Práctica No. 3.

Aplicaciones de circuitos combinacionales y secuenciales.

Combinational and secuential circuits applications.

Cree en tí y todo será posible.

Unidad Temática: II

Duración: 6.0 hrs (4 clases).
(Revisión 1er. y 2da. hora combinacionales, 3ra. y 4ta. hora secuenciales)

Unidad de Competencia Específica:
Utiliza los diferentes niveles de diseño para aplicaciones de los circuitos combinacionales y secuenciales en base a la tarjeta de desarrollo con el FPGA.

Competencia Específica:
Implementa aplicaciones de circuitos combinatorios y secuenciales descritos con HDL en una tarjeta de desarrollo con FPGA utilizando Gate Level y RTL. (i.e. switch, led, bicolor, RGB, 7-seg, LCD, encoder, motor PAP, servo con y sin tope, teclado PS2, etapas de pot, TLD, salida de mensaje de audio, JYSTK).

Contenido:
En esta práctica se implementarán los siguientes circuitos:

- Unidad Aritmético-Lógica (A+B, A-B, AxB, A/B, not A, A and B, A or B y A xor B, con A y B de 4 bits o más) con salida a led bicolor o RGB, salida a leds, salida a un display y selector de la I/O o dos displays, además de que por lo menos una de las entradas es un encoder.
- Control manual de motor a pasos unipolar con encoder mecánico rotatorio y encoder magnético u óptico con terminales de control (dirección/hold) y sensores de límite.
- Contador de pulso en alto en ms con salida a display hasta 9999 al presionar un botón, con beep y led RGB indicando el rango, mensaje de termino de conteo y reset.
- Control de velocidad y sentido de giro de un servomotor con encoder rotatorio.
- Control de posición de un servomotor (0°-180°) con encoder rotatorio.
- Codificador de teclado PS2 o USB a leds y display 7seg o LCD.
- Control de posición de 2 cámaras de 2GDL utilizando teclado.
- Reto 1: Implementar un medidor con SRF-05 de 0-400cm a 7-seg y alarma de 0 a 30cm.
- Reto 2: Implementar el control de posición de cámara con dos servos y jotstick.
- Reto 3: Implementar el control de posición de 2 cámaras dedos grados de libertad cada una con dos joystick.

Content:
In this practice, you will implement the next circuits:

- *Arithmetic-Logical Unit (A+B, A-B, AxB, A/B, not A, A and B, A or B and A xor B, with A and B of 4 bits or more) with bicolor or RGB led output, output to LEDs, output to display and selector of the I/O or two displays, at least one of the inputs is an encoder.*
- *Manual control of unipolar stepper motor with rotary mechanical encoder and magnetic or optical encoder with control terminals (direction/hold) and limit sensors.*
- *High pulse counter in ms with display output up to 9999 when pressing a button with beep sound output and RGB LED indicating the range, reset and end message.*
- *Servo motor speed/direction control with rotary encoder.*
- *Servo motor position control (0°-180°) with rotary encoder.*
- *PS2 or USB keyboard encoder to LEDs and 7seg or LCD display.*
- *Two cameras position controlof2DOF with keyboard.*
- *Challenge 1. Implement US dist. Meter (SRF-05) 0-400cm, 7-seg, 0 to 30cm an alarm.*
- *Challenge 2. Implement a 2DOF camera positioner, 2 servos and joystick.*
- *Challenge 3. Implement 2 cameras position control of 2DOF each one using 2 joysticks.*

Pre-reporte tres.
Previous work three.

Realizar los siguientes puntos como trabajo previo para el buen progreso de la práctica [**máximo 20 cuartillas** si se entrega impreso (Letra Times New Roman de 12ptos, interlineado sencillo)]:

1. Investigar los protocolos de comunicación abarcados en la práctica, así como diseñar las interfaces y etapas de potencia necesarias para conectar los componentes a la tarjeta de desarrollo.

2. Escribir los códigos de HDL (VHDL y Verilog) para las siguientes aplicaciones de los circuitos combinacionales y secuenciales utilizando top level design:

a) Unidad aritmético-lógica ALU para realizar las operaciones aritméticas de suma, resta, multiplicación y división (A+B, A-B, AxB, A/B) y las operaciones lógicas de negación, Y, O y O exclusiva (not A, A and B, A or B, A xor B), siendo las entradas A y B de por lo menos 4 bits, con salida C de 4 a 8 bits, empleando un selector "sel" de 3 bits funcionando de acuerdo a la tabla 3.1 y la figura 3.1. Por lo menos una de las entradas es un encoder rotatorio. Colocar un led bicolor o RGB el cual indique con un color **Rojo** que se está realizando una operación aritmética y con un color **Azul** una operación lógica.

Tabla 3.1. Relación de operaciones de la ALU

Operaciones	Sel	Operación	Observaciones @ A y B de 4 bits
Aritméticas Salida a leds y a display. Led RGB **Rojo**.	**0**00	C=A+B	C es de 5 bits (máx 30)
	001	C=A-B	C es de 4 bits (de -15 a 15)
	010	C=AxB	C es de 8 bits (máx 225)
	011	C=A/B	C es de 4 bits (solo enteros de 1 a 15)
Lógicas Salida a leds. Led RGB **Azul**.	**1**00	C=not(A)	C es de 4 bits
	101	C=A and B	C es de 4 bits
	110	C=A or B	C es de 4 bits
	111	C=A xor B	C es de 4 bits

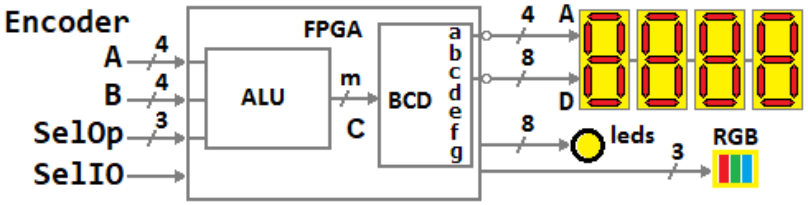

Figura 3.1. Diagrama a bloques de la ALU y su conexión.

b) Control de motor a pasos unipolar con encoder mecánico rotatorio y encoder magnético u óptico, tal que al girar cualquiera de los encoder, el motor girará en la misma dirección (horario o antihorario) además de encender un led RGB que indique la dirección. Incluir sensores de límite para proteger el giro de la carga mecánica con un indicador audible ("beep") que indique ciertas posiciones. Es posible colocar un selector para direccionar la señal de cada encoder. Observe la figura 3.2.

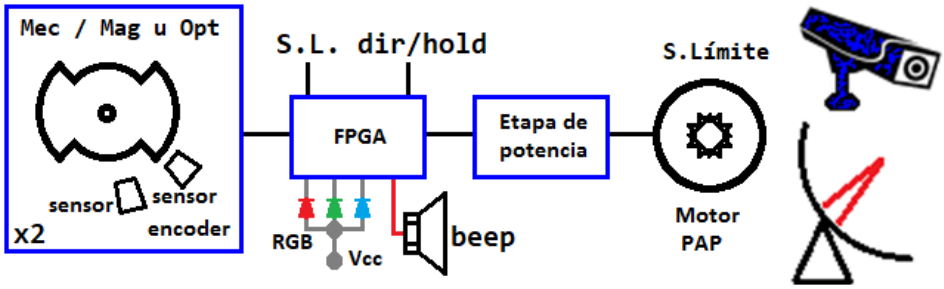

Figura 3.2. Diagrama a bloques del control del motor a pasos con encoder.

c) Contador de pulso en alto en ms, el cual al presionar el botón (push) el display empezará a contar las unidades en ms, cuando se suelta se detiene en un valor, pero si se presiona otra vez continua el conteo. Se cuenta con un botón de reset para reiniciar en cero en cualquier momento que se presione. El display puede contar hasta un máximo de 9,999 ms (9.999s) y cuando se alcance este valor se detendrá y sonará un mensaje audible a 1m *"límite alcanzado"*. Por medio de un led RGB se indica en que rango se encuentra el conteo ($0 < R \leq 3{,}333$, $3{,}333 < G \leq 6{,}666$, $6{,}666 < B \leq 9{,}999$) y al cambiar de rango un sonará un "BEEP" de cambio. Ver figura 3.3.

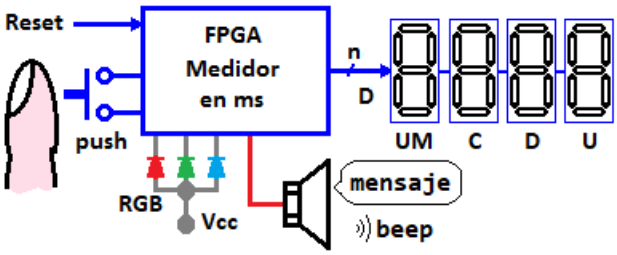

Figura 3.3. Esquema para el medidor de ms.

