Robótica

Trabajo Práctico Nº 2

Alumnos: Diniello, Felipe

Ortiz, Federico

Profesor: Ing. Giannetta, Hernán

Jefe de TP: Ing. Granzella, Eduardo Damián

Curso: R6055

Horario: Lunes 19:00 a 23:00

Año: 2013

Modelo dinamico del robot

Introducción

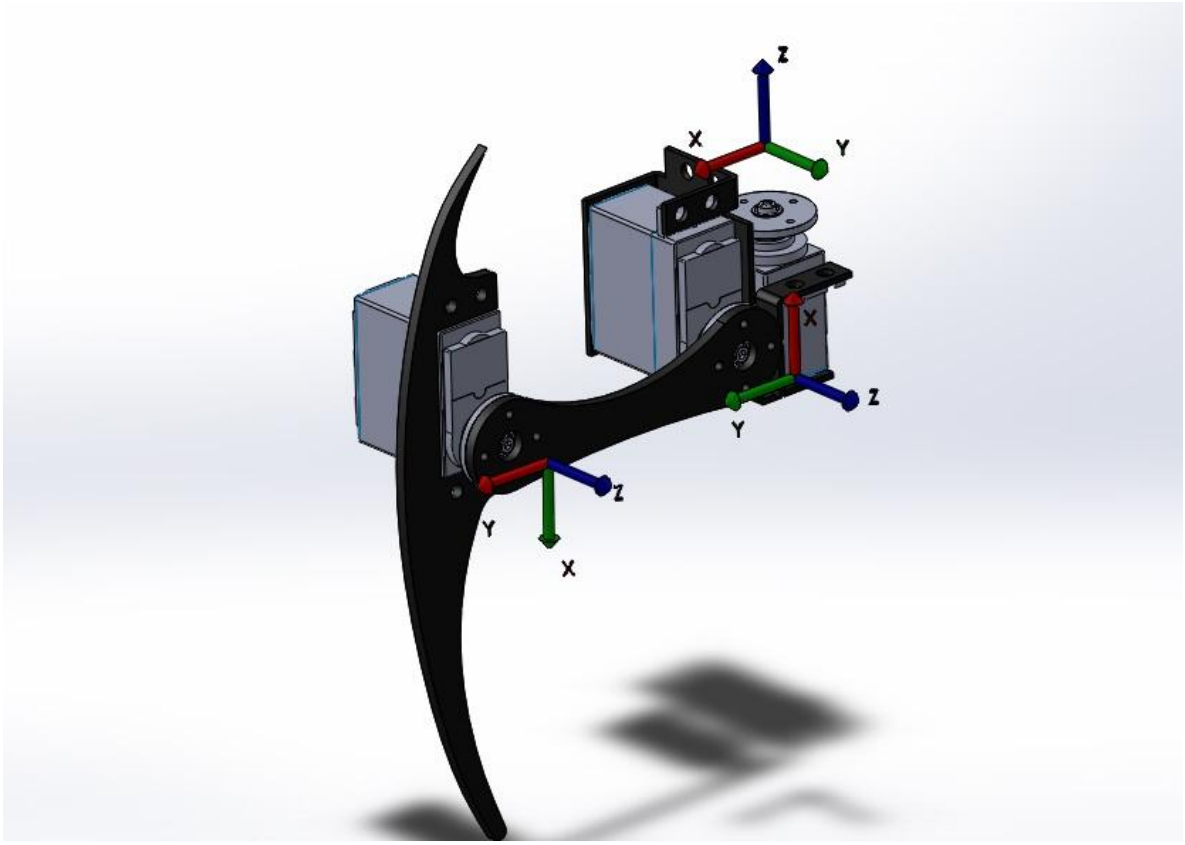
Utilizando el toolbox Hemero proporcionado por la cátedra buscamos la expresión de los torques de cada articulación necesarios para alcanzar la posición deseada.

Fue necesario recurrir a una herramienta de CAD para obtener el modelo de una extremidad y, simulando aluminio como material, obtener los parámetros mecánicos necesarios para completar la matriz de parámetros DyN.

Al ser las seis patas del robot exactamente iguales, realizamos el análisis dinamico para una sola.

