UTN FRBA
INGENIERÍA ELECTRÓNICA

ROBÓTICA

TRABAJO PRÁCTICO Nº 2

Análisis dinámico de un robot y su implementación en FPGA

DOCENTE: Ing. Hernán Giannetta
JTP: Ing. Damián Graznella

Matías Baldo
Fernando Valenzuela

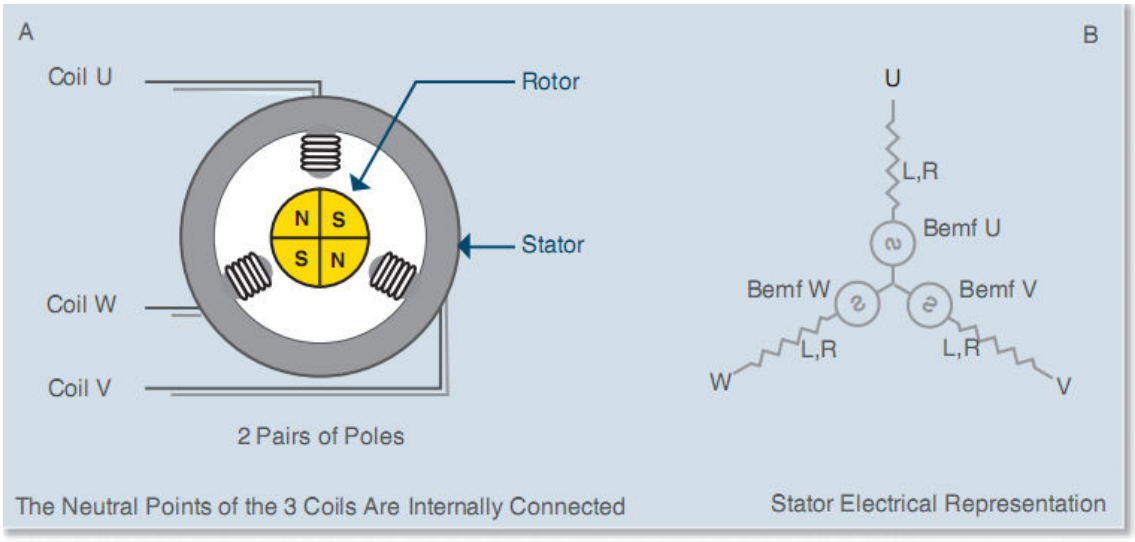
2010

Índice

CONTROL MOTOR BRUSHLESS MEDIANTE FPGA

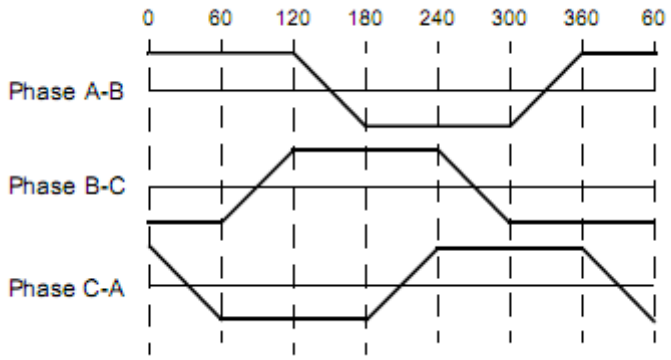
INTRODUCCIÓN MOTORES BRUSHLESS:

Los motores brushless se encuentran dentro de las ramas de los motores sincrónicos. El rotor de este consta de imanes permanentes y el estator de un bobinado formado por 2 o 3 fases, en nuestro caso 3 fases. Estos son motores de corriente continua donde su principal característica es que no existe un contacto mecánico para transferir energía, como sí sucede en los motores normales de DC.



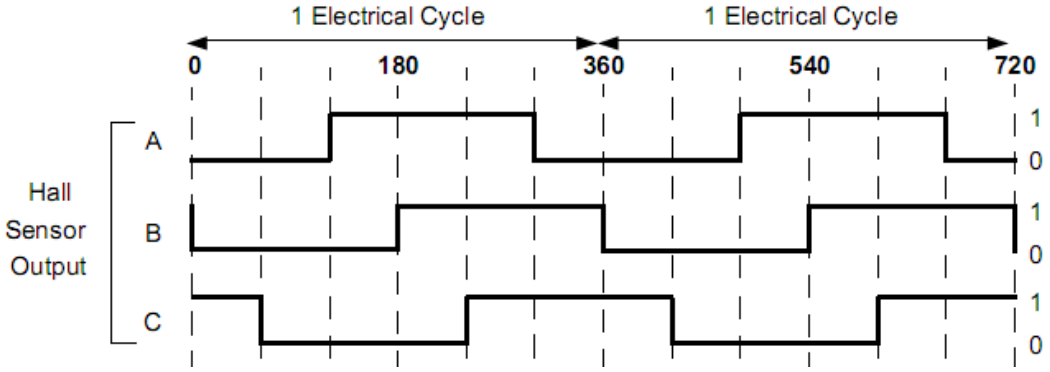
Para hacer que el motor gire hay que energizar los diferentes bobinados en una determinada secuencia. Más adelante desarrollaremos la secuencia a implementar, por ahora mencionaremos que en todo momento dos bobinas son excitadas al mismo tiempo mientras otra se mantiene desenergizada. También sucede que al ser motores sincrónicos es necesario respetar esta secuencia según la posición del rotor. Para determinar esta posición existen 2 métodos:

- Se mide la tensión inducida por el rotor sobre la bobina que no se excita. En este momento cabe mencionar que existen dos tipos de estatores en los motores brushless, en un caso la tensión generada (FEM) es senoidal y en otro caso es trapezoidal. En nuestro caso utilizaremos un motor trapezoidal.



Midiendo los cruces por cero podremos determinar la posición del rotor.

- En unos de los extremos del rotor se colocan 3 sensores de efectos hall. Estos están ubicados de tal manera que generan un codigo que identifica en un determinado rango la posición del rotor:



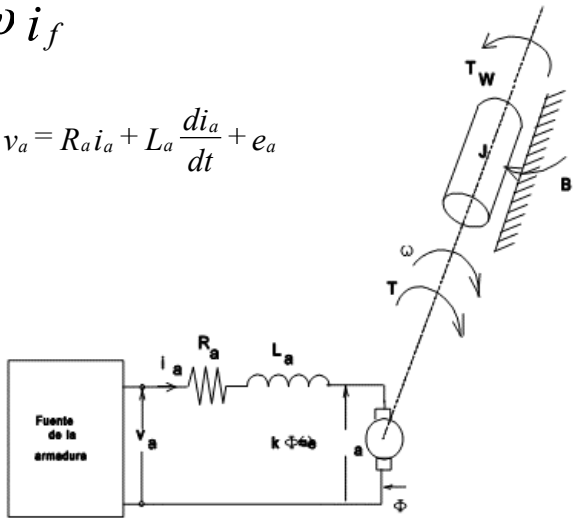
CARACTERÍSTICAS MECANICA/ELECTRICAS DEL MOTOR BRUSHLESS:

Como ya mencionamos anteriormente el motor brushless es considerado un motor de corriente continua. Las ecuaciones de este son muy parecidas a las de un motor de corriente continua, es necesario recalcar que poseen mas fases que este y el rotor no necesita excitación externa:

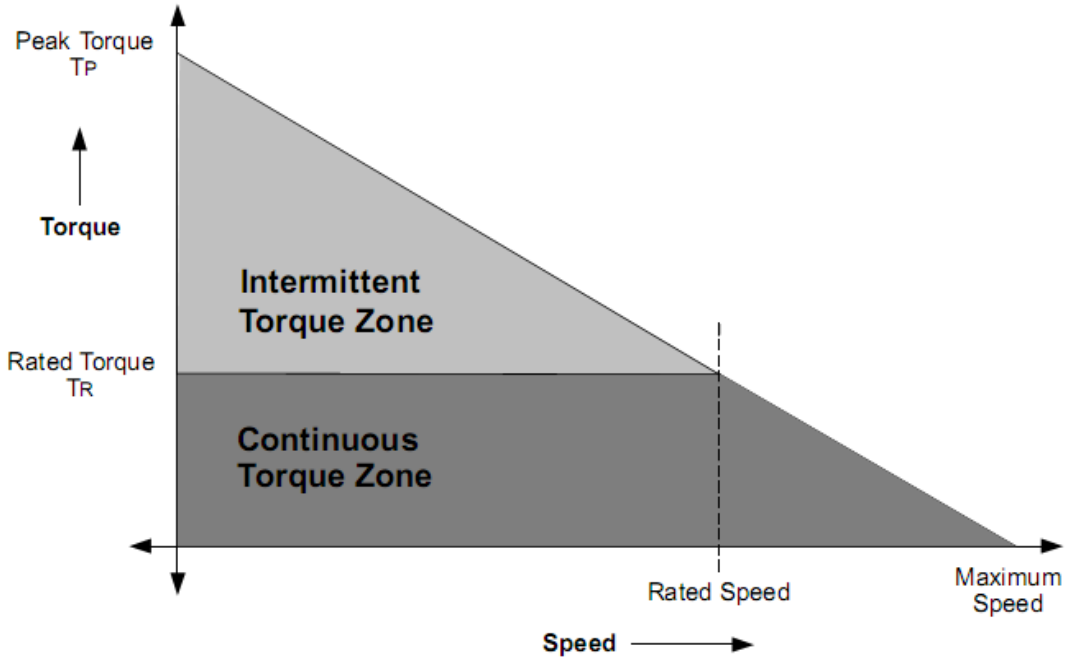
TORQUE: $T_m = k_t \phi i_a$

FEM: $e_a = k_v \omega i_f$

TENSION ARMADURA: $v_a = R_a i_a + L_a \frac{di_a}{dt} + e_a$



Con estas ecuaciones el control es lineal, haciendo más fácil su implementación en lazos de control. A continuación veremos una curva de comportamiento de estos motores:



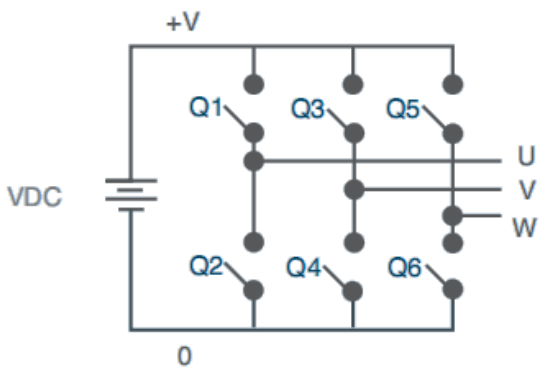
En esta gráfica se ven dos zonas de trabajo, una donde el torque máximo puede superar el valor nominal del motor. Esta zona de trabajo no debe ser continua ya que se estaría circulando por el estator una corriente mayor a la nominal, haciendo que el motor trabaje forzadamente. En la otra zona el torque puede variar de 0 al valor nominal, hasta superar la velocidad nominal del motor, a partir de aquí el torque se ve limitado por la velocidad-

CONTROL DEL MOTOR BRUSHLESS:

Para realizar la secuencia correcta utilizaremos la siguiente tabla sacada de una nota de aplicación de Atmel:

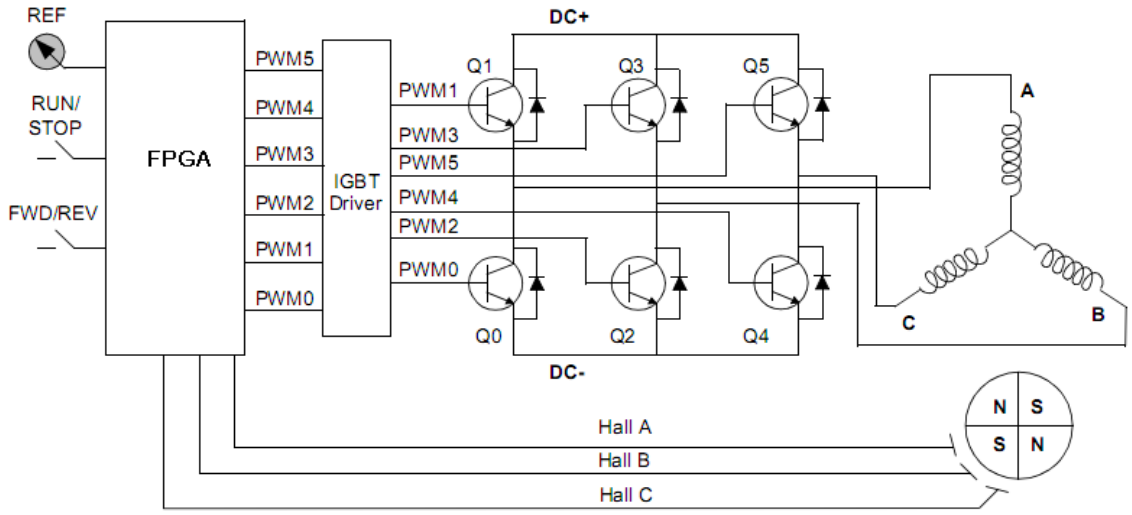
Hall Sensors	Clockwise (CW)			Counter Clockwise (CCW)		
H3-H2-H1	Step	Motor Voltage	Active Switches	Step	Motor Voltage	Active Switches
1-0-1	1	U - V	Q1 & Q4	4	V - U	Q3 & Q4
0-0-1	2	U - W	Q1 & Q6	5	W - U	Q5 & Q2
0-1-1	3	V - W	Q3 & Q6	6	W - V	Q5 & Q4
0-1-0	4	V - U	Q3 & Q2	1	U - V	Q1 & Q4
1-1-0	5	W - U	Q5 & Q2	2	U - W	Q1 & Q6
1-0-0	6	W - V	Q5 & Q4	3	V - W	Q3 & Q6

Donde Q1…Q6 corresponden a los dispositivos de conmutación (MOSFET, IGBT) ubicados de la siguiente forma:



ESQUEMA DE CONTROL:

A continuación mostramos un esquema donde se implementa un control sencillo de velocidad:



Se puede observar un puente H y sus respectivos drivers formado por 3 ramas donde excitan al estator del motor, luego se ven los 3 sensores de efecto hall los cuales indican la posición del rotor.