Consultar el boletín UPIITA número 63 (www.boletin.upiita.ipn.mx).

d) Control de velocidad (y sentido de giro) de un servomotor con pwm utilizando un encoder rotatorio. El movimiento del encoder en un sentido permitirá que el motor incremente su velocidad, a medida que se le dan más vueltas en el mismo sentido al encoder seguirá aumentando, pero si se dan vueltas en sentido contrario la velocidad disminuirá, ya sea hasta detenerse o bien para cambiar el sentido de giro e ir aumentando su velocidad en sentido contrario. La señal de control para el servo es una señal modulada en ancho de pulso (PWM), adecuada al tipo de motor adquirido. En el display se indica el número de la velocidad asignada, siendo 5 la máxima velocidad en sentido horario, -5 en antihorario y 0 para nula. Un diagrama a bloques se muestra en la figura 3.4.

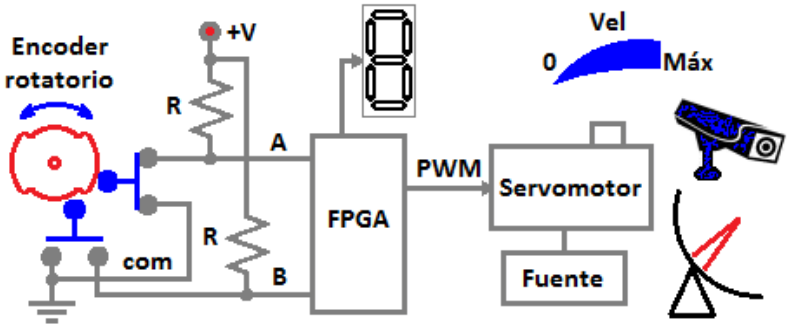

Figura 3.4. Diagrama a bloques para el control de velocidad del servo por PWM.

e) Control de posición de un servomotor con pwm utilizando un encoder rotatorio. El movimiento del encoder en un sentido cambiará la posición del servo en ese mismo sentido, tanto horario como antihorario. La señal de control para el servo es una señal modulada en ancho de pulso (PWM), adecuada al tipo de motor adquirido. En los displays se indica el ángulo en el que está el motor, cuyo rango es de 0° a 180° o ajustado al tipo de motor que se tiene, con una resolución mínima de 5°. Un diagrama a bloques se muestra en la figura 3.5.

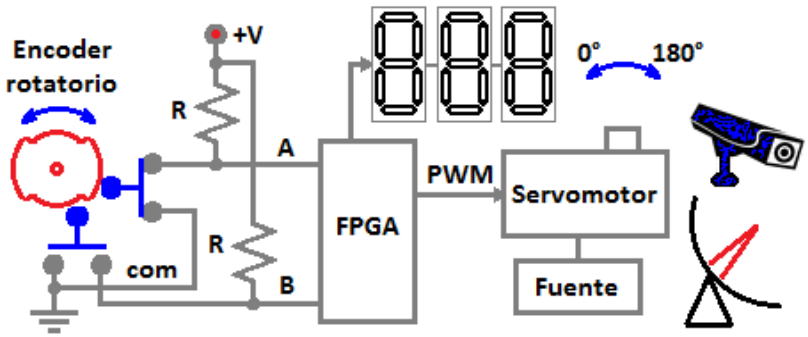

Figura 3.5. Diagrama a bloques para el control de posición del servo por PWM.

f) Codificador de teclado PS2 a leds y display 7seg o LCD (mínimo 30 teclas (números, letras, símbolos y funciones)). Cuando se presiona el teclado, se muestran los caracteres en el display y en código binario en los leds. Nota: es posible sustituir el teclado por USB, respetando dicho protocolo.

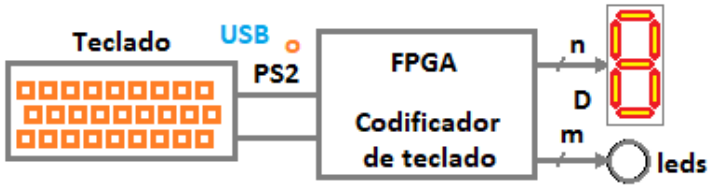

Figura 3.6. Diagrama a bloques para el control de posición del servo por PWM.

g) Control de posición de dos cámaras de seguridad de 2 GDL cada una, en el que las teclas de funciones permiten elegir cualquiera de las cámaras (F1 => cámara 1, F2 => cámara 2); una vez elegida la cámara con las flechas se permite controlar los movimientos arriba-abajo-izquierda-derecha.

NOTA 1: Todos los motores llevan alguna **carga** mecánica en el eje (banda, llantas, polea, sistema animatrónico, ventana o puerta a escala, gripper, grua, mecanismos, etc.), no se permiten cosas sencillas (hélices, clips, alambres, ligas, hojas de papel, etc.).

NOTA 2: En caso de no entregar el pre-reporte y/o de no traer el material (con los circuitos armados), los alumnos NO TENDRÁN derecho a entrar al laboratorio. Los circuitos deben de armarse de acuerdo con la indicación del archivo de restricciones. Coloque las preguntas con su numeración en los pre-reportes y reportes.

"No darse por vencido y ayudarse mutuamente en su aprendizaje es una característica de los alumnos UPIITA-IPN".

Material y Equipo.
Equipment and components.

Tarjeta de & software de desarrollo (Nexys/ISE-Vivado, DE2/Quartus II, etc.).
Componentes: resistencias 330 Ω, 120 Ω a ½ W, leds de diferentes colores (rojo, verde, amarillo, naranja, ámbar, azul, etc.), RGB, interruptores SPST, botón push NC o NO), etapas

de potencia para los motores, bocina o buzer, encoder rotatorio mecánico, óptico o magnético o sensores para hacerlos, grabador/reproductor de audio, motor a pasos unipolar y servomotor con tope y sin tope, display (7-seg, LCD, etc.), joystick analógico o digital, teclado con conexión a PS2 o USB. Mecanismos y cargas para motores.
Circuito de reloj (oscilador) externo o generador de señales con conectores.
Varios: Pinzas de punta y de corte, protoboard necesarios para los circuitos externos, cables para conexión con conectores (header macho, header hembra, caimán, BNC, etc.), cinta de aislar o termofit, fuente de alimentación regulada para alimentar los motores de forma externa, punta lógica o analizador de estados lógicos, multímetro, osciloscopio.

Nota: las cantidades de componentes depende de los diseños que se realicen, además de corroborar los componentes que se tengan en una tarjeta de desarrollo.

Introducción.
Introduction.

Encoder

Un encoder rotatorio mecánico consta de por lo menos dos interruptores que se cierran en diferentes posiciones cuando su eje gira. La secuencia generada en los interruptores A y B es: 00→10→11→01→00, lo que se conoce como código Gray. En la figura 3.7 se muestran los encoder y su diagrama interno. Es posible cambiar un poco el mecanismo y los sensores para tener un encoder rotatorio óptico o magnético, empleando optointerruptores y sensores de efecto hall junto con imanes, respectivamente.

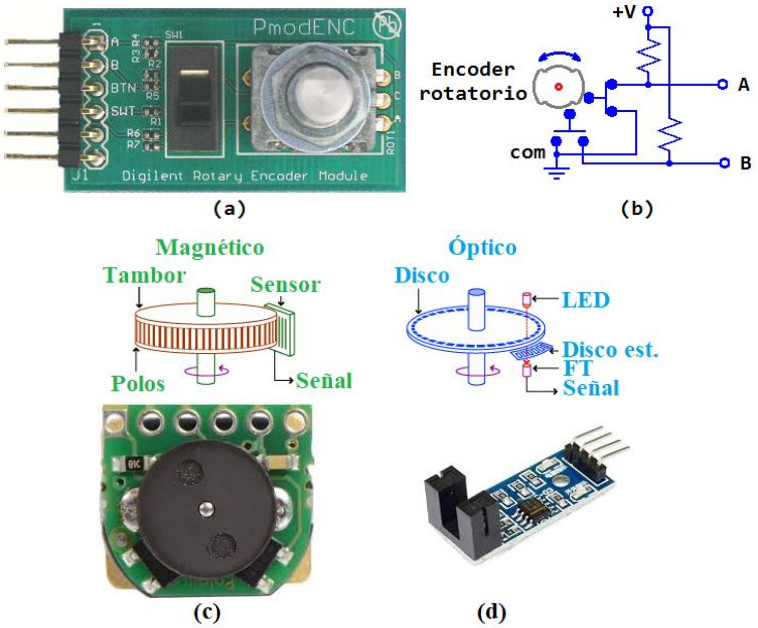

Figura 3.7. Foto del encoder mecánico (a), circuito eléctrico equivalente (b), encoder magnético (c) y encoder óptico (d).

PWM

La modulación de ancho de pulso también conocida como PWM permite cambiar el voltaje promedio de una señal, tal como se puede ver en la figura 3.8. Dicha señal es muy utilizada para el control de posición en un servomotor con tope o control de velocidad y sentido de giro en un servomotor de movimiento contínuo o sin tope, lo que se aprecia en la figura 3.9.

