UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Robótica

Tesis Final

Alumnos: Fuster, Alan Ezequiel

Hampel, Matías Rolf

Martinez, Matías Javier

Profesor: Ing. Giannetta, Hernán

Jefe de TP: Ing. Granzella, Eduardo Damián

Curso: R6055

Horario: Lunes 19:00 a 23:00

Año: 2012

Índice:

- # Cinemática

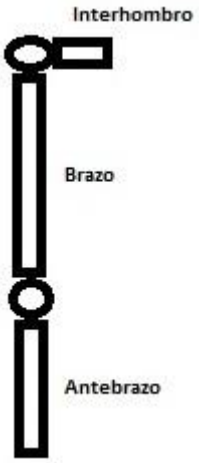
- ## Introducción a la cinemática

Para este proyecto de robótica elegimos un robot que imita los movimientos de un brazo humano. Para ello pensamos en una estructura como la siguiente:

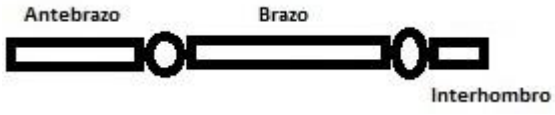
Como se puede observar en la imagen, el robot consta de tres ejes de libertad. Con esto, intentamos que nuestro robot alcance los puntos que un brazo humano puede alcanzar. A continuación se enumeran los tres ejes de libertad:

- 1º Eje de libertad: Se encuentra entre el interhombro y el brazo. Realiza una rotación total de 180º con el eje direccionado a 90º del interhombro.

  - Movimiento con un ángulo de 0º:

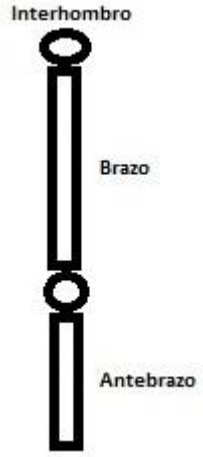


  - Movimiento con un ángulo de 90º:



- 2º Eje de libertad: Se encuentra entre el interhombro y el brazo. Realiza una rotación desde -60º a 180º con el eje coincidente con el interhombro.

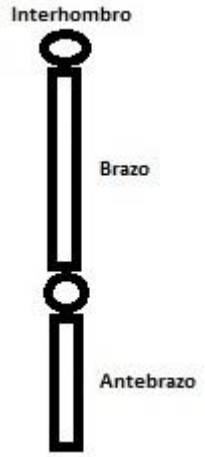
  - Movimiento con un ángulo de 0º:



  - Movimiento con un ángulo de 180º:

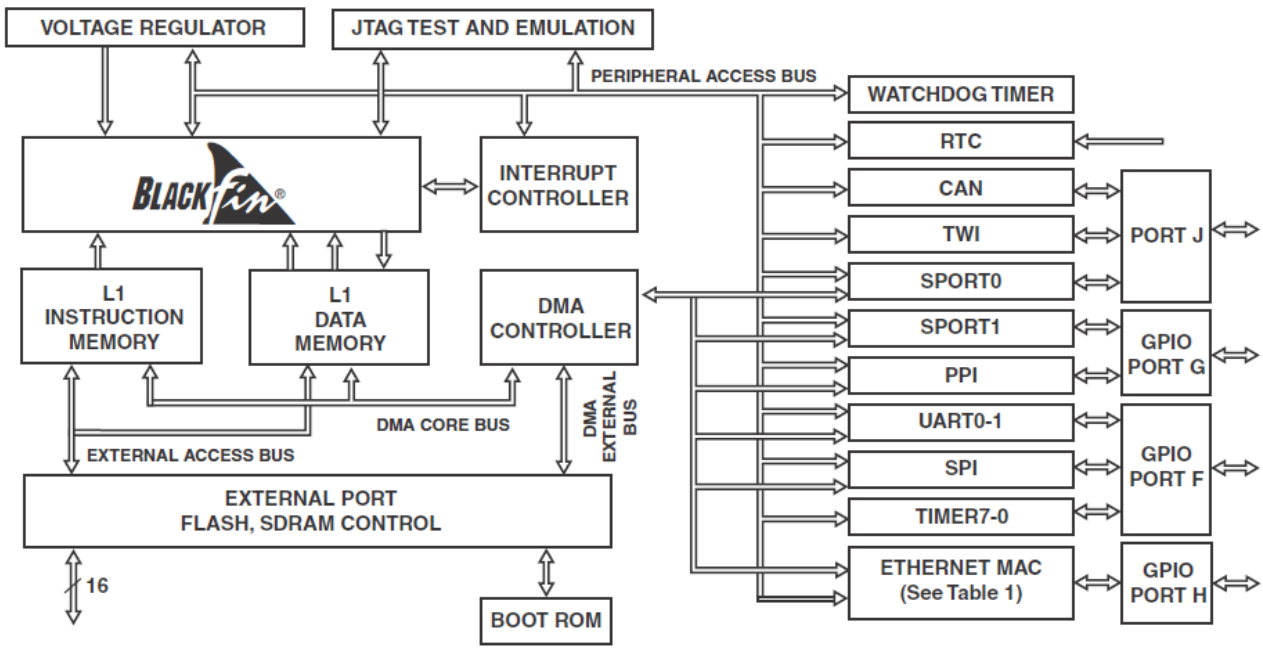
- 3º Eje de libertad: Se encuentra entre el brazo y el antebrazo. Realiza una rotación total de 180º con el eje direccionado a 90º del brazo.

  - Movimiento con un ángulo de 0º:



- Movimiento con un ángulo de 45º:

Para realizar la simulación de los movimientos y de la cinemática del robot se utilizó en primer lugar el programa Matlab, y luego se utilizó el IDE VisualDSP de los DSP Blackfin. Para este trabajo práctico utilizamos el DSP ADSP-BF537. Este utiliza una aritmética de punto fijo de 1.15 para los registro y de 9.31 para el acumulador. Este DSP contiene el siguiente diagrama:



A su vez, presenta las siguientes características:

- Up to 600MHz high performance Blackfin processor
- Three 16-bit MACs, two 40-bit ALUs, four 8-bit video ALUs, 40-bit shifter
- RISC-like register and instruction model for ease of programming and compiler-friendly support
- Advanced debug, trace, and performance monitoring
- Wide range of operating voltages
- Qualified for Automotive Applications
- Programmable on-chip voltage regulator
- 182-ball and 208-ball CSP_BGA packages

Memoria

- Up to 132K bytes of on-chip memory
- Instruction SRAM/cache and instruction SRAM
- Data SRAM/cache plus additional dedicated data SRAM
- Scratchpad SRAM
- External memory controller with glueless support for SDRAM and asynchronous 8-bit and 16-bit memories
- Flexible booting options from external flash, SPI and TWI memory or from SPI, TWI, and UART host devices
- Memory management unit providing memory protection

Periféricos

- IEEE 802.3-compliant 10/100 Ethernet MAC
- Controller area network (CAN) 2.0B interface
- Parallel peripheral interface (PPI), supporting ITU-R 656 video data formats
- 2 dual-channel, full-duplex synchronous serial ports
- (SPORTs), supporting 8 stereo I2S channels
- 12 peripheral DMAs, 2 mastered by the Ethernet MAC
- 2 memory-to-memory DMAs with external request lines
- Event handler with 32 interrupts inputs
- Serial peripheral interface (SPI) compatible
- 2 UARTs with IrDA support
- 2-wire interface (TWI) controller
- Eight 32-bit timer/counters with PWM support
- Real-time clock (RTC) and watchdog timer
- 32-bit core timer
- 48 general-purpose I/Os (GPIOs), 8 with high current drivers
- On-chip PLL capable of frequency multiplication
- Debug/JTAG interface

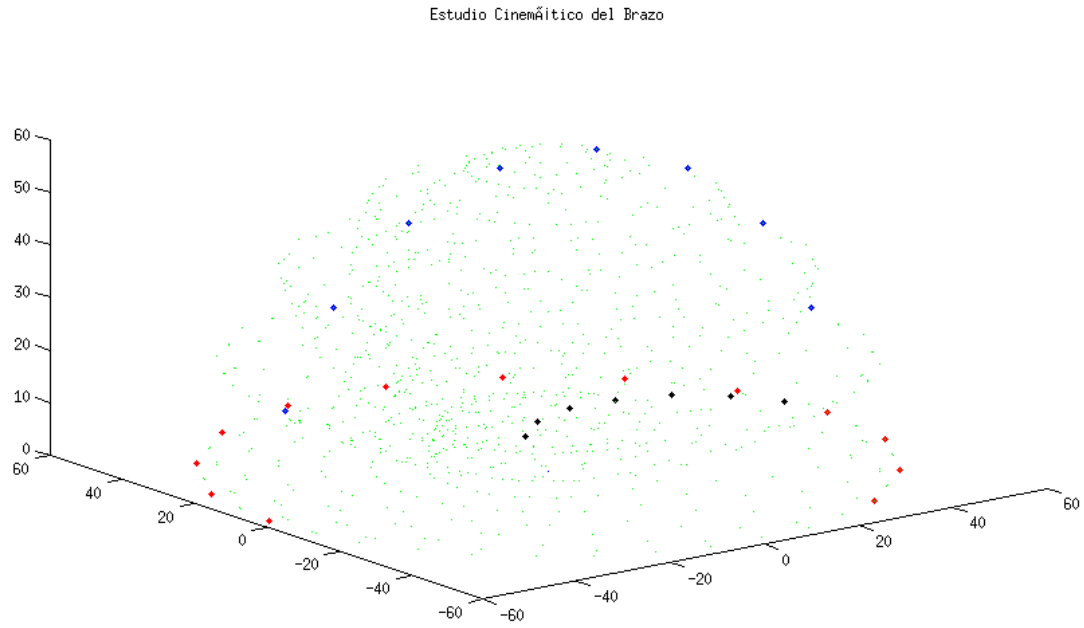
- **Simulación en Matlab**

Para simular la cinemática del robot se utilizaron las funciones para robots de Matlab creada y realizada por Peter Corke. Este conjunto de funciones se llama Robotics Toolbox. Mediante estas funciones pudimos obtener la matriz T de nuestro robot. El script utilizado fue el siguiente:

```
clc;
close all;
clear;
syms q1; % Hombro 1 Rotacion z1(360°), q1 positivo va para adelante
syms q2; % Hombro 2 Rotacion z2(180°), q2 positivo levanta el brazo
syms q3; % Codo Rotacion z3 (de 0° a menos de 180°), q3 positivo
levanta el antebrazo
l1=1; % Interhombro
l2=31; % Brazo (Humero)
l3=28.5; % Antebrazo (Cubito y Radio)
A=trotz(q1);
B=transl(0,0,l1);
C=trotx(pi/2);
D=trotz(q2);
E=transl(l2,0,0); % Trasladamos en +y hasta el codo
F=trotx(-pi/2); % Rotamos en y +90. Ahora las rotaciones en +z suben
el brazo
G=trotz(q3);
H=transl(l3,0,0);
T=A*B*C*D*E*F*G*H;
Res = T*[0;0;0;1];
plot3(0,0,0);
hold on;
tic;

for q1=-pi/3:pi/32:pi          %hombro1 de -60° a 180°
    for q2=0:pi/32:pi          %hombro2 de 0° a 180°
       for q3=0:pi/32:pi*5/6   %codo de 0° a 150°
            A=trotz(q1);
            B=transl(0,0,l1);
            C=trotx(pi/2);
            D=trotz(q2);
            E=transl(l2,0,0); % Trasladamos en +y hasta el codo
            F=trotx(-pi/2); % Rotamos en y +90. Ahora las rotaciones
en +z suben el brazo
            G=trotz(q3);
            H=transl(l3,0,0);

            T=A*B*C*D*E*F*G*H;
            res= T*[0;0;0;1];
            if(q2 == 0 && q3 == 0)
                plot3(res(1,1),res(2,1),res(3,1),'.-r');
            else if(q1 == 0 && q3 == 0)
                    plot3(res(1,1),res(2,1),res(3,1),'.-g');
                else if(q1 == 0 && q2 == 0)
                        plot3(res(1,1),res(2,1),res(3,1),'.-k');
                    else
                        plot3(res(1,1),res(2,1),res(3,1),'b');
                    end
                end
            end

       end
    end
```

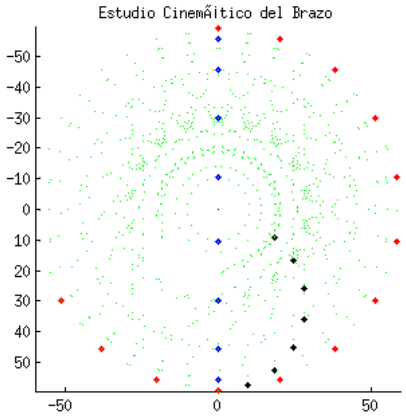
Luego de ejecutar el script obtuvimos las siguientes gráficas:

- Gráfico Nº1:

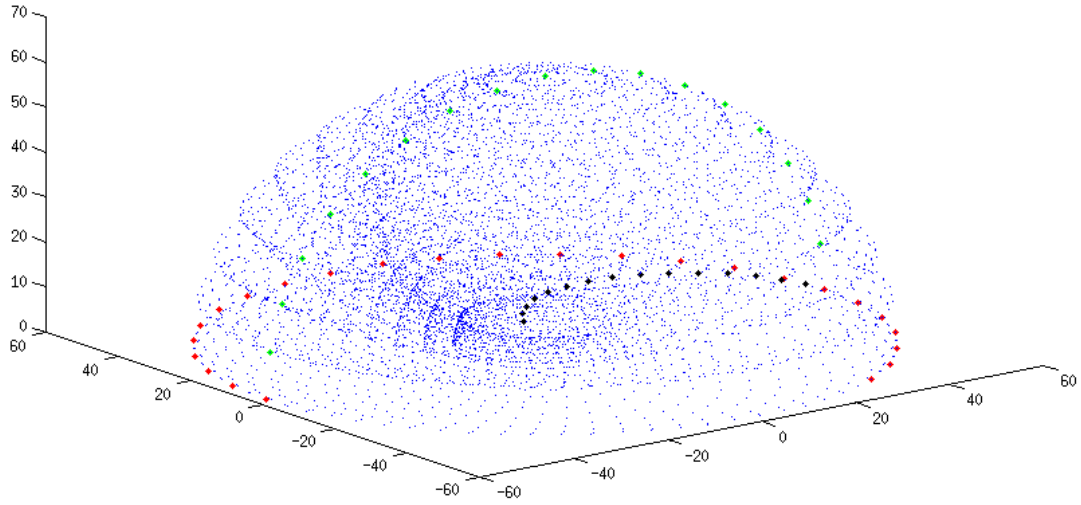


En este gráfico se encuentran destacados los tres movimientos que pueden realizar cada uno de los tres motores (ejes de libertad). Los puntos en color azul pertenecen al eje de libertad Nº1. Los puntos en color rojo pertenecen al eje de libertad Nº2. Los puntos en color negro pertenecen al eje de libertad Nº3.