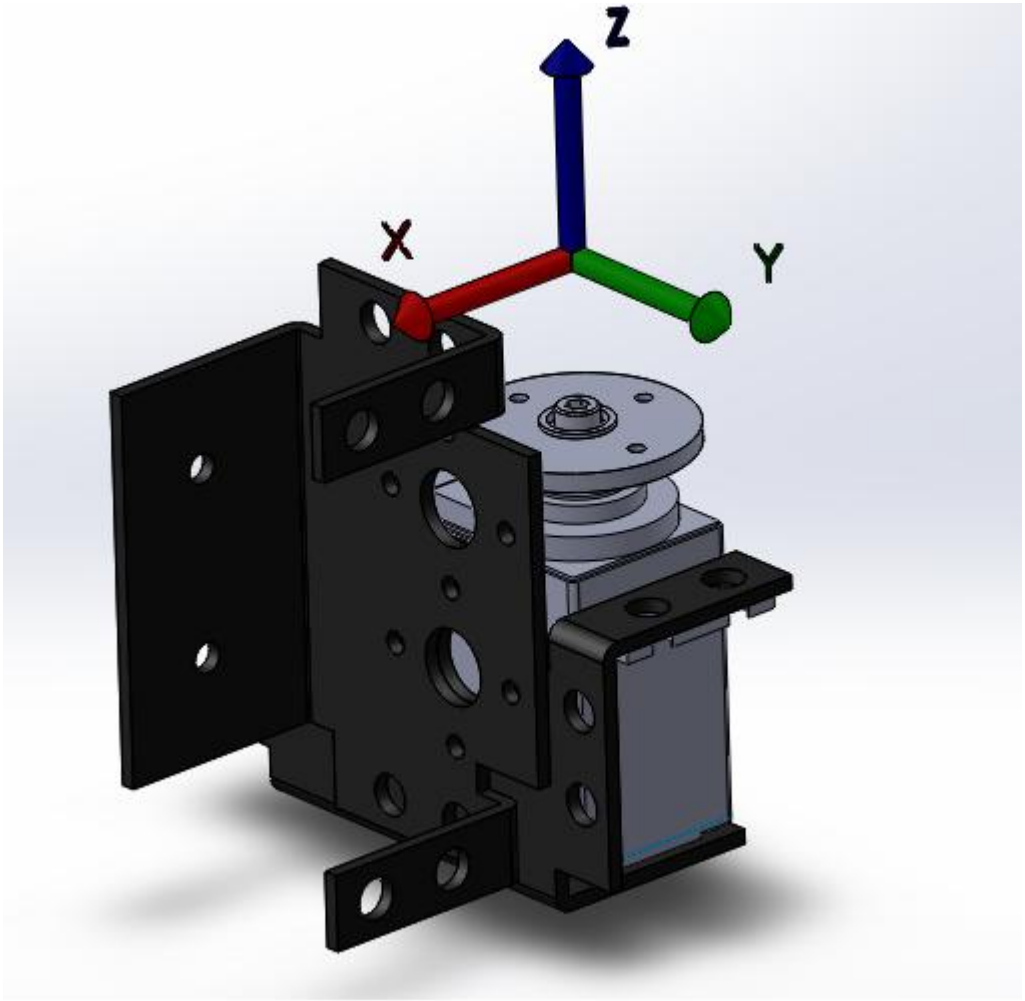
Parámetros mecanicos

Partiendo con el siguiente modelo



Simulamos que el material utilizado tiene una densidad de 1.18 gramos por centímetro cubico lo que nos da:

- Primera articulación:



Masa = 5.02 gramos

Volumen = 4250.63 milímetros cúbicos

Área de superficie = 6446.26 milímetros cuadrados

Centro de masa: (milímetros)

X = 8.20

Y = 9.53

Z = -17.87

Ejes principales de inercia y momentos principales de inercia: (gramos * milímetros cuadrados)

Medido desde el centro de masa.

Ix = (0.00, -1.00, 0.00)	Px = 974.33
Iy = (0.28, 0.00, 0.96)	Py = 1483.22
Iz = (-0.96, 0.00, 0.28)	Pz = 1982.42

Momentos de inercia: (gramos * milímetros cuadrados)

Obtenidos en el centro de masa y alineados con el sistema de coordenadas de resultados.