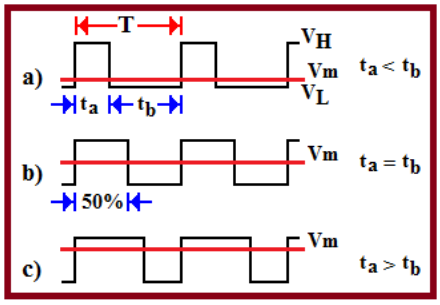

Figura 3.8. Voltaje medio de una señal. (a) Vm bajo, (b) Vm medio y (c) Vm alto.

Dicha señal es muy utilizada para el control de posición en un servomotor con tope o control de velocidad y sentido de giro en un servomotor de movimiento contínuo o sin tope, lo que se aprecia en la figura 3.9.

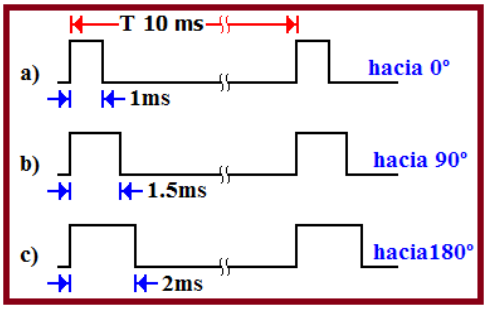

Figura 3.9. PWM que se controla la posición de un servomotor de forma genérica con una señal de T=10ms ó f=100Hz y señal en alto de 1ms a 2ms. (a) va a 0° con 1ms, (b) va a 90° con 1.5ms y (c) va a 180° con 2ms.

En seguida se presenta un ejemplo escrito en VHDL que al girar el eje del encoder en cualquier sentido, cambia la posición de un servomotor con tope, debido a que se modifica el PWM conectado a la señal de control. El cambio en el PWM se observa en un led testigo cambiando su intensidad y también se visualizan los cambios de posición en 5 leds de forma binario y en un display de 7 segmentos las 32 posiciones (0 a F y 0. a F.), a parte de los movimientos del servo.

Las características del servomotor FUTABA S3003 con el que se diseñó el programa son:

Ángulo	Tiempo en alto High
0°	0.3ms
180°	2.3ms

Tomando en cuenta que se tiene un reloj de 50MHz que da un periodo T=20ns, se calculan las veces que cabe el periodo en el tiempo en alto y en el incremento, quedando de la siguiente forma:

Ángulo	Tiempo en alto [ms] **high**	Constantes	Observaciones
0°	0.3ms (@cnt=0)	**high**/20ns=15000	Se llamará "**min**"
180°	2.3ms (@cnt=31)		
Incremento = (180°-0°)/31	Incremento = (2.3-0.3=2ms)/31 = 64.516us 64.516us/20ns = 3225.8	incremento=3225	Se llamará "**inc**"

El programa presenta la implementación de una función bajo el criterio de que se parte de un valor inicial (15000@cnt=0) en el conteo de "cnt" y los valores subsecuentes se incrementan en aproximadamente 6666 para 16 posiciones y 3225 para 32 posiciones, se puede modelar el valor del conteo del PWM que se comparará con en valor de cntPWM de la siguiente forma:
high <= 15000 + (cnt * 3225) para las 32 posiciones
high <= min + ((cnt)*(inc)) para el código;

cuyo resultado en la señal de PWM para el servo se muestra en la siguiente figura 3.