- Gráfico Nº2:

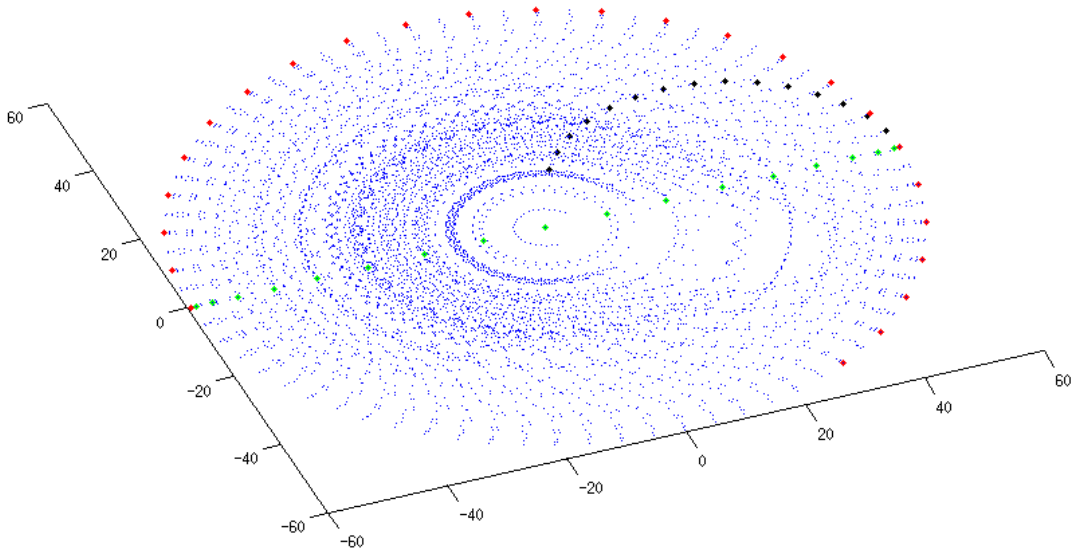


- Gráfico Nº3:



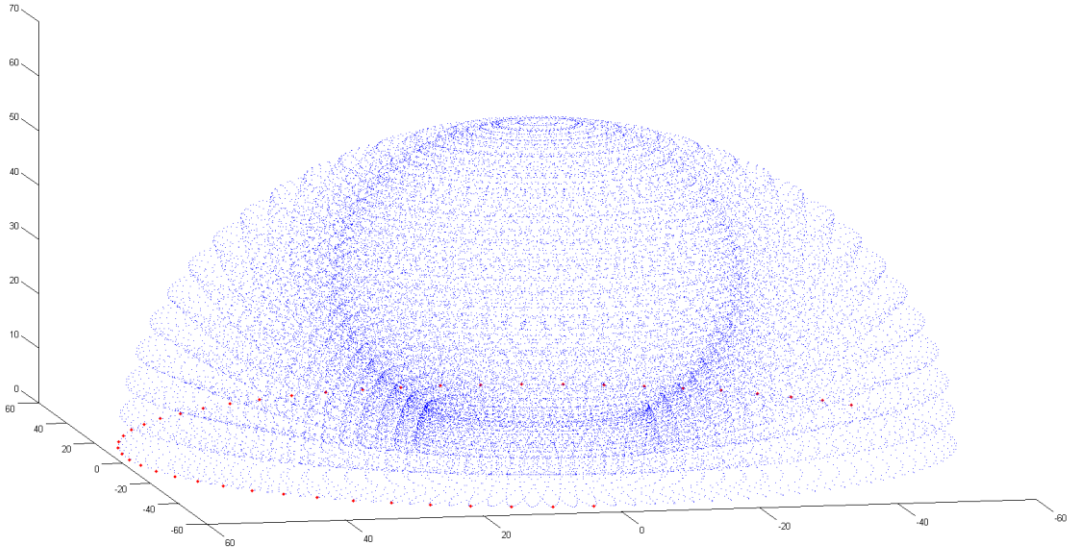
En este gráfico se muestran el total movimiento que puede realizar nuestro robot. Como se ve, el robot puede dibujar casi una semiesfera con el movimiento. Salvo en un sector particular donde el robot no se llega que es la zona atrás de la espalda de un hombre, donde su brazo tampoco llega.

- Gráfico Nº4:



En este gráfico se puede observar más claramente la zona donde el robot no puede alcanzar debido a sus características constructivas.

- Gráfico Nº5:



En este gráfico se observar una campana de puntos, cuyos puntos interiores este robot no puede alcanzar.

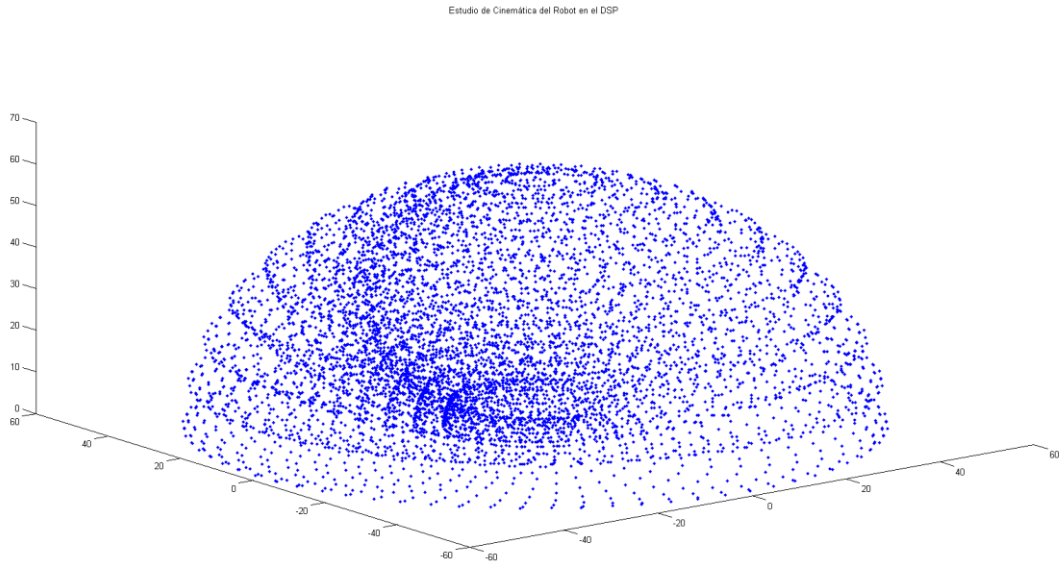
- **Simulación en VisualDSP**

Para realizar el programa para el DSP, utilizamos las fórmulas obtenidas en Matlab y las colocamos en dicho programa. El programa desarrollado es el siguiente:

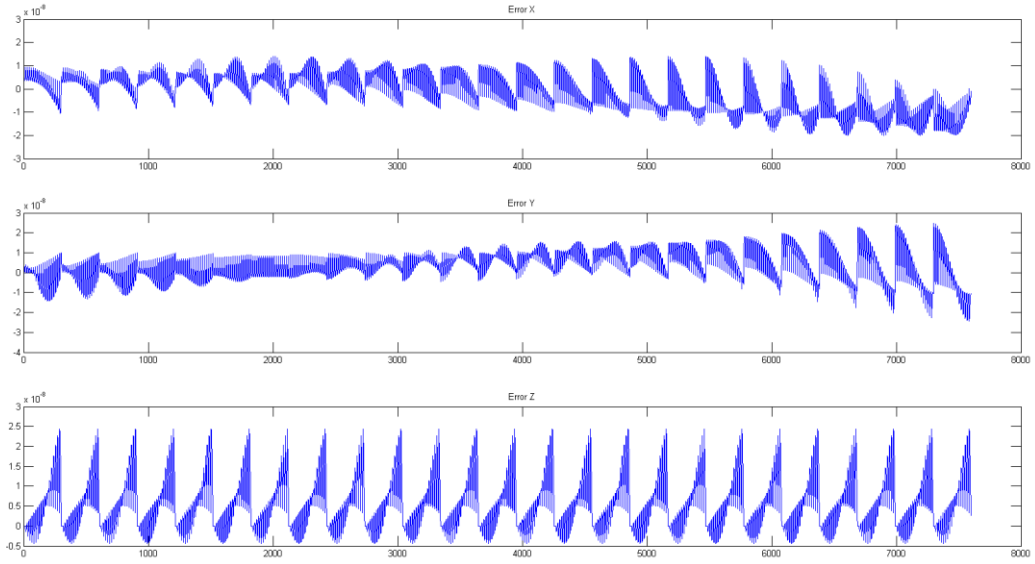
```
#include <stdio.h>
#include "Configuracion.h"
#include "MyDSPLibrary.h"
#include <math.h>

#define VECTOR_LONG     7600

int main(void)
{
        double q1, q2, q3, x[VECTOR_LONG], y[VECTOR_LONG], z[VECTOR_LONG];
        char * file_x =  "..\\x.dat";
        char * file_y =  "..\\y.dat";
        char * file_z = "..\\z.dat";
        int I = 0;
        for(q1=-3.141592654/3;q1<=3.141592654;q1+=3.141592654/18)
        {
                for(q2=0;q2<=3.141592654;q2+=3.141592654/18)
                {
                        for(q3=0;q3<=3.141592654*5/6;q3+=3.141592654/18)
                        {
                                x[i]=31*cos(q1)*cos(q2) – 28.5* sin(q1)*sin(q3) + 28.5 *cos(q1)*cos(q2)*cos(q3);

                                y[i] =31*cos(q2)*sin(q1) + 28.5*sin(q3)*cos(q1) + 28.5*cos(q3)*cos(q2)*sin(q1);

                                z[i] = 31*sin(q2) + 28.5*cos(q3)*sin(q2) + 1;
                                i++;
                        }
                }
        }
        MyWriteFile( file_x, x ,VECTOR_LONG) ;
        MyWriteFile( file_y, y, VECTOR_LONG) ;
        MyWriteFile( file_z, z, VECTOR_LONG) ;
}
```

Este programa obtiene todos los puntos de posición posibles del robot y a su vez, graba todos los puntos de un determinado eje (x, y o z) en archivos separados. Para luego leer dichos archivos en Matlab y comparar el cálculo en el DSP con el obtenido en dicho programa.

Al momento de graficar los puntos obtenidos en el DSP obtuvimos lo siguiente:



Como vemos en esta gráfica, los puntos obtenidos describen una semiesfera como lo obtenido anteriormente en Matlab. Luego se realizo la comparación de los valores obtenido en Matlab y VisualDSP y se obtuvo el gráfico de errores:



El error obtenido en los tres ejes esta en el orden de 10^{-18}, siendo este totalmente despreciable. Por lo tanto, se puede concluir que el programa desarrollado para el DSP se comporta igual que el script de Matlab. Por consiguiente, la adaptación al DSP fue exitosa.

- **Conclusiones del modelo cinemático del robot**

  - Pudimos comprobar los temas visto en clase en el desarrollo de esta práctica al implementar la cadena directa de cinemática en este robot.
  - Pudimos desarrollar exitosamente la cinemática de nuestro robot. De esta manera, se puede implementar en cualquier DSP.
  - Pudimos observar que nuestro robot tiene puntos a los que no puede llegar. O sea, puntos ciegos.
  - El error entre el DSP y Matlab es prácticamente despreciable.
  - Pudimos imitar exitosamente el movimiento del brazo humano.

Dinámica:

Cálculo del modelo dinámico del robot:

Para obtener el modelo dinámico de nuestro robot utilizaremos la herramienta para matlab Hemero, mediante una serie de funciones este toolbox logra entregarnos las expresiones de los torques que se le deben aplicar a cada articulación para ir a la posición deseada. Para lograr esto, la herramienta necesita que se le ingrese la matriz de parámetros DyN, para hallar los parámetros de dicha matriz realizamos el modelo del robot en el programa Solid Edge ST2.

La matriz DyN que debemos completar es la siguiente:

$$\alpha \;\; a \;\; \theta \;\; d \;\; \sigma \;\; masa \;\; rx \;\; ry \;\; rz \; lxx \; lyy \; lzz \; lxy \; lyz \; lxz \; Jm \; G \; B \;\; Tc+ \;\; Tc-$$

Estas 20 columnas deben ser completadas para cada una de las tres articulaciones, lo que resulta en una matriz de 3x20.

Las primeras cuatro columnas son los parámetros de Denavit-Hatemberg, los cuales fueron calculados en el tp 1:

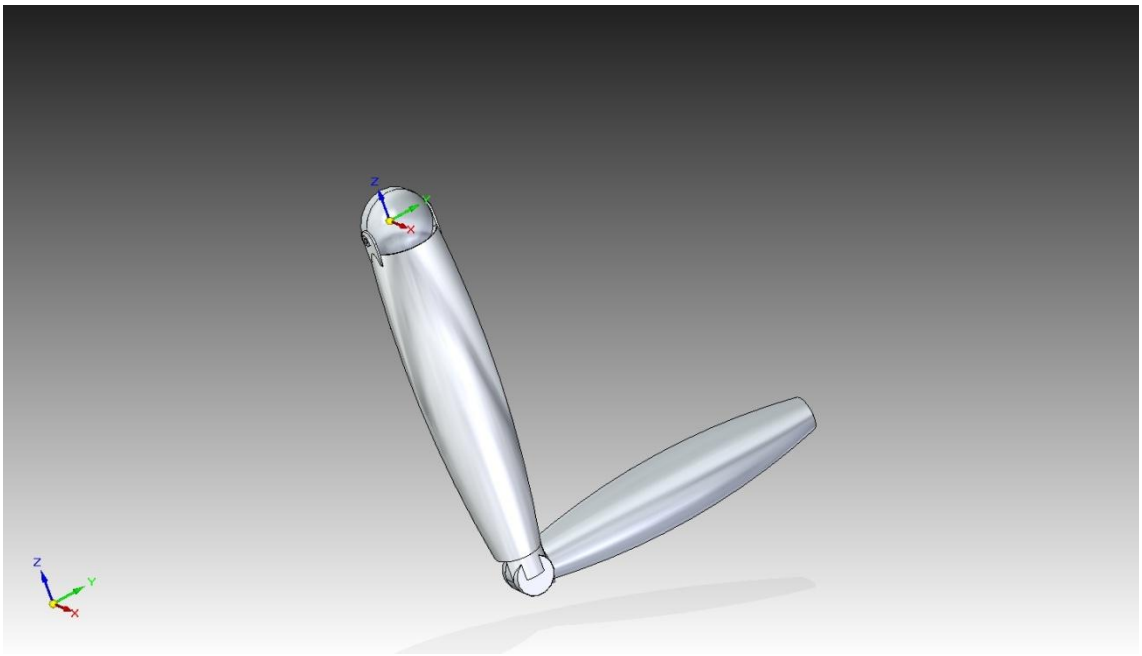
	α	a	θ	d
Inter-Hombro 1	90º	0	q1	L1
Inter-Hombro 2	-90º	L2	q2	0
Codo	0º	L3	q3	0

