UNIVERSIDAD TECNOLOGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Trabajo Practico N°2

# "Análisis dinámico de un robot e implementación en FPGA"

**Materia:** Robotica

**Integrantes:** Abaca, Daniel Alberto

Tunez, Diego

**Profesor:** Mas. Ing. Giannetta Hernan

**JTP:** Ing. Granzella Damian

## Obtención del Modelo dinámico:

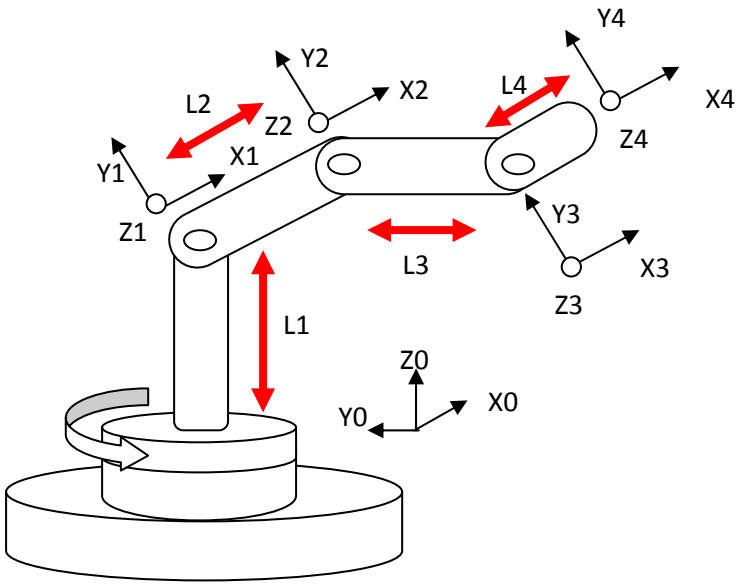
Se obtendrá el modelo dinámico a través del método Newton Euler utilizando el **toolbox Hemero** desarrollado por el señor Aníbal Ollero Baturone, quien explica su funcionamiento en el libro *Robótica: “Manipuladores y robots móviles”*.
Utilizando Matlab a través de la función **rne** permite obtener el modelo dinámico. Para ello simplemente hay que pasarle una matriz de parámetros **dyn (dinámica).**

Para hallar la matriz **dyn** necesitamos obtener algunos parámetros como ser las masas de las articulaciones y los parámetros de D.H que se calcularon en el trabajo práctico número 1 “Implementación de una Matriz Cinemática en DSP”.

Parámetros Denavit-Hartenberg(D.H):

Estos ya fueron calculados en el trabajo práctico número 1 que ahora pasamos solo a presentarlos.

En base al siguiente diagrama se obtiene los siguientes parámetros D.H:




| | $\theta$ | $d$ | $a$ | $\alpha$ |
|---|---|---|---|---|
| Articulación 1 | q1 | L1 | 0 | 90 grados |
| Articulación 2 | q2 | 0 | L2 | 0 |
| Articulación 3 | q3 | 0 | L3 | 0 |
| Articulación 4 | q4 | 0 | L4 | 0 |

Se necesita saber también otro parámetro que no está en la tabla anterior que se denomina **sigma(i)** este indicará el tipo de articulación (será **0** si es de rotación y **1** si por el contrario es prismática(traslacional)) el subíndice **i** indicara el numero de articulación para nuestro caso el valor de i estará **entre 1 a 4** (por ser cuatro grados de libertad) .

| | $sigma$ |
|---|---|
| Articulación 1 | 0 |
| Articulación 2 | 0 |
| Articulación 3 | 0 |
| Articulación 4 | 0 |

Obtención de Parámetros utilizando el programa T-FLEX:

Se obtendrán parámetros como ser por ejemplo la masa de las articulaciones a través del programa llamado **T-FLEX** Parametric CAD en este caso se descargo una versión estudiantil con algunas limitaciones, permitiendo a estudiantes la utilización del producto, para conocer sus beneficios y características. Este software une funcionalidades de modelado paramétrico 3D con el conjunto de herramientas de producción y dibujo paramétrico. Es interoperable con otros programas 3D y 2D, a través de los siguientes formatos: Parasolid, IGES, PASO, Rhino, DWG, DXF, SolidWorks, Solid Edge, Autodesk Inventor, etc.

Link de descarga (Versión estudiantil):

