ROBÓTICA
TP2 Dinámica

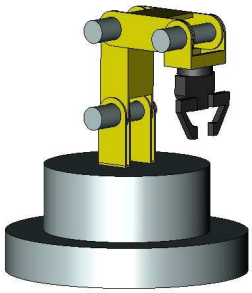
Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

GRUPO 1

Alumnos:
Hernán Muñoz
Fabián Zanella

PROFESOR:
Ing. GIANNETTA, Hernán

JTP:
Ing. GRANZELLA, Damián



FECHA DE ENTREGA:
04/07/2011

Índice

Introducción

Mediante el presenta trabajo práctico, se pretende calcular la dinámica del robot M5, por medio del método de Lagrange-Euler, para lo que se utilizará el toolbox de Hemero para Matlab y el diseño del robot en el software T-Flex CAD Student, para la extracción de sus parámetros característicos. De esta manera, se determinarán los torques que es necesario aplicar a los motores, dadas una posición, velocidad y aceleración determinadas.
Asimismo, se diseñará un programa en VHDL para su implementación en una FPGA, que controlará, a través de una interfaz de puente H, los motores del robot.

Marco teórico

Introducción a la dinámica de un robot

El modelo dinámico de un robot nos permite conocer la relación existente entre el movimiento y las fuerzas actuantes que provocan dicho movimiento.
A diferencia del modelo cinemático que solo representa el movimiento con respecto a un sistema de referencia, el modelo dinámico tiene en cuenta las fuerzas y pares aplicadas en
las articulaciones y además los parámetros dimensionales del robot como la longitud, masa e inercia de sus elementos.
Desde un punto de vista matemático, se relaciona la locación del robot (que está definida por sus variables articulares o por las coordenadas de localización de su extremo), y sus derivadas: velocidad y aceleración.
Un punto a tener en cuenta, es que la complejidad del modelo dinámico se acrecienta en gran medida con el aumento de grados de libertad (GDL).
El estudio dinámico del robot tiene como resultado un conjunto de ecuaciones matemáticas que describen la conducta dinámica del manipulador.
Tales ecuaciones de movimiento son útiles para simulación por PC, diseño de ecuaciones de control apropiadas para el robot y la evaluación del diseño y estructura cinemática del robot.
Debido al ya comentado aumento de complejidad a medida que crece el número de GDL no siempre es posible hallar una solución cerrada, es decir, un conjunto de ecuaciones diferenciales que al ser integradas nos den como resultado las fuerzas y torques a ser aplicadas para obtener un determinado desplazamiento con una cierta velocidad y aceleración, es decir, un único resultado.
Así, el modelo dinámico se debe resolver mediante cálculo numérico e iterativo (por Ej. Uso de series convergentes).
Cada eslabón de la cadena se trata como rígido y de masa concentrada, se usa el centro de gravedad.
Existen 2 formas básicas de encarar el problema del modelo dinámico:
Modelo Dinámico Directo: Expresa la evolución temporal de las coordenadas articulares del robot en función de las fuerzas y pares que intervienen.

Modelo Dinámico Inverso: Expresa las fuerzas y pares que intervienen en función de la evolución de las coordenadas articulares y sus derivadas.
Al hacer el análisis, se debe tener en cuenta las siguientes fuerzas intervinientes:
Inercia, gravedad, coriolis y centrípeta.
Existen distintos métodos para resolver el problema:
- Lagrange-Euler
- Newton-Euler
- Variables de estado
- Kane

Metodo Lagrange-Euler
Se basa en la mecánica Lagrangiana y parte de consideraciones energéticas.
Su obtención es sistemática pero consume mucho tiempo de cálculo, inútiles en control en tiempo real.
Se obtienen ecuaciones diferenciales cuya utilidad es el análisis y diseño de estrategias de control avanzadas y simulación.

Formulación de Lagrange-Euler
Se basa en la aplicación de la formulación Lagrangiana a la representación de Denavit-Hartenberg.
Puede utilizarse para:
Analizar y diseñar estrategias de control avanzado
Resolver los problemas dinámico directo e inverso
Problema dinámico directo: Dados los pares generalizados aplicados a cada articulación calcular las aceleraciones de los elementos. Integrando las aceleraciones se pueden obtener velocidades y coordenadas generalizadas.
Problema dinámico inverso: Dadas las coordenadas generalizadas y sus dos derivadas se calculan los pares generalizados. En este último caso gran cantidad de operaciones, por lo que no son útiles para realizar control en tiempo real.

Obtención del modelo dinámico de un robot mediante la formulación de Lagrange – Euler

Este planteamiento utiliza, por tanto, las matrices i-1Ai que relacionan el sistema de coordenadas de referencia del elemento i con el del elemento i-1. Se realizan en este caso operaciones de producto y suma innecesarias. Se trata de un procedimiento ineficiente desde el punto de vista computacional, donde tiene una complejidad de O(n4). Sin embargo, conduce a unas ecuaciones finales bien estructuradas donde aparecen de manera clara los diversos pares y fuerzas que intervienen en el movimiento.