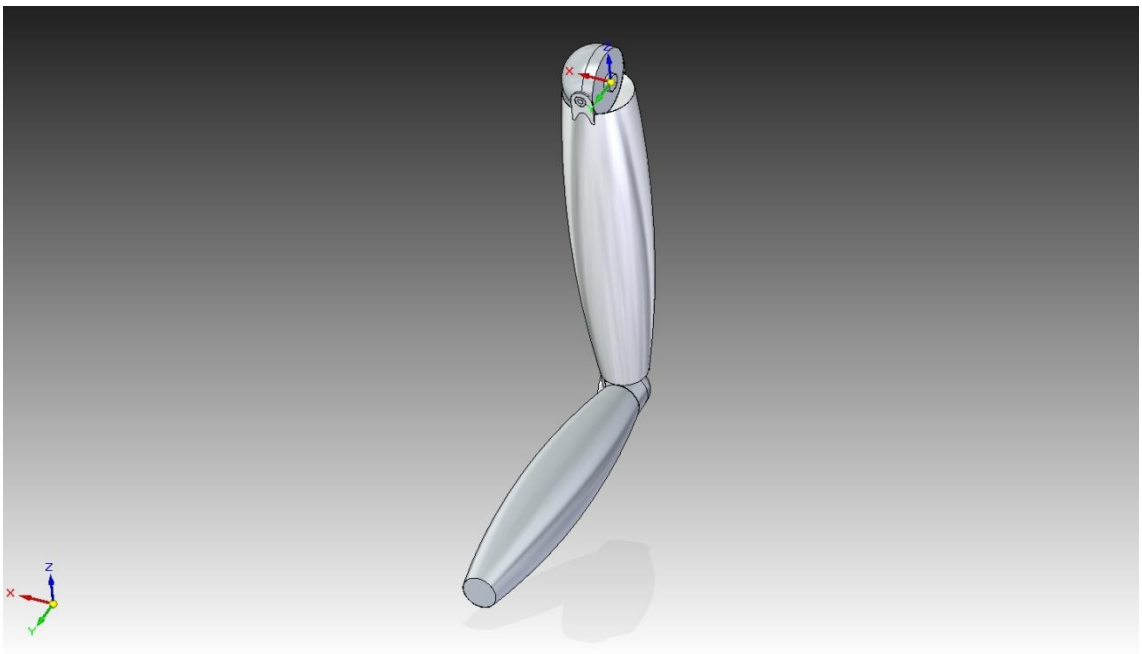
El parámetro σ será cero para las tres articulaciones dado que todos los movimientos serán de rotación y ninguno de traslación.

Cálculo de los parámetros mecánicos del robot:

Como dijimos anteriormente, la herramienta que utilizamos es el CAD Solid Edge ST2.

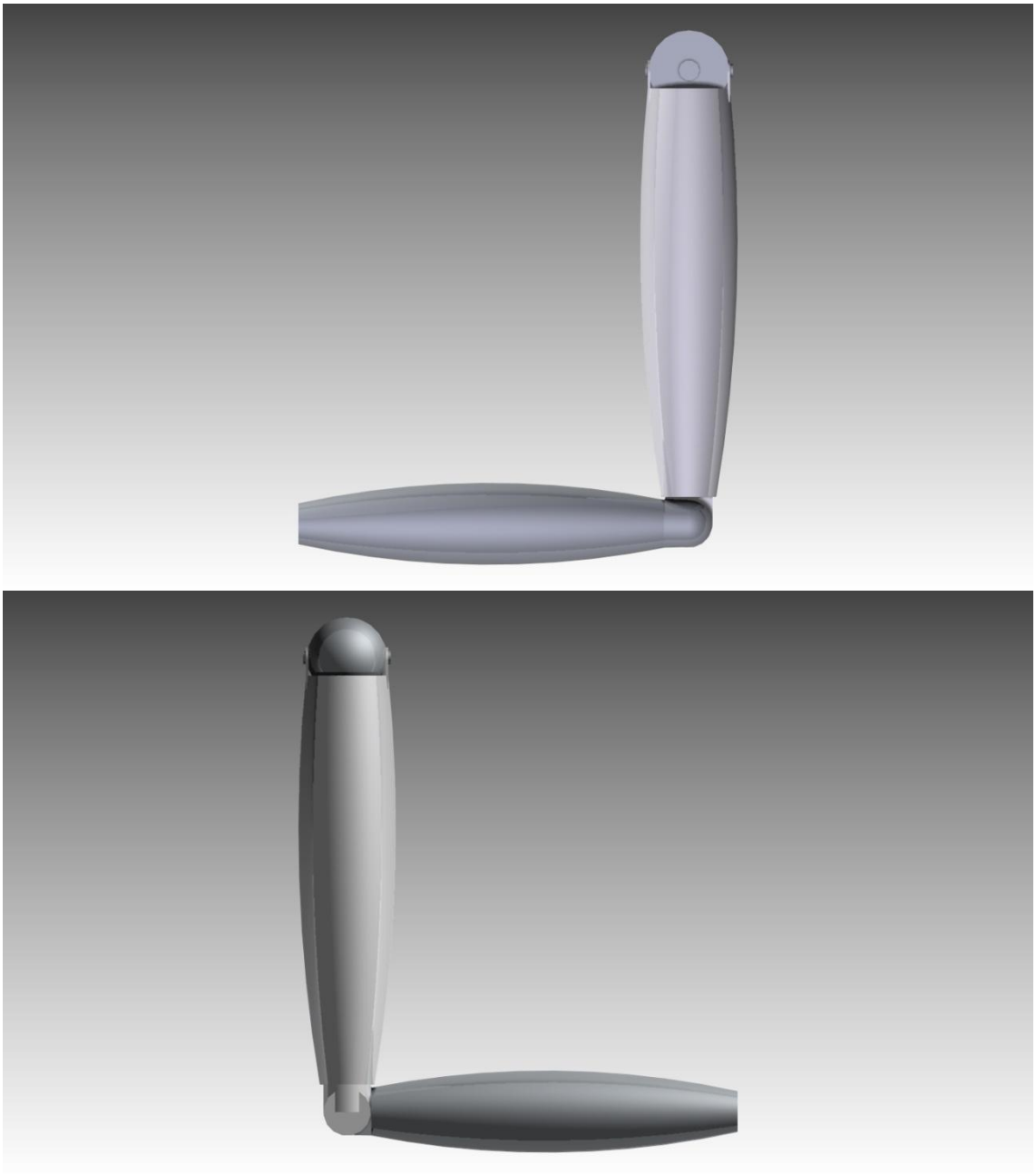
Para obtener dichos parámetros se procede a realizar las partes sólidas:





En estas dos imágenes se puede ver claramente dónde se encuentra nuestra referencia, ese será el inicio de nuestro sistema de ejes.

El modelo ya terminado es el siguiente:



Luego de realizar el diseño de las partes sólidas, se procede a especificar los materiales para poder así obtener los parámetros mecánicos buscados.

En nuestro caso el brazo será confeccionado en aluminio, obteniendo los siguientes resultados:

- Primera articulación:

Physical Properties Report

volume= 80769,557 mm^3

mass= 0,219 kg

With respect to the Global Coordinate System.

Center of Mass:

X= -0,17 mm

Y= -0,44 mm

Z= 20,52 mm

Center of Volume:

X= -0,17 mm

Y= -0,44 mm

Z= 20,52 mm

Mass Moments of Inertia:

Ixx= 0,00 kg-m^2

Iyy= 0,00 kg-m^2

Izz= 0,00 kg-m^2

Ixy= 0,00 kg-m^2

Ixz= 0,00 kg-m^2

Iyz= 0,00 kg-m^2

Principal Axes Orientation:

1= 0,000 0,000 1,000

2= 0,000 1,000 0,000

3= -1,000 0,000 0,000

With respect to the Principal Axes

Principal Moments of Inertia:

I1= 0,00 kg-m^2

I2= 0,00 kg-m^2

I3= 0,00 kg-m^2

Radii of Gyration:

K1= 19,25 mm

K2= 16,83 mm

K3= 16,72 mm

- Segunda articulación:

Physical Properties for Selected Parts

volume= 727325,572 mm^3

mass= 1,973 kg

With respect to the Global Coordinate System

Center of Mass:

X= -0,18 mm

Y= -133,76 mm

Z= 0,00 mm

Center of Volume:

X= -0,18 mm

Y= -133,76 mm

Z= 0,00 mm

Mass Moments of Inertia:

Ixx= 0,05 kg-m^2

Iyy= 0,00 kg-m^2

Izz= 0,05 kg-m^2

Ixy= 0,00 kg-m^2

Ixz= 0,00 kg-m^2

Iyz= 0,00 kg-m^2

Principal Axes Orientation:

X= 1,000 -0,008 0,000

Y= 0,000 0,000 1,000

Z= -0,008 -1,000 0,000

With respect to the Principal Axes

Principal Moments of Inertia:

I1= 0,01 kg-m^2

I2= 0,01 kg-m^2

I3= 0,00 kg-m^2

Radii of Gyration:

Rx= 77,42 mm

Ry= 77,41 mm

Rz= 25,03 mm

- Tercera articulación:

Physical Properties Report

volume= 585811,445 mm^3

mass= 1,589 kg

With respect to the Global Coordinate System.

Center of Mass:

X= 140,52 mm

Y= -1,78 mm

Z= -10,46 mm

Center of Volume:

X= 140,52 mm

Y= -1,78 mm

Z= -10,46 mm

Mass Moments of Inertia:

Ixx= 0,00 kg-m^2

Iyy= 0,04 kg-m^2

Izz= 0,04 kg-m^2

Ixy= 0,00 kg-m^2

Ixz= 0,00 kg-m^2

Iyz= 0,00 kg-m^2

Principal Axes Orientation:

1=	-0,013	0,008	1,000
2=	0,013	1,000	-0,008
3=	-1,000	0,013	-0,013

With respect to the Principal Axes

Principal Moments of Inertia:

I1= 0,01 kg-m^2

I2= 0,01 kg-m^2

I3= 0,00 kg-m^2

Radii of Gyration:

K1= 72,74 mm

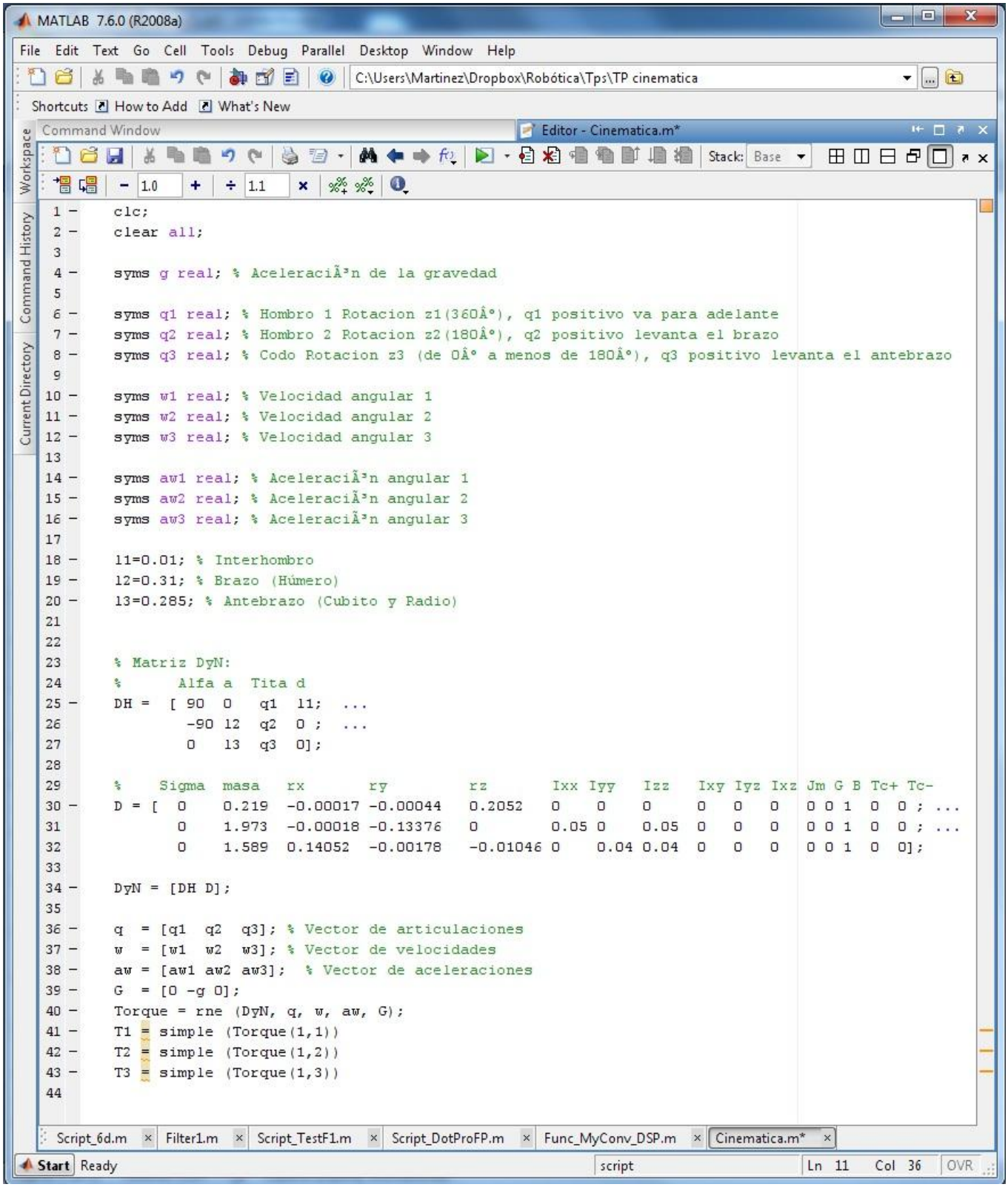
K2= 72,73 mm

K3= 18,76 mm

Ahora sí podemos confeccionar la matriz DyN, la cual se verá completa en el siguiente paso.