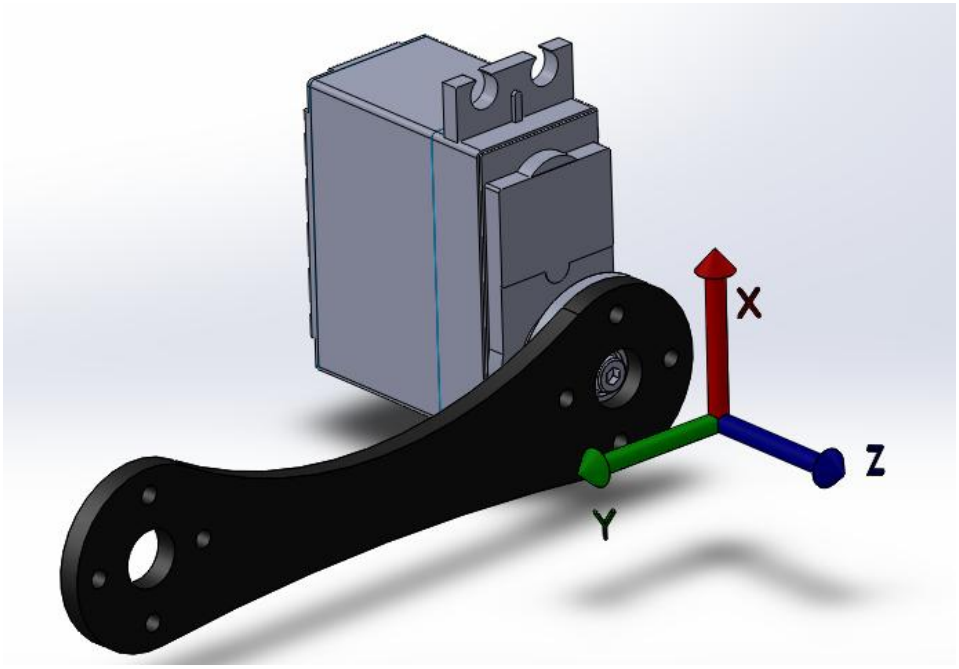
Lxx = 1944.40	Lxy = 0.00	Lxz = 132.42
Lyx = 0.00	Lyy = 974.33	Lyz = 0.00
Lzx = 132.42	Lzy = 0.00	Lzz = 1521.24

Momentos de inercia: (gramos * milímetros cuadrados)

Medido desde el sistema de coordenadas de salida.

Ixx = 4001.53	Ixy = 391.94	Ixz = -602.49
Iyx = 391.94	Iyy = 2913.14	Iyz = -854.15
Izx = -602.49	Izy = -854.15	Izz = 2314.00

- Segunda articulación:



Masa = 5.53 gramos

Volumen = 4688.17 milímetros cúbicos

Área de superficie = 4151.50 milímetros cuadrados

Centro de masa: (milímetros)

X = 38.10

Y = -1.65

Z = -1.50

Ejes principales de inercia y momentos principales de inercia: (gramos * milímetros cuadrados)

Medido desde el centro de masa.

Ix = (1.00, 0.00, 0.00) Px = 186.13

Iy = (0.00, 1.00, 0.00) Py = 4968.04

Iz = (0.00, 0.00, 1.00) Pz = 5145.88

Momentos de inercia: (gramos * milímetros cuadrados)

Obtenidos en el centro de masa y alineados con el sistema de coordenadas de resultados.

Lxx = 186.13 Lxy = 0.00 Lxz = 0.00

Lyx = 0.00 Lyy = 4968.04 Lyz = 0.00

Lzx = 0.00 Lzy = 0.00 Lzz = 5145.88

Momentos de inercia: (gramos * milímetros cuadrados)