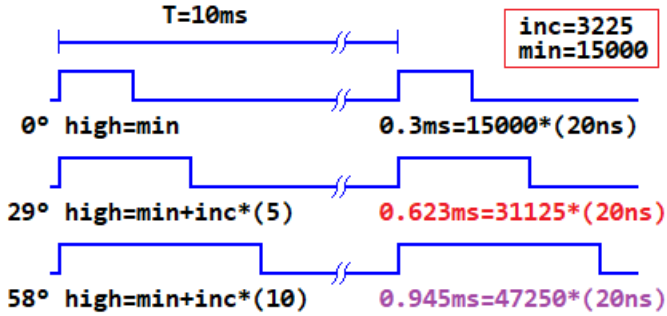

Figura 3.10. Tres ejemplos de la señal de PWM generadas por el código en VHDL.

```
----------------------------------------------------------------------------------------------------
-- Se presenta un generador de señal PWM para controlar un servomotor, utilizando un
-- encoder rotatorio mecánico para manipular su posición. El sentido de giro es CW y
-- CCW, tanto para el encoder como para el servomotor, teniendo un tope en ambos
-- sentidos. El tiempo en alto "high" se obtiene con la función Ta=K1+(conta*K2) o bien
-- high <= min + (cnt*inc), colocada como high = 15000 + (cnt * 3225).
-- Archivos:  encoder.vhd y encoder.ucf
----------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity encoder is
    Generic (    msb : integer := 5; --número de bits del contador
                 min : integer := 18332; --valor mínimo del contador para el tiempo en alto
```

```
                max : integer := 121686; --valor máximo del contador para el tiempo en alto
                inc : integer := 3334; -- incremento para el tiempo en alto
                N : integer := 15 ); --divisor
    Port (
        --reloj de 50MHz de la nexys2
                clk : in  STD_LOGIC;
        --reset asíncrono en alto en la nexys2 (BTN0) y en el encoder (Push)
                resetB,resetP: in STD_LOGIC;
        --señales A y B del encoder
                A,B : in  STD_LOGIC;
        --salida del contador de 5 bits cuando msb=5
                contador : out  STD_LOGIC_VECTOR (msb-1 downto 0);
        --salida a display de 7 segmentos + P
                display : out STD_LOGIC_VECTOR (7 downto 0):="00000001";
        --salida a los ánodos
                AN : out STD_LOGIC_VECTOR (0 to 3);
        --salidas de pwm para el servomotor y un led testigo
                servomotor,servoLED : out STD_LOGIC
                        );
end encoder;

architecture encoder of encoder is
--se utiliza una FSM moore para leer el encoder
type edos is (EA, EB, EC, ED);
signal EP: edos:=EA;

--declaración de señales
signal clkdiv: std_logic_vector(N downto 0);        --señal para el divisor
signal AB: std_logic_vector (1 to 2);               --señal que une las señales del encoder
signal cntPWM: integer range 1 to 500000 := 1;      --contador de 10ms @ clk=50MHz
                                                    -- o T=20ns
signal cnt: integer range 0 to 31:= 0;              --contador de 0 a 31
signal servo: std_logic;                            --señal de PWM para las salidas servos
signal high: integer range min to max := min;       --duración del tiempo en alto de la
                                                    -- señal PWM
------------ end signal declarations ------------
begin

AB <= a & b;            --unión (concatenación) de las señales del encoder
AN <= "1110";           --activación del ánodo del display uno

divisor:        --proceso del divisor
        process(clk,resetB,resetP)
        begin
                if    resetB = '1' or resetP = '1' then clkdiv <= (others => '0');
                elsif rising_edge(clk) then clkdiv <= clkdiv + 1;
                end if;
```

```
        end process divisor;

FSM:    --proceso que detecta los giros del encoder y genera la variable cnt
        process (clkdiv(N),resetB,resetP,cnt)
        begin
                if resetB = '1' or resetP = '1' then EP <= EA; cnt<=0;
                elsif rising_edge(clkdiv(N)) then
                        case EP is
                                when EA =>
                                 if    AB = "00" then EP <= EA; cnt<=cnt;  --hold
                                 elsif AB = "10" then EP <= EB;            --cw
                                                if    cnt = 31 then cnt <= 31;
                                                elsif cnt < 31 then cnt <= cnt + 1;
                                                else                cnt <= cnt;
                                                end if;
                                 elsif AB = "01" then EP <= ED;            --ccw
                                    if    cnt = 0 then cnt <= 0;
                                                elsif cnt > 0 then cnt <= cnt - 1;
                                                else                cnt <= cnt;
                                                end if;
                                 end if;
                                when EB => cnt <= cnt;                    --hold
                                 if    AB = "10" then EP <= EB;
                                 elsif AB = "11" then EP <= EC;
                                 elsif AB = "00" then EP <= EA;
                                 end if;
                                when EC =>
                                 if    AB = "11" then EP <= EC; cnt<=cnt;  --hold
                                 elsif AB = "01" then EP <= ED;            --cw
                                                if    cnt = 31 then cnt <= 31;
                                                elsif cnt < 31 then cnt <= cnt + 1;
                                                else                cnt <= cnt;
                                                end if;
                                 elsif AB = "10" then EP <= EB;            --ccw
                                                if    cnt = 0 then cnt <= 0;
                                                elsif cnt > 0 then cnt <= cnt - 1;
                                                else                cnt <= cnt;
                                                end if;
                                 end if;
                                when ED => cnt <= cnt;                    --hold
                                 if    AB = "01" then EP <= ED;
                                 elsif AB = "00" then EP <= EA;
                                 elsif AB = "11" then EP <= EC;
                                 end if;
                                when others => null;
                                end case;
                end if;
```

```
        end process FSM;

-- Proceso para generar la salida de PWM de 0.3 a 2.3 ms @ f=100Hz, cntPWM cuenta
-- de 1 a 500,000 que equivale a un periodo de 10ms, con 16 posiciones, high va de
-- min @ cnt=0 (0.3ms) hasta max @ cnt=15 (2.3ms).
--
-- Los valores de high para (cnt) con incrementos de 6666 son:
-- 0.3ms      (0),   0.4333ms  (1),   0.5666ms  (2),   0.7ms      (3),
-- 0.8333ms (4),   0.9666ms  (5),   1.1ms      (6),   1.2333ms  (7),
-- 1.3666ms (8),   1.5ms      (9),   1.6333ms  (10),  1.7666ms (11),
-- 1.9ms    (12),  2.0333ms  (13),  2.1666ms  (14),  2.3ms     (15)
--
-- Con 32 posiciones high va de min @ cnt=0 (0.3ms) hasta max @ cnt=31 (2.3ms).
-- Los valores de high para (cnt) con incrementos de 3225 son:
-- 0.3ms (0),       0.365ms (1),   0.429ms (2),   0.494ms (3),
-- 0.558ms (4),   0.623ms (5),   0.687ms (6),   0.752ms (7),
-- 0.816ms (8),   0.881ms (9),   0.945ms (10),  1.01ms (11),
-- 1.074s (12),   1.139ms (13),  1.203ms (14),  1.268ms (15)
-- 1.332ms (16),  1.397ms (17),  1.467ms (18),  1.526ms (19),
-- 1.59ms (20),   1.655ms (21),  1.719ms (22),  1.784ms (23),
-- 1.848ms (24),  1.913ms (25),  1.977ms (26),  2.042ms (27),
-- 2.106ms (28),  2.171ms (29),  2.235ms (30),  2.3ms (31)

ModulPulso:   --proceso que genera el pulso de salida e indica en el display un valor
        process(clk,servo)
        begin
        if rising_edge(clk) then cntPWM <= cntPWM + 1; --contador de 1 a 500,000
        high <= min + ((cnt)*(inc));
          if cntPWM <= high then     servo <= '1';
          else                       servo <= '0';
          end if;
         case cnt is          --orden:   abcdefgP-ánodo común, contador = salida a leds
           when 0   => display <= "00000011"; contador <= '0' & x"0";   -- 0    al display
           when 1   => display <= "10011111"; contador <= '0' & x"1";   -- 1    al display
           when 2   => display <= "00100101"; contador <= '0' & x"2";   -- 2    al display
           when 3   => display <= "00001101"; contador <= '0' & x"3";   -- 3    al display
           when 4   => display <= "10011001"; contador <= '0' & x"4";   -- 4    al display
           when 5   => display <= "01001001"; contador <= '0' & x"5";   -- 5    al display
           when 6   => display <= "01000001"; contador <= '0' & x"6";   -- 6    al display
           when 7   => display <= "00011111"; contador <= '0' & x"7";   -- 7    al display
           when 8   => display <= "00000001"; contador <= '0' & x"8";   -- 8    al display
           when 9   => display <= "00001001"; contador <= '0' & x"9";   -- 9    al display
           when 10 => display <= "00010001"; contador <= '0' & x"A";   -- A    al display
           when 11 => display <= "11000001"; contador <= '0' & x"B";   -- B    al display
           when 12 => display <= "01100011"; contador <= '0' & x"C";   -- C    al display
           when 13 => display <= "10000101"; contador <= '0' & x"D";   -- D    al display
           when 14 => display <= "01100001"; contador <= '0' & x"E";   -- E    al display
```

```
            when 15 => display <= "01110001"; contador <= '0' & x"F";   -- F    al display
            when 16 => display <= "00000010"; contador <= '1' & x"0";   -- 0.   al display
            when 17 => display <= "10011110"; contador <= '1' & x"1";   -- 1.   al display
            when 18 => display <= "00100100"; contador <= '1' & x"2";   -- 2.   al display
            when 19 => display <= "00001100"; contador <= '1' & x"3";   -- 3.   al display
            when 20 => display <= "10011000"; contador <= '1' & x"4";   -- 4.   al display
            when 21 => display <= "01001000"; contador <= '1' & x"5";   -- 5.   al display
            when 22 => display <= "01000000"; contador <= '1' & x"6";   -- 6.   al display
            when 23 => display <= "00011110"; contador <= '1' & x"7";   -- 7.   al display
            when 24 => display <= "00000000"; contador <= '1' & x"8";   -- 8.   al display
            when 25 => display <= "00001000"; contador <= '1' & x"9";   -- 9.   al display
            when 26 => display <= "00010000"; contador <= '1' & x"A";   -- A.   al display
            when 27 => display <= "11000000"; contador <= '1' & x"B";   -- B.   al display
            when 28 => display <= "01100010"; contador <= '1' & x"C";   -- C.   al display
            when 29 => display <= "10000100"; contador <= '1' & x"D";   -- D.   al display
            when 30 => display <= "01100000"; contador <= '1' & x"E";   -- E.   al display
            when 31 => display <= "01110000"; contador <= '1' & x"F";   -- F.   al display
            when others => display <= "11111101"; contador <= '0' & x"0";
          end case;
        end if;

        servomotor <= servo;        --salida de la señal PWM hacia el servomotor
        servoLED <= servo;          --salida de la señal PWM del led testigo

        end process ModulPulso;     -- fin del proceso

end encoder;                        -- fin de la arquitectura
--------------------------------------------------------------------------------------------

##################################################################
#                          UCF para la Nexys 2                   #
##################################################################
# reloj de 50MHz de la nexys2
net "clk"                   loc ="B8" ;    # 50MHz

# reset asíncrono en alto en la nexys2 (BTN0) y en el encoder (Push)
net "resetB"            loc ="B18" ;        # btn0
net "resetP"            loc ="L15" ;        # JA1

# señales A y B del encoder
net "A"                 loc ="M15" ;        # JA4
net "B"                 loc ="L17" ;        # JA3

# salida del contador de 5 bits cuando msb=5
net "contador(0)"       loc ="J14" ;        # LD0
net "contador(1)"       loc ="J15" ;        # LD1
net "contador(2)"       loc ="K15" ;        # LD2
```

```
net "contador(3)"     loc ="K14" ;        # LD3
net "contador(4)"     loc ="E17" ;        # LD4

# salida a display de 7 segmentos + P
net "display(7)"      loc ="L18" ;        # A
net "display(6)"      loc ="F18" ;        # B
net "display(5)"      loc ="D17" ;        # C
net "display(4)"      loc ="D16" ;        # D
net "display(3)"      loc ="G14" ;        # E
net "display(2)"      loc ="J17" ;        # F
net "display(1)"      loc ="H14" ;        # G
net "display(0)"      loc ="C17" ;        # P

# salida a los ánodos
net "AN(0)"           loc ="F17" ;   # AN0
net "AN(1)"           loc ="H17" ;   # AN1
net "AN(2)"           loc ="C18" ;   # AN2
net "AN(3)"           loc ="F15" ;   # AN3

# salidas de pwm para el servomotor y un led testigo
net "servomotor"      loc ="P18" ;   # JD4
net "servoLED"        loc ="R4" ;    # LD7

#Nexys2 Pmod Connector Pin Assignments
#Pmod JA              Pmod JB              Pmod JC              Pmod JD
#JA1:L15  JA7:K13     JB1:M13  JB7:P17     JC1:G15  JC7:H15     JD1:J13  JD7:K14
#JA2:K12  JA8:L16     JB2:R18  JB8:R16     JC2:J16  JC8:F14     JD2:M18  JD8:K15
#JA3:L17  JA9:M14     JB3:R15  JB9:T18     JC3:G13  JC9:G16     JD3:N18  JD9:J15
#JA4:M15  JA10:M16    JB4:T17  JB10:U18    JC4:H16  JC10:J12    JD4:P18  JD10:J14
```

Esta información se puede consultar en el boletín de la UPIITA números 29 y 34 (www.boletin.upiita.ipn.mx).

En la figura 3.11 se muestra el diagrama de conexión, y en la 3.12 fotos del funcionamiento del código presentado para 3 de las 32 posiciones (3, 2. y F.).

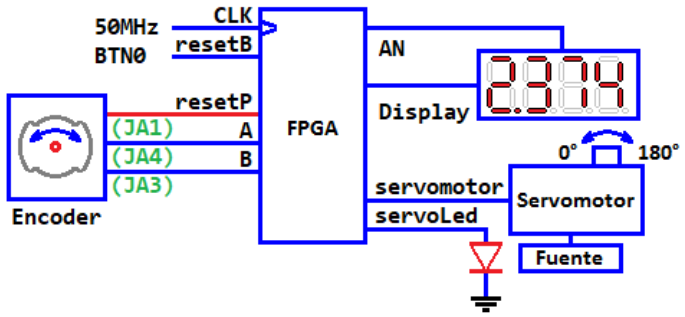

Figura 3.11. Diagrama de conexión para el control de posición de un servomotor.