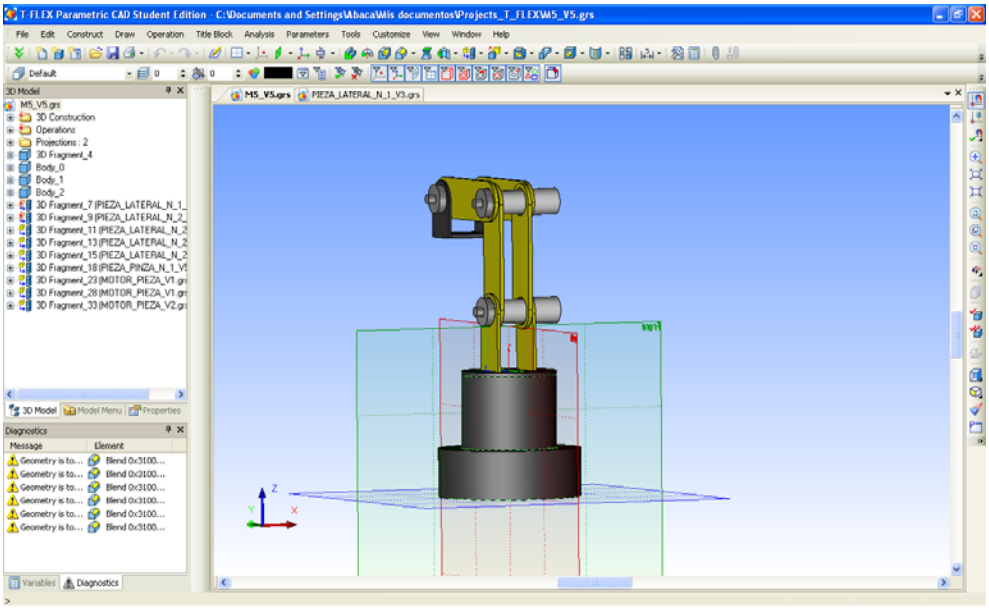
http://tflex.com/student/download.htm

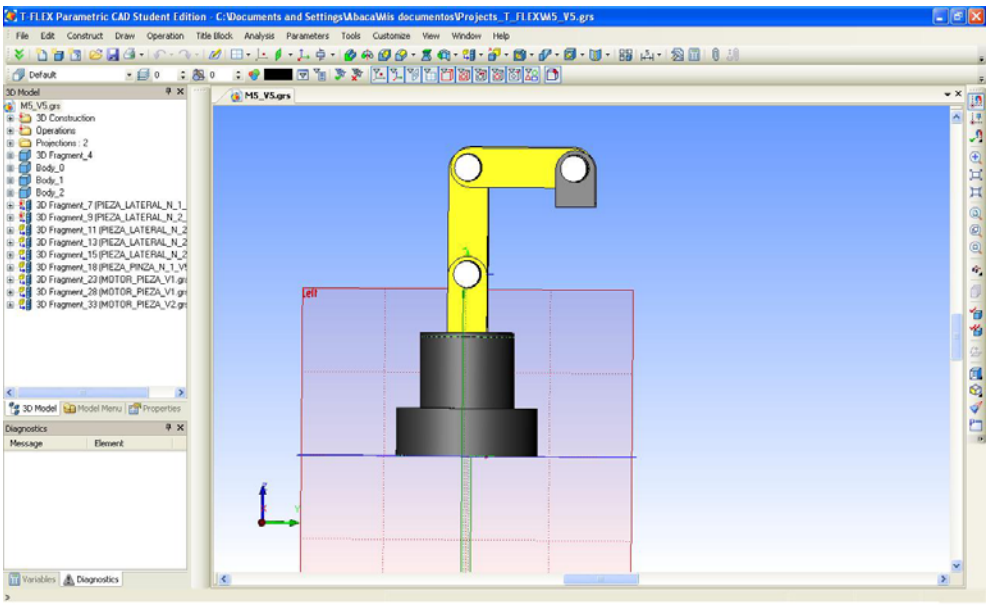
Volviendo al tema que nos interesa que es obtener **parámetros** como ser la masa de cada articulación. A continuación se indicara los pasos que se fueron realizando para cada articulación a fin de obtener los mismos.

En total son **5 pasos** que se detallan en la siguiente página:

**Pasos:**

1. Como primer paso se realizo el diseño del robot M5 muy aproximado y simplificado a modo de tener solamente un concepto general de todas las partes del robot M5 **sin tener en cuenta el gripper** dado que el estudio se baso solo en cuatro grados de libertad. A continuación se muestra dos imágenes generales.



**2.** Como primer cuerpo a hallar los parámetros partimos de la **articulación número 1**

Asignando adecuadamente las propiedades de nuestro cuerpo como ser el tipo de material que es plástico como se observa en la ventana **Body Properties**. Se obtienen los parámetros de nuestro interés como se muestra en la ventana de **Measure Element**.

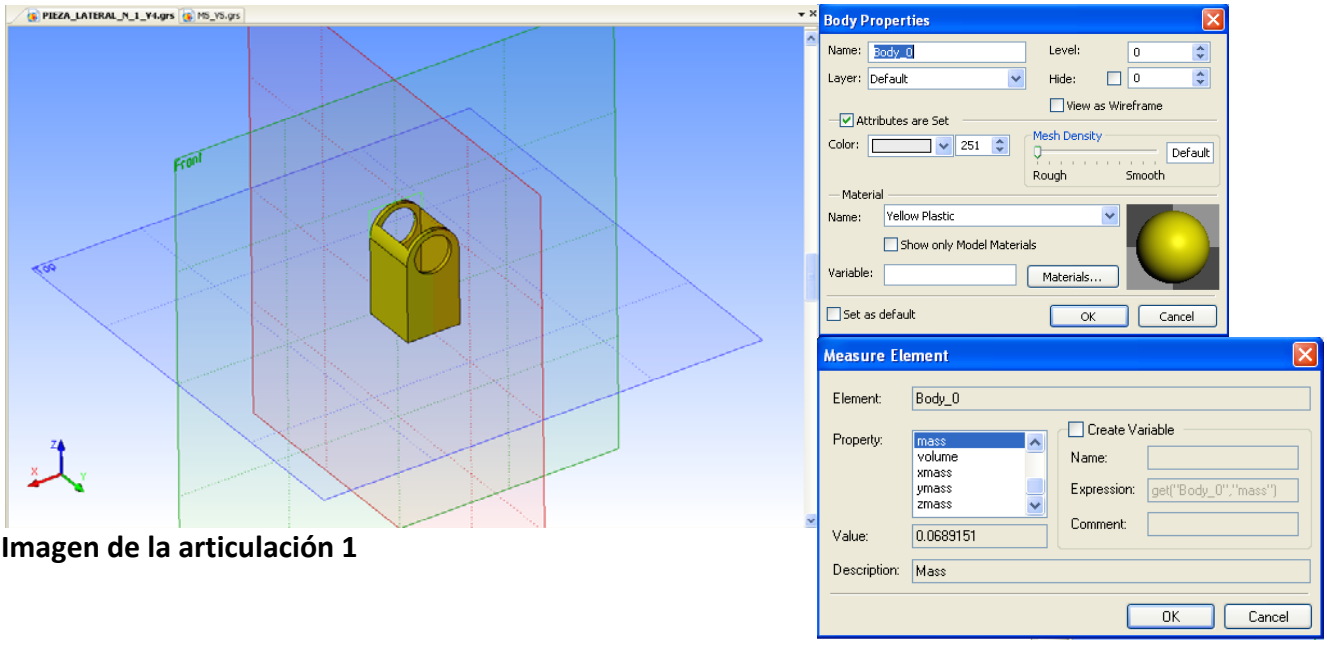



**Imagen de la articulación 1**

De la ventana de **Measure Element** se obtiene que la masa $\boldsymbol{M1 \cong 0.07Kg}$.