Y por último vemos un bloque que se encarga de la lógica de control. Este consta de una FPGA, la cual a partir de una referencia de velocidad externa ajusta las señales de excitación en un determinado duty haciendo que la tensión media del estator varie según la velocidad requerida. Tambíen podemos observar la posibilidad de cambiar el sentido de giro del motor.

IMPLEMENTACÍON EN VHDL DEL CONTROL MEDIANTE FPGA:

PROGRAMA FPGA:

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

ENTITY PWM_Brushless IS
  PORT (  clock, sentido              :in  std_logic;
          reset                       :in  std_logic;
          velocidad                    :in  std_logic_vector(7 downto 0);
          sensores_hall                :in  std_logic_vector(2 downto 0);
          gates                       :out std_logic_vector(0 to 5);
          salida_pwm                  :out std_logic);
END PWM_Brushless;

ARCHITECTURE Arch_PWMBrushless OF PWM_Brushless 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;
  SIGNAL reg_sensores_hall     : std_logic_vector(2 downto 0);
  SIGNAL cambio_sensor         : std_logic;
  SIGNAL tiempo_muerto         : std_logic_vector(3 downto 0);

  --Funcion: Incrementa 1 variable

  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;

BEGIN


 -- ACTUALIZA EL CONTADOR CON CADA CICLO DE CLOCK-> CARGA ENTRADA EN UN REGISTRO INTERNO
  PROCESS(reset,clock,reg_out)
        BEGIN
               IF (reset ='1') THEN
                      reg_out <="00000000";
               ELSIF (rising_edge(clock)) THEN
                              reg_out <= velocidad;
               END IF;
        END PROCESS;


  -- GENERA LOS TIEMPOS PARA EL DUTY:
  --  * Si la salida esta activa, decrementa cnt_out_int
  --  * Si la salida esta inactiva, incrementa cnt_out_int
  -- La suma de los dos tiempos forman el periodo del 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 <= 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;


-- Cuando cnt_out_int llega a uno de los limites (superior o inferior), rco_int
-- modifica su estado, de esta manera vuelve a inicializarse y si se encontraba sumando
-- pasa a restarse o viceversa.
 PROCESS(reset, cnt_out_int, rco_int, clock)
        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;


-- Con cada cambio de rco_int niega el estado anterior de pwm_int
```

```
        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;

              salida_pwm <= pwm_int;

-- Verifica que el PWM este activo, si lo esta, basandose en la posicion
-- de los sensores hall y el sentido activa la rama correcta del puente h
--  +Vbus   *--------------------
--               |      |      |
--              G1     G3     G5
--               |      |      |
--               |      |      |
--              G2     G4     G6
--               |      |      |
--   GND   *----------------------   donde gates = G1-G2-G3-G4-G5-G6

  PROCESS (clock,reset,pwm_int,sentido,sensores_hall)
  BEGIN

   IF ((pwm_int = '0') or (reset='1')) THEN
      gates <= "000000";
   ELSE
       IF (sentido = '0') THEN
         CASE sensores_hall IS
            WHEN "000" =>
                gates <= "000000";
            WHEN "001" =>
                gates <= "100001";
            WHEN "010" =>
                gates <= "011000";
            WHEN "011" =>
                gates <= "001001";
            WHEN "100" =>
                gates <= "000110";
            WHEN "101" =>
                gates <= "100100";
            WHEN "110" =>
                gates <= "010010";
            WHEN OTHERS =>
                gates <= "000000";
          END CASE;
        ELSE
          CASE sensores_hall IS
            WHEN "000" =>
                gates <= "000000";
            WHEN "001" =>
                gates <= "010010";
            WHEN "010" =>
                gates <= "100100";
            WHEN "011" =>
                gates <= "000110";
            WHEN "100" =>
                gates <= "001001";
            WHEN "101" =>
                gates <= "011000";
            WHEN "110" =>
                gates <= "100001";
            WHEN OTHERS =>
              gates <= "000000";
          END CASE;
        END IF;
    END IF;
  END PROCESS;

END Arch_PWMBrushless;
```

PROGRAMA TEST BENCH:

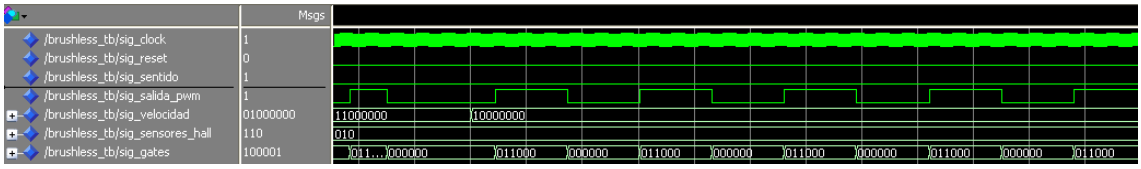
```
LIBRARY ieee;

use ieee.std_logic_1164.all;

ENTITY Brushless_TB IS

END Brushless_TB;

ARCHITECTURE Arch_Brushless_TB OF Brushless_TB IS

COMPONENT PWM_Brushless
  PORT (  clock, sentido              :in  std_logic;
          reset                       :in  std_logic;
          velocidad                   :in  std_logic_vector(7 downto 0);
          sensores_hall               :in  std_logic_vector(2 downto 0);
          gates                       :out std_logic_vector(0 to 5);
          salida_pwm                  :out std_logic);
END COMPONENT;

-- Internal signal declaration

SIGNAL sig_sentido               : std_logic;
SIGNAL sig_clock                 : std_logic;
SIGNAL sig_reset                 : std_logic;
SIGNAL sig_velocidad             : std_logic_vector(7 downto 0);
SIGNAL sig_gates                 : std_logic_vector(0 to 5);
SIGNAL sig_sensores_hall         : std_logic_vector(2 downto 0);
SIGNAL sig_salida_pwm            : std_logic;

shared variable ENDSIM: boolean:=false;
  constant clk_period:TIME:=100 ns;


BEGIN
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;
-- Instantiating top level design Component pwm_fpga

inst_PWM_Brushless : PWM_Brushless
PORT MAP(
               sentido => sig_sentido,
               clock  => sig_clock,
               reset  => sig_reset,
               velocidad      => sig_velocidad,
               sensores_hall => sig_sensores_hall,
               gates => sig_gates,
               salida_pwm => sig_salida_pwm
       );

stimulus_process: PROCESS

BEGIN

--RESET INICIAL
  sig_sentido <= '0';
       sig_reset <= '1';
       wait for 100 ns;
       sig_reset <= '0';

-- PRUEBO DIFERENTES VELOCIDADES CON UNA UNICA RAMA
  sig_sensores_hall <= "010";

--VELOCIDAD 1:
```

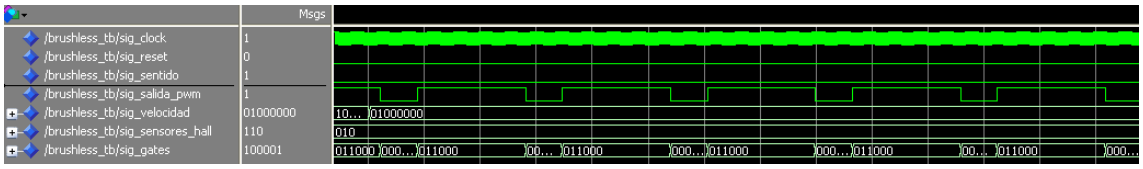
```
        sig_velocidad <= "11000000";
        wait for 300 us;
--VELOCIDAD 2:
        sig_velocidad <= "10000000";
        wait for 300 us;
--VELOCIDAD 3:
        sig_velocidad <= "01000000";
        wait for 300 us;

--PRUEBO A VELOCIDAD CONSTANTE EL GIRO DE LOS MOTORES
  sig_sentido <= '0';
  sig_sensores_hall <= "101";
  wait for 60us;
  sig_sensores_hall <= "001";
  wait for 60us;
  sig_sensores_hall <= "011";
  wait for 60us;
  sig_sensores_hall <= "010";
  wait for 60us;
  sig_sensores_hall <= "110";
  wait for 60us;
  sig_sensores_hall <= "100";
  wait for 60us;
  sig_sensores_hall <= "101";
  wait for 60us;
  sig_sensores_hall <= "001";
  wait for 60us;

  sig_sentido <= '1';
  sig_sensores_hall <= "010";
  wait for 60us;
  sig_sensores_hall <= "110";
  wait for 60us;
  sig_sensores_hall <= "100";
  wait for 60us;
  sig_sensores_hall <= "101";
  wait for 60us;
  sig_sensores_hall <= "001";
  wait for 60us;
  sig_sensores_hall <= "011";
  wait for 60us;
  sig_sensores_hall <= "010";
  wait for 60us;
  sig_sensores_hall <= "110";
  wait for 60us;


wait;


 END PROCESS stimulus_process;