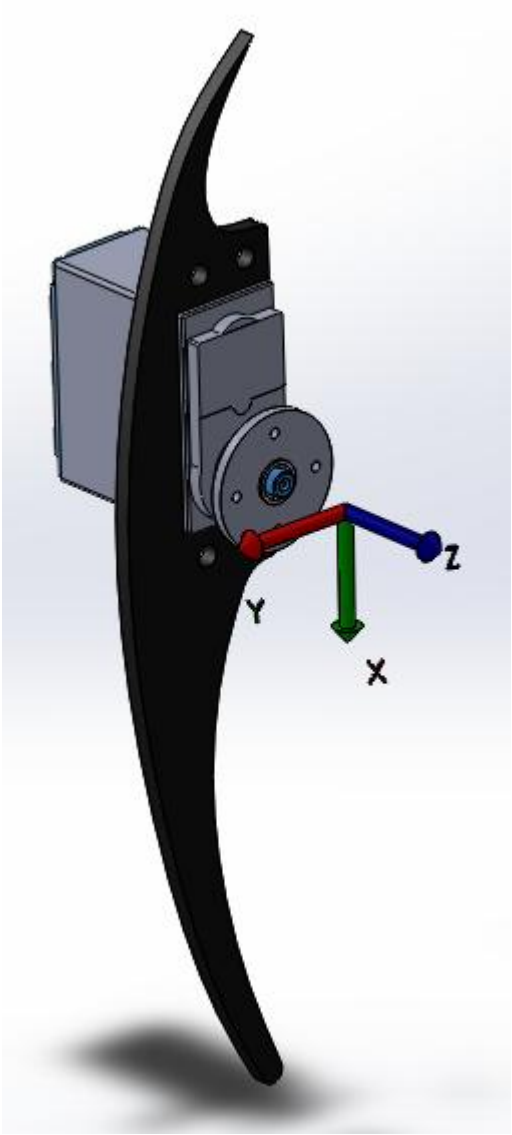
Medido desde el sistema de coordenadas de salida.

Ixx = 213.69 Ixy = -348.39 Ixz = -316.16

Iyx = -348.39 Iyy = 13010.86 Iyz = 13.72

Izx = -316.16 Izy = 13.72 Izz = 13191.36

- Tercera articulación:



Masa = 7.49 gramos

Volumen = 6347.04 milímetros cúbicos

Área de superficie = 5694.44 milímetros cuadrados

Centro de masa: (milímetros)

X = 17.65

Y = 8.54

Z = 1.50

Ejes principales de inercia y momentos principales de inercia: (gramos * milímetros cuadrados)

Medido desde el centro de masa.

Ix = (1.00, -0.01, 0.00)	Px = 379.28
Iy = (-0.01, -1.00, -0.00)	Py = 13000.68
Iz = (0.00, 0.00, -1.00)	Pz = 13368.73

Momentos de inercia: (gramos * milímetros cuadrados)

Obtenidos en el centro de masa y alineados con el sistema de coordenadas de resultados.

Lxx = 380.38	Lxy = -117.64	Lxz = 0.02
Lyx = -117.64	Lyy = 12999.59	Lyz = 0.00
Lzx = 0.02	Lzy = 0.00	Lzz = 13368.73

Momentos de inercia: (gramos * milímetros cuadrados)

Medido desde el sistema de coordenadas de salida.

Ixx = 943.67	Ixy = 1011.22	Ixz = 198.25
Iyx = 1011.22	Iyy = 15348.48	Iyz = 95.96
Izx = 198.25	Izy = 95.96	Izz = 16247.22

Obtención de torques del robot

Mediante el enfoque dinámico de Lagrange-Euler, buscamos obtener los torques que cada motor debe entregar al sistema para alcanzar los movimientos deseados. Estos quedaran definidos por la posición, velocidad y aceleración de cada articulación. Usando el toolbox de Hemero para matlab implementamos el siguiente scrypt.

```
syms q1 q2 q3 real
syms dq1 dq2 dq3 real
syms ddq1 ddq2 ddq3 real
syms n1 m1 m2 m3 real

q=[q1 q2 q3];
dq=[dq1 dq2 dq3];
ddq=[ddq1 ddq2 ddq3];

grav=[0 0 -9.8];


dyn_hexa= [ -pi/2  m1   q1   n1  0   5.02  8.2   9.5   -17.8  4001
2913  2314  392  -854 -602  0 1 0  0  0 ;

            0      m2   q2   0   0   5.53  38.1  -1.6  -1.5   214
13010 13200 -350 13   -316  0 1 0  0  0 ;

            0      m3   q3   0   0   7.48  17.6  8.54   1.5   943
15350 16300 1011 96    198  0 1 0  0  0 ]


tor=rne(dyn_hexa,q,dq,ddq,grav);
```