Obtención de torques del robot:

Se busca, mediante el enfoque dinámico de Lagrange-Euler, obtener los torques a aplicar a cada uno de los motores brushless del robot para lograr un movimiento determinado, por lo que los torques obtenidos quedarán en función de la posición, la velocidad y la aceleración. Para esto utilizaremos el toolbox de Hemero para Matlab, el script implementado es el siguiente:



Arrojando los siguientes resultados:

```
T1 =

(4991586901103*aw1)/10000000000000 -
(556191381*aw1*sin(2*q2))/1250000000000 -
(496811973*aw1*sin(2*q3))/1250000000000 +
(556191381*w1*w2)/625000000000 + (496811973*w1*w3)/625000000000
+ (701390283*aw1*cos(q2))/2500000000 +
(12783067*aw1*sin(q2))/78125000 -
(34506203547*aw1*cos(q2)^2)/1250000000000 -
(35685576739*aw1*cos(q3)^2)/500000000000 -
(47369679*aw2*sin(q2))/10000000000 +
(12783067*w1*w2*cos(q2))/78125000 -
(8060997*w1*w2*cos(q3))/5000000000 +
(35685576739*aw1*cos(q2)^2*cos(q3)^2)/250000000000 -
(701390283*w1*w2*sin(q2))/2500000000 +
(318182949*w1*w2*sin(q3))/2500000000 +
(173046867*aw1*cos(q2)*cos(q3))/1250000000 +
(4384051*aw1*cos(q2)*sin(q3))/2500000000 +
(4384051*aw1*cos(q3)*sin(q2))/2500000000  -
(47369679*w2^2*cos(q2))/10000000000 -
(556191381*w1*w2*cos(q2)^2)/312500000000 -
(496811973*w1*w2*cos(q3)^2)/312500000000 -
(496811973*w1*w3*cos(q2)^2)/312500000000 -
(496811973*w1*w3*cos(q3)^2)/312500000000 -
(173046867*aw1*sin(q2)*sin(q3))/1250000000 +
(34506203547*w1*w2*sin(2*q2))/1250000000000 +
(35685576739*w1*w2*sin(2*q3))/500000000000 +
(35685576739*w1*w3*sin(2*q2))/500000000000 +
(35685576739*w1*w3*sin(2*q3))/500000000000 +
(318182949*aw1*cos(q2)^2*cos(q3))/2500000000 +
(8060997*aw1*cos(q2)^2*sin(q3))/5000000000 -
(173046867*w1*w2*cos(q2)*sin(q3))/1250000000 -
(173046867*w1*w2*cos(q3)*sin(q2))/1250000000 -
(173046867*w1*w3*cos(q2)*sin(q3))/1250000000 -
(173046867*w1*w3*cos(q3)*sin(q2))/1250000000 -
(4384051*w1*w2*sin(q2)*sin(q3))/2500000000 -
(4384051*w1*w3*sin(q2)*sin(q3))/2500000000 +
(8060997*aw1*cos(q2)*cos(q3)*sin(q2))/5000000000 -
(2919468111*w2^2*cos(q2)*cos(q3))/1250000000000 -
(2919468111*w3^2*cos(q2)*cos(q3))/1250000000000 +
(8060997*w1*w2*cos(q2)^2*cos(q3))/2500000000 +
(8060997*w1*w3*cos(q2)^2*cos(q3))/5000000000 -
(318182949*aw1*cos(q2)*sin(q2)*sin(q3))/2500000000 -
(73963183*w2^2*cos(q2)*sin(q3))/2500000000000 -
(73963183*w2^2*cos(q3)*sin(q2))/2500000000000 -
(73963183*w3^2*cos(q2)*sin(q3))/2500000000000 -
(73963183*w3^2*cos(q3)*sin(q2))/2500000000000 -
(318182949*w1*w2*cos(q2)^2*sin(q3))/1250000000 -
(318182949*w1*w3*cos(q2)^2*sin(q3))/2500000000 +
(2919468111*w2^2*sin(q2)*sin(q3))/1250000000000 +
(2919468111*w3^2*sin(q2)*sin(q3))/1250000000000 - (47369679*aw1*
cos(90)*sin(q2))/5000000000 +
(496811973*aw1*cos(q2)*cos(q3)^2*sin(q2))/312500000000 +
(496811973*aw1*cos(q2)^2*cos(q3)*sin(q3))/312500000000 +
(496811973*w1*w2*cos(q2)^2*cos(q3)^2)/156250000000 +
(496811973*w1*w3*cos(q2)^2*cos(q3)^2)/156250000000 +
(73963183*aw2*cos(q2)*cos(q3))/2500000000000 +
```

```
(73963183*aw3*cos(q2)*cos(q3))/2500000000000 -
(2919468111*aw2*cos(q2)*sin(q3))/1250000000000 -
(2919468111*aw2*cos(q3)*sin(q2))/1250000000000 -
(2919468111*aw3*cos(q2)*sin(q3))/1250000000000 -
(2919468111*aw3*cos(q3)*sin(q2))/1250000000000 +
(4384051*w1*w2*cos(q2)*cos(q3))/2500000000 +
(4384051*w1*w3*cos(q2)*cos(q3))/2500000000 -
(73963183*aw2*sin(q2)*sin(q3))/2500000000000 -
(73963183*aw3*sin(q2)*sin(q3))/2500000000000 -
(35685576739*aw1*cos(q2)*cos(q3)*sin(q2)*sin(q3))/250000000000 -
(35685576739*w1*w2*cos(q2)*cos(q3)^2*sin(q2))/125000000000 -
(35685576739*w1*w2*cos(q2)^2*cos(q3)*sin(q3))/125000000000 -
(35685576739*w1*w3*cos(q2)*cos(q3)^2*sin(q2))/125000000000 -
(35685576739*w1*w3*cos(q2)^2*cos(q3)*sin(q3))/125000000000 -
(2919468111*w2*w3*cos(q2)*cos(q3))/625000000000 -
(73963183*w2*w3*cos(q2)*sin(q3))/1250000000000 -
(73963183*w2*w3*cos(q3)*sin(q2))/1250000000000 +
(2919468111*w2*w3*sin(q2)*sin(q3))/625000000000 -
(318182949*w1*w2*cos(q2)*cos(q3)*sin(q2))/1250000000 -
(318182949*w1*w3*cos(q2)*cos(q3)*sin(q2))/2500000000 -
(8060997*w1*w2*cos(q2)*sin(q2)*sin(q3))/2500000000 -
(8060997*w1*w3*cos(q2)*sin(q2)*sin(q3))/5000000000 -
(496811973*w1*w2*cos(q2)*cos(q3)*sin(q2)*sin(q3))/156250000000 -
(496811973*w1*w3*cos(q2)*cos(q3)*sin(q2)*sin(q3))/156250000000
```

```
T2 =

-(2045290720000*w1^2*cos(q2) - 178453056633*aw3 -
322666312500*aw2*sin(q3)^2 - 391704212158*aw2 -
2015249250*w3^2*cos(q3) + 275233500*w1^2*sin(q2) +
159091474500*w3^2*sin(q3) - 318182949000*aw2*cos(q3) -
159091474500*aw3*cos(q3) - 4030498500*aw2*sin(q3) -
2015249250*aw3*sin(q3) - 322666312500*aw2*cos(q3)^2 -
4030498500*w2*w3*cos(q3) + 318182949000*w2*w3*sin(q3) -
118758816*w1^2*cos(q2)^2 + 118758816*w1^2*sin(q2)^2 -
350970375000*w1^2*cos(q3)^2*sin(q2) -
350970375000*w1^2*sin(q2)*sin(q3)^2 + 887850000*g^2*cos(q2) -
659771200000*g^2*sin(q2) + 2192025500*w1^2*cos(q2)*cos(q3) -
173046867000*w1^2*cos(q2)*sin(q3) -
173046867000*w1^2*cos(q3)*sin(q2) -
2192025500*w1^2*sin(q2)*sin(q3) +
2015249250*w1^2*cos(q2)^2*cos(q3) - 159091474500*w1^2*
sin(90)^2*cos(q2)^2*sin(q3) - 558215700000*g^2*cos(q2)*cos(q3) +
11842419750*aw1*cos(q3)^2*sin(q2) -
7071050000*g^2*cos(q2)*sin(q3) - 7071050000*g^2*cos(q3)*sin(q2)
+ 11842419750*aw1*sin(q2)*sin(q3)^2 +
558215700000*g^2*sin(q2)*sin(q3) +
993623946*w1^2*cos(q2)^2*cos(q3)^2 -
993623946*w1^2*cos(q2)^2*sin(q3)^2 -
993623946*w1^2*cos(q3)^2*sin(q2)^2 -
2015249250*w1^2*cos(q3)^3*sin(q2)^2 +
993623946*w1^2*sin(q2)^2*sin(q3)^2 +
159091474500*w1^2*sin(q2)^2*sin(q3)^3 -
1132162500000*g^2*cos(q2)*cos(q3)^2 -
1132162500000*g^2*cos(q2)*sin(q3)^2 +
213250835899*w1^2*cos(q2)*sin(q2) - 73963183*aw1*cos(q2)*cos(q3)
+ 5838936222*aw1*cos(q2)*sin(q3) +
5838936222*aw1*cos(q3)*sin(q2) + 73963183*aw1*sin(q2)*sin(q3) -
144238428805*w1^2*cos(q2)*sin(q2)*sin(q3)^2 -
2015249250*w1^2*cos(q2)*sin(q2)*sin(q3)^3 +
178427883695*w1^2*cos(q3)*sin(q2)^2*sin(q3) -
2015249250*w1^2*cos(q3)*sin(q2)^2*sin(q3)^2 +
159091474500*w1^2*cos(q3)^2*sin(q2)^2*
sin(q3) - 159091474500*w1^2*cos(q2)*cos(q3)*sin(q2) -
2015249250*w1^2*cos(q2)*sin(q2)*sin(q3) -
501094196195*w1^2*cos(q2)*cos(q3)^2*sin(q2) -
159091474500*w1^2*cos(q2)*cos(q3)^3*sin(q2) -
178427883695*w1^2*cos(q2)^2*cos(q3)*sin(q3) -
3974495784*w1^2*cos(q2)*cos(q3)*sin(q2)*sin(q3) -
159091474500*w1^2*cos(q2)*cos(q3)*sin(q2)*sin(q3)^2 -
2015249250*w1^2*cos(q2)*cos(q3)^2*sin(q2)*sin(q3))/2500000000000
```

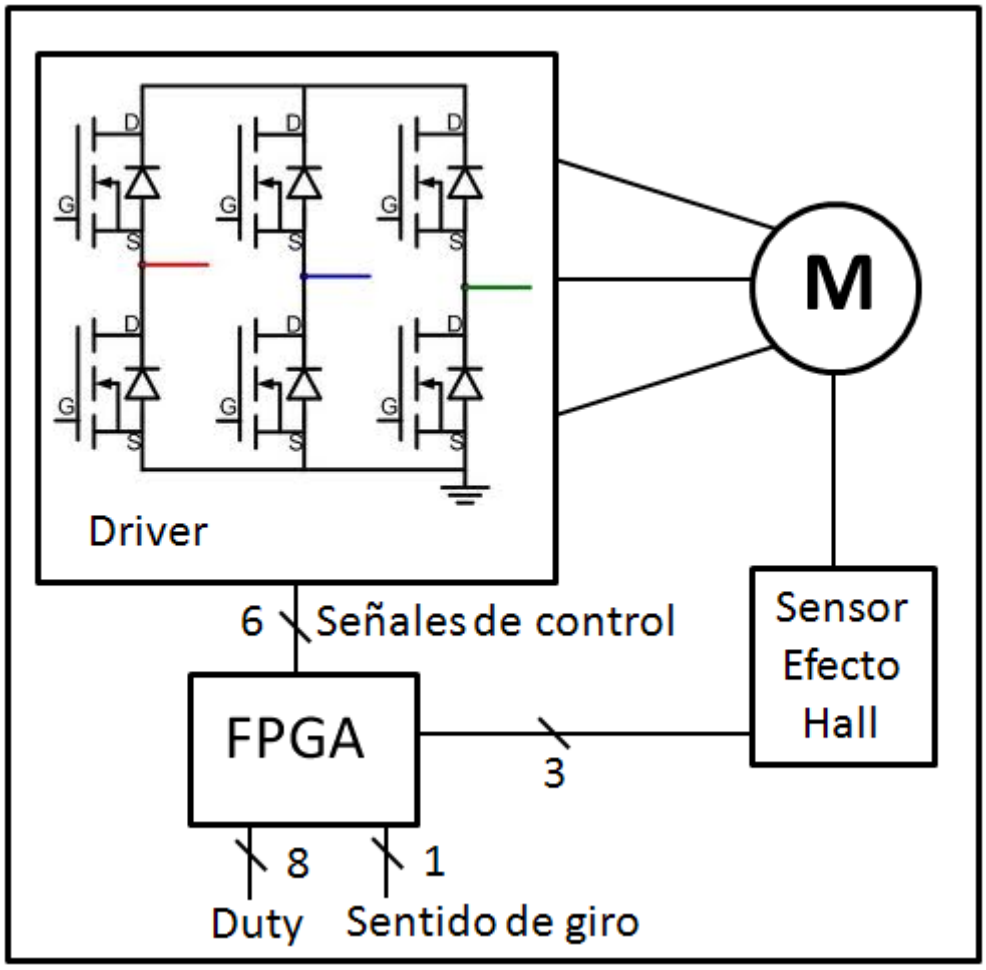
```
T3 = -(2015249250*w2^2*cos(q3) - 178453056633*aw3 -
178453056633*aw2 - 159091474500*w2^2*sin(q3)
     - 159091474500*aw2*cos(q3) - 2015249250*aw2*sin(q3) +
2192025500*w1^2*cos(q2)*cos(q3) -
173046867000*w1^2*cos(q2)*sin(q3)
     - 173046867000*w1^2*cos(q3)*sin(q2) -
2192025500*w1^2*sin(q2)*sin(q3)
     + 2015249250*w1^2*cos(q2)^2*cos(q3) -
159091474500*w1^2*cos(q2)^2*sin(q3)
     - 558215700000*g^2*cos(q2)*cos(q3) -
7071050000*g^2*cos(q2)*sin(q3) - 7071050000*g^2*cos(q3)*sin(q2)
     + 558215700000*g^2*sin(q2)*sin(q3) +
993623946*w1^2*cos(q2)^2*cos(q3)^2 -
993623946*w1^2*cos(q2)^2*sin(q3)^2 -
993623946*w1^2*cos(q3)^2*sin(q2)^2 +
993623946*w1^2*sin(q2)^2*sin(q3)^2 -
73963183*aw1*cos(q2)*cos(q3) + 5838936222*aw1*cos(q2)*sin(q3) +
5838936222*aw1*cos(q3)*sin(q2) + 73963183*aw1*sin(q2)*sin(q3) +
178427883695*w1^2*cos(q2)*sin(q2)*sin(q3)^2 +
178427883695*w1^2*cos(q3)*sin(q2)^2*sin(q3) -
159091474500*w1^2*cos(q2)*cos(q3)*sin(q2) -
2015249250*w1^2*cos(q2)*sin(q2)*sin(q3) -
178427883695*w1^2*cos(q2)*cos(q3)^2*sin(q2) -
178427883695*w1^2*cos(q2)^2*cos(q3)*sin(q3) -
3974495784*w1^2*cos(q2)*cos(q3)*sin(q2)*sin(q3))/2500000000000
```

Como se puede observar, los torques, quedaron en función de los parámetros antes mencionados.

