilidad al producto de ser más sencillo de operar, lo cual se ve beneficiado enormemente a la hora de ser vendido. Por otro lado, la creación de un nuevo lenguaje sencillo a partir de otro más complejo, nos da una posibilidad enorme a la hora de desarrollar nuevos productos programables y posicionarlos en el mercado.

DATOS TRAYECTORIA 1 DATOS TRAYECTORIA 1

Matlab			DSP		
x	y	z	x	y	z
126,5000	0,0000	228,0000	127	0	228
126,3290	0,0000	234,5750	126	0	235
125,8165	0,0000	241,1323	126	0	241
124,9639	0,0000	247,6541	125	0	248
123,7734	0,0000	254,1227	124	0	254
122,2483	0,0000	260,5207	122	0	261
120,3928	0,0000	266,8308	120	0	267
118,2118	0,0000	273,0359	118	0	273
115,7112	0,0000	279,1193	116	0	279
112,8977	0,0000	285,0644	113	0	285
109,7791	0,0000	290,8553	110	0	291
106,3637	0,0000	296,4763	106	0	296
102,6608	0,0000	301,9122	103	0	302
98,6803	0,0000	307,1482	99	0	307
94,4331	0,0000	312,1703	94	0	312
89,9306	0,0000	316,9648	90	0	317
89,9306	0,0000	316,9648	90	0	317
89,7389	0,0000	326,2645	90	0	326
89,0642	0,0000	335,5417	89	0	336
87,9082	0,0000	344,7712	88	0	345
86,2741	0,0000	353,9282	86	0	354
84,1662	0,0000	362,9878	84	0	363
81,5902	0,0000	371,9257	82	0	372
78,5532	0,0000	380,7176	79	0	381
75,0633	0,0000	389,3397	75	0	389
71,1300	0,0000	397,7688	71	0	398
66,7639	0,0000	405,9821	67	0	406
61,9768	0,0000	413,9573	62	0	414
56,7816	0,0000	421,6729	57	0	422
51,1924	0,0000	429,1081	51	0	429
45,2243	0,0000	436,2427	45	0	436
38,8935	0,0000	443,0575	39	0	443
38,8935	0,0000	443,0575	39	0	443
38,7935	0,0000	449,6340	39	0	450
38,3518	0,0000	456,1964	38	0	456
37,5696	0,0000	462,7270	38	0	463
36,4491	0,0000	469,2081	36	0	469
34,9932	0,0000	475,6222	35	0	476
33,2058	0,0000	481,9519	33	0	482
31,0919	0,0000	488,1802	31	0	488
28,6572	0,0000	494,2902	29	0	494
25,9081	0,0000	500,2654	26	0	500
22,8522	0,0000	506,0897	23	0	506
19,4977	0,0000	511,7472	19	0	512

15,8536	0,0000	517,2227	16	0	517
11,9299	0,0000	522,5014	12	0	523
7,7372	0,0000	527,5691	8	0	528
3,2867	0,0000	532,4120	3	0	532

DATOS TRAYECTORIA 2 Matlab			DATOS TRAYECTORIA 2 DSP		
x	y	z	x	y	z
126,5000	0,0000	228,0000	127	0	228
126,5000	0,0000	228,0000	127	0	228
126,5000	0,0000	228,0000	127	0	228
126,3290	0,0000	234,5750	126	0	235
132,3738	0,0000	241,6448	132	0	242
138,8955	0,0000	242,4974	139	0	242
137,9543	0,0000	249,6972	138	0	250
143,1618	0,0000	257,6909	143	0	258
149,4719	0,0000	259,5464	149	0	260
147,6302	0,0000	267,2728	148	0	267
151,8580	0,0000	276,0835	152	0	276
157,8032	0,0000	278,8969	158	0	279
154,9444	0,0000	287,0302	155	0	287
158,0648	0,0000	296,5289	158	0	297
163,5007	0,0000	300,2319	164	0	300
159,5253	0,0000	308,6324	160	0	309
161,4287	0,0000	318,6706	161	0	319
166,2233	0,0000	323,1731	166	0	323
161,0518	0,0000	331,6842	161	0	332
161,6501	0,0000	342,0960	162	0	342
165,6869	0,0000	347,2887	166	0	347
159,2627	0,0000	355,7393	159	0	356
158,4913	0,0000	366,3452	158	0	366
161,6723	0,0000	372,1021	162	0	372
153,9638	0,0000	380,3105	154	0	380
151,7843	0,0000	390,9205	152	0	391
154,0322	0,0000	397,1017	154	0	397
145,0347	0,0000	404,8792	145	0	405
141,4359	0,0000	415,2972	141	0	415
142,6962	0,0000	421,7526	143	0	422
132,4327	0,0000	428,9075	132	0	429
127,4322	0,0000	438,9347	127	0	439
127,6742	0,0000	445,5076	128	0	446
116,1963	0,0000	451,8496	116	0	452
109,8401	0,0000	461,2895	110	0	461
109,0580	0,0000	467,8200	109	0	468

96,4455	0,0000	473,1643	96	0	473
88,8084	0,0000	481,8263	89	0	482
87,0211	0,0000	488,1561	87	0	488
73,3814	0,0000	492,3275	73	0	492
64,5655	0,0000	500,0315	65	0	500
61,8164	0,0000	506,0067	62	0	506
47,2830	0,0000	508,8440	47	0	509
37,4164	0,0000	515,4245	37	0	515
33,7723	0,0000	520,9000	34	0	521
18,5027	0,0000	522,2595	19	0	522
7,7372	0,0000	527,5691	8	0	528
3,2867	0,0000	532,4120	3	0	532