Luego de ejecutarlo obtenemos las siguientes expresiones de torque:

T1=(10877173267*ddq1)/250000 + (10101023317*ddq2)/250000 + (1197658323*ddq3)/62500 - (504259*cos(q1))/1250 + (233681*sin(q1))/500 - (79849*sin(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/1250 - (1106*sin(q2)*((49*cos(q1))/5 - ddq1*m2))/125 + (79849*cos(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/1250 + (16456*sin(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/125 - (1106*cos(q2)*(m2*dq1^2 + (49*sin(q1))/5))/125 + (210693*(dq1 + dq2)*((8*dq1)/5 + (8*dq2)/5))/1000 - (1106*(dq1 + dq2)*((381*dq1)/10 + (381*dq2)/10))/125 + (210693*sin(q2)*(m2*dq1^2 + (49*sin(q1))/5))/1000 + (79849*(dq1 + dq2 + dq3)*((88*dq1)/5 + (88*dq2)/5 + (88*dq3)/5))/1250 - (16456*(dq1 + dq2 + dq3)*((427*dq1)/50 + (427*dq2)/50 + (427*dq3)/50))/125 + m2*(cos(q2)*((210693*ddq1)/1000 + (210693*ddq2)/1000 + (553*(dq1 + dq2)*((8*dq1)/5 + (8*dq2)/5))/100 + (553*sin(q2)*(m2*dq1^2 + (49*sin(q1))/5))/100 + cos(q3)*((16456*ddq1)/125 + (16456*ddq2)/125 + (16456*ddq3)/125 + (187*sin(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/25 - (187*(dq1 + dq2 + dq3)*((427*dq1)/50 + (427*dq2)/50 + (427*dq3)/50))/25 +