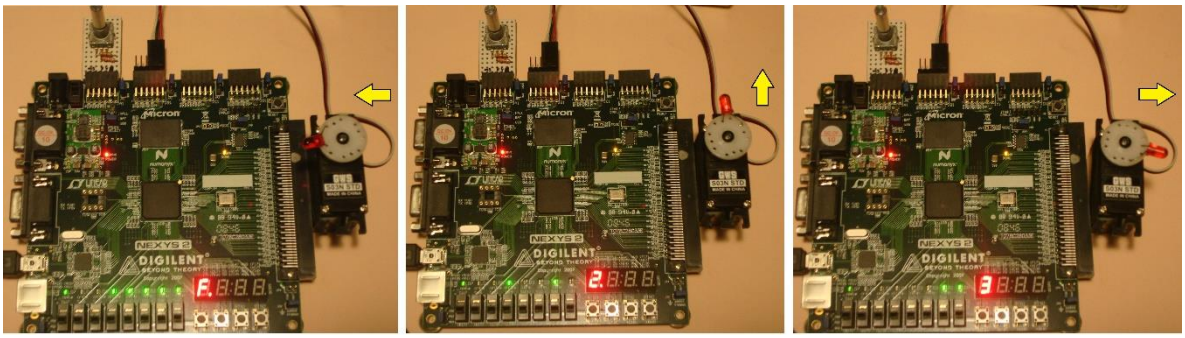

Figura 3.12. Posiciones del servo con encoder mecánico rotatorio.

Teclado y PS2

Un teclado comercial PS/2 de 104 teclas se encarga del rastreo de teclas, del sistema anti-rebotes y de la transmisión de datos utilizando una trama perfectamente establecida. El teclado PS/2 tiene un mapa de teclas a las que asigna un código, de uno o dos bytes, para cada una de ellas y que son los códigos que nos va a transmitir para indicarnos que se está pulsando una tecla determinada. A estos códigos se les llama códigos de rastreo de teclado, mostrados en la figura 3.13.

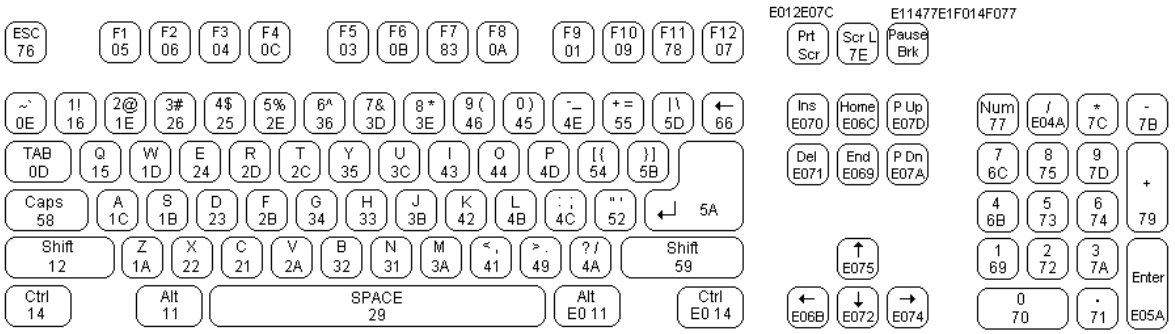

Figura 3.13. Teclado con los códigos de rastreo.

El teclado PS/2 envía el código de rastreo asociado a una tecla al ser esta pulsada, tantas veces como sea necesario si se mantiene pulsada con una cadencia tal como indique su tiempo de repetición, que es programable, y el mismo código de rastreo con el prefijo del byte F0h al ser soltada, también conocido como Break Code.

Por ejemplo: Para conseguir la letra G Mayúscula debemos pulsar la tecla Shift y mientras la mantenemos pulsada, pulsar la tecla G, soltar la tecla G y soltar la tecla Shift. Esa secuencia de pulsaciones nos va a hacer que el teclado transmita la siguiente secuencia de bytes:

Pulsar Shift -> 12h, (scan code de la tecla shift, ver figura 3.13)
Pulsar "G" -> 34h,
Soltar "G" -> F0h 34h
y soltar Shift -> F0h 12h

o escribiendo solo los bytes que vamos a recibir:

12h, 34h, F0h, 34h, F0h, 12h

En la siguiente tabla 3.2 se muestran las letras del abecedario y junto con el código de rastreo (scan code) y su dato en ASCII.

Tabla 3.2. Abecedario y código de rastreo (scan code)

Character	Scan Code (hex)	ASCII Character (hex)
A	1C	41
B	32	42
C	21	43
D	23	44
E	24	45
F	2B	46
G	34	47
H	33	48
I	43	49
J	3B	4ª
K	42	4B
L	4B	4C
M	3A	4D
N	31	4E
O	44	4F
P	4D	50
Q	15	51
R	2D	52
S	1B	53
T	2C	54
U	3C	55
V	2A	56
W	1D	57
X	22	58
Y	35	59
Z	1A	5ª

Un teclado PS2 también permite comandos, que por el momento no se presentarán en este documento y cuya información está disponible en libros y en la red.

Conexionado eléctrico:

El teclado PS/2 se conecta mediante cuatro hilos. Dos de ellos son para alimentación Vcc a 5V y GND, y otros dos para las señales Data y Clock. El pinout de los conectores Mini-DIN PS/2 tanto hembra (en la PC), como machos (en el teclado) se muestran en la figura 3.14 y en la tabla 3.3.

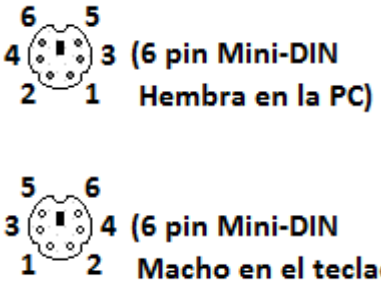

Figura 3.14. Conexionado del PS2.

Tabla 3.3. Asignación de pines y su descripción.

Pin	Nombre	Dirección	Descripción
1	DATA	↔	Pin de dato serial
2	N/C		No conectado
3	GND	−	Tierra
4	VCC	→	Alimentación
5	CLK	→	Reloj
6	N/C		No conectado

En el Teclado PS/2 las señales Data y Clock son de "colector abierto". Esto quiere decir que para establecer una comunicación eléctricamente correcta debemos nosotros suministrar voltaje para el nivel lógico alto, y es él el encargado de dar los correspondientes niveles lógicos bajos, tirando nuestra señal a GND cuando así sea necesario. Este tema se soluciona conectando dos resistencias de 10K entre dichas líneas y Vcc por lo que siempre tendrán nivel lógico alto, salvo cuando el teclado disponga lo contrario y lo tire a GND para dar lo correspondientes niveles lógicos bajos. Esto es lo que se llama conectar unas resistencias Pull-Up.

En la figura 3.15 se muestra como conectar las resistencias Pull-Up entre el teclado y el dispositivo programable.

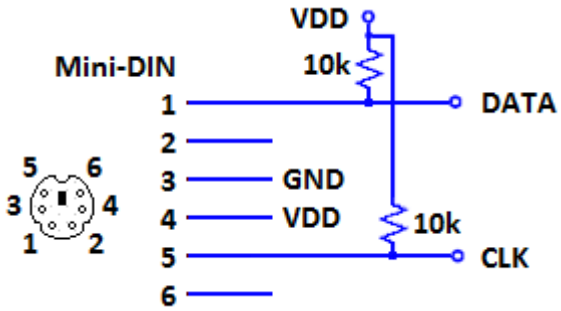

Figura 3.15. Conexión con resistencias de Pull-up.

Protocolo de comunicación PS/2:

El teclado PS/2 se comunica mediante un Protocolo Serie Síncrono, utiliza una señal de Clock que índica cuando están disponibles los correspondientes bits en la señal de Data.

En reposo la señal de Clock está a nivel alto; a cada pulso a nivel bajo corresponde un pulso a nivel alto o bajo en la señal de Data, que se traducen respectivamente como bits 0 ó 1 del dato a transmitir.

La trama completa se compone de 11 bits. Siendo el primero un bit de Start, a continuación, los 8 bits del Dato a transmitir enviándose primero el LSB (ó bit menos significativo), el décimo es el de paridad (usa la Impar, u Odd en inglés) y por último un bit de ACK o Stop,

como puede verse en la figur 3.16 el cronograma de esta trama de comunicación PS/2 Teclado (Keyboard) -> PC (host):

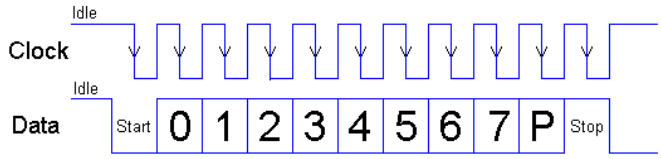

Figura 3.16. Trama de comunicación del teclado al host.