Algoritmo computacional para el modelo dinámico por Lagrange – Euler

- **L-E 1** : Asignar a cada eslabón un sistema de referencia de acuerdo a las normas de D-H.
- **L-E 2** : Obtener las matrices de transformacion 0**A**i para cada elemento i.
- **L-E 3** : Obtener las matrices **U**ij definidas por:

$$\mathbf{U}_{ij} = \frac{\partial^0 \mathbf{A}_i}{\partial q_j}$$

La derivada de la matriz de D-H 0**A**i respecto de la coordenada qi puede obtenerse fácilmente de manera computacional, mediante la expresión:

$$\frac{\partial^0 \mathbf{A}_i}{\partial q_j} = \begin{cases} {}^0\mathbf{A}_{j-1}\,\mathbf{Q}_j\,{}^{j-1}\mathbf{A}_i & \text{si } j \le i \\ [0] & \text{si } j > i \end{cases}$$

- **L-E 4** : Obtener las matrices **U**ijk definidas por:

$$\mathbf{U}_{ijk} = \frac{\partial \mathbf{U}_{ij}}{\partial q_k}$$

$$\frac{\partial \mathbf{U}_{ij}}{\partial q_k} = \frac{\partial}{\partial q_k}\left(\frac{\partial\,{}^0\mathbf{A}_i}{\partial q_j}\right) = \begin{cases} {}^0\mathbf{A}_{j-1}\mathbf{Q}_j\,{}^{j-1}\mathbf{A}_{k-1}\mathbf{Q}_k\,{}^{k-1}\mathbf{A}_i & \text{si } i \ge k \ge j \\ {}^0\mathbf{A}_{k-1}\mathbf{Q}_k\,{}^{k-1}\mathbf{A}_{j-1}\mathbf{Q}_j\,{}^{j-1}\mathbf{A}_i & \text{si } i \ge j \ge k \\ [0] & \text{si } k > i \text{ o } j > i \end{cases}$$

$$\mathbf{Q}_i = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Si la articulación i es de rotación

$$\mathbf{Q}_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Si la articulación i es de Traslación.

- **L-E 5** : Obtener las matrices de pseudoinercias **J**i para cada elemento, que vienen definidas por:

$$\mathbf{J}_i = \begin{bmatrix} \int x_i^2 dm & \int x_i y_i dm & \int x_i z_i dm & \int x_i dm \\ \int y_i x_i dm & \int y_i^2 dm & \int y_i z_i dm & \int y_i dm \\ \int z_i x_i dm & \int z_i y_i dm & \int z_i^2 dm & \int z_i dm \\ \int x_i dm & \int y_i dm & \int z_i dm & \int dm \end{bmatrix}$$

donde las integrales están extendidas al elemento i considerado, y (xi,yi,zi) son las coordenadas del diferencial de masa dm respecto al sistema de coordenadas del elemento.

- **L-E 6** :Obtener la matriz de inercias D = [dij] cuyos elementos vienen definidos por:

$$d_{ij} = \sum_{k=(\max i,j)}^{n} \text{Traza}\left(\mathbf{U}_{kj}\mathbf{J}_k\mathbf{U}_{ki}^T\right)$$

con i,j = 1,2, ...,n // n : numero de grados de libertad
Las matrices Ji y D son simétricas y semidefinidas positivas.

- **L-E 7** :Obtener los términos hikm definidos por:

$$h_{ikm} = \sum_{j=\max(i,k,m)}^{n} \text{Traza}\left(\mathbf{U}_{jkm}\mathbf{J}_j\mathbf{U}_{ji}^T\right)$$

con i,k,m = 1,2, ...,n
El termino hikm representa el efecto, en cuanto a fuerza o par, generado sobre el eslabón i como consecuencia del movimiento relativo entre los eslabones k y m. Se cumple que hikm=himk y que hiii=0.

- **L-E 8:** Obtener la matriz columna de fuerzas de Coriolis y centripeta H = [hi]T cuyos elementos vienen definidos por:

$$h_i = \sum_{k=1}^{n}\sum_{m=1}^{n} h_{ikm}\dot{q}_k\dot{q}_m$$

- **L-E 9:** Obtener la matriz columna de fuerzas de gravedad C = [ci]T cuyos elementos están definidos por:

$$c_i = \sum_{j=1}^{n} \left(- m_j \mathbf{g} \mathbf{U}_{ji}\, {}^{j}\mathbf{r}_j \right)$$

con i = 1,2,...,n
g: es el vector de gravedad expresado en el sistema de la base {S0} y viene expresado por (gx0 , gy0, gz0,0)
i**r**j : es el vector de coordenadas homogéneas del centro de masas del elemento j expresado en el sistema de referencia del elemento i.

- **L-E 10** : La ecuación dinámica del sistema será:

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

donde *τ* es el vector de fuerzas y pares motores efectivos aplicados sobre cada coordenada qi.