**3.** Parámetros de la **articulación numero 2**
Idem Paso 2

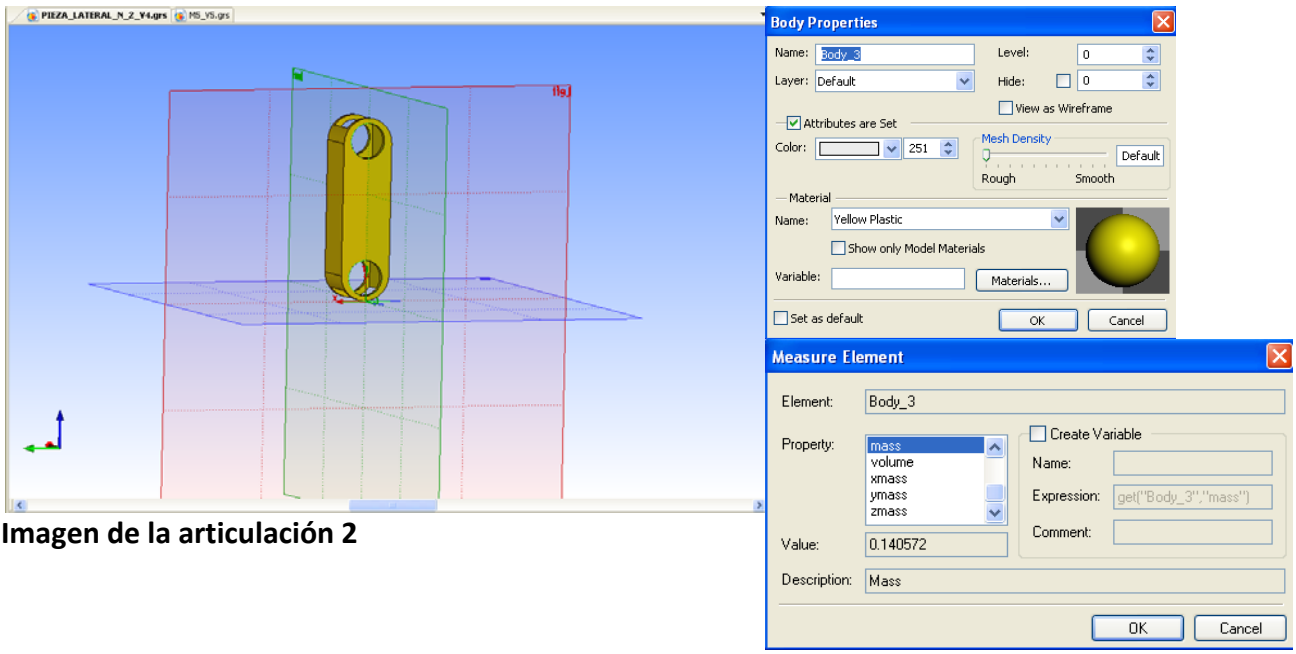



**Imagen de la articulación 2**

De la ventana de **Measure Element** se obtiene que la masa $\boldsymbol{M2 \cong 0.14Kg}$.

4. Parámetros de la **articulación numero 3**
   Idem Paso 2

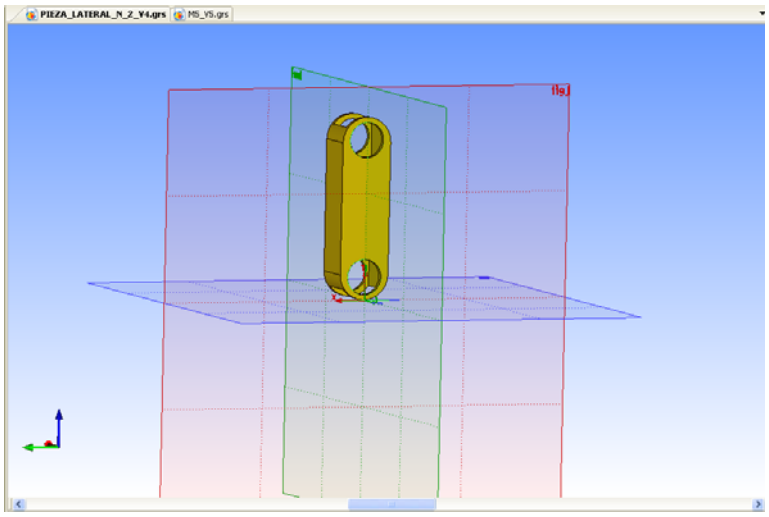
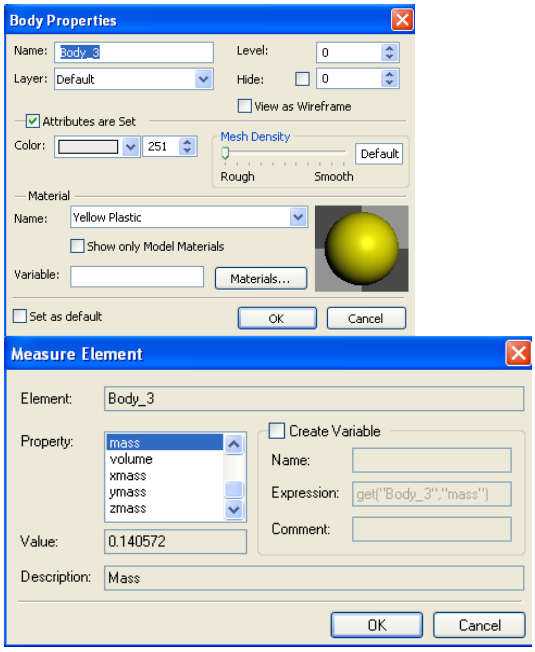



**Imagen de la articulación 3**




De la ventana de **Measure Element** se obtiene que la masa $\boldsymbol{M3 \cong 0.14Kg}$.