END Arch_Brushless_TB;
```

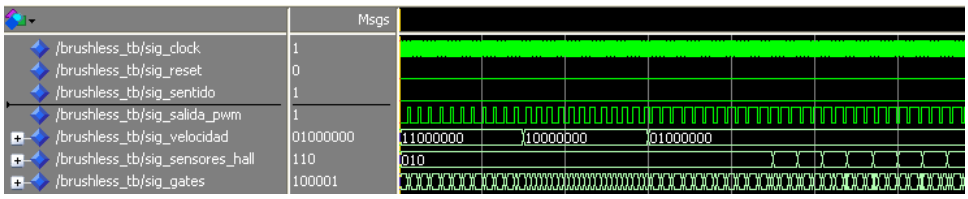
Para la implementación del código nos basamos de una nota de aplicación de Atmel, en la cual se implementa una modulación simple de PWM. Nosotros modificamos este código para module la salida de la rama que se encuentra activa. A la vez leemos el los sensores de efecto hall y el sentido para aplicar de forma correcta la secuencia necesaria para el puente H.

SIMULACIÓN:

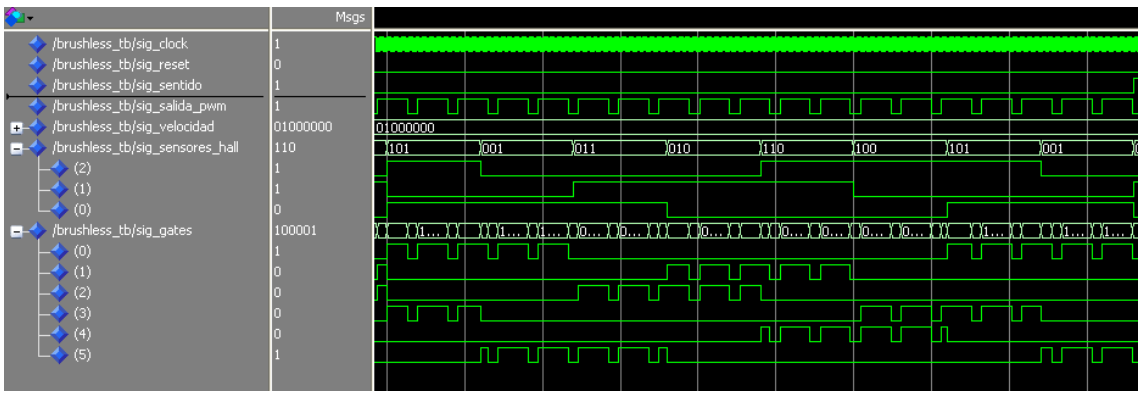
Como primer paso se probó el modulo de PWM con diferentes velocidades:



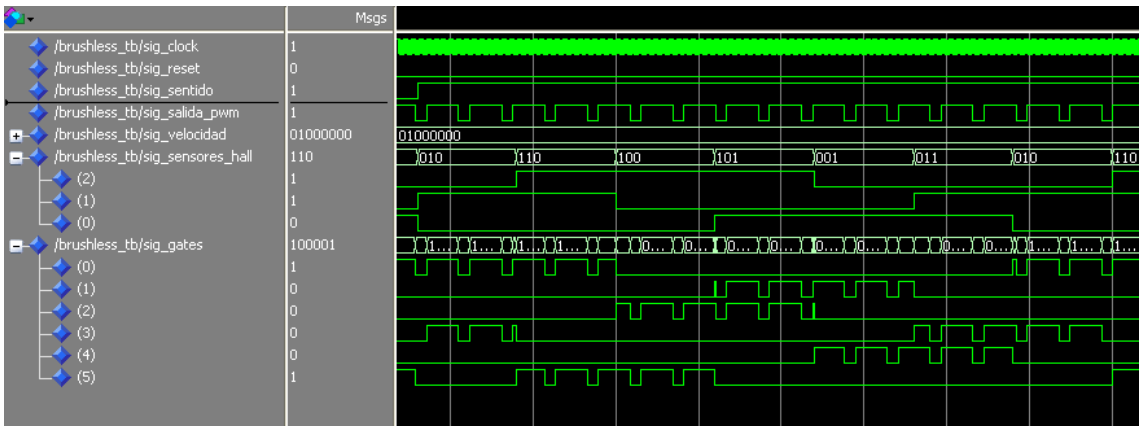




Luego, a velocidad constante se empezó a probar la lógica de conmutación. Para esto, se generó las señales correspondientes a los sensores de efectos hall:



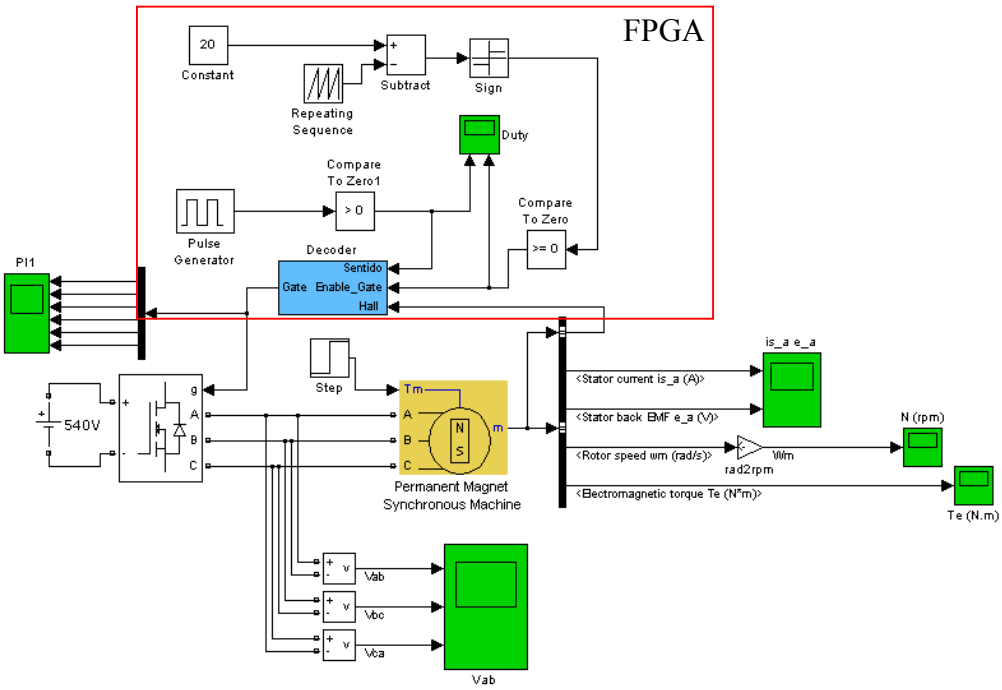
Hall Sensors	Clockwise (CW)		
H3-H2-H1	Step	Motor Voltage	Active Switches
1-0-1	1	U - V	Q1 & Q4
0-0-1	2	U - W	Q1 & Q6
0-1-1	3	V - W	Q3 & Q6
0-1-0	4	V - U	Q3 & Q2
1-1-0	5	W - U	Q5 & Q2
1-0-0	6	W - V	Q5 & Q4



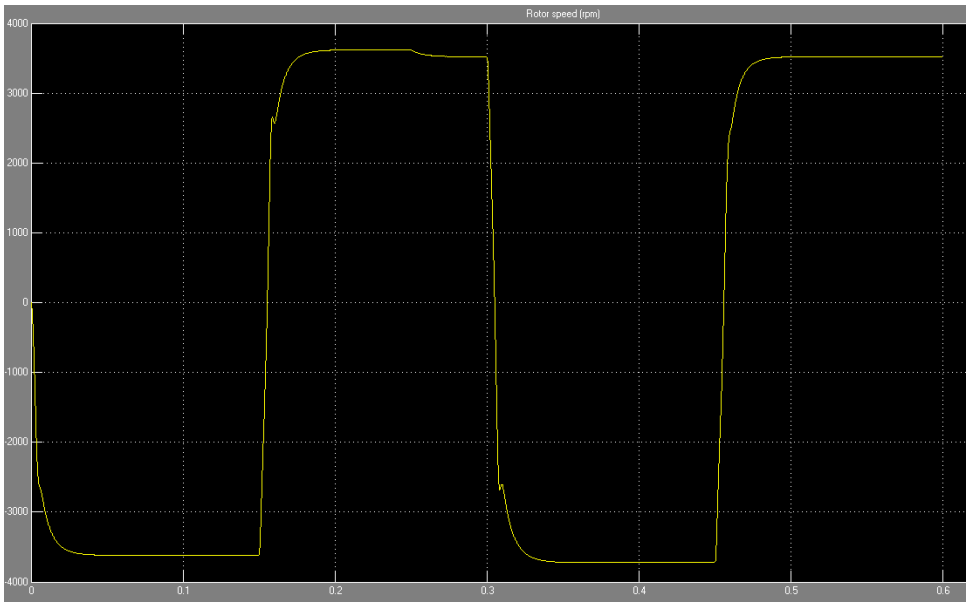
Hall Sensors	Counter Clockwise (CCW)		
H3-H2-H1	Step	Motor Voltage	Active Switches
1-0-1	4	V - U	Q3 & Q4
0-0-1	5	W - U	Q5 & Q2
0-1-1	6	W - V	Q5 & Q4
0-1-0	1	U - V	Q1 & Q4
1-1-0	2	U - W	Q1 & Q6
1-0-0	3	V - W	Q3 & Q6

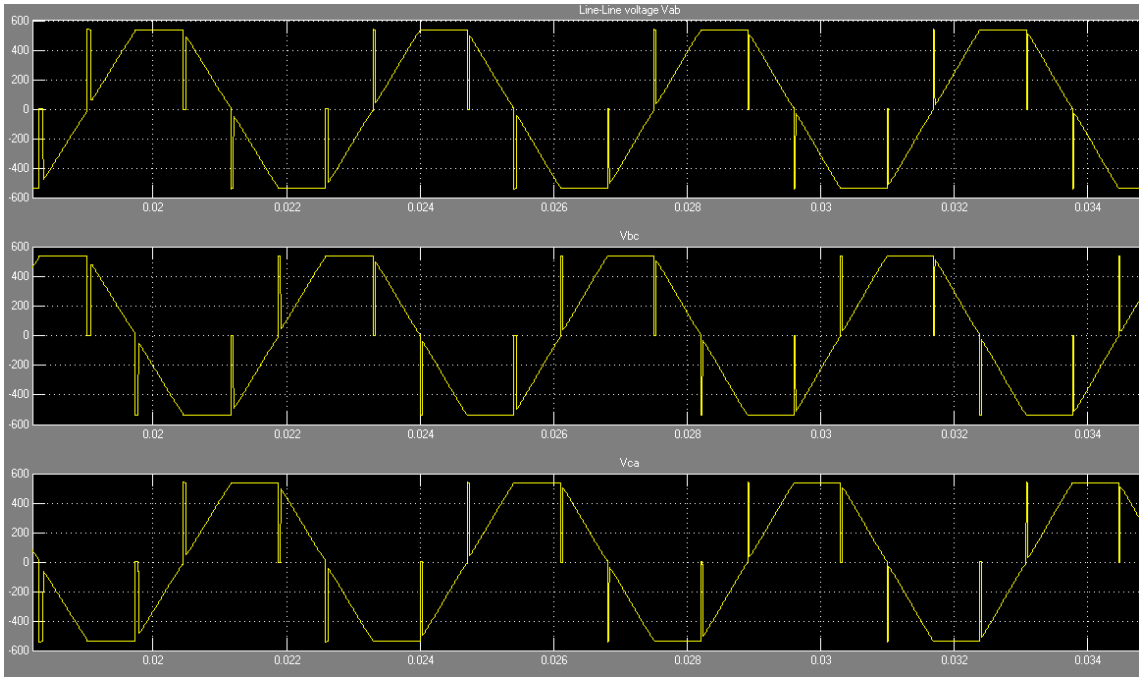
SIMULACIÓN MOTOR BRUSHLESS EN MATLAB – LAZO ABIERTO:

Se simulará en Matlab un motor brushless en vació y con carga. Se utilizo el siguiente modelo en Simulink:

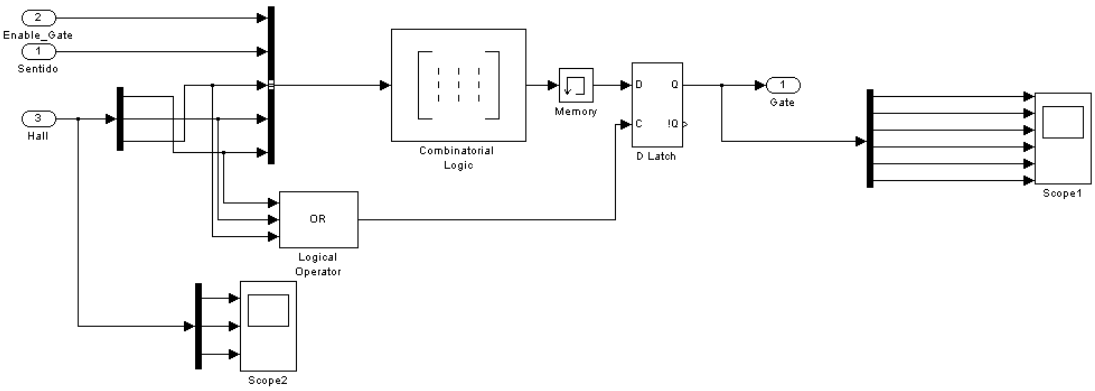


Lo que anteriormente vimos en VHDL es simulado por los bloques recuadrados. La simulación consta de aplicar máxima velocidad en un sentido y revertírselo en dos condiciones diferentes, con carga y sin carga.