El teclado PS/2 admite también comandos enviados desde el PC con el mismo formato, el protocolo es bidireccional, el cual se indica mediante la señal de Clock al colocarla en nivel bajo durante 160 uS, y la señal de Data a bajo unos 35 uS después de del Clock. Posteriomente se espera la señal del Clock generada por el Teclado. Esto indica que el teclado está dispuesto para recibir nuestro comando. Después de detectar el primer pulso de Clock, a partir del siguiente se comienza a enviar el byte (los 8 bits de comando) con el correspondiente pulso en bajo, iniciando por el LSB, y después el bit de paridad impar (el número de unos en los datos más el de paridad deber ser impar, o sea 1 si el número de unos es par y cero si el total de unos es impar). Por último, hay que esperar el nivel bajo de ACK del teclado, tras dos pulsos de reloj, mostrado en la figura 3.17.

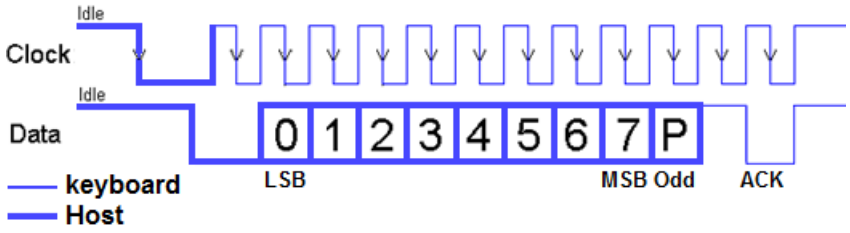

Figura 3.17. Trama de comunicación del host al teclado.

A continuación, se muestran tres códigos para el teclado PS2 escritos en VHDL, que utilizan ciclos for con estructura de registro de corrimiento y con FSM.

```
-------------------------------------------------------------------------------------------
-- Primer código basado en un registro de corrimiento
-------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.all;

Entity Reg_Der_S_P is
    generic (n: in integer:= 33); --Trabaja a 11x3 para asegurar que no haya perdida de datos
    port (CLK_T: in std_logic; --reloj del ps2
            RST : in std_logic; --reset
            R : in std_logic; --dato del ps2
            Q : out std_logic_vector (8 downto 1)); --salida
endReg_Der_S_P;
```

```
Architecture Registro of Reg_Der_S_P is
    signal Qn, Qp: std_logic_vector (n-1 downto 0);
begin

Combinacional: process (Qp,R)
Begin
Qn(n-1) <= R;
    For i in n-2 downto 0 loop
    Qn(i) <= Qp(i+1);
    end loop;
Q <= Qn(8 downto 1);
end process Combinacional;

Secuencial: process (CLK_T,RST)
Begin
if (RST = '1') then Qp<= (others => '0'); -- Reset manda 0 a todas las salidas
elsif (CLK_T'event and CLK_T = '1') then Qp<=Qn; -- se revisa si hay flanco de subida
end if;
end process Secuencial;
end Registro;
-- asignaciones de variables
--Qn(0) <= start
--Qn(8) ->Qn (1) <= Data
--Qn(9) <= Parity
--Qn(10) <= Stop


------------------------------------------------------------------------------------------
-- Segundo código
------------------------------------------------------------------------------------------
-- Código de teclado PS2 implementado con FSM
-- teclado con PS2
-- se toma la lectura al presionar una tecla empleando una FSM
-- http://www.pantechsolutions.net/cpld-fpga-boards/
-- ps-2-interfacing-with-spartan-3e-webserver

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--declaración de la entidad
entity key is
port( ps2data: in std_logic; --dato del PS2
     ps2clk: in std_logic; --reloj del PS2
     leds : out std_logic_vector(7 downto 0); --salida a leds
                D: out std_logic_vector(7 downto 0); --salida a display
```

```
                anodos: out std_logic_vector (1 to 4)); --habilitación de los ánodos
end key;

--declaración de la arquitectura con FSM
architecture keyboard of key is
type state is (state1,state2,state3,state4,state5,state6,state7,state8,state9,state10,state11);
signal EP,EF : state; --EP=estado presente, EF=estado futuro
signal store : std_logic_vector(7 downto 0):="00000000"; --señal para guardar el dato
signal start,parity,stop : std_logic; --bit de inicio, bit de paridad y bit de paro
begin

--proceso para la lectura de los datos del teclado por el PS2
process(ps2clk,ps2data)
begin
--actualización del estado presenteef
        if ps2clk'event and ps2clk = '1' then EP <= EF;
        end if;
--guardado de datos
        if ps2clk'event and ps2clk = '0' then
                if    EP = state1   then start    <= ps2data;     EF <= state2;
                elsif EP = state2  then store(0) <= ps2data;  EF <= state3;
                elsif EP = state3  then store(1) <= ps2data;  EF <= state4;
                elsif EP = state4  then store(2) <= ps2data;  EF <= state5;
                elsif EP = state5  then store(3) <= ps2data;  EF <= state6;
                elsif EP = state6  then store(4) <= ps2data;  EF <= state7;
                elsif EP = state7  then store(5) <= ps2data;  EF <= state8;
                elsif EP = state8  then store(6) <= ps2data;  EF <= state9;
                elsif EP = state9  then store(7) <= ps2data;  EF <= state10;
                elsif EP = state10 then parity   <= ps2data;  EF <= state11;
                elsif EP = state11 then stop     <= ps2data;  EF <= state1;
                end if;
        end if;
end process; --termina de leer los datos del teclado

--proceso para mandar el dato leido a leds
process(store)
begin
        leds <= store;
end process; --termina el envío de datos a leds

--proceso para visualizar el dato en el display de la nexys2
process(store)
begin
anodos <= "1110";
--asigna al display el dato del teclado guardado en la variable store
--                                                abcdefgP
        if      store = X"16"  then    D <= "10011111";     -- 1
```

```
        elsif   store = X"1E"  then   D <= "00100101";    -- 2
        elsif   store = X"26"  then   D <= "00001101";    -- 3
        elsif   store = X"25"  then   D <= "10011001";    -- 4
        elsif   store = X"2E"  then   D <= "01001001";    -- 5
        elsif   store = X"36"  then   D <= "01000001";    -- 6
        elsif   store = X"3D"  then   D <= "00011111";    -- 7
        elsif   store = X"3E"  then   D <= "00000000";    -- 8
        elsif   store = X"46"  then   D <= "00011001";    -- 9
        elsif   store = X"45"  then   D <= "00000011";    -- 0
        elsif   store = X"4E"  then   D <= "10111111";    -- '
        elsif   store = X"55"  then   D <= "10100101";    -- ¿
        elsif   store = X"15"  then   D <= "00000010";    -- q primer linea de letras
        elsif   store = X"1D"  then   D <= "11111101";    -- w
        elsif   store = X"24"  then   D <= "01100001";    -- e
        elsif   store = X"2D"  then   D <= "11110101";    -- r
        elsif   store = X"2C"  then   D <= "11100001";    -- t
        elsif   store = X"35"  then   D <= "10001001";    -- y
        elsif   store = X"3C"  then   D <= "11000111";    -- u
        elsif   store = X"43"  then   D <= "11011111";    -- i
        elsif   store = X"44"  then   D <= "11000101";    -- o
        elsif   store = X"4D"  then   D <= "00110001";    -- p
        elsif   store = X"54"  then   D <= "10111111";    -- ´
        elsif   store = X"5B"  then   D <= "11111101";    -- +
        elsif   store = X"1C"  then   D <= "00010001";    -- a segunda linea de letras
        elsif   store = X"1B"  then   D <= "01001001";    -- s
        elsif   store = X"23"  then   D <= "10000101";    -- d
        elsif   store = X"2B"  then   D <= "01110001";    -- f
        elsif   store = X"34"  then   D <= "00001001";    -- g
        elsif   store = X"33"  then   D <= "10010001";    -- h
        elsif   store = X"3B"  then   D <= "10000111";    -- j
        elsif   store = X"42"  then   D <= "11111101";    -- k
        elsif   store = X"4B"  then   D <= "11100011";    -- l
        elsif   store = X"4C"  then   D <= "01010101";    -- ñ
        elsif   store = X"52"  then   D <= "01100011";    -- {
        elsif   store = X"5D"  then   D <= "00001111";    -- }
        elsif   store = X"1A"  then   D <= "11111101";    -- z tercer linea de letras
        elsif   store = X"22"  then   D <= "11111101";    -- x
        elsif   store = X"21"  then   D <= "11100101";    -- c
        elsif   store = X"2A"  then   D <= "11111101";    -- v
        elsif   store = X"32"  then   D <= "11000001";    -- b
        elsif   store = X"31"  then   D <= "11010101";    -- n
        elsif   store = X"2A"  then   D <= "11111111";    -- m
        elsif   store = X"41"  then   D <= "11111111";    -- ,
        elsif   store = X"49"  then   D <= "11111110";    -- .
        elsif   store = X"4A"  then   D <= "11111101";    -- -
--      elsif   store = ""  then      D <= ""; --
        else                          D <= X"FF"; -- APAGA LOS DISPLAY
```

```
        end if;
end process; --termina de leer los datos del teclado

end keyboard; --termina la arquitectura


----------------------------------------------------------------------------------------------
-- Tercer código
----------------------------------------------------------------------------------------------
-- Lectura del código de rastreo de un teclado conectado al puerto PS/2 de la Nexys 2
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--Señales de la entidad
--reloj y dato del PS2, reset con un interruptor SW0 o botón
--8 leds de la tarjeta para sacar el código hexadecimal asignado a cada tecla
entity teclado is
    Port ( clk,dato,rst: in std_logic;
           leds : out std_logic_vector(7 downto 0));
end teclado;

architecture Arq_teclado of teclado is
--se declaran las señales reg, que es un registro de 22 bits
--para recibir los codigos de rastreo al soltar la tecla
--y tmp que se utiliza como un contador hasta 22 (0 a 21)
signal reg: std_logic_vector(21 downto 0);
signal tmp: std_logic_vector(4 downto 0);
begin

--Proceso para guardar los datos enviados del teclado
--por medio de un registro de corrimiento a la derecha
process (clk,rst,dato,reg)
begin
        if rst='1' then reg <= (others=>'0');
        elsif clk'event and clk ='0' then reg <= dato & reg(21 downto 1);
        end if;
end process;

--Proceso que habilita el contador tmp para la recepción del dato del teclado
conteo:process (clk,rst,tmp)
begin
--si reset = '1' o tmp=22, se resetea la señal tmp (contador)
        if rst='1' or tmp = "10110" then tmp <= (others=>'0');
--se habilita el contador con flanco de subida del reloj
        elsif clk'event and clk ='1' then tmp <= tmp + 1;
```

```
        end if;
end process conteo;

--Proceso para la salida de datos a los leds
led:process (rst,reg,tmp)
begin
        if rst='1' then leds <= (others=>'0');
--se verifica que ya se hayan mandado los 22 números de trama al soltar la tecla
        elsif tmp = "10110" then
--al soltar la tecla se envía el código de rastreo F0 "break code" y el codigo de la tecla
        elsif reg(8 downto 1)= X"F0" then leds <= reg(19 downto 12);
        else                                leds <= (others=>'0');
        end if;
end process led;
end Arq_teclado;
```