Método de Newton – Euler

La formulación de Newton-Euler utiliza las ecuaciones de equilibrio de fuerzas y torques (pares):

$$\sum F = m\,\dot{v}$$

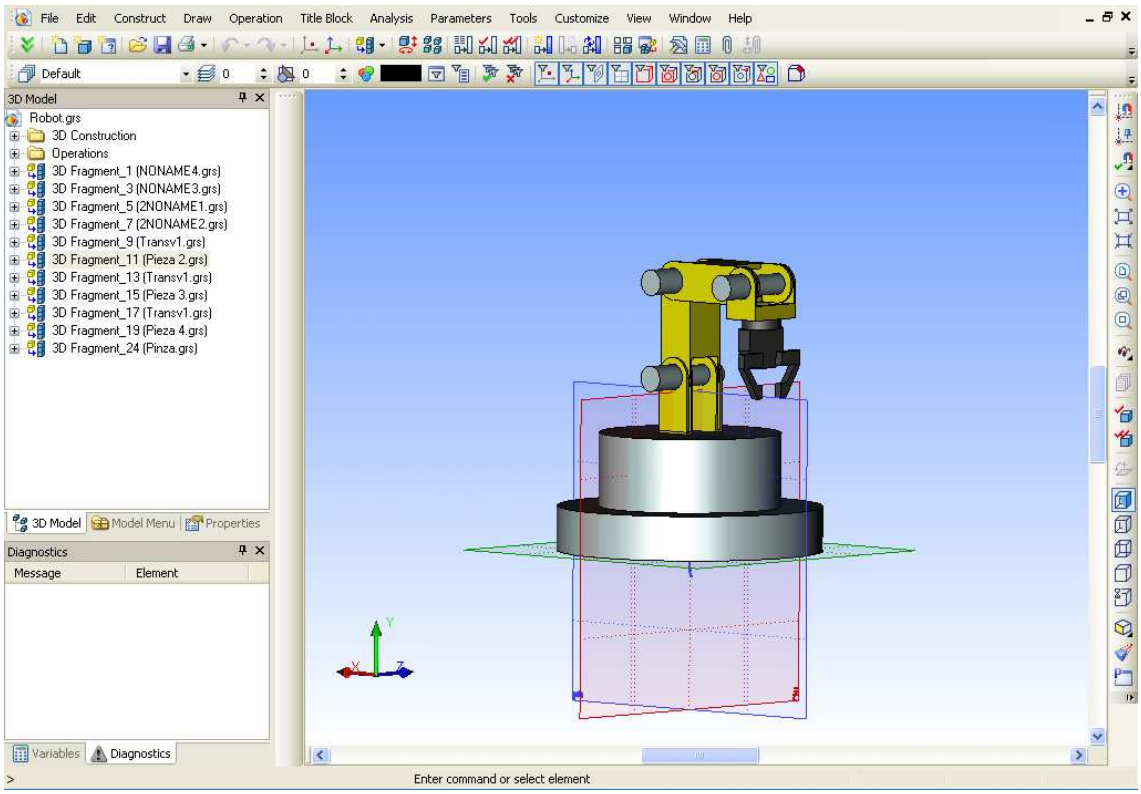
$$\sum T = I\,\dot{\omega} + \omega \times (I\,\omega)$$

El modelo computacional es el siguiente:

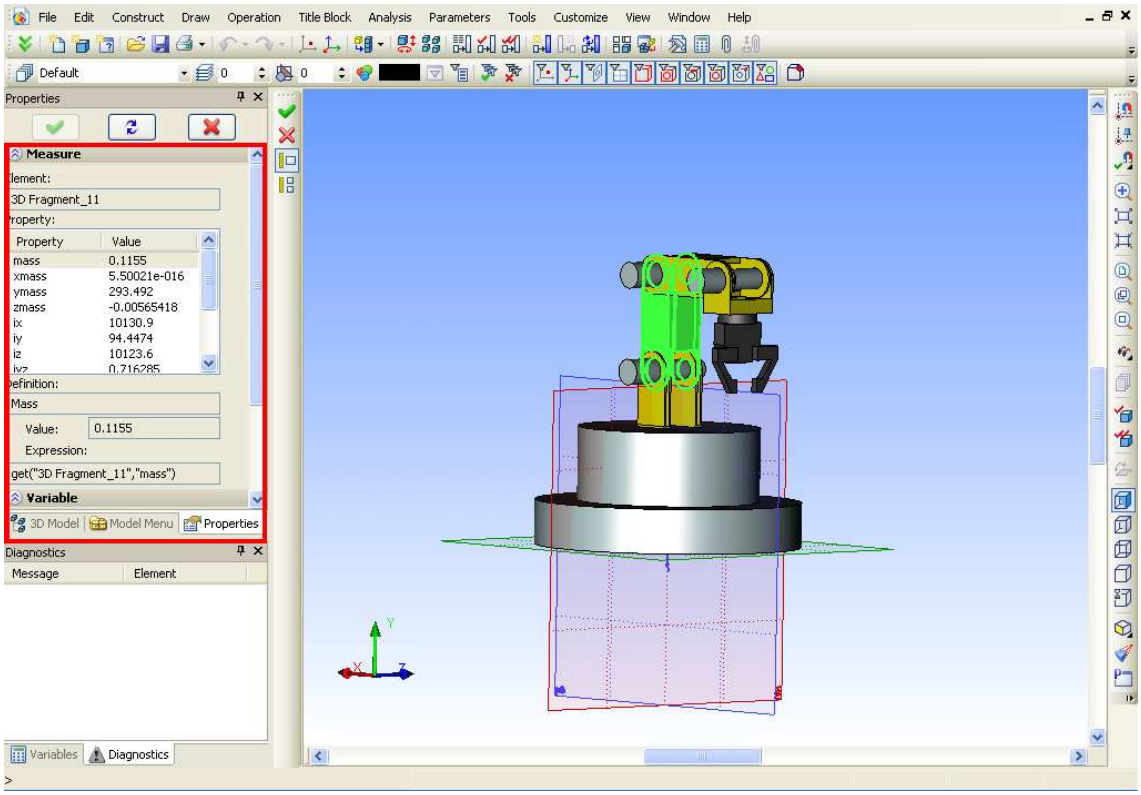
- **N-E 1**. Asignar a cada eslabón un sistema de referencia de acuerdo con las normas de D-H.
- **N-E 2**. Obtener las matrices de rotación i-1**R**i y sus inversas i**R**i-1 = (i-**R**i) - 1 = (i-1**R**i)T.
- **N-E 3**. Establecer las condiciones iniciales para el sistema de base S0..
- **N-E 4**. Obtener la velocidad angular del sistema { Si}
- **N-E 5**. Obtener la aceleración angular del sistema { Si }.
- **N-E 6**. Obtener la aceleración lineal del sistema i.
- **N-E 7**. Obtener la aceleración lineal del centro de gravedad del eslabón i:
- **N-E 8**. Obtener la fuerza ejercida sobre el eslabón i:
- **N-E 9**. Obtener el par ejercido sobre el eslabón i:
- **N-E 10**. Obtener la fuerza o torque (par) aplicado a la articulación i.

Obtención de parámetros por medio del T-Flex

Se realizó el modelo del robot M5 en el software T-Flex:



Para ahorrar el cálculo de los centros de masa, se determinará que los mismos se encuentran en las articulaciones de las piezas del robot. De la misma manera, no se tendrán en cuenta los momentos de inercia. El procedimiento para determinar los parámetros correspondientes a la masa de las articulaciones del robot es el siguiente: Una vez realizado el modelado del robot con el software T-Flex, se procederá a hacer click derecho sobre el nombre de la pieza a medir en la ventana “3D Model” a la izquierda de la pantalla y se seleccionará la opción “Measure”. La misma desplegará una barra en donde figuran la masa, centro de masa, momentos de inercia, etc.



De esta manera, se obtienen las masas de las articulaciones:

m1 = 0.0689 Kg

m2 = 0.1155 Kg

m3 = 0.1221 Kg

m4 = 0.1840 Kg

Recordando que sólo se tendrán en cuenta las articulaciones que aportan a la posición del muñón, por lo que se dejará afuera la articulación de rotación de la pinza.

Simulación con Matlab

Para esta simulación y la obtención de los torques necesarios (de acuerdo a posición, velocidad y aceleración) se utilizará el toolbox de Hemero para Matlab. Una de las funciones del mismo es:

TAU = rne(DYN, Q, QD, QDD)

La que calcula el modelo dinámico completo del manipulador mediante el método recursivo de Newton-Euler, recibiendo como parámetros los valores (o variables simbólicas en este caso, que luego serán reemplazadas por valores) de posición, velocidad y aceleración de las correspondientes articulaciones y una matriz dinámica que recopila valores característicos de las mismas y que deberá ser construida de la siguiente manera:

- Habrá una fila por cada enlace que tenga el manipulador.
- Cada fila tendrá el siguiente formato:

1)	**alpha(i-1)**	Parámetros de Denavit-Hartenberg.
2)	**a(i-1)**	
3)	**theta(i)**	
4)	**d(i)**	
5)	**sigma(i)**	Tipo de articulación; 0 si es de rotación y 1 si es prismática.
6)	**masa**	Masa del enlace i.
7)	**rx**	Centro de masas del enlace respecto al cuadro de referencia de dicho enlace.
8)	**ry**	
9)	**rz**	
10)	**Ixx**	Elementos del tensor de inercia referido al centro de masas del enlace.
11)	**Iyy**	
12)	**Izz**	
13)	**Ixy**	
14)	**Iyz**	
15)	**Ixz**	
16)	**Jm**	Inercia de la armadura.
17)	**G**	Velocidad de la articulación / velocidad del enlace.
18)	**B**	Fricción viscosa, referida al motor.
19)	**Tc+**	Fricción de Coulomb (rotación positiva), referida al motor.
20)	**Tc-**	Fricción de Coulomb (rotación negativa), referida al motor.