1. Parámetros de la **articulación numero 4**
   Idem Paso 2

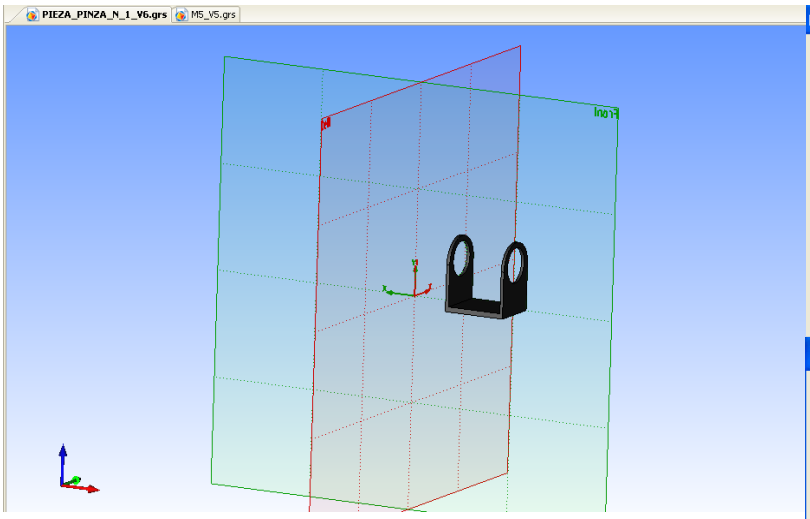
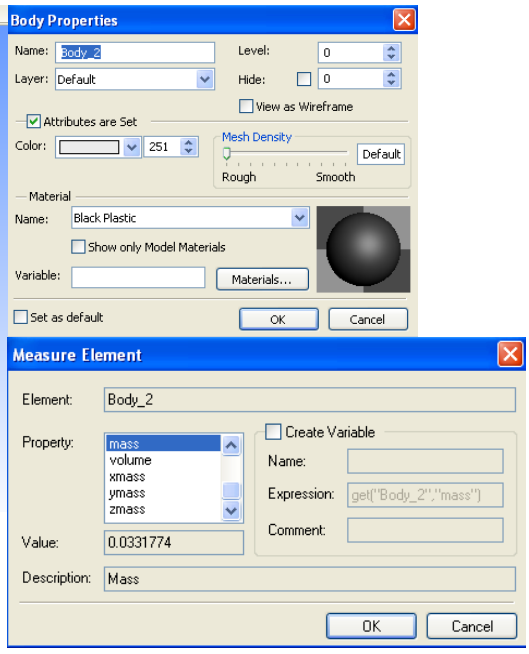



**Imagen de la articulación 4**




De la ventana de **Measure Element** se obtiene que la masa $\boldsymbol{M4 \cong 0.033Kg}$.

Aplicación de Matlab para la obtención del modelo dinámico mediante el método Newton-Euler:

Obtenidos los parámetros necesarios con el método de D.H y con el programa T-FLEX estamos en condiciones de calcular el modelo dinámico a través de la herramienta matemática Matlab utilizado el **toolbox Hemero** nombrado anteriormente.

A continuación se detallan los parámetros del robot M5, necesarios para el Toolbox Hemero:

| | $Masas$ |
|---|---|
| Articulación 1 | M1 = 0.07kg |
| Articulación 2 | M2 = 0.14kg |
| Articulación 3 | M3 = 0.14kg |
| Articulación 4 | M4 = 0.033Kg |

| | $Longitudes$ |
|---|---|
| Articulación 1 | l1 = 0.07m |
| Articulación 2 | l2 = 0.12m |
| Articulación 3 | l3 = 0.12m |
| Articulación 4 | l4 = 0.05m |

Centros de masas:

Para facilitar la resolución del problema se supondrá que las masas M1, M2, M3 y M4 están concentradas en los extremos de dichas articulaciones de nuestro robot M5.

$$lcm_1 = \begin{bmatrix} 0 \\ 0 \\ 0.08 \end{bmatrix} \; lcm_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \; lcm_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \; lcm_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Determinación de la matriz dyn:

Para resolver las ecuaciones se utiliza una matriz llamada **dyn**. Dicha matriz posee nx20 elementos, en donde n es la cantidad de articulaciones del robot a estudiar.

**Las columnas** de dicha matriz **dyn** se conforman con los siguientes datos:

Columna 1: alpha (i-1): Parámetros de Denavit-Hartenberg

Columna 2: a (i-1)

Columna 3: theta (i)

Columna 4: d (i)

Columna 5: sigma (i): Tipo de articulación; 0 si es de rotación y 1 si es prismática

Columna 6: masa: Masa del enlace i

Columna 7: rx: Centro de masas del enlace respecto al cuadro de referencia de dicho enlace
Columna 8: ry
Columna 9: rz

Columna 10: Ixx: Elementos del tensor de inercia referido al centro de masas del enlace
Columna 11: Iyy
Columna 12: Izz
Columna 13: Ixy
Columna 14: Iyz
Columna 15: Ixz

Columna 16: Jm: Inercia de la armadura

Columna 17: G: Velocidad de la articulación / velocidad del enlace

Columna 18: B: Fricción viscosa, referida al motor

Columna 19: Tc+: Fricción de Coulomb (rotación positiva), referida al motor
Columna 20: Tc-: Fricción de Coulomb (rotación negativa), referida al motor

En nuestro caso, la matriz **dyn** de **4x20 (no considerar la primera fila)** correspondiente sería la siguiente:

$$\begin{pmatrix} \boldsymbol{\alpha} & \boldsymbol{a} & \boldsymbol{\theta} & \boldsymbol{d} & \boldsymbol{sigma} & \boldsymbol{masa} & \boldsymbol{rx} & \boldsymbol{ry} & \boldsymbol{rz} & \boldsymbol{lxx} & \boldsymbol{lyy} & \boldsymbol{lzz} & \boldsymbol{lxy} & \boldsymbol{lyz} & \boldsymbol{lxz} & \boldsymbol{Jm} & \boldsymbol{G} & \boldsymbol{B} & \boldsymbol{Tc+} & \boldsymbol{Tc-} \\ \mathbf{90} & \mathbf{0} & \boldsymbol{t1} & \mathbf{0.07} & \mathbf{0} & \mathbf{0.07} & \mathbf{0} & \mathbf{0} & \mathbf{0.08} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0.12} & \boldsymbol{t2} & \mathbf{0} & \mathbf{0} & \mathbf{0.14} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0.12} & \boldsymbol{t3} & \mathbf{0} & \mathbf{0} & \mathbf{0.14} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0.05} & \boldsymbol{t4} & \mathbf{0} & \mathbf{0} & \mathbf{0.033} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

**Matriz dyn**

Simulación en Matlab utilizando el toolbox Hemero:

El objetivo es obtener las ecuaciones de **cupla, torque o par** necesarios para los **cuatro motores brushless** que operan las cuatro articulaciones de nuestro robot M5 (por considerar solo cuatro grados de libertad) a fin de saber que cupla debo aplicarle a cada motor a fin de obtener una respuesta satisfactoria al requerimiento solicitado a nuestro robot M5.