Implementación del controlador del motor para FPGA

Se implementó un controlador para un motor del tipo Brushless utilizando como driver un puente H completo trifásico, como controlador un FPGA para el cual se ah escrito el código correspondiente y un sensor de efecto Hall para realimentar la posición del motor.

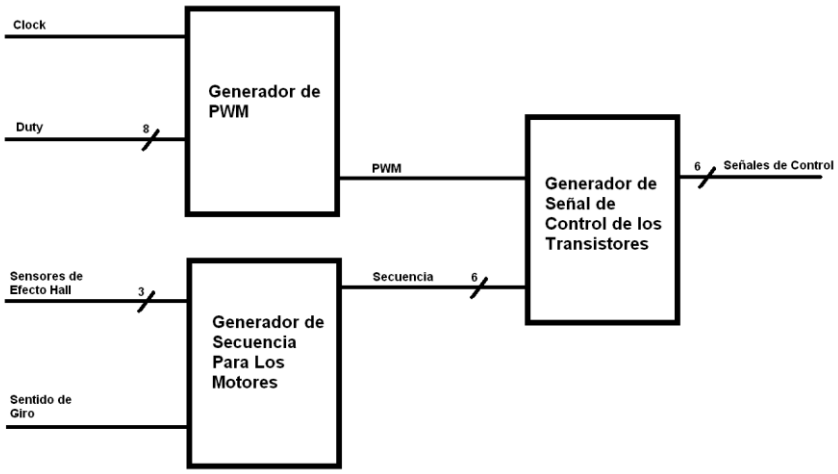
Esquema del sistema:



El sistema que se encarga de comandar el motor se comunica con el FPGA brindándole los datos de duty y sentido de giro. El FPGA generara las señales correspondiente para controlar el puente H. El sensor de efecto Hall realimentara al FPGA la posición del motor, para que este sepa que par de transistores del puente activar.

Las señales de control de los transistores son del tipo PWM para poder variar la velocidad del motor. La velocidad está dada por el ancho de los pulsos del PWM generado. Este ancho es regulado por el sistema de control externo a través de las líneas de “Duty”. Aparte el sistema de control externo debe de indicar el sentido de giro.

El diagrama en bloques del controlador implementado en el FPGA es el siguiente:



Los códigos implementados:

Código que realiza la interconexión entre las 3 unidades creadas (Generador de PWM, Generador de secuencia y generador de señal de control).

```
--************************************************
-- Archivo: Control_Motor.vhd
-- Created : 19/07/12
--************************************************

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

--**************************************
-- Declaro la entidad de Mayor jerarquia
--**************************************

 ENTITY Control_Motor Is

  PORT (DutyControl :in STD_LOGIC_VECTOR (7 downto 0);

       Sensores:in STD_LOGIC_VECTOR (2 downto 0);

       ClockPwm, ResetPwm, Sentido_de_Giro :in  STD_LOGIC;

       Pwm1,Pwm2,Pwm3,Pwm4,Pwm5,Pwm6 :out STD_LOGIC

       );

  END Control_Motor;
```

```
--**************************************
-- Declaro el comportamiento de la entidad
--**************************************

 ARCHITECTURE ArchMotorControl of Control_Motor IS

 --**************************************
-- Defino que componentes voy a utilizar
--**************************************

COMPONENT Secuenciador Is

  PORT (SensoresHall: in  std_logic_vector(2 downto 0);

       Sentido_Giro: in std_logic;

       Q1, Q2, Q3, Q4, Q5, Q6 :out STD_LOGIC

       );

END COMPONENT;


COMPONENT Logica_Transistores IS

PORT (  INl1,INl2,INl3,INl4,INl5,INl6,PWM           :in  STD_LOGIC;

       OUT1,OUT2,OUT3,OUT4,OUT5,OUT6       :out STD_LOGIC

    );

END COMPONENT;


COMPONENT PWM_fpga

PORT (

  clock: in std_logic;
  reset: in std_logic;
  data_value: in std_logic_vector(7 downto 0);
  pwm: out std_logic

);

END COMPONENT;



--**************************************
```

```
-- Declaracion de Señales Internas
--************************************

SIGNAL LinePwm, T1, T2, T3, T4, T5, T6: STD_LOGIC;

SIGNAL SDutyControl : STD_LOGIC_VECTOR (7 downto 0);

SIGNAL SSecuencia : STD_LOGIC_VECTOR (1 downto 0);

SIGNAL SClockPwm, SResetPwm :  STD_LOGIC;

SIGNAL SPwm1,SPwm2,SPwm3,SPwm4,SPwm5,SPwm6 : STD_LOGIC;

 BEGIN


--************************************
-- Describo interconexion de los bloques
--************************************

BloquePwm : PWM_fpga

  PORT MAP

    (

    clock => ClockPwm,
    reset => ResetPwm,
    data_value => DutyControl,
    pwm => LinePwm

    );

 BloqueSecuenciador : Secuenciador

  PORT MAP

    (

    Sentido_Giro => Sentido_de_Giro,
    SensoresHall => Sensores,
    Q1 => T1,
    Q2 => T2,
    Q3 => T3,
    Q4 => T4,
    Q5 => T5,
    Q6 => T6

    );

BloqueLogica : Logica_Transistores
```

```
 PORT MAP

  (

  INl1 => T1,
  INl2 => T2,
  INl3 => T3,
  INl4 => T4,
  INl5 => T5,
  INl6 => T6,
  PWM => LinePwm,
  OUT1 => Pwm1,
  OUT2 => Pwm2,
  OUT3 => Pwm3,
  OUT4 => Pwm4,
  OUT5 => Pwm5,
  OUT6 => Pwm6

  );

END ArchMotorControl;
```

Código generador del PWM.

```
--************************************************
-- Archivo: PWm_fpga.vhd
-- Created : 19/07/12
--************************************************

 library IEEE;

use       ieee.std_logic_1164.all;

use     ieee.std_logic_arith.all;

use       ieee.std_logic_unsigned.all ;


ENTITY PWM_fpga IS

PORT (  clock,reset                 :in  STD_LOGIC;

      Data_value                    :in  std_logic_vector(7 downto 0);

      pwm                   :out STD_LOGIC

 );END PWM_fpga;

ARCHITECTURE arch_pwm OF PWM_fpga IS

 SIGNAL reg_out           : std_logic_vector(7 downto 0);
```

```
SIGNAL cnt_out_int                  : std_logic_vector(7 downto 0);
SIGNAL pwm_int, rco_int     : STD_LOGIC;
 BEGIN
   -- Registro de 8 bits que guarda el valor para determinar el ancho del pulso.
  PROCESS(clock,reg_out,reset)
 BEGIN
          IF (reset ='1') THEN
                    reg_out <="00000000";
          ELSIF (rising_edge(clock)) THEN
                    reg_out <= data_value;
          END IF;
 END PROCESS;
-- Contador up y down de 8 bits. Este contador cuenta con la entrada clock y genera la señal
 -- de cuenta terminal cuando se alcanza el maximo valor del registro o se pasa de 0 a al
maximo valor
 -- en caso de que se decremente. Esta señal se utiliza para generar el PWM.
 PROCESS (clock,cnt_out_int,rco_int,reg_out)
 BEGIN
 IF (rco_int = '1') THEN
               cnt_out_int <= reg_out;
  ELSIF  rising_edge(clock) THEN
        IF  (rco_int  =  '0'  and  pwm_int  ='1'  and  cnt_out_int  <"11111111") THEN
      cnt_out_int <= cnt_out_int+1;
         ELSE
           IF  (rco_int  ='0'  and  pwm_int  ='0'  and  cnt_out_int  > "00000000") THEN
     cnt_out_int <= cnt_out_int-1;
     END IF;
 END IF;
 END IF;
 END PROCESS;
```

```
-- Logica para generar la señal RCO.
 PROCESS(cnt_out_int, rco_int, clock,reset)
 BEGIN
        IF (reset ='1') THEN
                rco_int <='1';
   ELSIF rising_edge(clock) THEN
                IF ((cnt_out_int = "11111111") or (cnt_out_int ="00000000")) THEN
                        rco_int <= '1';
      ELSE
                        rco_int <='0';
      END IF;
    END IF;
END PROCESS;
 -- Cambiamos el estado del flip-flopr para generar el PWM.
  PROCESS (clock,rco_int,reset)
 BEGIN
        IF (reset = '1') THEN
                pwm_int <='0';
        ELSIF rising_edge(rco_int) THEN
                pwm_int <= NOT(pwm_int);
        ELSE
                pwm_int <= pwm_int;
        END IF;
END PROCESS;
 pwm <= pwm_int;
  END arch_pwm;
```

Código generador de secuencias para los motores.

```
--************************************************
-- Archivo: Secuenciador.vhd
```

```
-- Created : 19/07/12
--************************************************

 library IEEE;

USE  ieee.std_logic_1164.all;

USE  ieee.std_logic_arith.all ;


ENTITY Secuenciador Is

 PORT (SensoresHall: in  std_logic_vector(2 downto 0);

      Sentido_Giro: in std_logic;

      Q1, Q2, Q3, Q4, Q5, Q6 :out STD_LOGIC

      );

END Secuenciador;

 ARCHITECTURE ArchSecuenciador of Secuenciador IS

 BEGIN

          PROCESS(SensoresHall,Sentido_Giro)

                    BEGIN

                    IF(Sentido_Giro = '0') THEN -- Para sentido horario

                    CASE SensoresHall IS

                              WHEN "001" =>

                                        Q1 <= '1';
                                        Q3 <= '0';
                                        Q5 <= '0';
                                        Q6 <= '1';
                                        Q4 <= '0';
                                        Q2 <= '0';

                              WHEN "000" =>

                                        Q1 <= '1';
                                        Q3 <= '0';
                                        Q5 <= '0';
                                        Q6 <= '0';
                                        Q4 <= '1';
                                        Q2 <= '0';

                              WHEN "100" =>
```

```
                Q1 <= '0';
                Q3 <= '0';
                Q5 <= '1';
                Q6 <= '0';
                Q4 <= '1';
                Q2 <= '0';

        WHEN "110" =>

                Q1 <= '0';
                Q3 <= '0';
                Q5 <= '1';
                Q6 <= '0';
                Q4 <= '0';
                Q2 <= '1';

        WHEN "111" =>

                Q1 <= '0';
                Q3 <= '1';
                Q5 <= '0';
                Q6 <= '0';
                Q4 <= '0';
                Q2 <= '1';

        WHEN "011" =>

                Q1 <= '0';
                Q3 <= '1';
                Q5 <= '0';
                Q6 <= '1';
                Q4 <= '0';
                Q2 <= '0';

        WHEN OTHERS => NULL;

         END CASE;

END IF;

         IF(Sentido_Giro = '1') THEN -- Para sentido antihorario

         CASE SensoresHall IS

WHEN "011" =>

                Q1 <= '0';
                Q3 <= '0';
                Q5 <= '1';
                Q6 <= '0';
                Q4 <= '1';
```

```
                    Q2 <= '0';

WHEN "111" =>

                    Q1 <= '1';
                    Q3 <= '0';
                    Q5 <= '0';
                    Q6 <= '0';
                    Q4 <= '1';
                    Q2 <= '0';

WHEN "110" =>

                    Q1 <= '1';
                    Q3 <= '0';
                    Q5 <= '0';
                    Q6 <= '1';
                    Q4 <= '0';
                    Q2 <= '0';

WHEN "100" =>

                    Q1 <= '0';
                    Q3 <= '1';
                    Q5 <= '0';
                    Q6 <= '1';
                    Q4 <= '0';
                    Q2 <= '0';

WHEN "000" =>

                    Q1 <= '0';
                    Q3 <= '1';
                    Q5 <= '0';
                    Q6 <= '0';
                    Q4 <= '0';
                    Q2 <= '1';

WHEN "001" =>

                    Q1 <= '0';
                    Q3 <= '0';
                    Q5 <= '1';
                    Q6 <= '0';
                    Q4 <= '0';
                    Q2 <= '1';

WHEN OTHERS => NULL;

END CASE;
```

```
                    END IF;

            END PROCESS;

END ArchSecuenciador;
```

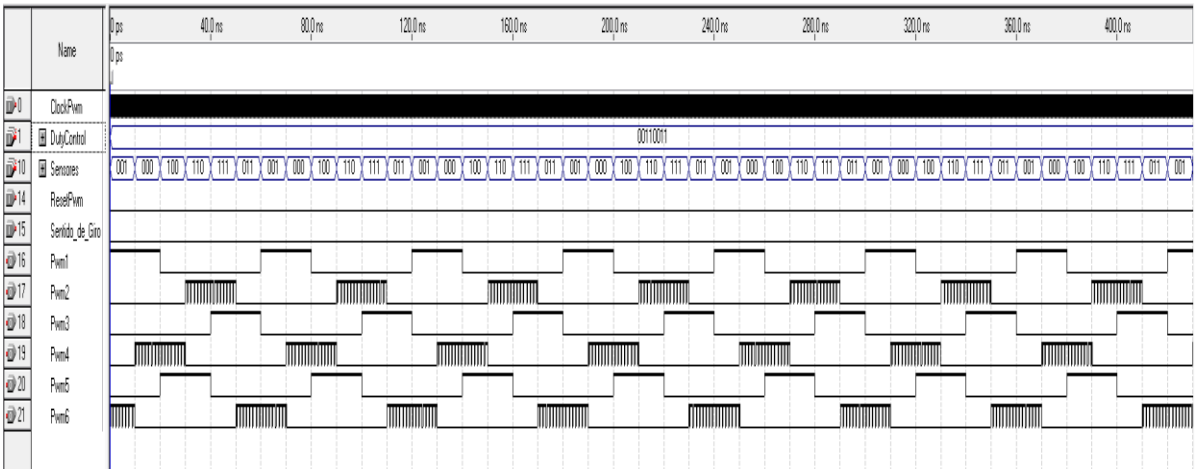
Generador de señales de control de los transistores.

```
--************************************************
-- Archivo: Logica_Transistores.vhd
-- Created : 19/07/12
--************************************************

 library IEEE;

USE  ieee.std_logic_1164.all;

USE  ieee.std_logic_arith.all ;

ENTITY Logica_Transistores IS

PORT (  INl1,INl2,INl3,INl4,INl5,INl6,PWM          :in  STD_LOGIC;

      OUT1,OUT2,OUT3,OUT4,OUT5,OUT6       :out STD_LOGIC

   );

END Logica_Transistores;

 ARCHITECTURE arch_logica OF Logica_Transistores IS

 BEGIN

OUT1<= INl1;

OUT3<= INl3;

OUT5<= INl5;

OUT2<= INl2 and PWM;

OUT4<= INl4 and PWM;

OUT6<= INl6 and PWM;

END arch_logica;
```

Compilación y simulación:

Se utilizó el programa Quartus de Altera para realizar la compilación y simulación del código.