"Si tienes alguna duda, apóyate con tu profesor"

Desarrollo.
Procedure

Recuerde *MOSTRAR* sus circuitos funcionando a su profesor para la valoración del trabajo de laboratorio (TL2).

1. Implementar una unidad aritmético-lógica **ALU**, usando diseño de alto nivel, que realice las operaciones aritméticas A+B, A-B, AxB, A/B y las operaciones lógicas not A, A and B, A or B, A xor B, siendo las entradas A y B de por lo menos 4 bits, con salida aritmética "C" de 4 a 8 bits, empleando un selector "sel" de la operación de 3 bits de acuerdo a la tabla 3.1 y la figura 3.1 (repetidas abajo). Por lo menos una de las entradas es un encoder mecánico rotatorio. Un led bicolor o RGB en **Rojo** indica una operación aritmética y en **Azul** una operación lógica. El lenguaje en el que se implementa lo define el profesor. Reportar los códigos comentados, simulaciones, fotos del funcionamiento con texto explicativo de lo que sucede.
   *Implement an ALU arithmetic-logic unit, using a top level design, that performs the arithmetic operations A+B, A-B, AxB, A/B and the logical operations not A, A and B, A or B, A xor B , being the inputs A and B of at least 4 bits, with output C of 4 to 8 bits, using a "sel" selector of the 3-bit operation according to table 3.1 and figure 3.1 (repeated below). At least one of the inputs is a rotative mechanical encoder. A two-color led or RGB in **Red** indicates an arithmetic operation and in **Blue** a logical operation. The language in which it is implemented is defined by the teacher. Report the commented codes, simulations, operation photos with explanatory text of what happens.*

Tabla 3.1. Relación de operaciones de la ALU

Operaciones	selOp	Operación	Observaciones @ A y B de 4 bits
Aritméticas Salida a leds y a display. Led RGB **Rojo**.	**0**00	C=A+B	C es de 5 bits (máx 30)
	001	C=A-B	C es de 4 bits (de -15 a 15)
	010	C=AxB	C es de 8 bits (máx 225)
	011	C=A/B	C es de 4 bits (solo enteros de 1 a 15)
Lógicas Salida a leds. Led RGB **Azul**.	**1**00	C=not(A)	C es de 4 bits
	101	C=A and B	C es de 4 bits
	110	C=A or B	C es de 4 bits
	111	C=A xor B	C es de 4 bits

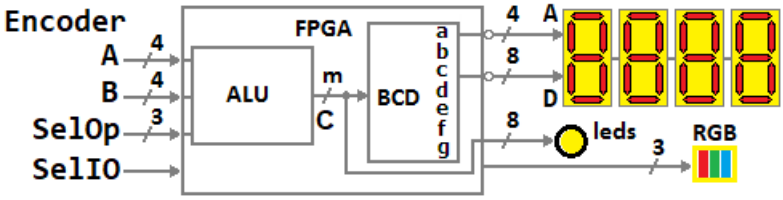

Figura 3.1. Diagrama a bloques de la ALU y su conexión.

Challenge 1 (substitute points 1 and 3).
Implement an ultrasonic distance meter from 0 to 400cm using the HC-05 or SRF-05 or similar, with 7-segment display output. Within the range of 0 to 30cm a proximity alarm is activated, which can be enabled and disabled with a switch. Report the internal parts (dividers, counters, BCD, etc.) with a block diagram, codes (HDL and UCF) and photos.

2. Implementar un circuito de control de giro para un **motor a pasos unipolar** con encoder mecánico rotatorio y encoder magnético u óptico, tal que al girar cualquiera de los encoder, la carga mecánica girará en la misma dirección (horario o antihorario) además de encender un led RGB que indique la dirección. Incluir sensores de límite para proteger el giro de la carga mecánica y un indicador audible ("beep") que indique ciertas posiciones. Es posible colocar un selector para direccionar la señal de cada encoder, funcionando de acuerdo a la figura 3.2 (repetida abajo). El motor deberá llevar una **carga** en el eje (banda, llantas, polea, sistema animatrónico, ventana o puerta a escala, etc., no se permiten cosas sencillas como hélices, clips, alambres, ligas, hojas de papel, etc.). El lenguaje en el que se implementa lo define el profesor. Reportar los códigos comentados, fotos del funcionamiento con texto explicativo de lo que sucede.
*Implement a control circuit for a unipolar stepper motor with rotary mechanical encoder and magnetic or optical encoder, such that when turning any of the encoders, the motor will turn in the same direction (clockwise or counterclockwise)) as well as turning on an RGB LED that indicate the address. Include limit sensors to protect the mechanical load rotation or an audible indicator ("beep") that indicates certain positions. It is possible to place a selector to address the signal of each encoder, working according to figure 3.2 (repeated below). The motor must have a **load** on the shaft (belt, wheels, pulley, animatronic system, window or door to scale, etc., simple things such as propellers, clips, wires, links, sheets of paper, etc. are not allowed). The language in which it is*

implemented is defined by the teacher. Report the commented codes, simulations, operation photos with explanatory text of what happens

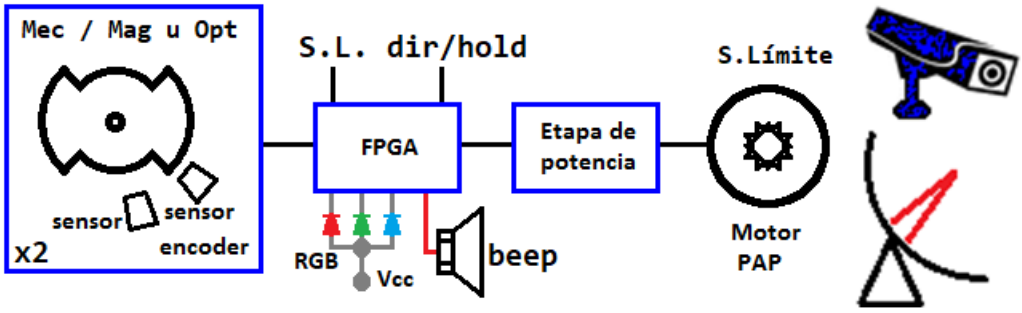

Figura 3.2. Diagrama a bloques del control del motor a pasos con encoder.