Así pues para un robot con n enlaces, la matriz DYN tendría dimensiones nx20. Todos los ángulos deberán ser introducidos en radianes. El resto de parámetros de la matriz podrán tener las unidades que se deseen, siempre que

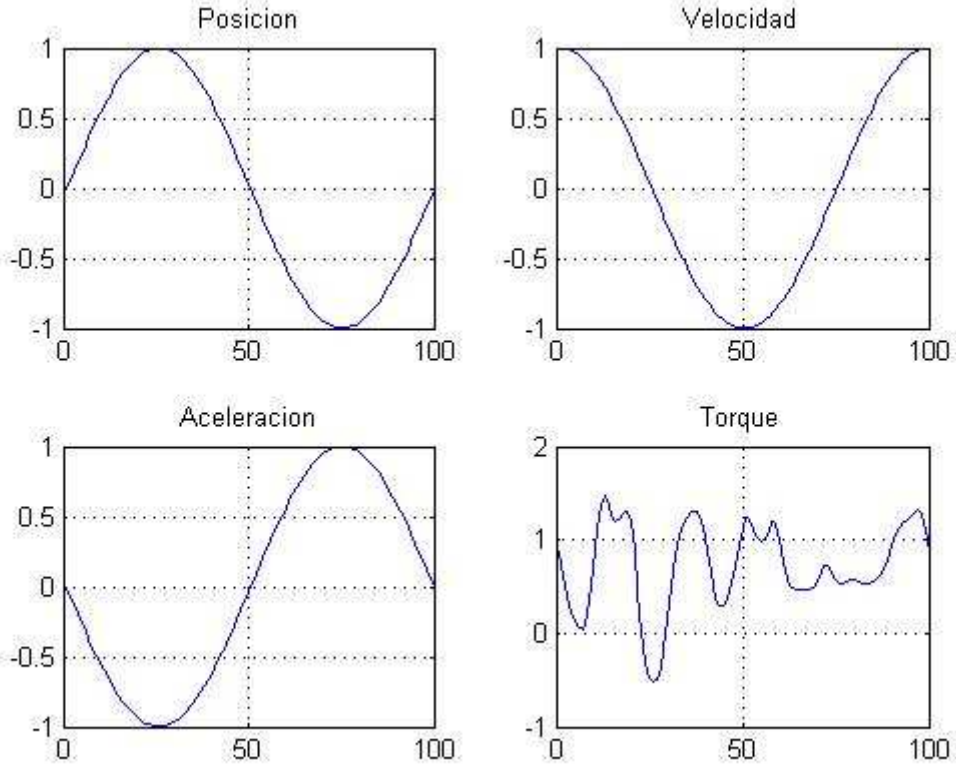
se sea coherente en el uso de dichas unidades. Es decir que si se introducen las masas en Kg y los centros de masas en metros, al escribir el tensor de inercia se deberá expresar en Kg m2. En este caso, utilizaremos Kg y metros. De esta manera, se escribió el siguiente programa:

```
clc
clear all

pil = sym(pi);

syms q1 q2 q3 q4 real;
syms qd1 qd2 qd3 qd4 real;
syms qdd1 qdd2 qdd3 qdd4 real;
g=9.8;

dyn=[
     pil/2  0       q1+pil/2   0.07    0   0.0689  0   0
     0.07   0   0   0   0   0   0   0   1   0   0   0;

     0.127   q2+pil/2   0       0   0.1155  0   0   0
     0   0   0   0   0   0   0   1   0   0   0;

     0      0.127   q3-pil/2   0       0   0.1221  0   0
     0      0   0   0   0   0   0   0   1   0   0   0;

     pil/2  0.14    q4         0       0   0.1840  0   0
     0      0   0   0   0   0   0   0   1   0   0   0;
     ];


 q=[q1 q2 q3 q4];
 qd=[qd1 qd2 qd3 qd4];
 qdd=[qdd1 qdd2 qdd3 qdd4];
 gravedad=[0 0 -g];
 torques = rne(dyn, q, qd, qdd, gravedad);
 simple(torques);
```