Los resultados obtenidos en la simulación fueron.



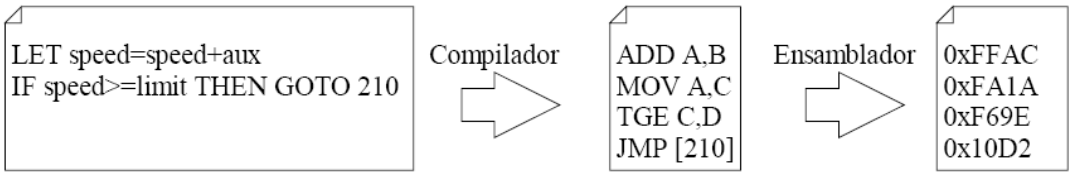
Como se puede observar en la imagen se han graficado los datos del duty, señal del sensor de efecto Hall, el sentido de giro y las 6 señales para el puente H. Se puede observar que todo el sistema cumple con las especificaciones del mismo.

Compilador:

Introducción:

Para el desarrollo del compilador utilizaremos la herramienta ANTLR (ANother Tool for Language Recognition) la cual está escrita en Java, pero acepta múltiples lenguajes de programación.

El compilador es el encargado de llevar un lenguaje de alto nivel a uno de bajo nivel, el cual, en condiciones ideales, debe aprovechar las funciones específicas del hardware con el que se está trabajando para así lograr una codificación óptima. Lugo el ensamblador será el encargado de traducir nuevamente ese lenguaje en código máquina, para ser comprendido por el hardware.

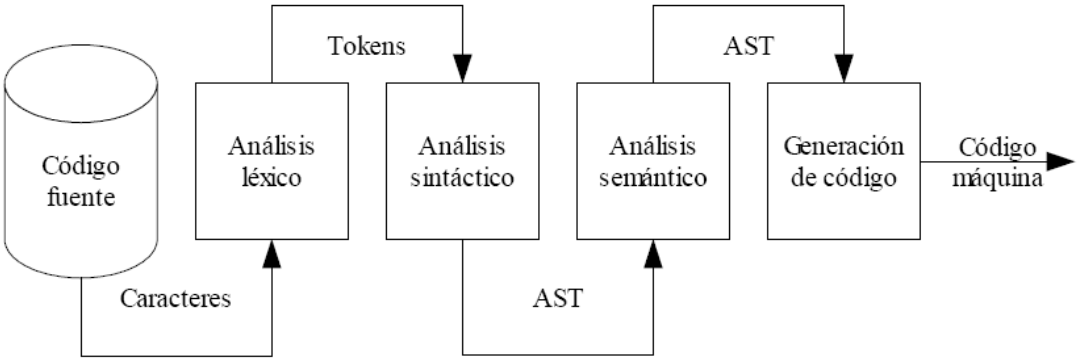


Funcionamiento de un compilador

Funcionamiento de un ensamblador

Fases del proceso de compilación:

Lo primero que debe hacer un compilador es comprobar que la información que se le suministra pertenece a su lenguaje (no hay errores léxicos, sintácticos ni semánticos). Si es así, el intérprete debe representar de alguna manera la información que le se le suministró para poder trabajar con ella, y finalmente traducir dicha información a código máquina.
Un esquema de dicho funcionamiento es el que se muestra en la siguiente figura.



Estructura básica de un compilador

A primera vista podemos distinguir que hay dos tipos de elemento en dicho gráfico: los "elementos activos" (figuras cerradas) y los "flujos de datos", representados como flechas que unen los diferentes elementos activos. Si entre los elementos activos "A" y "B" hay una flecha llamada "C", eso quiere decir que "A" produce el flujo de datos "C", que es usado por "B".
Analicemos brevemente cada elemento y cada flujo:

• Código fuente: Es información almacenada en la memoria de un ordenador. Suele tratarse de uno o varios ficheros de texto, normalmente en el disco duro de la máquina5. En estos ficheros hay cierta información cuyo fin es provocar ciertas acciones en una máquina objetivo (que puede no ser la que está "interpretándolos"). Para ello, los ficheros son leídos del disco y pasados a la memoria, conformando el flujo denominado "Caracteres".

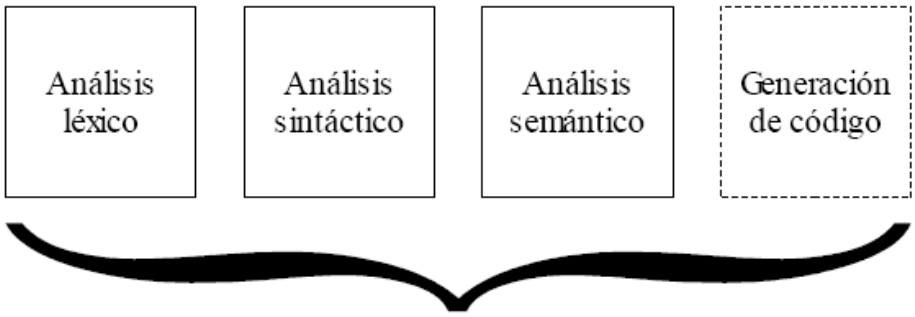
• Análisis léxico: Esta fase tiene que ver con el "vocabulario" del que hablábamos más arriba. El proceso de análisis léxico agrupa los diferentes caracteres de su flujo de entrada en tokens. Los tokens son los símbolos léxicos del lenguaje; se asemejan mucho a las palabras del lenguaje natural. Los tokens están identificados con símbolos (tienen "nombres") y suelen contener información adicional (como la cadena de caracteres que los originó, el fichero en el que están y la línea donde comienzan, etc). Una vez son identificados, son transmitidos al siguiente nivel de análisis. El programa que permite realizar el análisis léxico es un analizador léxico. En inglés se le suele llamar scanner o lexer.

• Análisis sintáctico: En la fase de análisis sintáctico se aplican las reglas sintácticas del lenguaje analizado al flujo de tokens. En caso de no haberse detectado errores, el intérprete representará la información codificada en el código fuente en un Árbol de Sintaxis Abstracta, que no es más que una representación arbórea de los diferentes patrones sintácticos que se han encontrado al realizar el análisis, salvo que los elementos innecesarios (signos de puntuación, paréntesis) son eliminados. En adelante llamaremos AST a los Árboles de Sintaxis Abstracta.
El código que permite realizar el análisis sintáctico se llama "analizador sintáctico". En inglés se le llama parser, que significa "iterador" o directamente analyzer ("analizador").

• Análisis semántico: El análisis semántico del árbol AST empieza por detectar incoherencias a nivel sintáctico en el AST. Si el AST supera esta fase, es corriente enriquecerlo para realizar un nuevo análisis semántico. Es decir, es corriente efectuar varios análisis semánticos, cada uno centrado en aspectos diferentes. Durante éstos análisis el árbol es enriquecido y modificado.
Cualquier herramienta que realice un análisis semántico será llamada "analizador semántico" en este texto. En la bibliografía inglesa suelen referirse a los analizadores semánticos como tree parsers (o "iteradores de árboles").

• Generación de código: En esta fase se utiliza el AST enriquecido, producto del proceso de análisis semántico, para generar código máquina.

Finalmente ANTLR es capaz de actuar a tres niveles a la vez (cuatro si tenemos en cuenta la generación de código):



Comparación de las herramientas

El uso de una sola herramienta para todos los niveles tiene varias ventajas. La más importante es la "estandarización": con ANTLR basta con comprender el paradigma de análisis una vez para poder implementar todas las fases de análisis.

Compilador para nuestro brazo robótico:

En este ejemplo el objetivo será que a partir de nuestro propio lenguaje (código fuente) sea traducido y o analizado para lograr otro lenguaje en este caso ‘C’ para Visual DSP a través de nuestro compilador haciendo uso de sus analizadores (léxico, sintáctico y semántico).