Luego, para calcular las correspondientes cuplas para cada articulación, se realizo el siguiente script en Matlab:

```
%Considerese ahora el empleo de la herramienta MATLAB. La funcion rne
%devuelve como resultado los pares ejercidos en cada articulacion.

clear
clc

%Para obtener el modelo dinamico simbolico bastan las siguientes lineas:

syms t1 t2 t3 t4 real; %Variables articulares tita 1, tita 2, tita 3, tita 4
syms td1 td2 td3 td4 real; %Velocidades articulares tita' 1, tita' 2, tita' 3, tita' 4
syms tdd1 tdd2 tdd3 tdd4 real; %Aceleraciones articulares tita''1, tita''2, tita''3 y tita''4
syms g real; %Aceleracion de la gravedad

%Escribimos la matriz con los parametros dinamicos del manipulador

dyn=[90 0 t1 0.07 0 0.07 0 0 0.08 0 0 0 0 0 0 0 1 0 0 0;
     0 0.12 t2 0 0 0.14 0 0 0 0 0 0 0 0 0 0 1 0 0 0;
     0 0.12 t3 0 0 0.14 0 0 0 0 0 0 0 0 0 0 1 0 0 0;
     0 0.05 t4 0 0 0.033 0 0 0 0 0 0 0 0 0 0 1 0 0 0];

q=[t1 t2 t3 t4]; %Vector de variables articulares
qd = [td1 td2 td3 td4]; %Vector de velocidades articulares
qdd = [tdd1 tdd2 tdd3 tdd4]; %Vector de aceleraciones articulares
grav = [0 -g 0]; %Vector de aceleracion de la gravedad
tau = rne(dyn, q, qd, qdd, grav)
simple (tau)

%De esta forma se obtiene una expresion simbolica de los pares.
%Tambien es posible evaluar cada uno de los terminos que intervienen en la
%expresion del par por separado. Para ello se escribe:

%M = inertia (dyn, q) %Matriz de masas M(tita)
%G = gravedad (dyn, q, grav) %Termino gravitatorio G(tita)
%V = coriolis (dyn, q, qd) %Terminos centrifugos y de coriolis V(tita, tita')
```

Con esto obtenemos un vector de 1 fila con 4 columnas, en donde están las cuplas de los motores brushless de cada articulación.
Al ejecutar estas instrucciones en Matlab, obtuvimos los siguientes pares motores, en forma simbólica. O sea, que de acuerdo a los valores de **posición, velocidad y aceleración** que le necesitemos obtener del robot, podemos calcular los pares motores que se deberán aplicar a las articulaciones del robot.

Resultados obtenidos con Matlab:

Expresión del torque (T1) para la articulación numero 1:

**T1** = (70809*tdd1)/10000000 + (25737*tdd2)/10000000 + (33*tdd3)/400000 - (1557*td2^2*sin(t2))/625000 - (99*td3^2*sin(t3))/500000 + (519*g*cos(t1 + t2 - 90))/50000 + (519*g*cos(t1 + t2 + 90))/50000 + (99*tdd1*cos(t2 + t3))/250000 + (99*tdd2*cos(t2 + t3))/500000 + (99*tdd3*cos(t2 + t3))/500000 + (1557*tdd1*cos(t2))/312500 + (1557*tdd2*cos(t2))/625000 + (99*tdd1*cos(t3))/250000 + (99*tdd2*cos(t3))/250000 + (99*tdd3*cos(t3))/500000 + (33*g*cos(t1 + t2 + t3 - 90))/40000 + (33*g*cos(t1 + t2 + t3 + 90))/40000 - (99*td2^2*sin(t2 + t3))/500000 - (99*td3^2*sin(t2 + t3))/500000 + (939*g*cos(t1 - 90))/50000 + (939*g*cos(t1 + 90))/50000 - (99*td1*td2*sin(t2 + t3))/250000 - (99*td1*td3*sin(t2 + t3))/250000 - (99*td2*td3*sin(t2 + t3))/250000 - (1557*td1*td2*sin(t2))/312500 - (99*td1*td3*sin(t3))/250000 - (99*td2*td3*sin(t3))/250000

Expresión del torque (T2) para la articulación numero 2:

**T2** = (25737*tdd1)/10000000 + (25737*tdd2)/10000000 + (33*tdd3)/400000 + (1557*td1^2*sin(t2))/625000 - (99*td3^2*sin(t3))/500000 + (519*g*cos(t1 + t2 - 90))/50000 + (519*g*cos(t1 + t2 + 90))/50000 + (99*tdd1*cos(t2 + t3))/500000 + (1557*tdd1*cos(t2))/625000 + (99*tdd1*cos(t3))/250000 + (99*tdd2*cos(t3))/250000 + (99*tdd3*cos(t3))/500000 + (33*g*cos(t1 + t2 + t3 - 90))/40000 + (33*g*cos(t1 + t2 + t3 + 90))/40000 + (99*td1^2*sin(t2 + t3))/500000 - (99*td1*td3*sin(t3))/250000 - (99*td2*td3*sin(t3))/250000

Expresión del torque (T3) para la articulación numero 3:

**T3** = (33*tdd1)/400000 + (33*tdd2)/400000 + (33*tdd3)/400000 + (99*td1^2*sin(t3))/500000 + (99*td2^2*sin(t3))/500000 + (99*tdd1*cos(t2 + t3))/500000 + (99*tdd1*cos(t3))/500000 + (99*tdd2*cos(t3))/500000 + (33*g*cos(t1 + t2 + t3 - 90))/40000 + (33*g*cos(t1 + t2 + t3 + 90))/40000 + (99*td1^2*sin(t2 + t3))/500000 + (99*td1*td2*sin(t3))/250000

Expresión del torque (T4) para la articulación numero 4:

**T4** = 0

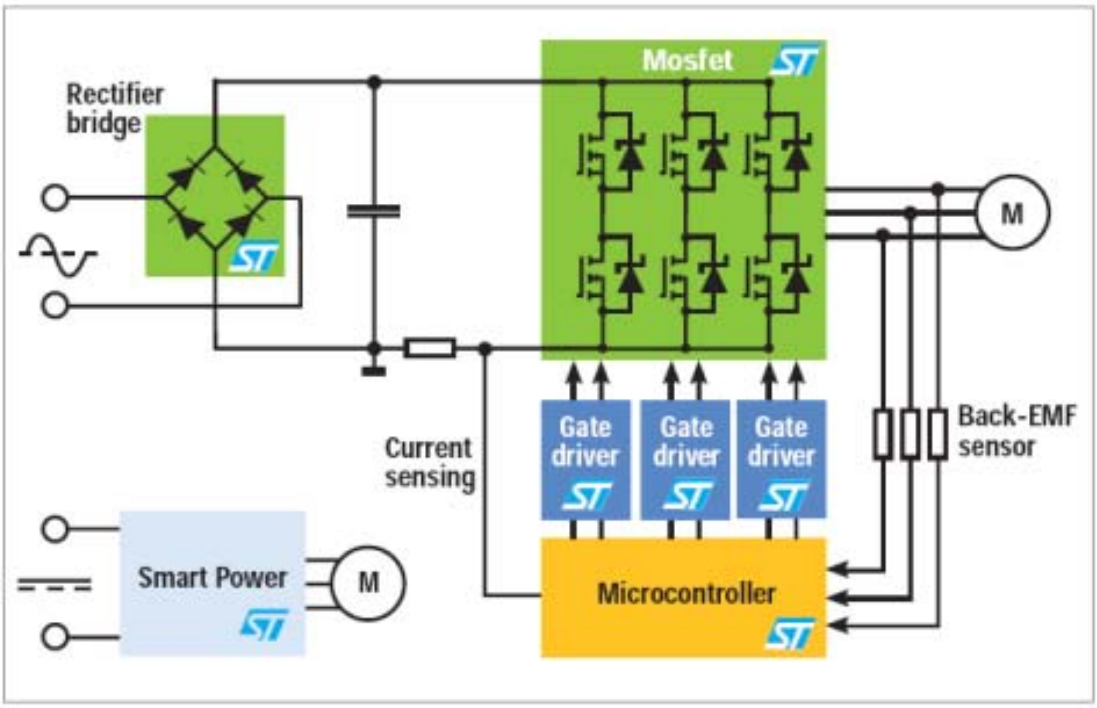
Finalmente se observa que las expresiones de los torque quedan en función de los siguientes parámetros:

**t1, t2, t3, t4: Variables articulares tita1, tita2, tita3, tita4**
**td1, td2, td3, td4: Velocidades articulares tita'1, tita'2, tita'3, tita'4**
**tdd1, tdd2, tdd3, tdd4: Aceleraciones articulares tita''1, tita''2, tita''3 y tita''4**
**g=$9,8\frac{m}{s^2}$: Aceleración de la gravedad**

**Aclaración**: La última articulación tiene el par motor igual a cero, ya que el extremo del robot está sin carga.

## Implementación en código VHDL:

En esta etapa del trabajo, debemos implementar un sistema de control para un motor Brushless mediante PWM. Esto, se realizará con el uso de FPGA y un lenguaje descriptor de hardware (VHDL).




*High-frequency three-phase brushless DC motor drive*

Para realizar este control, hay que:

- Generar el PWM: Debe haber un módulo encargado de generar una señal de PWM. Este módulo recibe un dato (generalmente, de un microcontrolador) y entrega una señal rectangular, cuyo ciclo de actividad, depende del dato ingresado.

- Controlar el puente H: Hay que elegir que transistores deben activarse siguiendo la secuencia adecuada. Los sensores de efecto Hall permiten conocer la posición del rotor, y dependiendo de los cambios de estado de los mismos, se sabe cuando hay que activar cada transistor.

Para generar la señal de PWM, se utilizará el código del ejemplo dado por Atmel. Dicho código, se encuentra en:

http://www.atmel.com/dyn/resources/prod_documents/DOC2324.PDF

---

Dicho código es:

**---- pwm_fpga.vhd**

```
library IEEE;
USE     ieee.std_logic_1164.all;
USE    ieee.std_logic_arith.all ;
USE     work.user_pkg.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

 -- 8 BIT DATA REGISTER TO STORE THE MARKING VALUES .
 -- THE MARKING VALUES WILL DETERMINE THE DUTY CYCLE OF PWM OUTPUT
```

```
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;

-- 8 BIT UPDN COUNTER. COUNTS UP OR DOWN BASED ON THE PWM_INT SIGNAL AND GENERATES
-- TERMINAL COUNT WHENEVER COUNTER REACHES THE MAXIMUM VALUE OR WHEN IT TRANSISTS
-- THROUGH ZERO. THE TERMINAL COUNT WILL BE USED AS INTERRUPT TO AVR FOR GENERATING
-- THE LOAD SIGNAL.
-- INC and DEC are the two functions which are used for up and down counting. They are defined in sepearate user_pakge library

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 <= INC(cnt_out_int);
    ELSE
       IF (rco_int ='0' and pwm_int ='0' and cnt_out_int > "00000000") THEN
```

```
                cnt_out_int <= DEC(cnt_out_int);
            END IF;
        END IF;
    END IF;
END PROCESS;

-- Logic to generate RCO signal

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;

-- TOGGLE FLIP FLOP TO GENERATE THE PWM OUTPUT.

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;
```