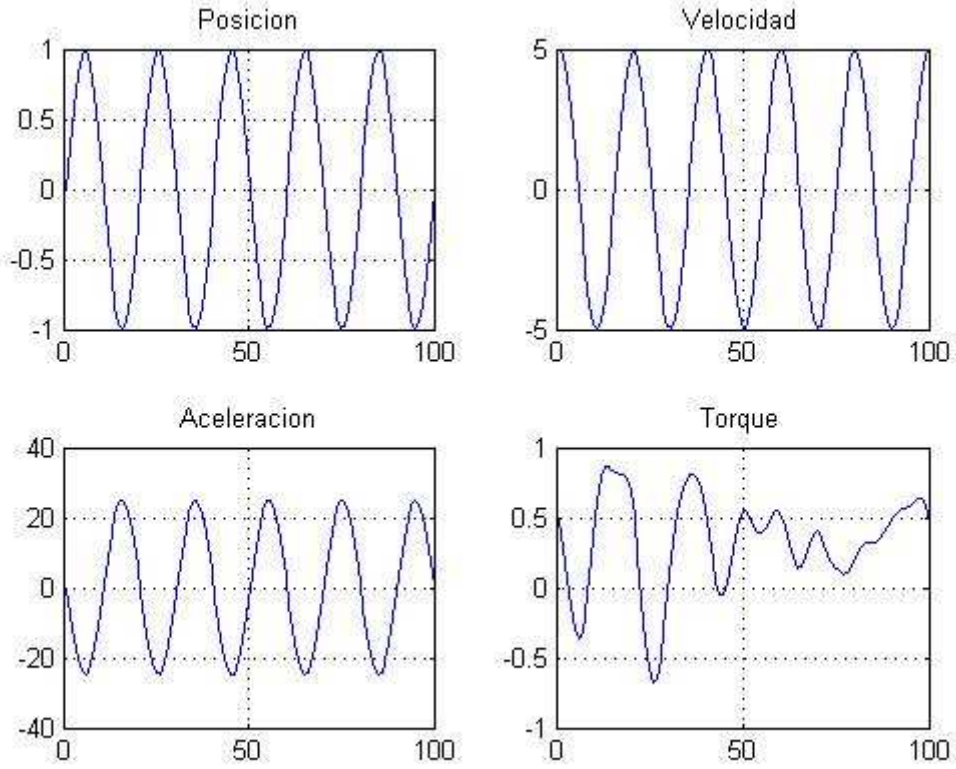
Obteniéndose los torques para cada articulación. Los mismos no son representados aquí por cuestiones de espacio y simplicidad.
Luego, se simuló el comportamiento del torque, a partir de las siguientes funciones de q, elegidas de forma arbitraria:

```
 q1 = 'sin(t)'
q2 = 'sin(5*t)'
q3 = 'cos(3*t)'
q4 = 'cos(2*t)'
```