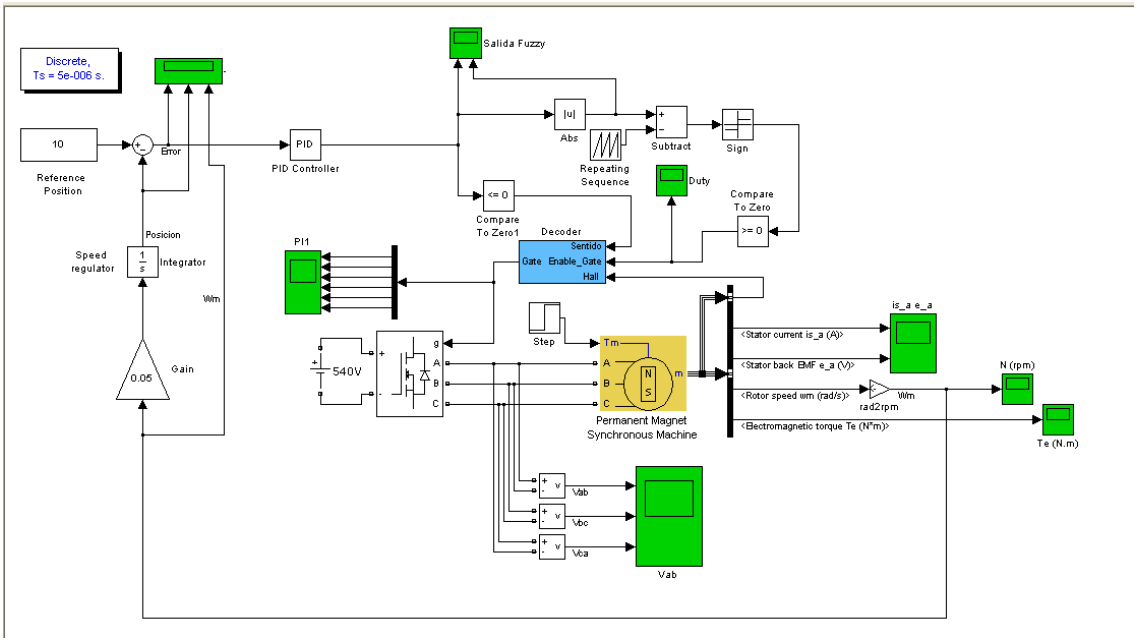
El bloque llamado Decoder se implementó de la siguiente manera:



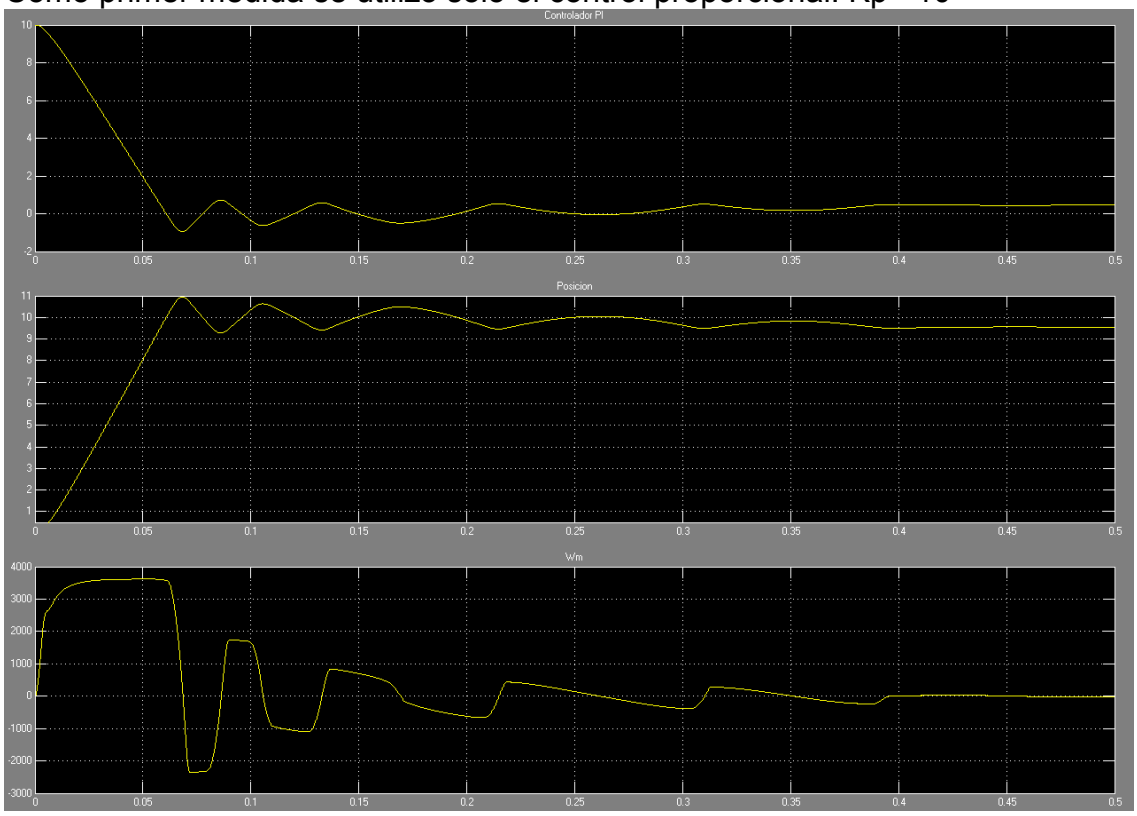
Donde el bloque Combinational Logic, contiene las secuencias necesarias para el control del puente H.

SIMULACIÓN CONTROL DE POSICIÓN – LAZO CON PID:

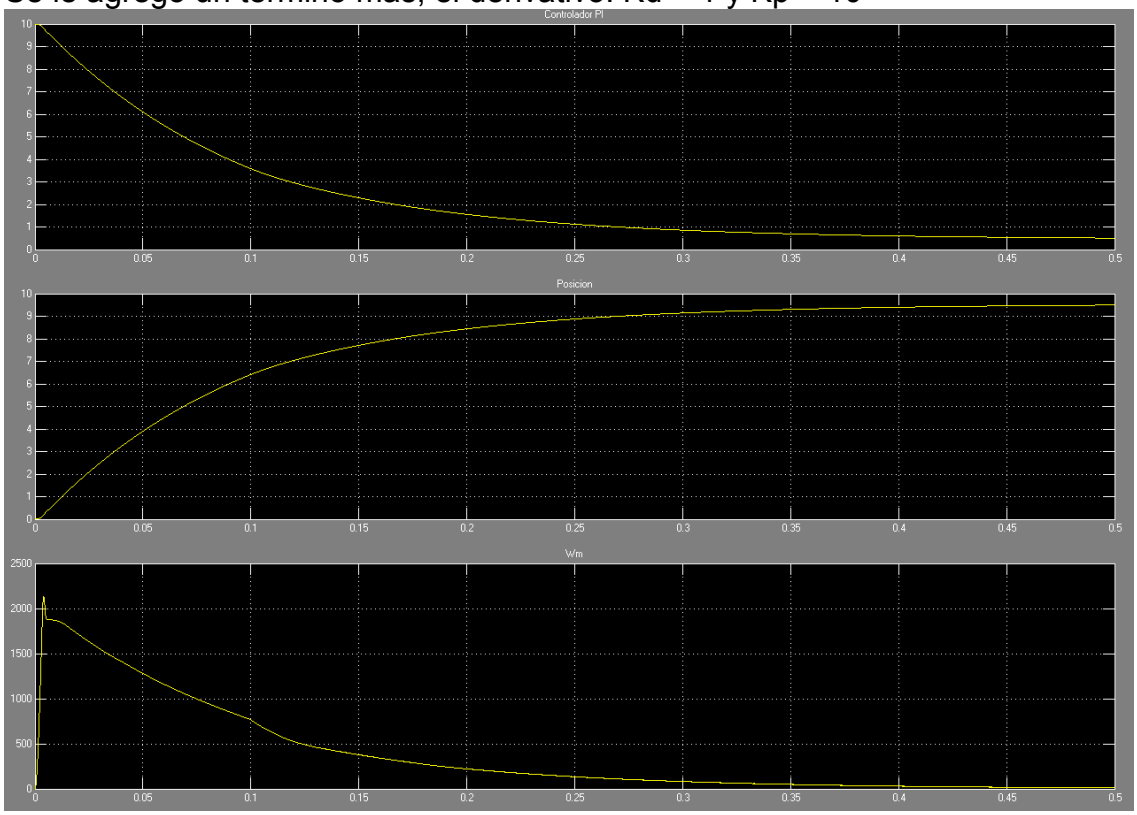
A continuación se planteó resolver un sistema de control de posición mediante un lazo PID.



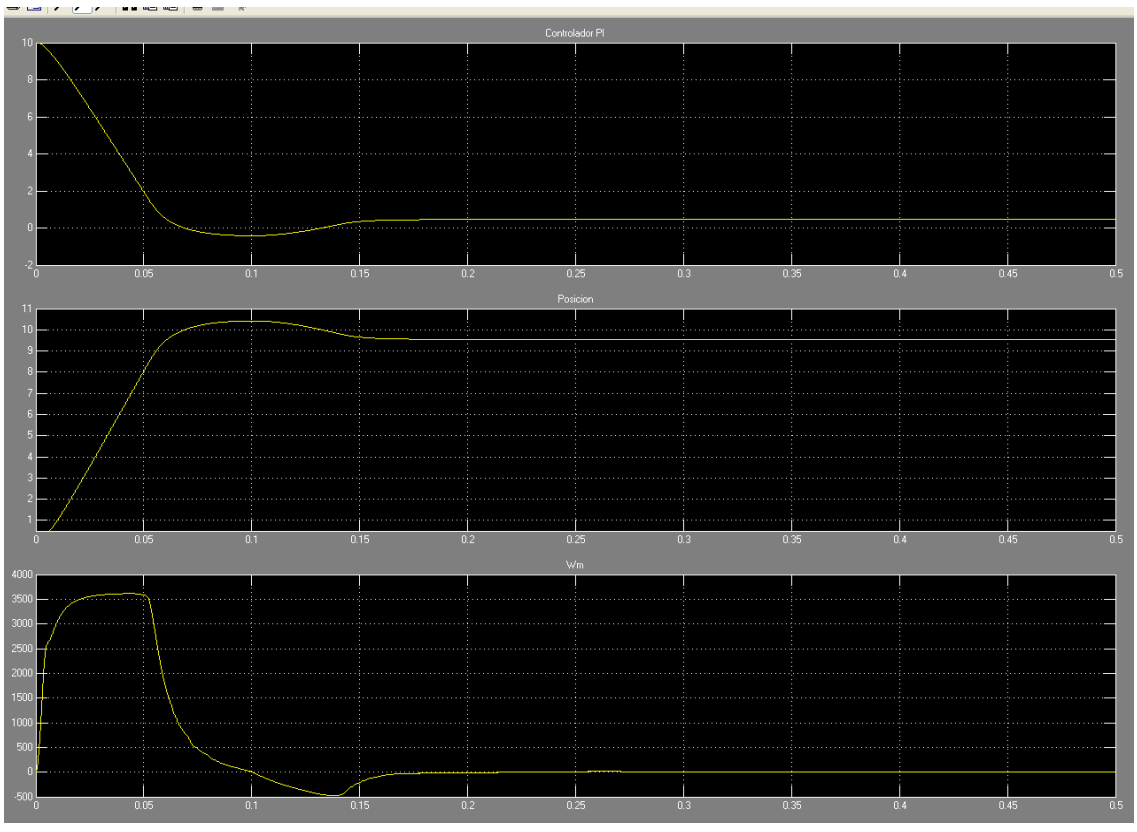
Como primer medida se utilizo solo el control proporcional: Kp =10



Se le agregó un termino mas, el derivativo: Kd = 1 y Kp = 10

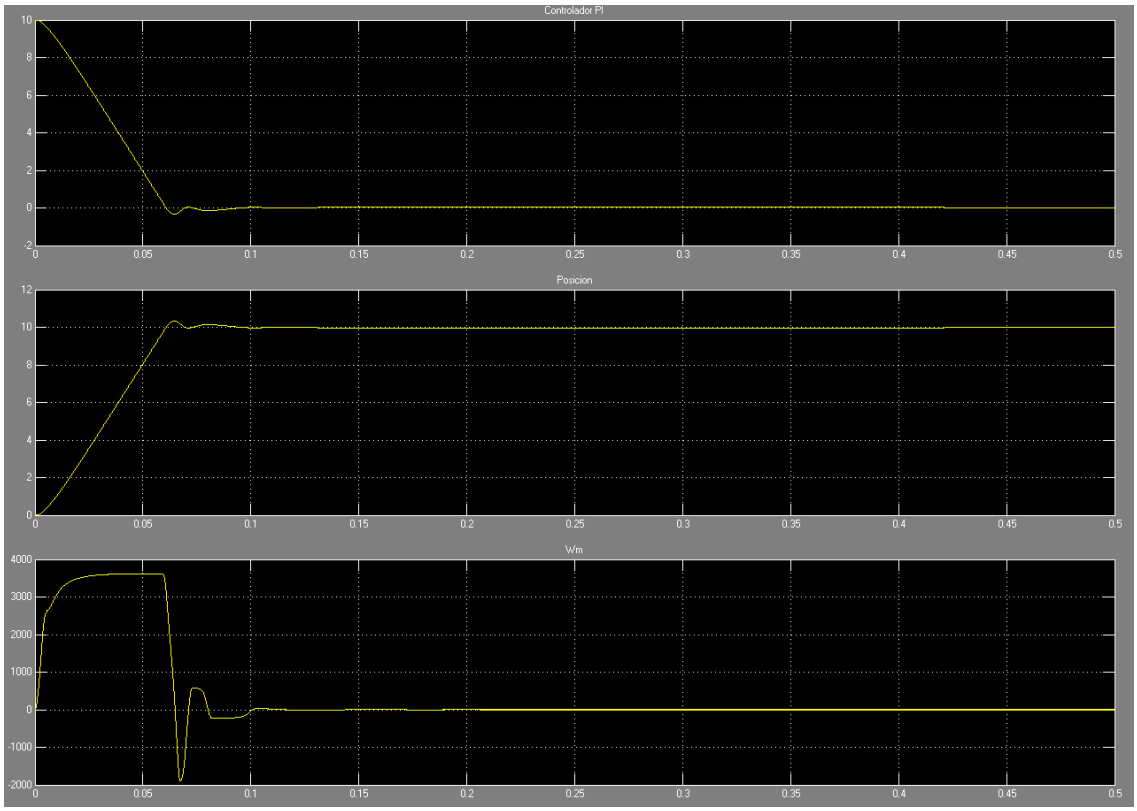


Kd = 0.1 y Kp =10



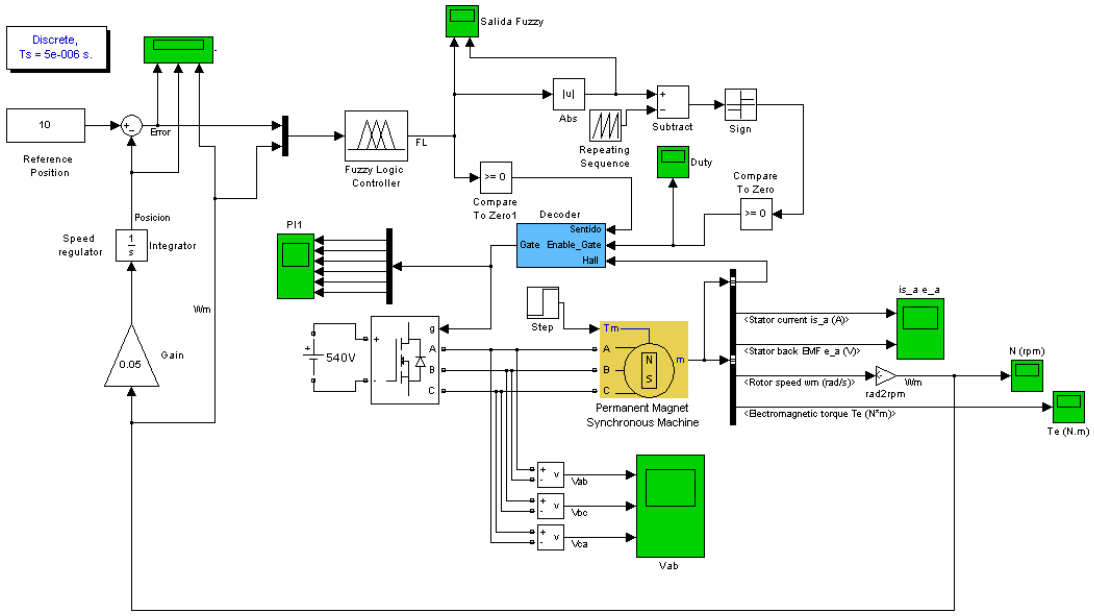
Jugando con diferentes valores, entre ellos con el integral para reducir el error estacionario llegamos a estos valores y a estas respuestas:

Kp = 50
Ki = 10
Kd = 0.1



SIMULACIÓN CONTROL DE POSICIÓN - LAZO CON LOGICA DIFUSA:

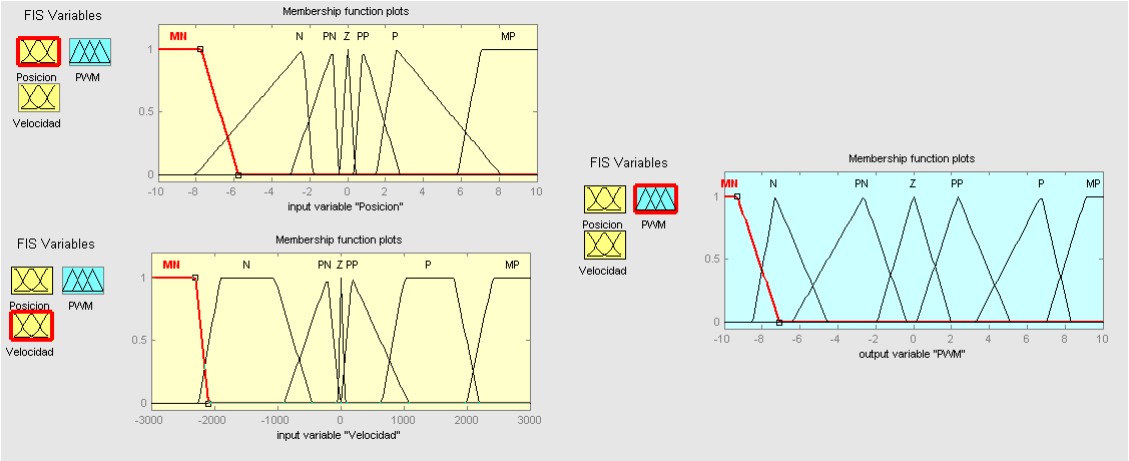
También se planteó resolver el control de posición utilizando lógica difusa como medio de control. Las reglas que se plantearon utilizan las variables de velocidad y posición como medio de control. El diagrama en bloques del control es:



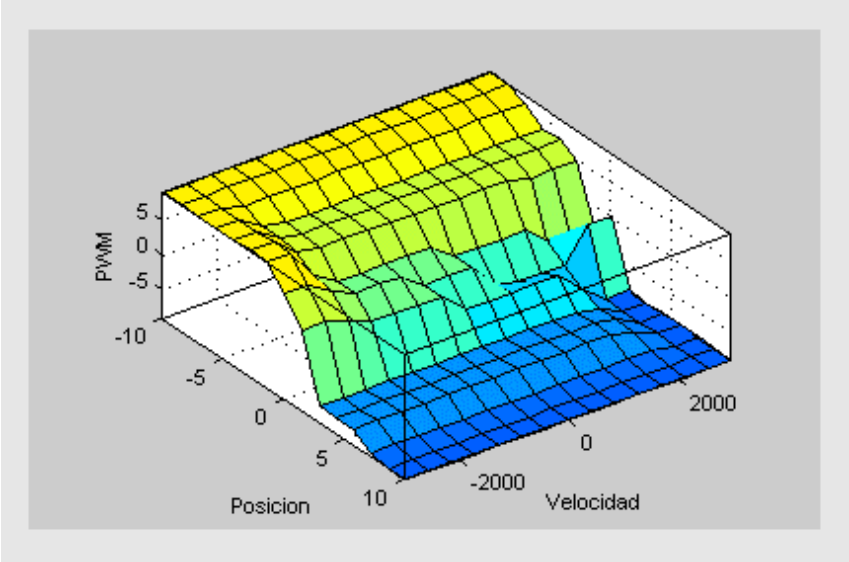
A continuación mostraremos la tabla con las reglas utilizadas:

	MN	N	PN	Z	PP	P	MP	Posición
MN	MP	MP	MP	P	PP	N	MN	
N	MP	P	PP	PP	Z	N	MN	
PN	MP	P	PP	Z	Z	N	MN	
Z	MP	P	PP	Z	PN	N	MN	
PP	MP	P	Z	Z	PN	N	MN	
P	MP	P	Z	PN	PN	N	MN	
MP	MP	P	PN	N	MN	MN	MN	
Velocidad								

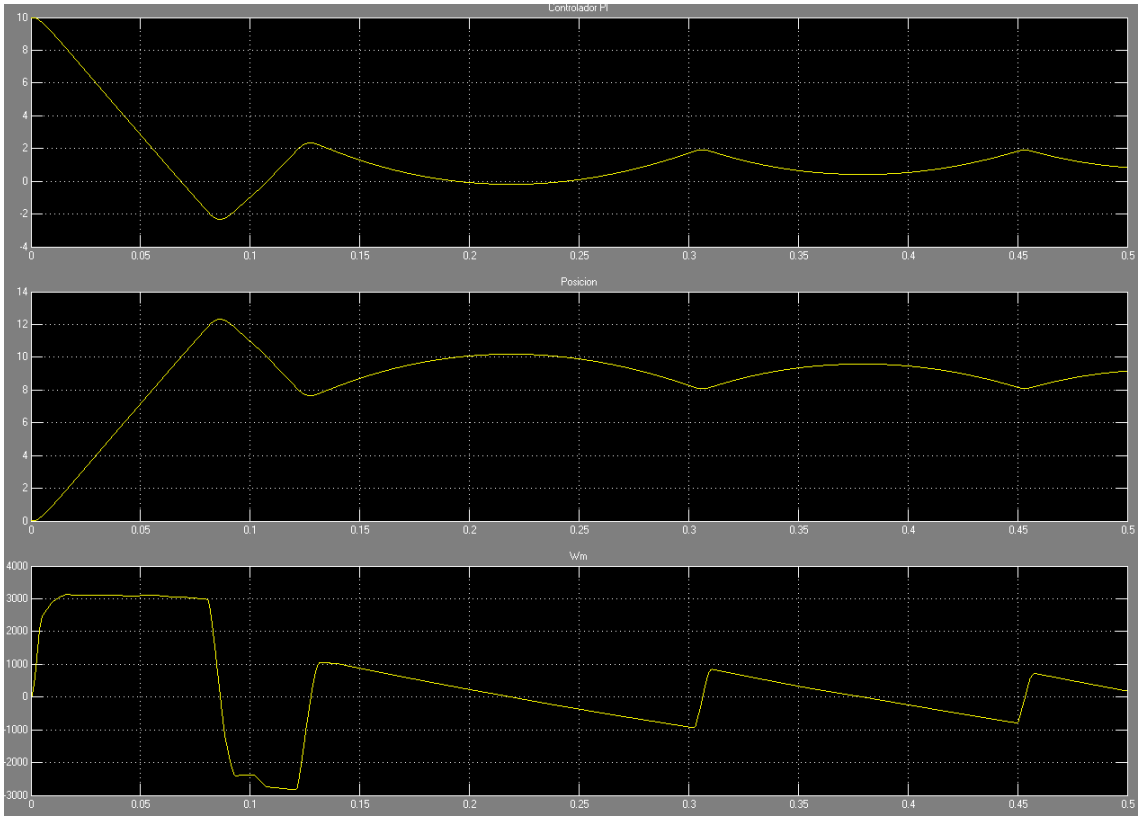
MN	Muy negativa
N	Negativa
PN	Poco negativa
Z	Cero
PP	Poco positiva
P	Positiva
MP	Muy positiva



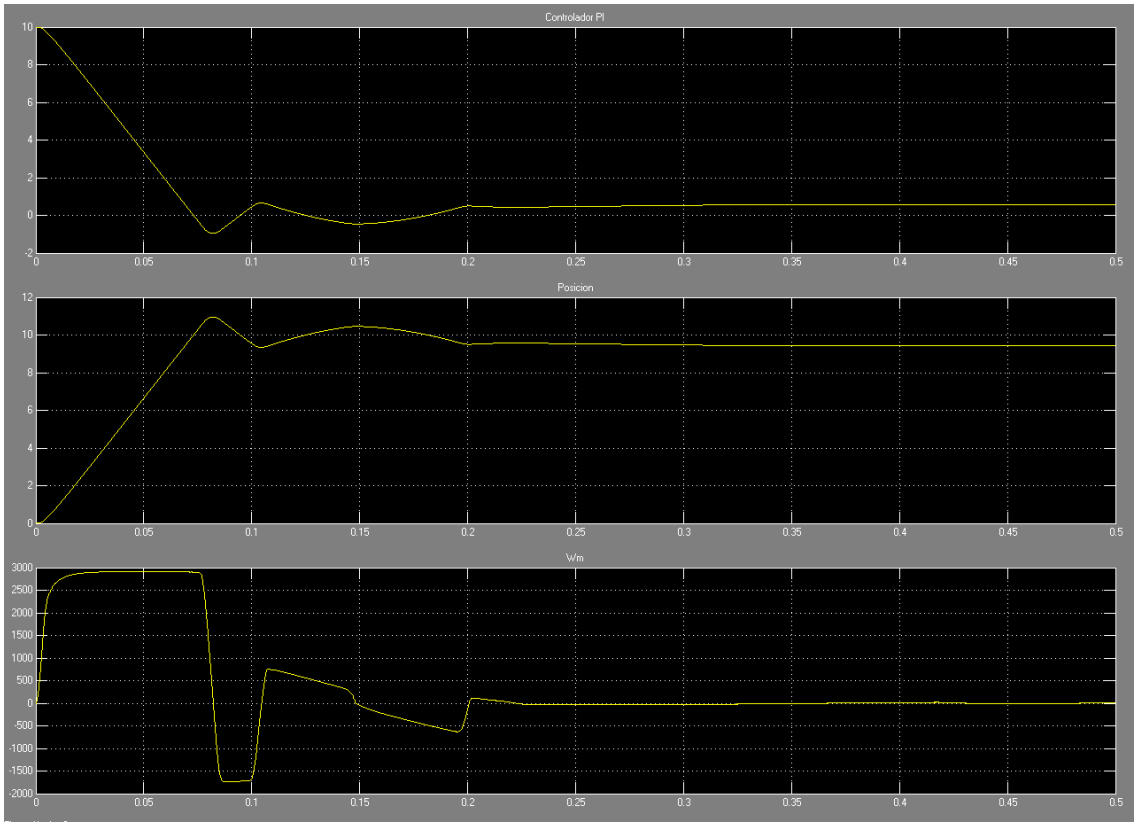