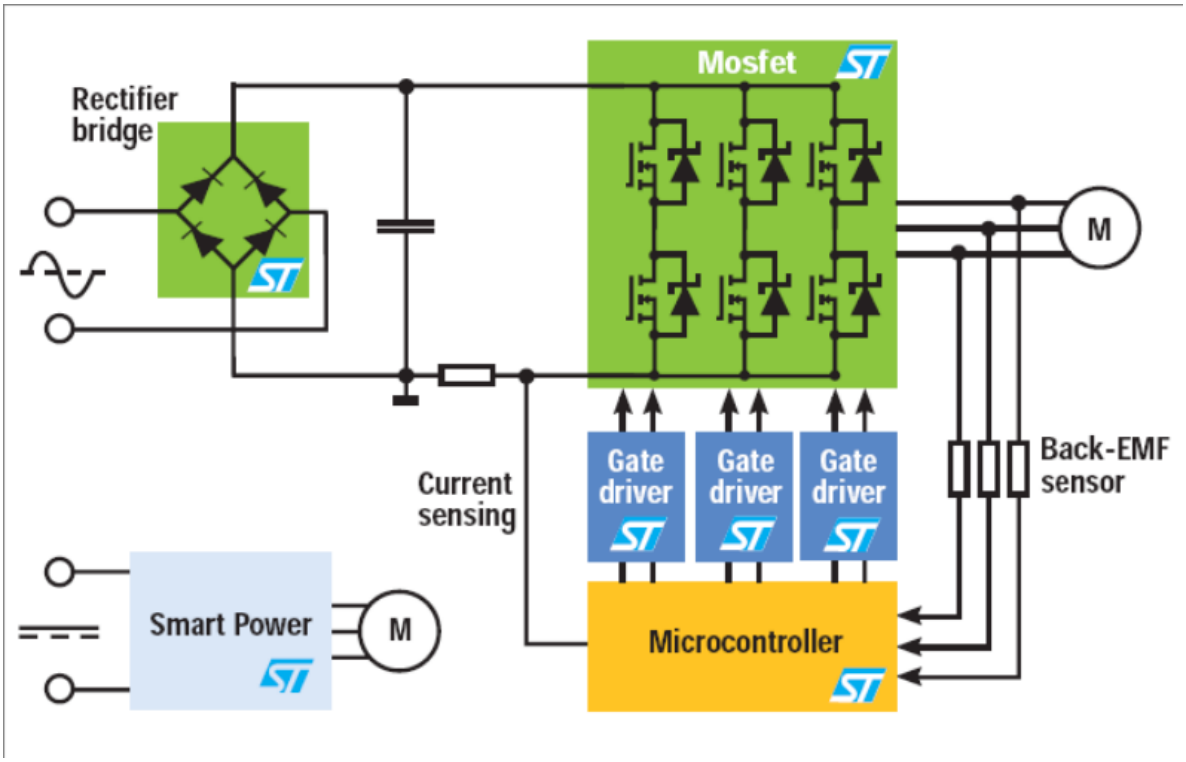
(187*cos(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/25) - sin(q3)*((79849*ddq1)/1250 + (79849*ddq2)/1250 + (79849*ddq3)/1250 - (187*sin(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/25 + (187*cos(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/25 + (187*(dq1 + dq2 + dq3)*((88*dq1)/5 + (88*dq2)/5 + (88*dq3)/5))/25) - (553*cos(q2)*((49*cos(q1))/5 - ddq1*m2))/100) - sin(q2)*((553*sin(q2)*((49*cos(q1))/5 - ddq1*m2))/100 - (1106*ddq2)/125 - (1106*ddq1)/125 + (553*cos(q2)*(m2*dq1^2 + (49*sin(q1))/5))/100 + (553*(dq1 + dq2)*((381*dq1)/10 + (381*dq2)/10))/100 + cos(q3)*((79849*ddq1)/1250 + (79849*ddq2)/1250 + (79849*ddq3)/1250 - (187*sin(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/25 + (187*cos(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/25 + (187*(dq1 + dq2 + dq3)*((88*dq1)/5 + (88*dq2)/5 + (88*dq3)/5))/25) + sin(q3)*((16456*ddq1)/125 + (16456*ddq2)/125 + (16456*ddq3)/125 + (187*sin(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/25 - (187*(dq1 + dq2 + dq3)*((427*dq1)/50 + (427*dq2)/50 + (427*dq3)/50))/25 + (187*cos(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/25))) + m3*(cos(q3)*((16456*ddq1)/125 + (16456*ddq2)/125 + (16456*ddq3)/125 + (187*sin(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/25 - (187*(dq1 + dq2 + dq3)*((427*dq1)/50 + (427*dq2)/50 + (427*dq3)/50))/25 + (187*cos(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/25) - sin(q3)*((79849*ddq1)/1250 + (79849*ddq2)/1250 + (79849*ddq3)/1250 - (187*sin(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/25 + (187*cos(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/25 + (187*(dq1 + dq2 + dq3)*((88*dq1)/5 + (88*dq2)/5 + (88*dq3)/5))/25)) + (16456*cos(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/125 - (210693*cos(q2)*((49*cos(q1))/5 - ddq1*m2))/1000

T2=(10101023317*ddq1)/250000 + (10101023317*ddq2)/250000 + (1197658323*ddq3)/62500 - (79849*sin(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/1250 - (1106*sin(q2)*((49*cos(q1))/5 - ddq1*m2))/125 + (79849*cos(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/1250 + (16456*sin(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/125 - (1106*cos(q2)*(m2*dq1^2 + (49*sin(q1))/5))/125 + (210693*(dq1 + dq2)*((8*dq1)/5 + (8*dq2)/5))/1000 - (1106*(dq1 + dq2)*((381*dq1)/10 + (381*dq2)/10))/125 + (210693*sin(q2)*(m2*dq1^2 + (49*sin(q1))/5))/1000 + (79849*(dq1 + dq2 + dq3)*((88*dq1)/5 + (88*dq2)/5 + (88*dq3)/5))/1250 - (16456*(dq1 + dq2 + dq3)*((427*dq1)/50 + (427*dq2)/50 + (427*dq3)/50))/125 + m3*(cos(q3)*((16456*ddq1)/125 + (16456*ddq2)/125 + (16456*ddq3)/125 + (187*sin(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/25 - (187*(dq1 + dq2 + dq3)*((427*dq1)/50 + (427*dq2)/50 + (427*dq3)/50))/25 + (187*cos(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/25) - sin(q3)*((79849*ddq1)/1250 + (79849*ddq2)/1250 + (79849*ddq3)/1250 - (187*sin(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/25 + (187*cos(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/25 + (187*(dq1 + dq2 + dq3)*((88*dq1)/5 + (88*dq2)/5 + (88*dq3)/5))/25)) + (16456*cos(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/125 - (210693*cos(q2)*((49*cos(q1))/5 - ddq1*m2))/1000