Donde INC y DEC, se implementan en el siguiente código (también, de Atmel):

```
--user_pkg.vhd
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

PACKAGE user_pkg IS
   function INC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
   function DEC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
END user_pkg ;

PACKAGE BODY user_pkg IS
```

```
function INC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR is
   variable XV: STD_LOGIC_VECTOR(X'LENGTH - 1 downto 0);
   begin
  XV := X;
  for I in 0 to XV'HIGH LOOP
     if XV(I) = '0' then
       XV(I) := '1';
       exit;
     else XV(I) := '0';
     end if;
  end loop;

return XV;
end INC;

function DEC(X: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR is
   variable XV: STD_LOGIC_VECTOR(X'LENGTH - 1 downto 0);
   begin
  XV := X;
  for I in 0 to XV'HIGH LOOP
     if XV(I) = '1' then
       XV(I) := '0';
       exit;
     else XV(I) := '1';
     end if;
```

```
 end loop;


return XV;

end DEC;


END user_pkg;
```

Una vez obtenida la generación de PWM en función de los datos de entrada, hay que controlar en puente H.

Dependiendo del valor de las entradas de los sensores Hall, hay que elegir que transistores se van a activar, y que transistor va a recibir los pulsos de PWM. La activación/desactivación de cada transistor, se realiza según el siguiente diagrama de estados:

| Estado | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | PwmSignal | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | PwmSignal |
| 2 | 0 | 0 | 1 | 0 | 0 | PwmSignal |
| 3 | 0 | PwmSignal | 1 | 0 | 0 | 0 |
| 4 | 0 | PwmSignal | 0 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | PwmSignal | 1 | 0 |

El código es el siguiente:

**--Driver motor fpga**

```
library IEEE;

USE ieee.std_logic_1164.all;

USE ieee.std_logic_arith.all;


ENTITY driver_motor_fpga IS

PORT (  ClockDriver    :in  STD_LOGIC;
```

```
        ResetDriver    :in  STD_LOGIC;
        SentidoGiro    :in  STD_LOGIC;
    DataDriver         :in  std_logic_vector(7 downto 0);
    SensoresHall       :in  std_logic_vector(2 downto 0);
    Q5Q3Q1             :out std_logic_vector(2 downto 0);
    Q2                 :out STD_LOGIC;
    Q4                 :out STD_LOGIC;
    Q6                 :out STD_LOGIC
  );
END driver_motor_fpga;

ARCHITECTURE Arch_Driver_motor_fpga OF driver_motor_fpga IS

 --Este componente, es el definido en pwm_fpga.vhd del ejemplo hecho por ATMEL
 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;

SIGNAL PwmSignal: STD_LOGIC;

BEGIN
  -- Conexion del modulo pwm
  ModuloPWM: pwm_fpga
```

```
        PORT MAP(
                        clock => ClockDriver,
                        reset => ResetDriver,
                        data_value => DataDriver,
                        pwm => PwmSignal
                        );

PROCESS(SensoresHall,SentidoGiro, PwmSignal)
 BEGIN
   IF(SentidoGiro = '0') THEN --En caso de que gire en sentido horario

     CASE SensoresHall IS
       WHEN "001" =>
        Q5Q3Q1 <= "001";
        Q6 <= PwmSignal;
        Q4 <= '0';
        Q2 <= '0';
       WHEN "000" =>
        Q5Q3Q1 <= "001";
        Q6 <= '0';
        Q4 <= PwmSignal;
        Q2 <= '0';
       WHEN "100" =>
        Q5Q3Q1 <= "100";
        Q6 <= '0';
        Q4 <= PwmSignal;
```

```
        Q2 <= '0';
      WHEN "110" =>
        Q5Q3Q1 <= "100";
        Q6 <= '0';
        Q4 <= '0';
        Q2 <= PwmSignal;
      WHEN "111" =>
        Q5Q3Q1 <= "010";
        Q6 <= '0';
        Q4 <= '0';
        Q2 <= PwmSignal;
      WHEN "011" =>
        Q5Q3Q1 <= "010";
        Q6 <= PwmSignal;
        Q4 <= '0';
        Q2 <= '0';
      WHEN OTHERS => NULL;
  END CASE;
END IF;

IF(SentidoGiro = '1') THEN --En caso de que gire en sentido antihorario

  CASE SensoresHall IS
       WHEN "011" =>
         Q5Q3Q1 <= "100";
         Q6 <= '0';
```

```
    Q4 <= PwmSignal;
    Q2 <= '0';
WHEN "111" =>
    Q5Q3Q1 <= "001";
    Q6 <= '0';
    Q4 <= PwmSignal;
    Q2 <= '0';
WHEN "110" =>
    Q5Q3Q1 <= "001";
    Q6 <= PwmSignal;
    Q4 <= '0';
    Q2 <= '0';
WHEN "100" =>
    Q5Q3Q1 <= "010";
    Q6 <= PwmSignal;
    Q4 <= '0';
    Q2 <= '0';
WHEN "000" =>
    Q5Q3Q1 <= "010";
    Q6 <= '0';
    Q4 <= '0';
    Q2 <= PwmSignal;
WHEN "001" =>
    Q5Q3Q1 <= "100";
    Q6 <= '0';
    Q4 <= '0';
```

```
            Q2 <= PwmSignal;
          WHEN OTHERS => NULL;
        END CASE;
    END IF;
 END PROCESS;


 END ARCHITECTURE Arch_Driver_motor_fpga;
```

---

Ahora, para poder simularlo, se crea otra entidad, que use un componente driver_motor_fpga, y se generen las señales necesarias para probar su funcionamiento.

Para esto, se le va a generar una secuencia a los sensores Hall, para simular que el motor está girando.