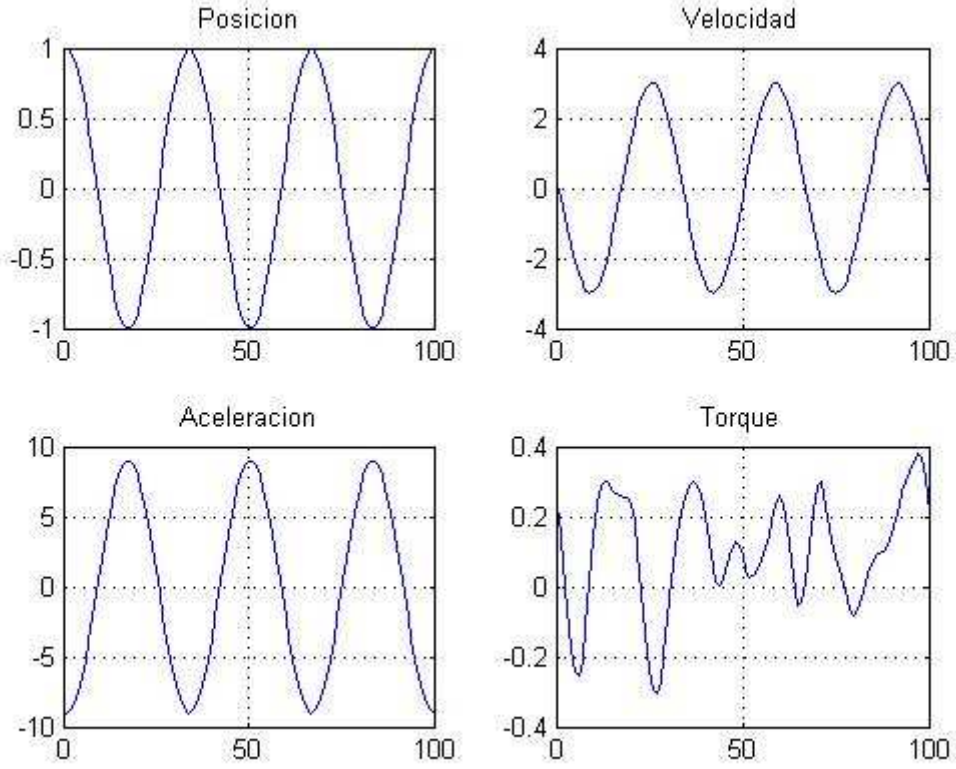
q1


Posicion
Velocidad
Aceleracion
Torque

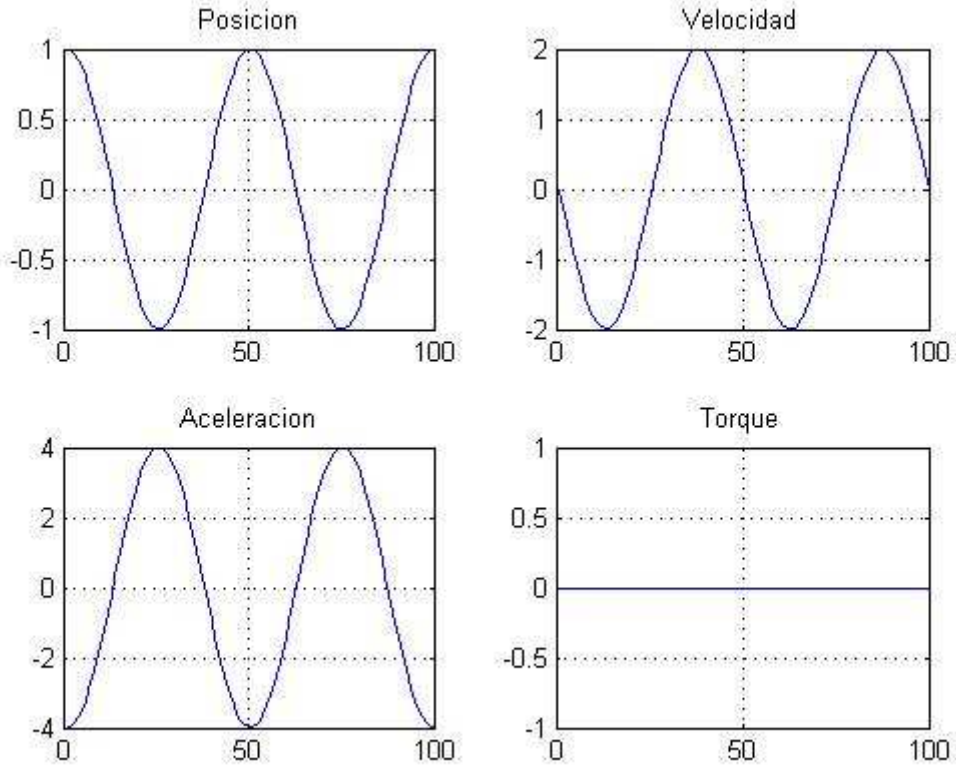
q2


Posicion
Velocidad
Aceleracion
Torque

q3


Posicion
Velocidad
Aceleracion
Torque

q4


Posicion
Velocidad
Aceleracion
Torque

El torque correspondiente a la última articulación es 0 porque el brazo no sostiene ninguna carga.

Programación en VHDL

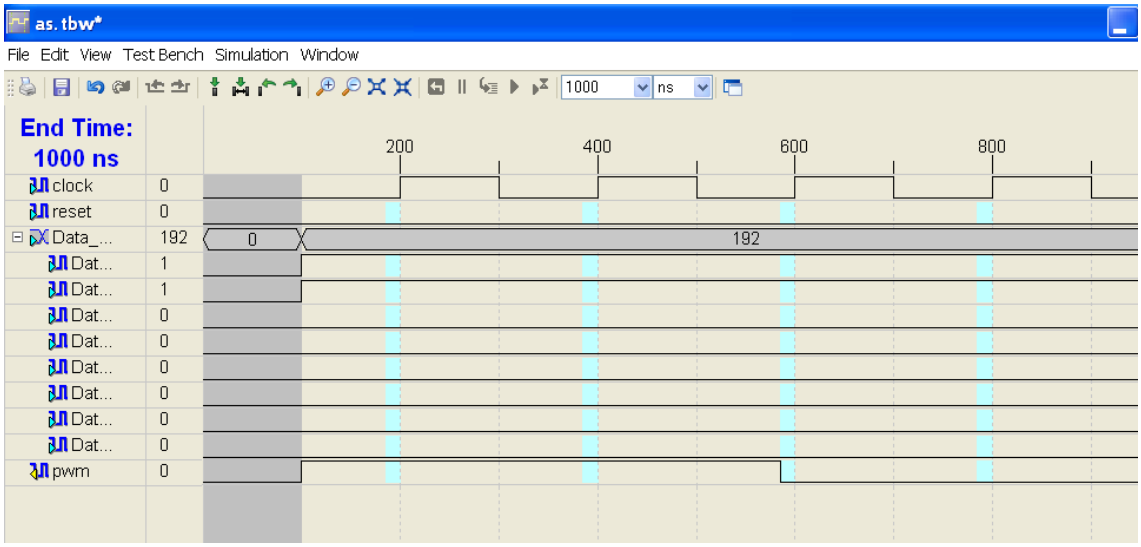
Finalmente, luego de obtener las ecuaciones de los torques, se procede a programar una FPGA para controlar los cuatro motores de CC con PWM, a través de un puente H.

```
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;
```