La superficie generada por estas reglas es:



En este caso aplicamos un escalón para evaluar la respuesta del sistema, este nos arrojo los siguientes resultados:



Podemos observar una respuesta del sistema bastante pobre, jugando un poco con las reglas pudimos cambiar la respuesta de este:



Aunque temporalmente tenemos una mejor respuesta, en este caso podemos ver que no llega en ningún momento a la posición deseada. Para lograr un mejor resultado es necesario ajustar mejor las reglas y el peso de cada una de ellas.

ESTUDIO DINÁMICO: ONE-LEGGED ROBOT

A continuación trabajaremos sobre el modelo dinámico de un robot que simula los movimientos de una pierna. Para esto partiremos de un trabajo realizado denominado “Study on One-legged robot jumping”.

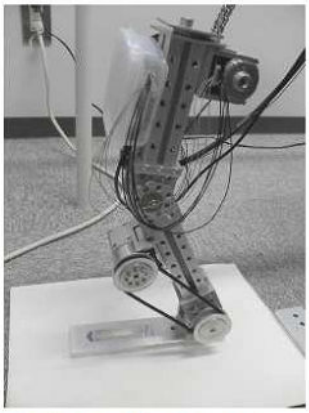


Figure 1: Test One-legged Robot.

Este paper nos entrega las ecuaciones dinámicas del sistema y algunos valores de las propiedades físicas del robot. Nosotros lo que haremos será realizar un modelo 3D en SolidWork a partir de alguna las dimensiones proporcionadas en el paper, luego como material para las piezas se utilizará aluminio y a partir de estos valores crearemos un nuevo conjunto de propiedades fisicas. Luego se realizaran diferentes simulaciones de comportamiento del robot y se dimensionará los motores a utilizar.

MODELIZACION DEL ROBOT:

Nuestro modelo del robot se basará en algunas de las dimensiones proporcionadas en el paper:

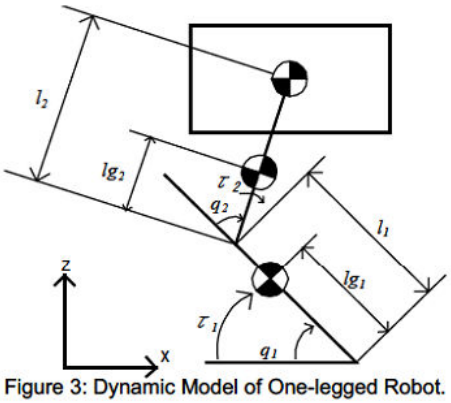
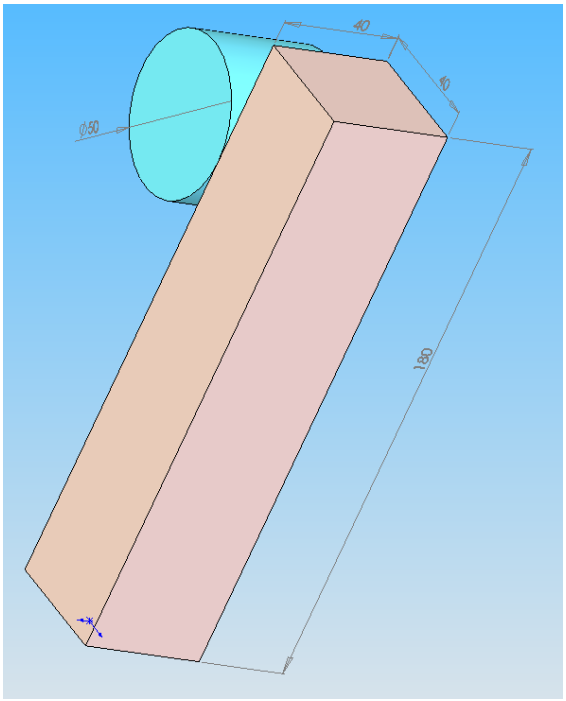
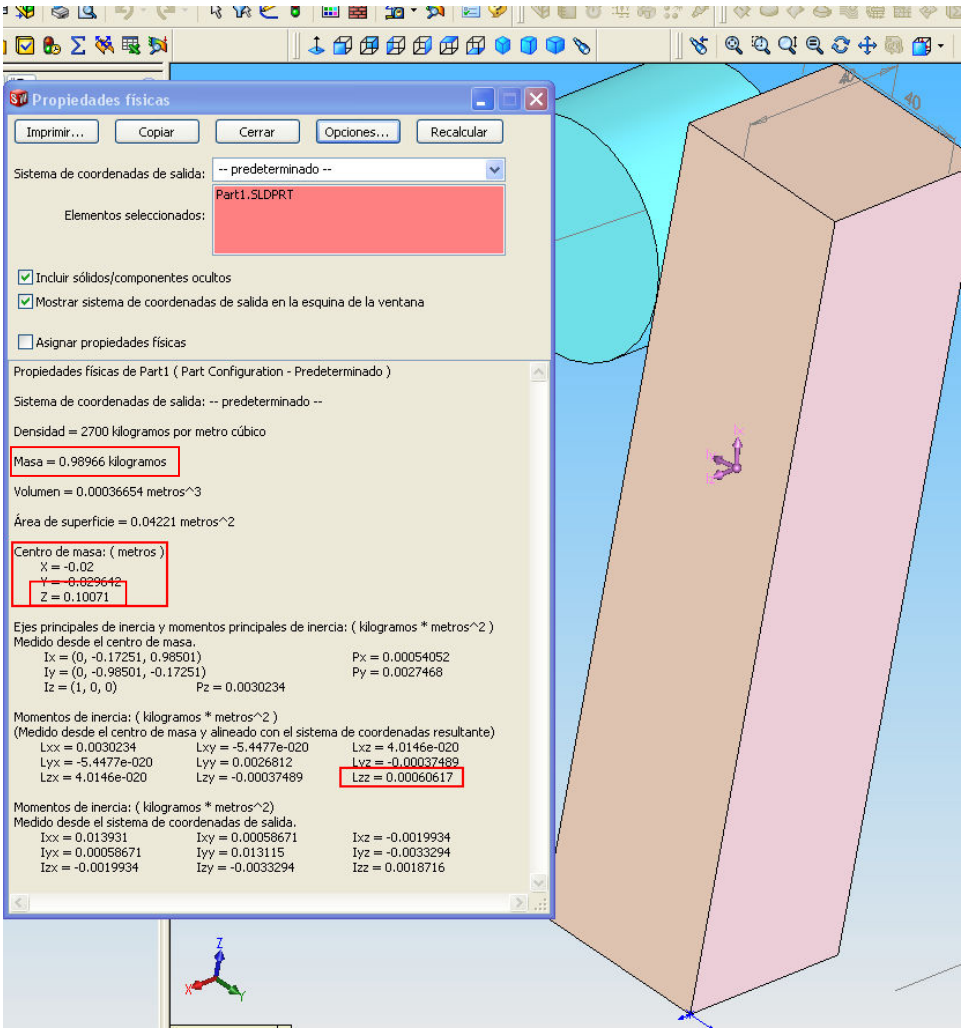


Figure 3: Dynamic Model of One-legged Robot.

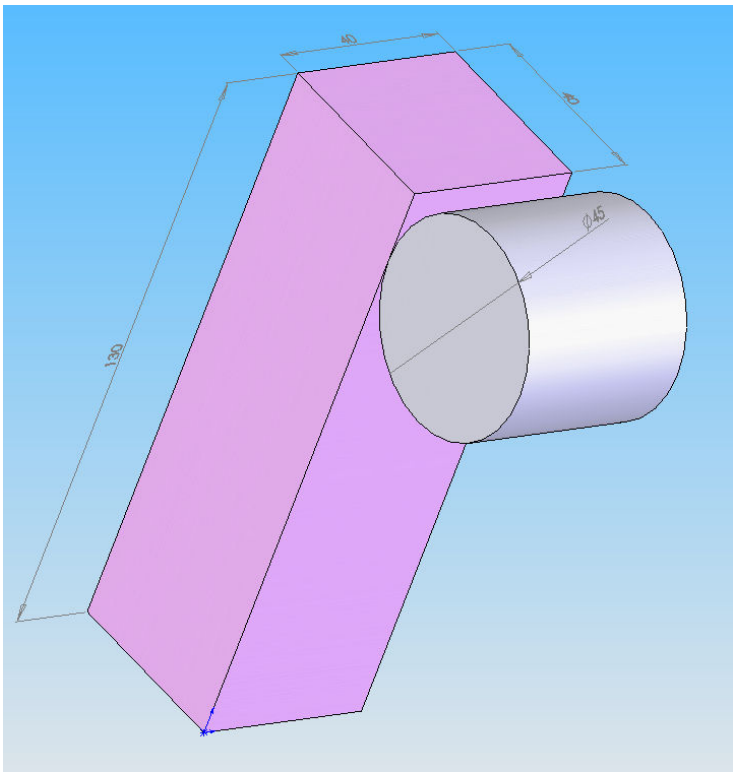
Mass(kg)	1.336(kg)
Width	0.18(m)
Depth	0.04(m)
Height	0.36(m)