El código es:

---

**-- test_control.vhd**

```
LIBRARY ieee;

use ieee.std_logic_1164.all;

ENTITY test_control IS

END test_control;


ARCHITECTURE Arch_Test_control OF test_control IS


COMPONENT driver_motor_fpga

PORT (  ClockDriver    :in  STD_LOGIC;

          ResetDriver     :in  STD_LOGIC;

          SentidoGiro     :in  STD_LOGIC;

          DataDriver      :in  std_logic_vector(7 downto 0);
```

```
        SensoresHall  :in  std_logic_vector(2 downto 0);
        Q5Q3Q1        :out std_logic_vector(2 downto 0);
        Q2            :out STD_LOGIC;
        Q4            :out STD_LOGIC;
        Q6            :out STD_LOGIC
    );
END COMPONENT;

-- Internal signal declaration
SIGNAL sig_clock       : std_logic;
SIGNAL sig_reset       : std_logic;
SIGNAL sig_sentido     : std_logic;
SIGNAL sig_data_value          : std_logic_vector(7 downto 0);
SIGNAL sig_HALLs       : std_logic_vector(2 downto 0);
SIGNAL Signal_Q2, Signal_Q4, Signal_Q6      : std_logic;
SIGNAL Sig_Q5Q3Q1 : std_logic_vector(2 downto 0);
shared variable ENDSIM: boolean:=false;
constant clk_period:TIME:=200 ns;

BEGIN
-- Genera el clock
clk_gen: process
   BEGIN
   If ENDSIM = false THEN
     sig_clock <= '1';
     wait for clk_period/2;
```

```
    sig_clock <= '0';
    wait for clk_period/2;
  else
    wait;
  end if;
 end process;

inst_control_motor : driver_motor_fpga
PORT MAP(
  ClockDriver => sig_clock,
  ResetDriver => sig_reset,
  SentidoGiro => sig_sentido,
  DataDriver  => sig_data_value,
  SensoresHall => sig_HALLs,
  Q2 => Signal_Q2,
  Q4 => Signal_Q4,
  Q6 => Signal_Q6,
  Q5Q3Q1 => Sig_Q5Q3Q1
 );

stimulus_process: PROCESS
-- Se dejan tiempos de 8mseg entre cada cambio de estado, para que gire a, aprox. 1200 RPM
BEGIN
 sig_sentido <= '0'; -- primero giro en sentido horario
 sig_reset <= '1';
 wait for 500 ns;
```

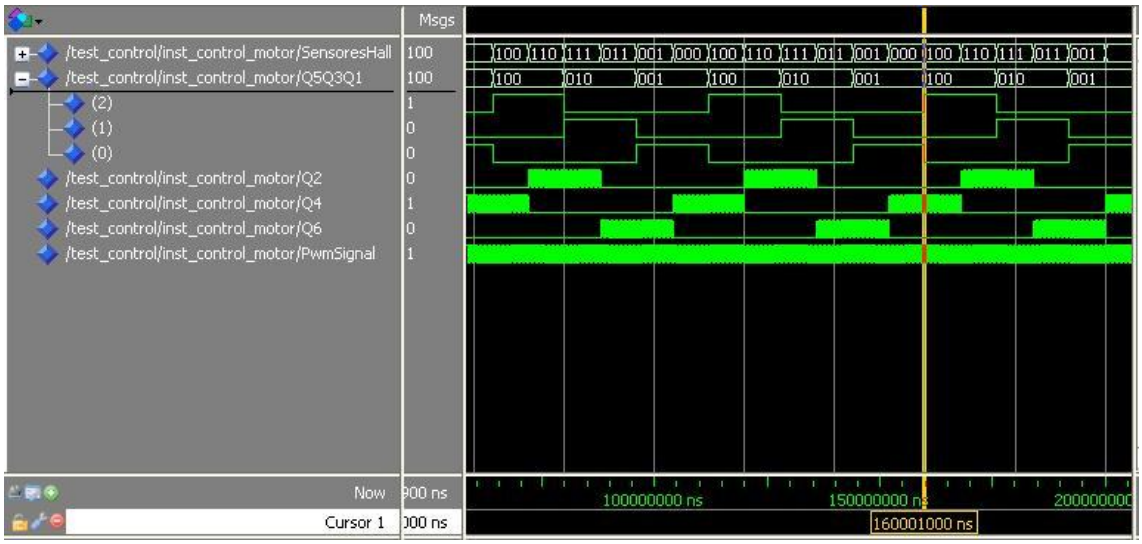
```
 sig_reset <= '0';
 sig_data_value <= "10000000";  -- Para el duty
 wait for 500 ns;
 for i in 1 to 5 loop   --simulo 5 vueltas del motor(sentido horario)aprox 1200RPM
   sig_HALLs <= "001";
   wait for 8 ms;
   sig_HALLs <= "000";
   wait for 8 ms;
   sig_HALLs <= "100";
   wait for 8 ms;
   sig_HALLs <= "110";
   wait for 8 ms;
   sig_HALLs <= "111";
   wait for 8 ms;
   sig_HALLs <= "011";
   wait for 8 ms;
end loop;
wait for 1000 ns;
sig_sentido <= '1'; -- Ahora, va a girar en sentido antihorario
for i in 1 to 5 loop   --simulo 5 vueltas del motor(sentido antihorario)
    sig_HALLs <= "011";
    wait for 7 ms;
    sig_HALLs <= "111";
    wait for 7 ms;
    sig_HALLs <= "110";
    wait for 7 ms;
```

```
    sig_HALLs <= "100";
    wait for 7 ms;
    sig_HALLs <= "000";
    wait for 7 ms;
    sig_HALLs <= "001";
    wait for 7 ms;
  end loop;
wait;
 END PROCESS stimulus_process;
END Arch_Test_control;
```

---

Resultados de la simulación:

Al realizar la simulación, se hizo el gráfico de: los sensores Hall, los transistores del puente H, y la señal a la salida del módulo PWM. Los resultados, son:



## Conclusiones:

A través de este trabajo práctico pudimos experimentar la potencialidad de un software de modelado parametrico 3D llamado T-FLEX que nos simplifica muchísimo la complejidad de sacar parámetros físicos de un modelo mecánico sofisticado.

Además al introducirnos en un lenguaje nuevo para nosotros como ser el VHDL nos permitió entender y poder simular como es el funcionamiento lógico de un motor Brushless que componen a un robot.

Bibliografía:


- “Robótica Manipuladores y robots móviles” de Ollero Baturone, Anibal
- www.atmel.com