T3=(1197658323*ddq1)/62500 + (1197658323*ddq2)/62500 + (1197658323*ddq3)/62500 - (79849*sin(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/1250 + (79849*cos(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/1250 + (16456*sin(q3)*(sin(q2)*((49*cos(q1))/5 - ddq1*m2) + m3*(dq1 + dq2)^2 + cos(q2)*(m2*dq1^2 + (49*sin(q1))/5)))/125 + (79849*(dq1 + dq2 + dq3)*((88*dq1)/5 + (88*dq2)/5 + (88*dq3)/5))/1250 - (16456*(dq1 + dq2 + dq3)*((427*dq1)/50 + (427*dq2)/50 + (427*dq3)/50))/125 + (16456*cos(q3)*(sin(q2)*(m2*dq1^2 + (49*sin(q1))/5) + m3*(ddq1 + ddq2) - cos(q2)*((49*cos(q1))/5 - ddq1*m2)))/125

Como podemos ver, cada torque depende de la posición, velocidad y aceleración de cada articulación. Además, se aprecia que a medida que avanzamos de un eslabón al siguiente (por ejemplo de T3 a T2) se obtienen relaciones cada vez más complejas por la dependencia entre sí. Es por esto que la expresión de T1 es mucho más compleja que la de T3.

Implementacion de PWM utilizando una FPGA



Para realizar este control necesitamos:

· Generar el PWM: Un módulo encargado de generar una señal de PWM. A partir de un dato que recibe debe proporcionar una señal cuadrada cuyo ciclo de actividad es variable.

· 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.

El código necesario para implementar esto es el siguiente:

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;
```

transistores.vhd

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


ENTITY driver_mosfet IS
PORT (
                        S1 :out STD_LOGIC;
                        S2 :out STD_LOGIC;
                        S3 :out STD_LOGIC;
                        S4 :out STD_LOGIC;
                        S5 :out STD_LOGIC;
                        S6 :out STD_LOGIC;
                        hall_sensor :in  std_logic_vector(2 downto 0);
                        reset :in STD_LOGIC;
                        clock :in STD_LOGIC;
                        pwm_in :in STD_LOGIC
    );


END driver_mosfet;


ARCHITECTURE arch_driver OF driver_mosfet IS
-- Internal signal declaration
BEGIN
Cntrl_PWM_puenteH: PROCESS (reset, hall_sensor, pwm_in) IS
BEGIN
```

```
        IF (reset = '1') THEN
                s1<= '0'; s2<= '0'; s3<= '0'; s4<= '0'; s5<= '0'; s6<= '0';
        ELSE
                CASE hall_sensor IS
                        WHEN "100" =>                   --Estado 1
                                s1<= '1'; s2<= '0'; s3<= '0'; s4<= pwm_in; s5<= '0'; s6<= '0';

                        WHEN "110" =>                   --Estado 2
                                s1<= '1'; s2<= '0'; s3<= '0'; s4<= '0'; s5<= '0'; s6<= pwm_in;

                        WHEN "010" =>                   --Estado 3
                                s1<= '0'; s2<= '0'; S3<= '1'; s4<= '0'; s5<= '0'; s6<= pwm_in;

                        WHEN "011" =>                   --Estado 4
                                s1<= '0'; s2<= pwm_in; s3<= '1'; s4<= '0'; s5<= '0'; s6<= '0';

                        WHEN "001" =>                   --Estado 5
                                s1<= '0'; s2<= pwm_in; s3<= '0'; s4<= '0'; s5<= '1'; s6<= '0';

                        WHEN "101" =>                   --Estado 6
                                s1<= '0'; s2<= '0'; s3<= '0'; s4<= pwm_in; s5<= '1'; s6<= '0';

                        WHEN OTHERS =>
                                s1<= '0'; s2<= '0'; s3<= '0'; s4<= '0'; s5<= '0'; s6<= '0';
                END CASE;
        END IF;

END PROCESS;
```

```
END ARCHITECTURE arch_driver;
```