l_0 (Body link)	0.18(m)
l_1 (2nd thigh length)	0.18(m)
l_2 (thigh length)	0.13(m)
m_0 (body mass)	0.235(kg)
m_1 (2nd thigh mass)	0.465(kg)
m_2 (thigh mass)	0.450(kg)
M_p (frame mass)	0.185(kg)
I_1	2nd thigh inertia
I_2	thigh inertia
lg_1	gravity point of 2nd thigh
lg_2	gravity point of thigh

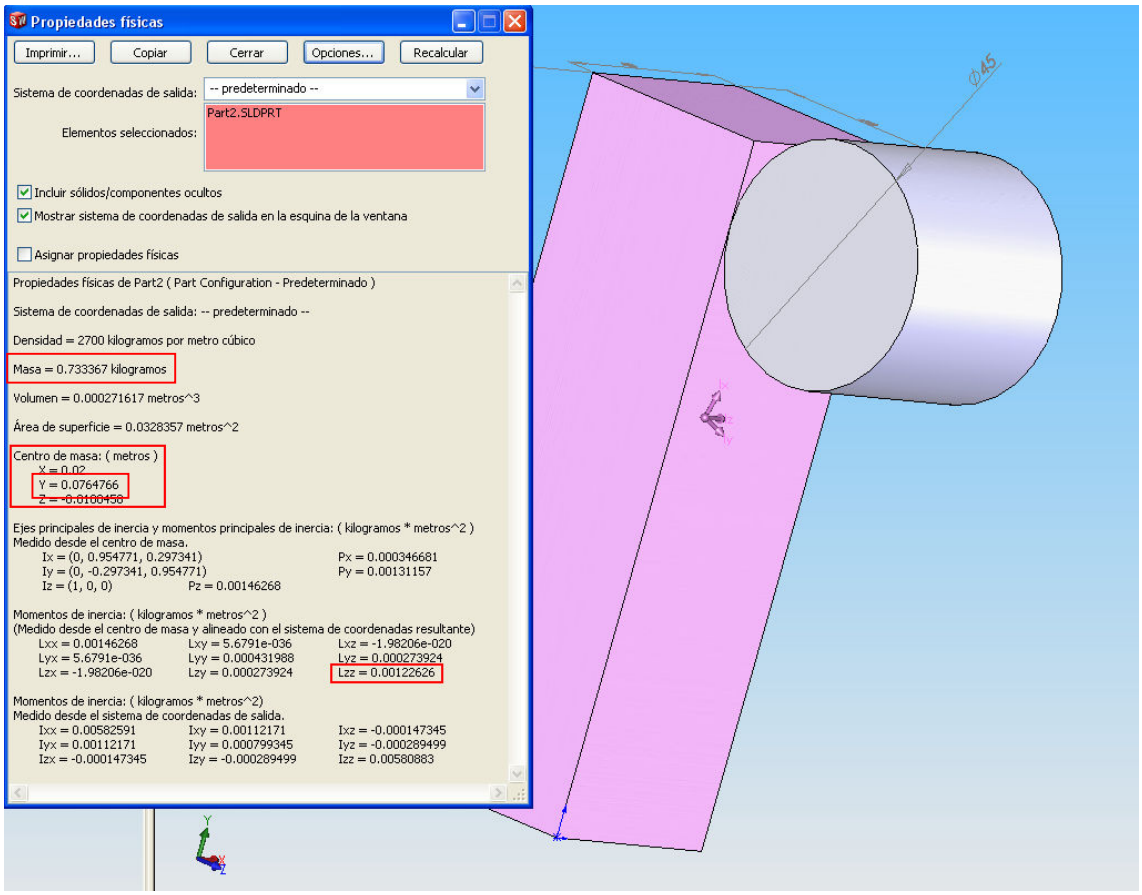
PRIMERA EXTREMIDAD:





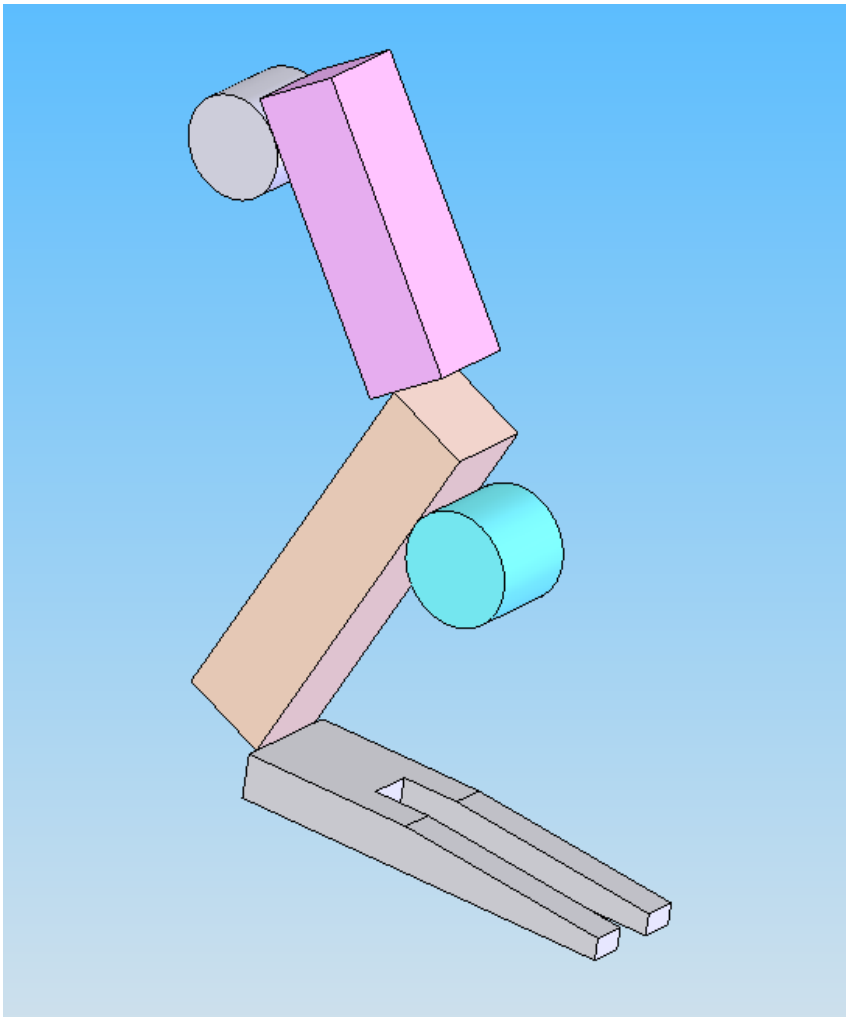
SEGUNDA EXTREMIDAD:





En los gráficos donde se ven los valores de las propiedades físicas de cada cuerpo se marcó los resultados que nos interesarán para la simulación de los movimientos, estos son la ubicación del centro de masa y el momento de inercia sobre el eje que nos interesa, es decir, el eje sobre el cual debe girar. Como material se utilizo para todo el cuerpo una aleación de aluminio. Para una modelizacíon más rigurosa es necesario mejorar como se modelizó la distribución de masa en la pieza, para eso es necesario disponer de un plano completo de cada pieza y asignarle a cada una de estas el material correcto

ENSAMBLADO COMPLETO DEL ROBOT:



SIMULACIÓN DEL ROBOT:

A partir del modelo alcanzado anteriormente pasaremos a realizar diferentes simulaciones y de esta manera encontrar que requerimientos debe cumplir el motor a utilizar.

Simbolo	Valor
l1	0,18 [m]
l2	0,13 [m]
m1	0,986[kg]
m2	0,733 [kg]
lg1 (eje Z)	0,1 [m]
lg2 (eje Y)	0,076 [m]
I1	0,000606 [kg.m^2]
I2	0,00122 [kg.m^2]

Para esto se utilizo un script hecho en Matlab en el cual se implementaron las ecuaciones provistas en el paper y los valores físicos simulados en SolidWork. Este script arroja como resultado el par que debe generar el motor según el angulo de las 2 articulaciones bajo una determinada velocidad y aceleracion angular.