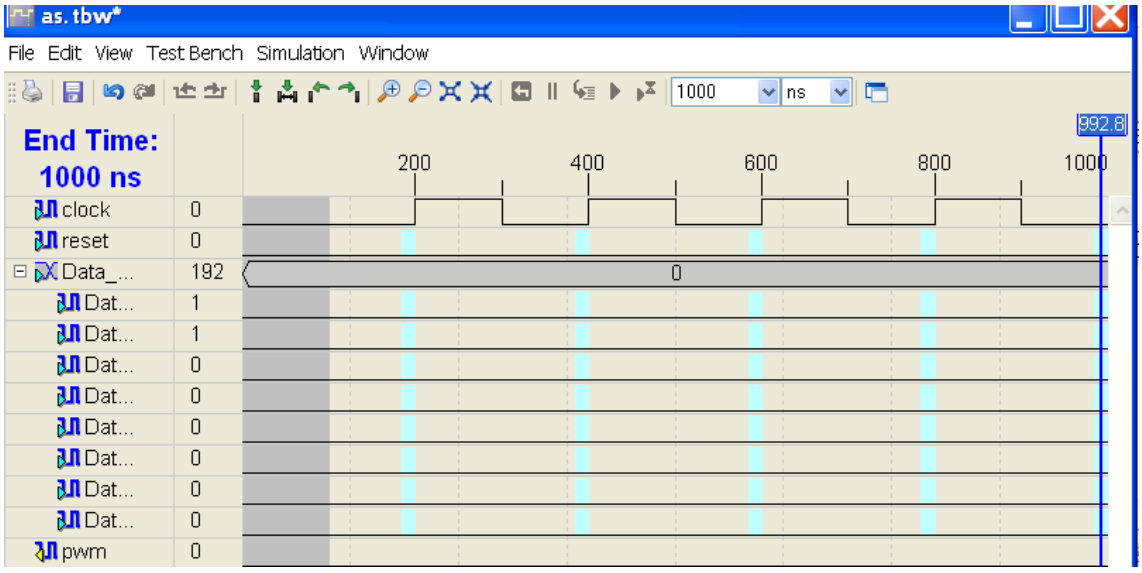
```
                END PROCESS;

 -- 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;
```

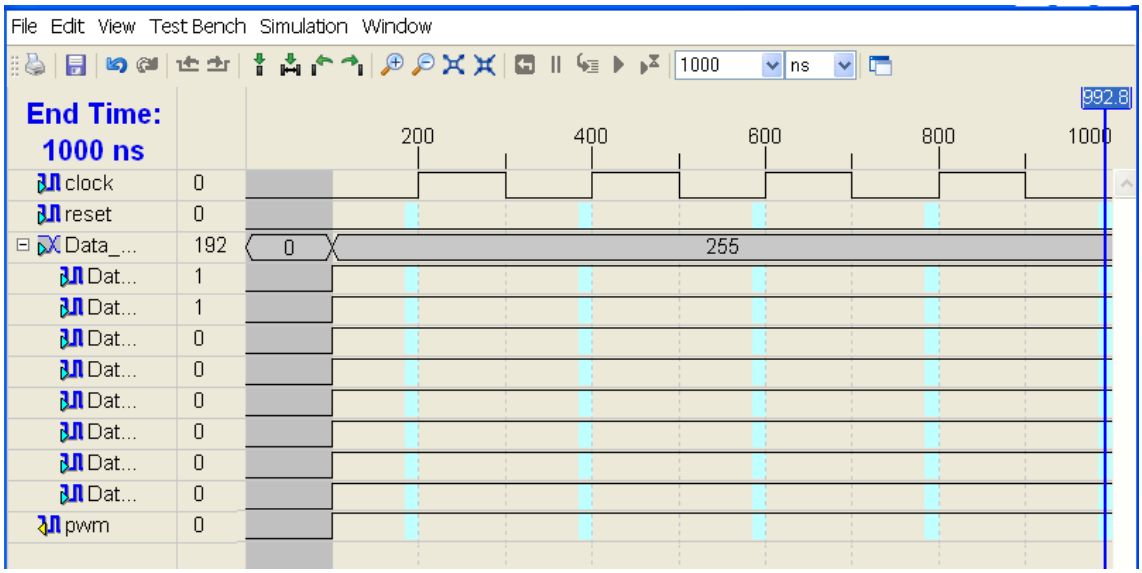
Ejemplo 1 cuando data signal tiene un valor de 192 decimado sin signo:



Ejemplo 2 cuando data signal tiene un valor de 0 decimado sin signo:



Ejemplo 3 cuando data signal tiene un valor de 0 decimado sin signo:



Conclusiones

El presente trabajo lleva a cabo el desarrollo dinámico del robot M5 y se demuestra a través de gráficos el desarrollo tanto de posición, velocidad, torque y aceleración, siendo consistente con valores teóricos y teniendo un comportamiento esperado.
Pudimos ver el comportamiento del PWM a través de una simulación en Xilinx mostrando una respuesta esperada para la salida. Asimismo, tomamos varios valores para poder ratificar más exactamente el comportamiento óptimo que se obtiene para el desarrollo practico.