Código:

```
grammar SimpleCalc;

tokens
{

ALIAS='ALIAS';
AND='AND';
ARRAY='ARRAY';
ASSOCIATIVE='ASSOCIATIVE';
BEGIN='BEGIN';
BINDINGS='BINDINGS';
BY = 'BY';
CASE = 'CASE';
CONST = 'CONST';
DEFINITION = 'DEFINITION';
DIV = 'DIV';
DO = 'DO';
ELSE = 'ELSE';
ELSIF = 'ELSIF';
END = 'END';
EXIT = 'EXIT';
FOR = 'FOR';
FROM = 'FROM';
IF = 'IF';
IMPLEMENTATION = 'IMPLEMENTATION';
IMPORT = 'IMPORT';
IN = 'IN';
LOOP = 'LOOP';
MINUS = '-' ;
MOD = 'MOD';
MODULE = 'MODULE';
NOT = 'NOT';
OF = 'OF';
OPAQUE = 'OPAQUE';
OR = 'OR';
PLUS='+';
POINTER = 'POINTER';
PROCEDURE = 'PROCEDURE';
PRODUCT = '*';
RECORD = 'RECORD';
REPEAT = 'REPEAT';
RETURN = 'RETURN';
SET = 'SET';
THEN = 'THEN';
TO = 'TO';
TYPE = 'TYPE';
UNTIL = 'UNTIL';
```

```
VAR = 'VAR';
VARIADIC = 'VARIADIC';
WHILE = 'WHILE';
MOVEXYZ = 'MOVEXYZ';
MOVEX = 'MOVEX';
MOVEY = 'MOVEY';
MOVEZ = 'MOVEZ';
MOVERELX = 'MOVERELX';
MOVERELY = 'MOVERELY';
MOVERELZ = 'MOVERELZ';
START = 'START';
STOP = 'STOP';

}

@header
{
	import java.util.HashMap;
	import java.io.*;
	import java.io.FileWriter;
	import java.io.IOException;
}

@members
{
	Integer i=0;
	String filename = "robotica_tp3.c";
	HashMap variables = new HashMap();//Tabla de java para almacenar variables
	HashMap valores = new HashMap();
	static FileWriter salida;

	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 : ((asig1)|(asig2)|(asig3)|(asig4)|(asig5)|(asig6)|(asig7)|(asig8)|(asig9)|(asig))+;//ver
diagrama que asi puedo reconocer varias expresiones
asig : ID '=' op {System.out.println($ID.text +"="+ $op.value);
variables.put($ID.text, new Integer($op.value));};
//para debug
op returns [int value]
```

```
:
e=factor {$value = $e.value;}
(PLUS e=factor {$value += $e.value;}
| MINUS e=factor {$value -= $e.value;}
| PRODUCT e=factor {$value *= $e.value;}
)*
;
factor returns [int value]
:
NUMBER {$value = Integer.parseInt($NUMBER.text);}//Con esto acepta un entero

        |          ID
{
Integer v = (Integer)variables.get($ID.text);//Con esto acepta letras como enteros
valores.put(i++, new Integer(v));
//System.out.println("valor1="+(Integer)valores.get(i));
if ( v!=null ) $value = v.intValue();
else System.err.println("Variable no definida: "+$ID.text);
};

asig2: MOVEXYZ LEFT_PAREN x=factor COMA y=factor COMA z=factor COMA
vel=factor RIGHT_PAREN
{
        if(i==1){
                String str = "\tmove_xyz(" + x + ',' + y + ',' + z + ',' + vel + ");\n";
                try{
                        salida = new FileWriter(filename, true);
                        BufferedWriter out = new BufferedWriter(salida);
                        out.write(str);
                        out.close();
                }
                catch (IOException e){
                        e.printStackTrace();
                        return;
                }

                System.out.println("move_xyz(" + x + ',' + y + ',' + z + ',' + vel + ");");
        }
};

asig3: MOVEX LEFT_PAREN x=factor COMA vel=factor RIGHT_PAREN
{
        if(i==1){
                String str = "\tmove_x(" + x + ','  + vel + ");\n";
                try{
                        salida = new FileWriter(filename, true);
                        BufferedWriter out = new BufferedWriter(salida);
                        out.write(str);
                        out.close();
                }
                catch (IOException e){
                        e.printStackTrace();
                        return;
                }

                System.out.println("move_x(" + x + ','  + vel + ");\n");
        }
```

```
};

asig4: MOVEY LEFT_PAREN y=factor COMA vel=factor RIGHT_PAREN
{
	if(i==1){
		String str = "\tmove_y(" + y + ','  + vel + ");\n";
		try{
			salida = new FileWriter(filename, true);
			BufferedWriter out = new BufferedWriter(salida);
			out.write(str);
			out.close();
		}
		catch (IOException e){
			e.printStackTrace();
			return;
		}

		System.out.println("move_y(" + y + ','  + vel + ");\n");

	}
};

asig5: MOVEZ LEFT_PAREN z=factor COMA vel=factor RIGHT_PAREN
{
	if(i==1){
		String str = "\tmove_z(" + z + ','  + vel + ");\n";
		try{
			salida = new FileWriter(filename, true);
			BufferedWriter out = new BufferedWriter(salida);
			out.write(str);
			out.close();
		}
		catch (IOException e){
			e.printStackTrace();
			return;
		}

		System.out.println("move_z(" + z + ','  + vel + ");\n");
	}
};

asig6: MOVERELX LEFT_PAREN x=factor COMA vel=factor RIGHT_PAREN
{
	if(i==1){
		String str = "\tmoverel_x(" + x + ','  + vel + ");\n";
		try{
			salida = new FileWriter(filename, true);
			BufferedWriter out = new BufferedWriter(salida);
			out.write(str);
			out.close();
		}
		catch (IOException e){
			e.printStackTrace();
			return;
		}

		System.out.println("moverel_x(" + x + ','  + vel + ");\n");
```

```
        }
};

asig7: MOVERELY LEFT_PAREN y=factor COMA vel=factor RIGHT_PAREN
{
        if(i==1){
                String str = "\tmoverel_y(" + y + ','  + vel + ");\n";
                try{
                        salida = new FileWriter(filename, true);
                        BufferedWriter out = new BufferedWriter(salida);
                        out.write(str);
                        out.close();
                }
                catch (IOException e){
                        e.printStackTrace();
                        return;
                }

                System.out.println("moverel_y(" + y + ','  + vel + ");\n");
        }
};

asig8: MOVERELZ LEFT_PAREN z=factor COMA vel=factor RIGHT_PAREN
{
        if(i==1){
                String str = "\tmoverel_z(" + z + ','  + vel + ");\n";
                try{
                        salida = new FileWriter(filename, true);
                        BufferedWriter out = new BufferedWriter(salida);
                        out.write(str);
                        out.close();
                }
                catch (IOException e){
                        e.printStackTrace();
                        return;
                }

                System.out.println("moverel_z(" + z + ','  + vel + ");\n");
        }
};

asig1   :       START
{
        String str =
"#include <stdio.h>\n"+
"#include  \"MyDSPLibrary.h\"\n"+
"#include <string.h>\n"+
"\n"+
"#define  RESOLUCION  100\n"+
"\n"+
"static unsigned int numero_mov=1;\n"+
"static double x0=0, y0=0, z0=0;\n"+
"\n"+
"void move_xyz(double xf, double yf, double zf, unsigned long vel);\n"+
"void move_x(double xf, unsigned long vel);\n"+
"void move_y(double yf, unsigned long vel);\n"+
"void move_z(double zf, unsigned long vel);\n"+
```

```
"void moverel_xyz(double xrel, double yrel, double zrel, unsigned long vel);\n"+
"void moverel_x(double xrel, unsigned long vel);\n"+
"void moverel_y(double yrel, unsigned long vel);\n"+
"void moverel_z(double zrel, unsigned long vel);\n"+
"\n"+
"int main(void)\n"+
"{\n";

        try{
                salida = new FileWriter(filename, true);
                BufferedWriter out = new BufferedWriter(salida);
                out.write(str);
                out.close();
                i = 1;
        }
        catch (IOException e){
                e.printStackTrace();
                return;
}
};

asig9   :          STOP
{
        String str =
"\treturn 0;\n"+
"}\n"+
"\n"+
"void move_xyz(double xf, double yf, double zf, unsigned long vel)\n"+
"{\n"+
"\tdouble x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];\n"+
"\tdouble v[3], t, t_max;\n"+
"\tchar file_x[15], file_y[15], file_z[15];\n"+
"\tunsigned int i=0;\n"+
"\tsprintf(file_x,\"..\\\\x\%u.dat\",numero_mov);\n"+
"\tsprintf(file_y,\"..\\\\y\%u.dat\",numero_mov);\n"+
"\tsprintf(file_z,\"..\\\\z\%u.dat\",numero_mov);\n"+
"\tnumero_mov++;\n"+
"\tv[0] = xf-x0;\n"+
"\tv[1] = yf-y0;\n"+
"\tv[2] = zf-z0;\n"+
"\tt_max = (xf-x0)/v[0];\n"+
"\tt=t_max/RESOLUCION;\n"+
"\twhile(i<RESOLUCION)\n"+
"\t{\n"+
"\t\tx[i] = x0 + v[0] * t;\n"+
"\t\ty[i] = y0 + v[1] * t;\n"+
"\t\tz[i] = z0 + v[2] * t;\n"+
"\t\tt+=t_max/RESOLUCION;\n"+
"\t\ti++;\n"+
"\t}\n"+
"\tx0 = xf;\n"+
"\ty0 = yf;\n"+
"\tz0 = zf;\n"+
"\tMyWriteFile( file_x, x ,RESOLUCION);\n"+
"\tMyWriteFile( file_y, y, RESOLUCION);\n"+
"\tMyWriteFile( file_z, z, RESOLUCION);\n"+
"}\n"+
```

```
"\n"+
"void move_x(double xf, unsigned long vel)\n"+
"{\n"+
"\tdouble x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];\n"+
"\tdouble t, t  max;\n"+
"\tchar file_x[15], file_y[15], file_z[15];\n"+
"\tunsigned int i=0;\n"+
"\tsprintf(file_x,\"..\\\\x\%u.dat\",numero_mov);\n"+
"\tsprintf(file_y,\"..\\\\y\%u.dat\",numero_mov);\n"+
"\tsprintf(file_z,\"..\\\\z\%u.dat\",numero_mov);\n"+
"\tnumero_mov++;\n"+
"\tt_max = xf-x0;\n"+
"\tt=t_max/RESOLUCION;\n"+
"\twhile(i<RESOLUCION)\n"+
"\t{\n"+
"\t\tx[i] = x0+t;\n"+
"\t\ty[i] = y0;\n"+
"\t\tz[i] = z0;\n"+
"\t\tt+=t_max/RESOLUCION;\n"+
"\t\ti++;\n"+
"\t}\n"+
"\tx0 = xf;\n"+
"\tMyWriteFile( file_x, x ,RESOLUCION);\n"+
"\tMyWriteFile( file_y, y, RESOLUCION);\n"+
"\tMyWriteFile( file_z, z, RESOLUCION);\n"+
"}\n"+
"\n"+
"void move_y(double yf, unsigned long vel)\n"+
"{\n"+
"\tdouble x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];\n"+
"\tdouble t, t_max;\n"+
"\tchar file_x[15], file_y[15], file_z[15];\n"+
"\tunsigned int i=0;\n"+
"\tsprintf(file_x,\"..\\\\x\%u.dat\",numero_mov);\n"+
"\tsprintf(file_y,\"..\\\\y\%u.dat\",numero_mov);\n"+
"\tsprintf(file_z,\"..\\\\z\%u.dat\",numero_mov);\n"+
"\tnumero_mov++;\n"+
"\tt_max = yf-y0;\n"+
"\tt=t_max/RESOLUCION;\n"+
"\twhile(i<RESOLUCION)\n"+
"\t{\n"+
"\t\tx[i] = x0;\n"+
"\t\ty[i] = y0+t;\n"+
"\t\tz[i] = z0;\n"+
"\t\tt+=t_max/RESOLUCION;\n"+
"\t\ti++;\n"+
"\t}\n"+
"\ty0 = yf;\n"+
"\tMyWriteFile( file_x, x ,RESOLUCION);\n"+
"\tMyWriteFile( file_y, y, RESOLUCION);\n"+
"\tMyWriteFile( file_z, z, RESOLUCION);\n"+
"}\n"+
"\n"+
"void move_z(double zf, unsigned long vel)\n"+
"{\n"+
"\tdouble x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];\n"+
"\tdouble t, t_max;\n"+
```

```
"\tchar file_x[15], file_y[15], file_z[15];\n"+
"\tunsigned int i=0;\n"+
"\tsprintf(file_x,\"..\\\\x\%u.dat\",numero_mov);\n"+
"\tsprintf(file_y,\"..\\\\y\%u.dat\",numero_mov);\n"+
"\tsprintf(file  z,\"..\\\\z\%u.dat\",numero_mov);\n"+
"\tnumero_mov++;\n"+
"\tt_max = zf-z0;\n"+
"\tt=t_max/RESOLUCION;\n"+
"\twhile(i<RESOLUCION)\n"+
"\t{\n"+
"\t\tx[i] = x0;\n"+
"\t\ty[i] = y0;\n"+
"\t\tz[i] = z0+t;\n"+
"\t\tt+=t_max/RESOLUCION;\n"+
"\t\ti++;\n"+
"\t}\n"+
"\tz0 = zf;\n"+
"\tMyWriteFile( file_x, x ,RESOLUCION);\n"+
"\tMyWriteFile( file_y, y, RESOLUCION);\n"+
"\tMyWriteFile( file_z, z, RESOLUCION);\n"+
"}\n"+
"\n"+
"void moverel_xyz(double xrel, double yrel, double zrel, unsigned long vel)\n"+
"{\n"+
"\tdouble x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];\n"+
"\tdouble v[3], t, t_max, xf, yf, zf;\n"+
"\tchar file_x[15], file_y[15], file_z[15];\n"+
"\tunsigned int i=0;\n"+
"\tsprintf(file_x,\"..\\\\x\%u.dat\",numero_mov);\n"+
"\tsprintf(file_y,\"..\\\\y\%u.dat\",numero_mov);\n"+
"\tsprintf(file_z,\"..\\\\z\%u.dat\",numero_mov);\n"+
"\tnumero_mov++;\n"+
"\txf = x0+xrel;\n"+
"\tyf = y0+yrel;\n"+
"\tzf = z0+zrel;\n"+
"\tv[0] = xf-x0;\n"+
"\tv[1] = yf-y0;\n"+
"\tv[2] = zf-z0;\n"+
"\tt_max = (xf-x0)/v[0];\n"+
"\tt=t_max/RESOLUCION;\n"+
"\twhile(i<RESOLUCION)\n"+
"\t{\n"+
"\t\tx[i] = x0 + v[0] * t;\n"+
"\t\ty[i] = y0 + v[1] * t;\n"+
"\t\tz[i] = z0 + v[2] * t;\n"+
"\t\tt+=t  max/RESOLUCION;\n"+
"\t\ti++;\n"+
"\t}\n"+
"\tx0 = xf;\n"+
"\ty0 = yf;\n"+
"\tz0 = zf;\n"+
"\tMyWriteFile( file_x, x ,RESOLUCION);\n"+
"\tMyWriteFile( file_y, y, RESOLUCION);\n"+
"\tMyWriteFile( file_z, z, RESOLUCION);\n"+
"}\n"+
"\n"+
"void moverel_x(double xrel, unsigned long vel)\n"+
```

```
"{\n"+
"\tdouble x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];\n"+
"\tdouble t, t_max, xf;\n"+
"\tchar file_x[15], file_y[15], file_z[15];\n"+
"\tunsigned int i=0;\n"+
"\tsprintf(file_x,\"..\\\\x\%u.dat\",numero_mov);\n"+
"\tsprintf(file_y,\"..\\\\y\%u.dat\",numero_mov);\n"+
"\tsprintf(file_z,\"..\\\\z\%u.dat\",numero_mov);\n"+
"\tnumero_mov++;\n"+
"\txf = x0+xrel;\n"+
"\tt_max = xf-x0;\n"+
"\tt=t_max/RESOLUCION;\n"+
"\twhile(i<RESOLUCION)\n"+
"\t{\n"+
"\t\tx[i] = t+x0;\n"+
"\t\ty[i] = y0;\n"+
"\t\tz[i] = z0;\n"+
"\t\tt+=t_max/RESOLUCION;\n"+
"\t\ti++;\n"+
"\t}\n"+
"\tx0 = xf;\n"+
"\tMyWriteFile( file_x, x ,RESOLUCION);\n"+
"\tMyWriteFile( file_y, y, RESOLUCION);\n"+
"\tMyWriteFile( file_z, z, RESOLUCION);\n"+
"}\n"+
"\n"+
"void moverel_y(double yrel, unsigned long vel)\n"+
"{\n"+
"\tdouble x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];\n"+
"\tdouble t, t_max, yf;\n"+
"\tchar file_x[15], file_y[15], file_z[15];\n"+
"\tunsigned int i=0;\n"+
"\tsprintf(file_x,\"..\\\\x\%u.dat\",numero_mov);\n"+
"\tsprintf(file_y,\"..\\\\y\%u.dat\",numero_mov);\n"+
"\tsprintf(file_z,\"..\\\\z\%u.dat\",numero_mov);\n"+
"\tnumero_mov++;\n"+
"\tyf = y0+yrel;\n"+
"\tt_max = yf-y0;\n"+
"\tt=t_max/RESOLUCION;\n"+
"\twhile(i<RESOLUCION)\n"+
"\t{\n"+
"\t\tx[i] = x0;\n"+
"\t\ty[i] = t+y0;\n"+
"\t\tz[i] = z0;\n"+
"\t\tt+=t_max/RESOLUCION;\n"+
"\t\ti++;\n"+
"\t}\n"+
"\ty0 = yf;\n"+
"\tMyWriteFile( file_x, x ,RESOLUCION);\n"+
"\tMyWriteFile( file_y, y, RESOLUCION);\n"+
"\tMyWriteFile( file_z, z, RESOLUCION);\n"+
"}\n"+
"\n"+
"void moverel_z(double zrel, unsigned long vel)\n"+
"{\n"+
"\tdouble x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];\n"+
"\tdouble t, t_max, zf;\n"+
```

```
"\tchar file_x[15], file_y[15], file_z[15];\n"+
"\tunsigned int i=0;\n"+
"\tsprintf(file_x,\"..\\\\x\%u.dat\",numero_mov);\n"+
"\tsprintf(file_y,\"..\\\\y\%u.dat\",numero_mov);\n"+
"\tsprintf(file  z,\"..\\\\z\%u.dat\",numero_mov);\n"+
"\tnumero_mov++;\n"+
"\tzf = z0 + zrel;\n"+
"\tt_max= zf-z0;\n"+
"\tt=t_max/RESOLUCION;\n"+
"\twhile(i<RESOLUCION)\n"+
"\t{\n"+
"\t\tx[i] = x0;\n"+
"\t\ty[i] = y0;\n"+
"\t\tz[i] = t+z0;\n"+
"\t\tt+=t_max/RESOLUCION;\n"+
"\t\ti++;\n"+
"\t}\n"+
"\tz0 = zf;\n"+
"\tMyWriteFile( file_x, x ,RESOLUCION);\n"+
"\tMyWriteFile( file_y, y, RESOLUCION);\n"+
"\tMyWriteFile( file_z, z, RESOLUCION);\n"+
"}\n";

        try{
                salida = new FileWriter(filename, true);
                BufferedWriter out = new BufferedWriter(salida);
                out.write(str);
                out.close();
                i = 0;
        }
        catch (IOException e){
                e.printStackTrace();
                return;
        }
};

/*------------------------------------------------------------------
* LEXER RULES
*------------------------------------------------------------------*/
ID : ('a'..'z'|'A'..'Z')+ ;
WHITESPACE : ( '\t' | ' ' | '\r' | '\n'| '\u000C' )+ { $channel = HIDDEN;} ;
LEFT_PAREN: '(';
//LIST: 'list';
//PRINT: 'print';
RIGHT_PAREN: ')';
COMA :',';
//VARIABLES: 'variables'; // for list command
//SIGN: '+' | '-';
NUMBER: FLOAT|INTEGER;
fragment FLOAT:INTEGER '.' '0'..'9'+;
fragment INTEGER: '0' | '1'..'9' '0'..'9'*;
//fragment INTEGER: '0' | SIGN? '1'..'9' '0'..'9'*;
NAME: LETTER (LETTER | DIGIT | '_')*;
STRING_LITERAL: '"' NONCONTROL_CHAR* '"';
fragment NONCONTROL_CHAR: LETTER | DIGIT | SYMBOL | SPACE;
fragment LETTER: LOWER | UPPER;
fragment LOWER: 'a'..'z';
```

```
fragment UPPER: 'A'..'Z';
fragment DIGIT: '0'..'9';
fragment SPACE: ' ' | '\t';
fragment SYMBOL: '!' | '#'..'/' | ':'..'@' | '['..'`' | '{'..'~';
```

Como puede observarse estos serán los comandos a utilizar para lograr el movimiento de nuestro robot (funciones interpretadas por el compilador), las posiciones y velocidades son a modo de ejemplo:

- START
- MOVEXYZ(10,5,4,1)
- MOVEXYZ(20,35,44,1)
- MOVEX(10,1)
- MOVEY(5,1)
- MOVEZ(4,1)
- MOVERELX(10,1)
- MOVERELY(-2,1)
- MOVERELZ(20,1)
- STOP