```
%Variables fisicas del robot
syms l1;
syms l2;
syms m1;
syms m2;
syms lg1;
syms lg2;
syms I1;
syms I2;

%Declaracion de variables varias
syms Q1;
syms Q2;
syms S1;
syms S2;
syms S12;
syms C1;
syms C2;
syms C12;

g=9.83;
l1 = 0.18;
l2 = 0.13;
m1 = 0.986;
m2 = 0.733;
lg1=0.1;
lg2=0.076;
I1=0.000606;
I2=0.00122;
Mp = 0.185;

dQ1= 2*pi;  %1rad/seg -> 1 vuelta por segundo
```

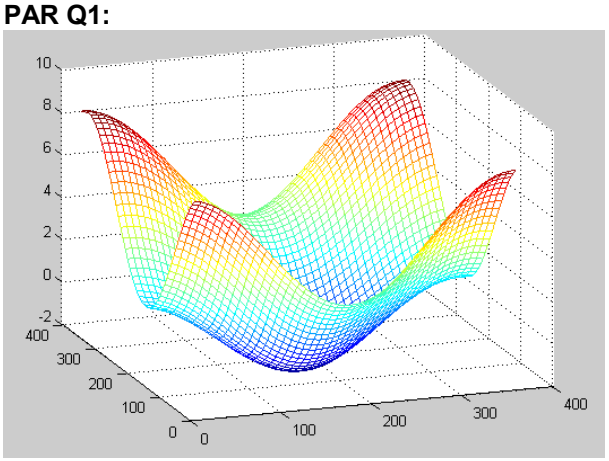
```
dQ2= 2*pi;  %1rad/seg -> 1 vuelta por segundo

d2Q1= 2*pi/0.1;  %aceleracion angular: en 100ms alcanza la velocidad
max
d2Q2= 2*pi/0.1;  %aceleracion angular: en 100ms alcanza la velocidad
max%

S1 = sin(Q1);
S2 = sin(Q2);
S12 = sin(Q1+Q2);

C1 = cos(Q1);
C2 = cos(Q2);
C12 = cos(Q1+Q2);

M=[m1*lg1^2+I1+m2*(l1^2+lg2^2+2*l1*l2*C2)+I2   m2*(lg2^2+l1*lg2*C2)+I2
;
   m2*(lg2^2+l1*lg2*C2)+I2                     m2*lg2^2+I2
];


C=[-2*m2*l1*lg2*S2*dQ1*dQ2-m2*l1*lg2*S2*dQ2^2  ;
    m2*(lg2^2+l1*lg2*dQ1*dQ2)                 ];

G=[m1*g*lg1*C1+m2*g*(l1*C1+lg2*C12)  ;
    m2*g*lg2*C12                   ];

J=[-l1*S1-l2*S2    -l2*S12  ;
    l1*C1+l2*C12   l2*C12 ];

T= M*[d2Q1;d2Q2]+C+G;

tamano = 50;
T1 = zeros(tamano,tamano);
T2 = zeros(tamano,tamano);
escalaq1 = zeros(tamano,tamano);
escalaq2 = zeros(tamano,tamano);

for N1 = 1:1:tamano
    for N2 = 1:1:tamano
        Q1 = 2*pi/tamano*N1;
        Q2 = 2*pi/tamano*N2;
        escalaq1(N1,N2)= 360/tamano*N1;
        escalaq2(N1,N2)= 360/tamano*N2;
        T1(N1,N2)= eval(T(1));
        T2(N1,N2)= eval(T(2));
        plot(T1);
    end
end
```

CONDICION 1:

```
Velocidad angular Q1 = 1rad/seg          ->1 vuelta por segundo
Velocidad angular Q2 = 1rad/seg
Aceleración angular Q1 = 10 rad/seg^2  ->en 100ms alcanza la velocidad maxima
Aceleración angular Q2 = 10 rad/seg^2
```

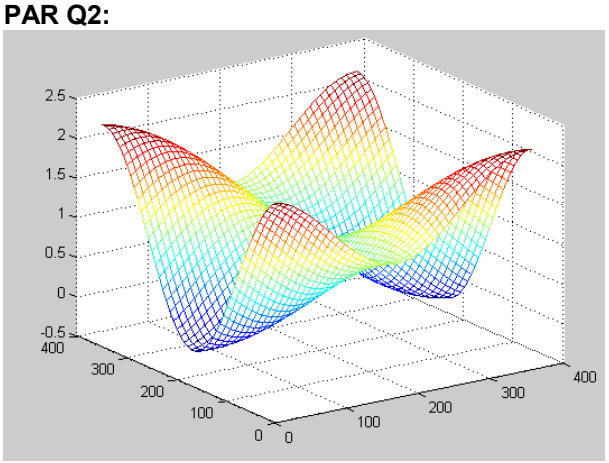
```
dQ1= 2*pi;
dQ2= 2*pi;
d2Q1= 2*pi/0.1;
d2Q2= 2*pi/0.1;
```

PAR Q1:



PAR MAXIMO: 8 N.m

PAR Q2:



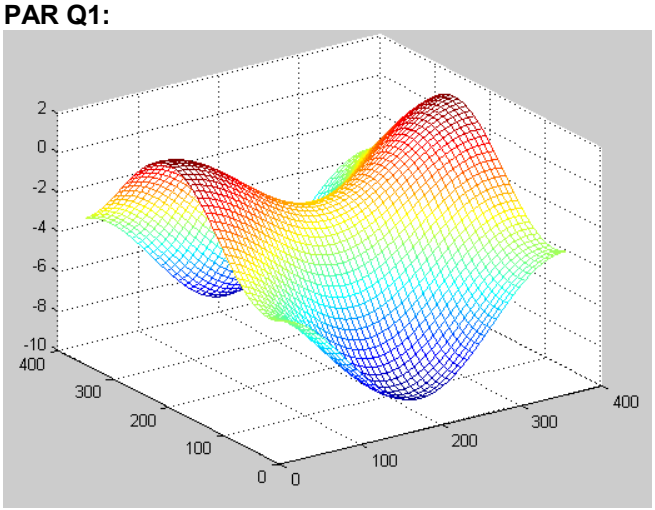
PAR MAXIMO: 2,3 N.m

CONDICION 2:

```
Velocidad angular Q1 = 1rad/seg          ->1 vuelta por segundo
Velocidad angular Q2 = 1rad/seg
Aceleración angular Q1 = -10 rad/seg^2  ->en 200ms alcanza la velocidad maxima
Aceleración angular Q2 = -10 rad/seg^2    contraria
```

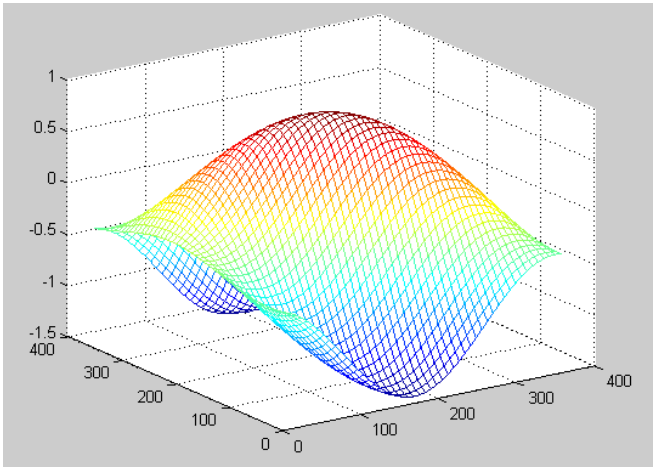
```
dQ1= 2*pi;
dQ2= 2*pi;
d2Q1= 2*pi/0.1;
d2Q2= 2*pi/0.1;
```

PAR Q1:



PAR MAXIMO: 8,5 N.m

PAR Q2:



PAR MAXIMO: 1,5 N.m

ELECCIÓN MOTOR:

Como la mayoria de los motores son de una mayor velocidad de la que se necesita en esta aplicación, se debera utilizar una reduccion donde disminuye la relacion de velocidades pero a la vez aumenta la de torque.

A partir de las simulaciones hechas establecemos que necesitamos motores con las siguientes características:

MOTOR 1:

w = 1rad/s = 60 R.P.M.
T = 9 N.m
P = T.w = 9 W

MOTOR 2:

w = 1rad/s = 60 R.P.M.
T = 2.5 N.m
P = T.w = 2.5W

La elección de los motores a utilizar se hará a partir de la potencia anteriormente mencionada, los requerimientos de velocidad y par se lograran con reducciones.

MOTOR 1:

Φ37-20211-12 BLDC Overview

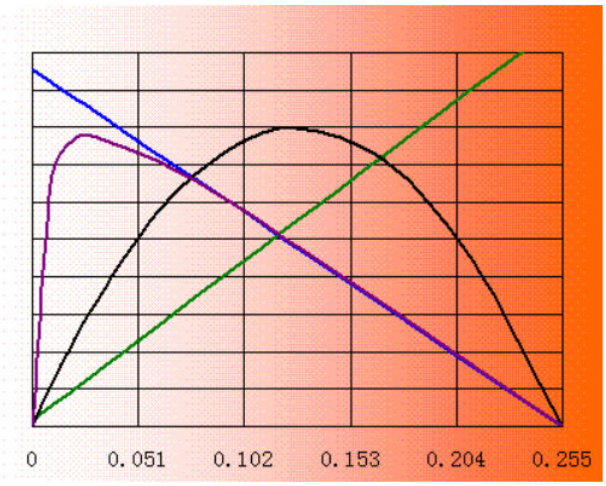
- Three Phase, Six Step, Full Wave, Y-Circuit
- Sintered Nd-Fe-B Permanent Magnet Rotor
- Hall Sensor / Sensorless
- Step (Cogging)
- Slot

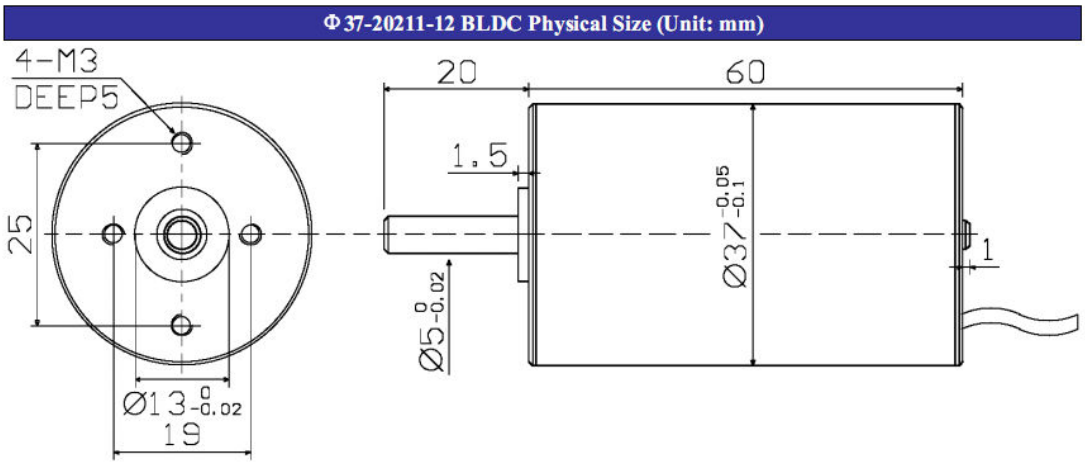
Parameters	Φ37-20211-12 BLDC Absolute Maximum Ratings	Unit
Continuous Current	5.5	A
Speed	10000	rpm
Winding and Rotor Temperature	-20 to +150	℃
Ambient Temperature	-20 to +85	℃

Notice: The Absolute Maximum Ratings are those values beyond which the safety of the device cannot be guaranteed

Parameters	Φ37-20211-12 BLDC Intrinsic Characteristics (20℃)		Unit
Resistance (Including 0.5m Line)	0.71		Ohm
Speed-Torque Gradient	18700		rpm/Nm
Torque Constant	0.024		Nm/A
Speed Constant	398		rpm/V
Back-EMF Constant	2.5		mV/rpm
Rotor Magnetic Poles	4		Poles
Ball Bearing No Load Continuous Life (At Nominal Voltage)	30000	(Please order if have special requirement)	Hours
Weight (Including 0.5m Line)	Approximate	250	g

Parameters	Φ37-20211-12 BLDC Performance Characteristics (20℃)					Unit
Nominal Voltage	12					V
Maximum Output Power	32					W
(See Curves Below)	No Load Point	Some Loaded Points Performance				
Output Torque	0	0.03	0.04	0.05	0.06	Nm
Output Speed	4770	4210	4020	3840	3650	rpm
Input Current	0.17	1.42	1.84	2.26	2.68	A
Output Power	0	13	17	20	23	W
Efficiency	0	78	76	74	71	%
Free-convection Cooling	If the shell temperature of the motor is higher than 85℃, fan or other cooling equipments must be installed. Otherwise the motor may be damaged by hotness.					





El motor principal se sobredimensiono debido a que como se verá mas adelante un motor de menor potencia tiene aproximadamente el mismo tamaño y peso.

MOTOR 2:

Φ38-20136-12 Brushless DC Motor Product Datasheet

Φ38-20136-12 BLDC Overview

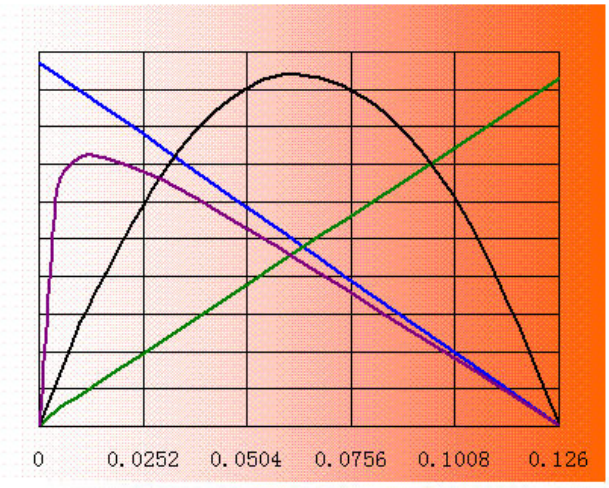
- Three Phase, Six Step, Full Wave, Y-Circuit
- Sintered Nd-Fe-B Permanent Magnet Rotor
- Hall Sensor / Sensorless
- Stepless (Very Low Cogging)
- Slot

Parameters	Φ38-20136-12 BLDC Absolute Maximum Ratings	Unit
Continuous Current	2.1	A
Speed	5000	rpm
Winding and Rotor Temperature	-20 to +150	℃
Ambient Temperature	-20 to +85	℃

Notice: The Absolute Maximum Ratings are those values beyond which the safety of the device cannot be guaranteed

Parameters	Φ38-20136-12 BLDC Intrinsic Characteristics (20 ℃)		Unit
Resistance (Including 0.5m Line)	2.75		Ohm
Speed-Torque Gradient	26970		rpm/Nm
Torque Constant	0.0305		Nm/A
Speed Constant	284		rpm/V
Back-EMF Constant	3.48		mV/rpm
Rotor Magnetic Poles	2		Poles
Ball Bearing No Load Continuous Life (At Nominal Voltage)	30000	(Please order if have special requirement)	Hours
Weight (Including 0.5m Line)	Approximate	255	g

Parameters	Φ38-20136-12 BLDC Performance Characteristics (20 ℃)					Unit
Nominal Voltage	12					V
Maximum Output Power	11.3					W
(See Curves Below)	No Load Point	Some Loaded Points Performance				
Output Torque	0	0.01	0.02	0.03	0.04	Nm
Output Speed	3410	3140	2870	2600	2330	rpm
Input Current	0.05	0.38	0.71	1.04	1.36	A
Output Power	0	3.3	6.0	8.2	9.8	W
Efficiency	0	72	71	66	60	%
Free-convection Cooling	If the shell temperature of the motor is higher than 85℃, fan or other cooling equipments must be installed. Otherwise the motor may be damaged by hotness.					



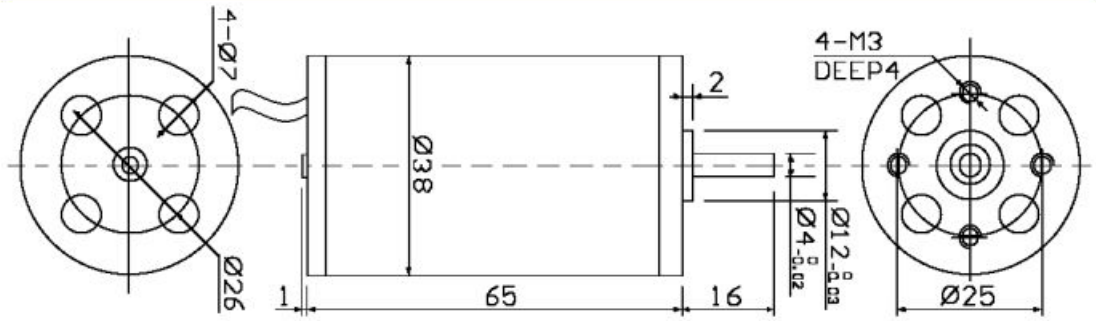
η(%)	P(W)	I(A)	N(rpm)
100	12	4.5	3500

Speed vs. Torque: ----------
Current vs. Torque: ----------
Output Power vs. Torque: ----------
Efficiency vs. Torque: ----------

Short Term Operation Range:

Torque(Nm)

Φ38-20136-12 BLDC Physical Size (Unit: mm)



CONCLUSIONES:

En el desarrollo del trabajo logramos implementar un control de velocidad para un motor brushless en VHDL y luego simular el comportamiento del bloque en Simulink para integrar un sistema de control de posición.

El uso de un FPGA como core lógico de alimentación al motor introduce una importante ventaja dado que al ser un procesamiento por hardware el tiempo de respuesta es mínimo comparado con un microcontolador, donde la lógica es administrada por software.

Luego de desarrollar el módulo en FPGA utilizamos un sistema similar pero implementado en Matlab para simular el comportamiento de un motor brushless y controlar su posición mediante dos controladores, uno PID y otro Difuso, lo que nos dio la oportunidad de integrar y comparar dos metodologías de trabajo aprendidas en Sistemas de Control e Inteligencia Artificial. Pudimos ver que el sistema de control con un PID tiene una respuesta temporal y estacionaria óptima, pero el control difuso permite ser optimizado de forma mas intuitiva y resulta más robusto frente a cambios en los parámetros propios del motor o distintas condiciones de carga.

Por último tomamos como base el paper llamado “Study on One-legged Robot Jumping” y realizamos una primera aproximación a la modelación física de dicho robot para así obtener todos los parámetros descriptos en el paper y así poder simular su comportamiento. Fueron graficados los torques de los motores en cuestión bajo distintas condiciones de contorno, para luego proceder a la elección de motores comerciales que cumplan los requerimientos impuestos por el modelo.

Podemos ver que esta práctica extiende el alcance de la práctica 1, dando una comprensión más completa sobre los manipuladores y brindándonos más herramientas para llevar a la práctica los modelos estudiados.