Éste será el código generado por nuestro compilador, quedando listo para poder compilar el C y ser descargado en el DSP de Blackfinn:

```
#include <stdio.h>
#include  "MyDSPLibrary.h"
#include <string.h>

#define  RESOLUCION  100

static unsigned int numero_mov=1;
static double x0=0, y0=0, z0=0;

void move_xyz(double xf, double yf, double zf, unsigned long vel);
void move_x(double xf, unsigned long vel);
void move_y(double yf, unsigned long vel);
void move_z(double zf, unsigned long vel);
void moverel_xyz(double xrel, double yrel, double zrel, unsigned long vel);
void moverel_x(double xrel, unsigned long vel);
void moverel_y(double yrel, unsigned long vel);
void moverel_z(double zrel, unsigned long vel);
```

```
int main(void)
{
	move_xyz(10,5,4,1);
	move_xyz(20,35,44,1);
	move  x(10,1);
	move_y(5,1);
	move_z(4,1);
	moverel_x(10,1);
	moverel_y(0,1);
	moverel_z(20,1);
	return 0;
}

void move_xyz(double xf, double yf, double zf, unsigned long vel)
{
	double x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];
	double v[3], t, t_max;
	char file_x[15], file_y[15], file_z[15];
	unsigned int i=0;
	sprintf(file_x,"..\\x%u.dat",numero_mov);
	sprintf(file_y,"..\\y%u.dat",numero_mov);
	sprintf(file_z,"..\\z%u.dat",numero_mov);
	numero_mov++;
	v[0] = xf-x0;
	v[1] = yf-y0;
	v[2] = zf-z0;
	t_max = (xf-x0)/v[0];
	t=t_max/RESOLUCION;
	while(i<RESOLUCION)
	{
		x[i] = x0 + v[0] * t;
		y[i] = y0 + v[1] * t;
		z[i] = z0 + v[2] * t;
		t+=t_max/RESOLUCION;
		i++;
	}
	x0 = xf;
	y0 = yf;
	z0 = zf;
	MyWriteFile( file_x, x ,RESOLUCION);
	MyWriteFile( file_y, y, RESOLUCION);
	MyWriteFile( file_z, z, RESOLUCION);
}
```

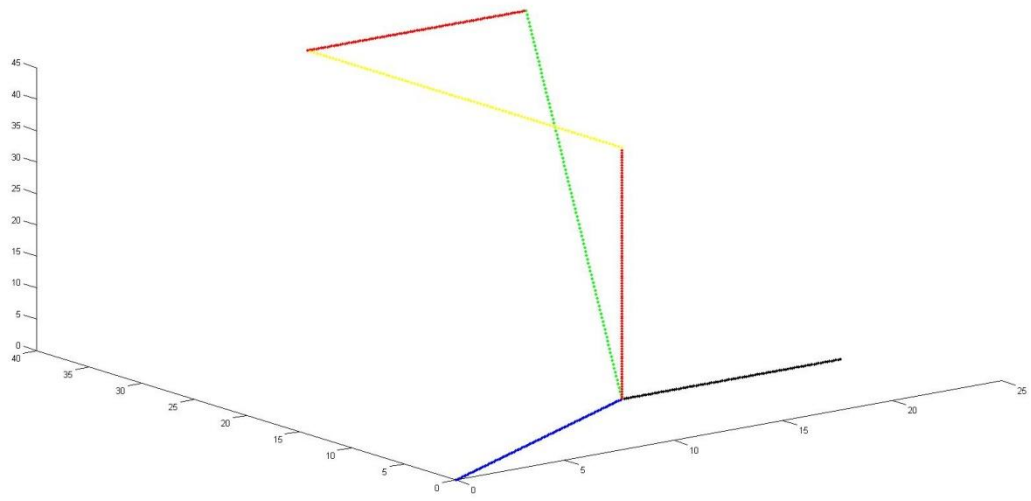
```
void move_x(double xf, unsigned long vel)
{
	double x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];
	double t, t_max;
	char file  x[15], file_y[15], file_z[15];
	unsigned int i=0;
	sprintf(file_x,"..\\x%u.dat",numero_mov);
	sprintf(file_y,"..\\y%u.dat",numero_mov);
	sprintf(file_z,"..\\z%u.dat",numero_mov);
	numero_mov++;
	t_max = xf-x0;
	t=t_max/RESOLUCION;
	while(i<RESOLUCION)
	{
		x[i] = x0+t;
		y[i] = y0;
		z[i] = z0;
		t+=t_max/RESOLUCION;
		i++;
	}
	x0 = xf;
	MyWriteFile( file_x, x ,RESOLUCION);
	MyWriteFile( file_y, y, RESOLUCION);
	MyWriteFile( file_z, z, RESOLUCION);
}

void move_y(double yf, unsigned long vel)
{
	double x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];
	double t, t_max;
	char file_x[15], file_y[15], file_z[15];
	unsigned int i=0;
	sprintf(file_x,"..\\x%u.dat",numero_mov);
	sprintf(file_y,"..\\y%u.dat",numero_mov);
	sprintf(file_z,"..\\z%u.dat",numero_mov);
	numero_mov++;
	t_max = yf-y0;
	t=t_max/RESOLUCION;
	while(i<RESOLUCION)
	{
		x[i] = x0;
		y[i] = y0+t;
		z[i] = z0;
		t+=t_max/RESOLUCION;
		i++;
	}
	y0 = yf;
	MyWriteFile( file_x, x ,RESOLUCION);
	MyWriteFile( file_y, y, RESOLUCION);
	MyWriteFile( file_z, z, RESOLUCION);
}
```

```
void move_z(double zf, unsigned long vel)
{
	double x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];
	double t, t_max;
	char file  x[15], file_y[15], file_z[15];
	unsigned int i=0;
	sprintf(file_x,"..\\x%u.dat",numero_mov);
	sprintf(file_y,"..\\y%u.dat",numero_mov);
	sprintf(file_z,"..\\z%u.dat",numero_mov);
	numero_mov++;
	t_max = zf-z0;
	t=t_max/RESOLUCION;
	while(i<RESOLUCION)
	{
		x[i] = x0;
		y[i] = y0;
		z[i] = z0+t;
		t+=t_max/RESOLUCION;
		i++;
	}
	z0 = zf;
	MyWriteFile( file_x, x ,RESOLUCION);
	MyWriteFile( file_y, y, RESOLUCION);
	MyWriteFile( file_z, z, RESOLUCION);
}

void moverel_xyz(double xrel, double yrel, double zrel, unsigned long vel)
{
	double x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];
	double v[3], t, t_max, xf, yf, zf;
	char file_x[15], file_y[15], file_z[15];
	unsigned int i=0;
	sprintf(file_x,"..\\x%u.dat",numero_mov);
	sprintf(file_y,"..\\y%u.dat",numero_mov);
	sprintf(file_z,"..\\z%u.dat",numero_mov);
	numero_mov++;
	xf = x0+xrel;
	yf = y0+yrel;
	zf = z0+zrel;
	v[0] = xf-x0;
	v[1] = yf-y0;
	v[2] = zf-z0;
	t_max = (xf-x0)/v[0];
	t=t_max/RESOLUCION;
	while(i<RESOLUCION)
	{
		x[i] = x0 + v[0] * t;
		y[i] = y0 + v[1] * t;
		z[i] = z0 + v[2] * t;
		t+=t_max/RESOLUCION;
		i++;
	}
	x0 = xf;
	y0 = yf;
	z0 = zf;
	MyWriteFile( file_x, x ,RESOLUCION);
	MyWriteFile( file_y, y, RESOLUCION);
```

```
	MyWriteFile( file_z, z, RESOLUCION);
}

void moverel_x(double xrel, unsigned long vel)
{
	double x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];
	double t, t_max, xf;
	char file_x[15], file_y[15], file_z[15];
	unsigned int i=0;
	sprintf(file_x,"..\\x%u.dat",numero_mov);
	sprintf(file_y,"..\\y%u.dat",numero_mov);
	sprintf(file_z,"..\\z%u.dat",numero_mov);
	numero_mov++;
	xf = x0+xrel;
	t_max = xf-x0;
	t=t_max/RESOLUCION;
	while(i<RESOLUCION)
	{
		x[i] = t+x0;
		y[i] = y0;
		z[i] = z0;
		t+=t_max/RESOLUCION;
		i++;
	}
	x0 = xf;
	MyWriteFile( file_x, x ,RESOLUCION);
	MyWriteFile( file_y, y, RESOLUCION);
	MyWriteFile( file_z, z, RESOLUCION);
}

void moverel_y(double yrel, unsigned long vel)
{
	double x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];
	double t, t_max, yf;
	char file_x[15], file_y[15], file_z[15];
	unsigned int i=0;
	sprintf(file_x,"..\\x%u.dat",numero_mov);
	sprintf(file_y,"..\\y%u.dat",numero_mov);
	sprintf(file_z,"..\\z%u.dat",numero_mov);
	numero_mov++;
	yf = y0+yrel;
	t_max = yf-y0;
	t=t_max/RESOLUCION;
	while(i<RESOLUCION)
	{
		x[i] = x0;
		y[i] = t+y0;
		z[i] = z0;
		t+=t_max/RESOLUCION;
		i++;
	}
	y0 = yf;
	MyWriteFile( file_x, x ,RESOLUCION);
	MyWriteFile( file_y, y, RESOLUCION);
	MyWriteFile( file_z, z, RESOLUCION);
}
```

```
void moverel_z(double zrel, unsigned long vel)
{
	double x[RESOLUCION], y[RESOLUCION], z[RESOLUCION];
	double t, t_max, zf;
	char file  x[15], file_y[15], file_z[15];
	unsigned int i=0;
	sprintf(file_x,"..\\x%u.dat",numero_mov);
	sprintf(file_y,"..\\y%u.dat",numero_mov);
	sprintf(file_z,"..\\z%u.dat",numero_mov);
	numero_mov++;
	zf = z0 + zrel;
	t_max= zf-z0;
	t=t_max/RESOLUCION;
	while(i<RESOLUCION)
	{
		x[i] = x0;
		y[i] = y0;
		z[i] = t+z0;
		t+=t_max/RESOLUCION;
		i++;
	}
	z0 = zf;
	MyWriteFile( file_x, x ,RESOLUCION);
	MyWriteFile( file_y, y, RESOLUCION);
	MyWriteFile( file_z, z, RESOLUCION);
}
```

Una vez compilado nuestro código con nuestro compilador, descargamos el .c generado en el DSP de blackfinn, utilizando el Visual DSP. Antes de probar el hardware se procede a simular el dispositivo para ver si obtenemos el comportamiento deseado, para ello levantamos los archivos generados por el Visual DSP en el Matlab y se procede a graficar las trayectorias realizadas por el robot. El resultado es el siguiente:



Como se puede observar logramos tener una gráfica de la trayectoria, la cual concuerda con lo pedido al robot. Dado que a este le dimos como parámetro una posición en el espacio y logramos describir una trayectoria, logramos plantear la cinemática inversa del robot. Quedando solamente para un posterior estudio el cálculo de la matriz homogénea para calcular los movimientos articulares efectivos para realizar dicha trayectoria.

Script utilizado en Matlab para la simulación:

```
clc;
close all;
clear;


file_x = 'C:\VisualDSP\TP3Robo\x1.dat';
file_y = 'C:\VisualDSP\TP3Robo\y1.dat';
file_z = 'C:\VisualDSP\TP3Robo\z1.dat';

x = func_FileBinary2Signal(file_x,'double') ;
y = func_FileBinary2Signal(file_y,'double') ;
z = func_FileBinary2Signal(file_z,'double') ;

plot3(0,0,0,'.-b');
hold on;
k =1;
for k = 1: 100
    plot3(x(1,k),y(1,k),z(1,k),'.-b');
end


file_x = 'C:\VisualDSP\TP3Robo\x2.dat';
file_y = 'C:\VisualDSP\TP3Robo\y2.dat';
file_z = 'C:\VisualDSP\TP3Robo\z2.dat';

x = func_FileBinary2Signal(file_x,'double') ;
y = func_FileBinary2Signal(file_y,'double') ;
z = func_FileBinary2Signal(file_z,'double') ;

k =1;
for k = 1: 100
    plot3(x(1,k),y(1,k),z(1,k),'.-g');
end


file_x = 'C:\VisualDSP\TP3Robo\x3.dat';
file_y = 'C:\VisualDSP\TP3Robo\y3.dat';
file_z = 'C:\VisualDSP\TP3Robo\z3.dat';

x = func_FileBinary2Signal(file_x,'double') ;
y = func_FileBinary2Signal(file_y,'double') ;
z = func_FileBinary2Signal(file_z,'double') ;

k =1;
for k = 1: 100
    plot3(x(1,k),y(1,k),z(1,k),'.-r');
end


file_x = 'C:\VisualDSP\TP3Robo\x4.dat';
```

```
file_y = 'C:\VisualDSP\TP3Robo\y4.dat';
file_z = 'C:\VisualDSP\TP3Robo\z4.dat';

x = func_FileBinary2Signal(file_x,'double') ;
y = func_FileBinary2Signal(file_y,'double') ;
z = func_FileBinary2Signal(file_z,'double') ;

k =1;
for k = 1: 100
    plot3(x(1,k),y(1,k),z(1,k),'.-y');
end


file_x = 'C:\VisualDSP\TP3Robo\x5.dat';
file_y = 'C:\VisualDSP\TP3Robo\y5.dat';
file_z = 'C:\VisualDSP\TP3Robo\z5.dat';

x = func_FileBinary2Signal(file_x,'double') ;
y = func_FileBinary2Signal(file_y,'double') ;
z = func_FileBinary2Signal(file_z,'double') ;

k =1;
for k = 1: 100
    plot3(x(1,k),y(1,k),z(1,k),'.r');
end


file_x = 'C:\VisualDSP\TP3Robo\x6.dat';
file_y = 'C:\VisualDSP\TP3Robo\y6.dat';
file_z = 'C:\VisualDSP\TP3Robo\z6.dat';

x = func_FileBinary2Signal(file_x,'double') ;
y = func_FileBinary2Signal(file_y,'double') ;
z = func_FileBinary2Signal(file_z,'double') ;

k =1;
for k = 1: 100
    plot3(x(1,k),y(1,k),z(1,k),'.-k');
end



hold off;
```

Conclusiones:

Como conclusión final podemos decir que logramos integrar en este proyecto todos los alcances de la materia, con los cuales estamos en condiciones de realizar el planteo del desarrollo de un robot desde cero dado que en el presente estudio planteamos lo planos del robot, calculamos la cinemática del mismo, definiendo un entorno de trabajo y definiendo también las zonas prohibidas del robot para evitar colisiones. Luego planteamos la dinámica del robot, teniendo en cuenta las características constructivas, pudiendo así calcular los torques necesarios para mover efectivamente el robot. Finalmente con el compilador logramos establecer el juego de funciones que se le proveerán al programador para poder programar eficientemente nuestro robot. Todo esto lo contrastamos además con el respaldo de la poderosa herramienta de cálculo Matlab, con la cual pudimos corroborar que nuestro trabajo fue realizado exitosamente.