3. Implementar en TLD un **<u>contador de pulso en alto en ms</u>** con salida a display, el cual al presionar un botón (push) se visualiza el conteo en el display y cuando se suelta el botón se detiene en un valor, pero si se presiona otra vez continua el conteo. Se cuenta con un botón de reset para reiniciar en cero en cualquier momento que se presione. El display puede contar hasta un máximo de 9,999 ms (9.9 s) y cuando se alcance este valor se detendrá y sonará un mensaje en una bocina (*"valor máximo", "ya me llené"*, etc.). Por medio de un led RGB se indica en que rango en que se encuentra el conteo ($0 < R \leq 3{,}333$, $3{,}333 < G \leq 6{,}666$, $6{,}666 < B \leq 9{,}999$). Cada vez que se cambia de rango sonará un *"BEEP"* en un buzer o en la bocina. En la figura 3.3 se muestra el diagrama a bloques, en la 3.18 un ejemplo del funcionamiento y en la 3.19 una forma de hacer el TLD. El lenguaje en el que se implementa lo define el profesor. Reportar los códigos comentados, simulaciones, fotos del funcionamiento con texto explicativo de lo que sucede.
Implement in TLD a pulse counter in high in ms with output to display, which when pressing a button (push) the count is displayed on the display and when the button is released it stops at a value, but if it is pressed again continue the count. It has a reset button to restart at zero any time it is pressed. The display can count to a maximum of 9,999 ms (9.9 s) and when this value is reached it will stop and a message ("maximum value", "I am already filled") will sound in a speaker, etc. By means of a RGB led it is indicated in what rank the count is ($0 < R \leq 3{,}333$, $3{,}333 < G \leq 6{,}666$, $6{,}666 < B \leq 9{,}999$). Each time the range is changed, a "BEEP" will sound in a buzer or speaker. The block diagram is shown in figure 3.3, an example of the operation in figure 3.18 and one way to implement the TLD in figure 3.19. The language in which it is implemented is defined by the teacher. Report the commented codes, simulations, operation photos with explanatory text of what happens.

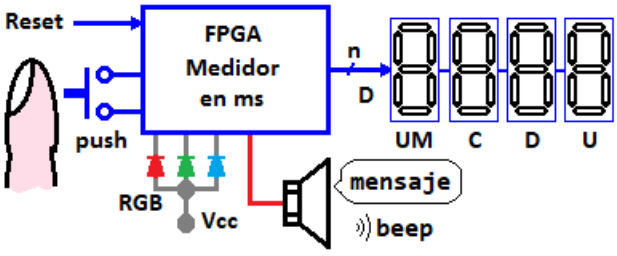

Figura 3.3. Esquema para el medidor de ms.

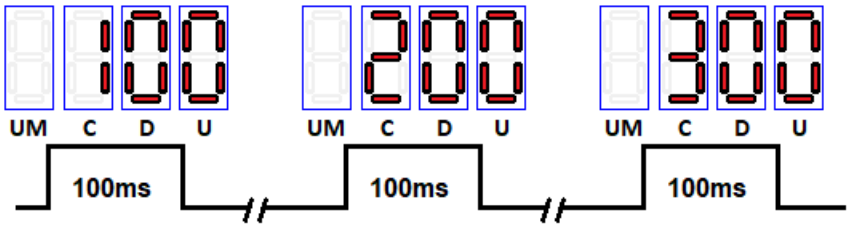

Figura 3.18. Ejemplo de funcionamiento cuando se presiona 3 veces 100ms.

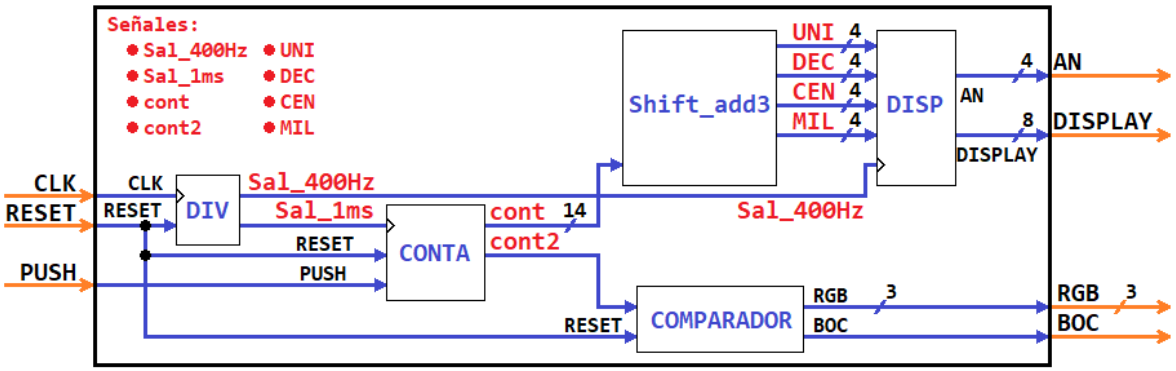

Figura 3.19. Estructura de bloques para el medidor de ms en TLD.

Consultar el boletín UPIITA número 63 (www.boletin.upiita.ipn.mx o http://www.boletin.upiita.ipn.mx/index.php/component/content/article/11-numeros/1427-numero-63).

4. Implementar un control de velocidad y sentido de giro de un **servomotor (sin tope)** con encoder rotatorio (óptico o magnético, ver figura 3.4). El movimiento del encoder en un sentido permitirá que el motor incremente su velocidad, mientras se le den más vueltas seguirá aumentando, pero si se dan vueltas en sentido contrario la velocidad disminuirá, ya sea hasta detenerse o bien para cambiar el sentido de giro e ir aumentando su velocidad en sentido contrario. En el display (ver figura 3.20) se muestra la velocidad en niveles numéricos (-5 a 5). Se utiliza una señal PWM adecuada al motor adquirido, **con carga** en el eje (banda, llantas, polea, sistema animatrónico, ventana o puerta a escala, etc., no se permiten cosas sencillas como hélices, clips, alambres, ligas, hojas de papel, etc.). El lenguaje en el que se implementa lo define el el profesor. Reportar los códigos comentados, simulaciones, fotos del funcionamiento con texto explicativo de lo que sucede.

   *Implement a speed control and direction of rotation of a servomotor (without stop) with rotary encoder (optical or magnetic, see figure 3.4). The movement of the encoder in one direction will allow the engine to increase its speed, as more turns are made in the same direction the encoder will continue to increase, but if they are turned in the opposite direction the speed will decrease, either to stop or to change the direction of rotation and increase its speed in the opposite direction. The display (see figure 3.20) shows the speed in numerical levels (-5 to 5). A suitable PWM signal is used for the motor purchased, **with load** on the shaft (belt, wheels, pulley, animatronic system, window or*

door to scale, etc., simple things such as propellers, clips, wires, rubber bands, paper sheet, etc. are not allowed). The language in which it is implemented is defined by the teacher. Report the commented codes, simulations, operation photos with explanatory text of what happens.

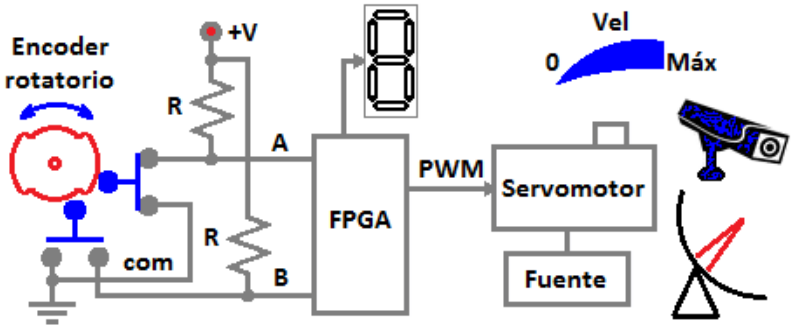

Figura 3.4. Diagrama a bloques para el control de velocidad del servo sin tope por PWM.

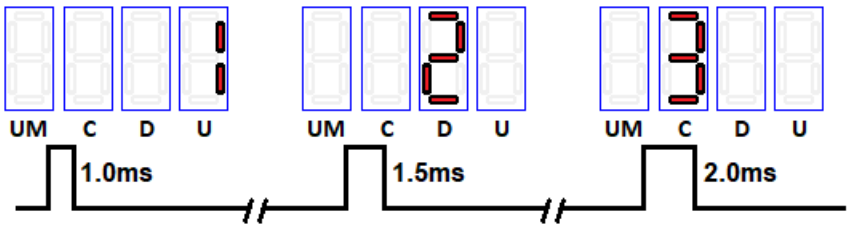

Figura 3.20. Ejemplo del valor del display para el servo con tope.

5. Implementar un control de posición para un **servomotor con tope** cambiando la señal de control (PWM) adecuada al tipo de motor, por medio de un encoder rotatorio mecánico (óptico o magnético), mostrado en la figura 3.5. El movimiento del encoder en un sentido cambiará la posición del servo en ese mismo sentido, tanto horario como antihorario. Los displays (ver figura 3.21) indican el ángulo del motor (0°-180°), con resolución mínima de 5°. El motor deberá llevar una **carga** en el eje (banda, llantas, polea, sistema animatrónico, ventana o puerta a escala, etc., no se permiten cosas sencillas como hélices, clips, alambres, ligas, hojas de papel, etc.). El lenguaje en el que se implementa lo define el el profesor. Reportar los códigos comentados, simulaciones, fotos del funcionamiento con texto explicativo de lo que sucede.
*Implement a position control for a servo motor with stop by changing the control signal (PWM) suitable to the motor type, by means of a mechanical rotary encoder (optical or magnetic), shown in figure 3.5. The movement of the encoder in one direction will change the position of the servo in that same direction, both clockwise and counterclockwise. The displays (see figure 3.21) indicate the angle of the motor (0°-180°), with a minimum resolution of 5°. The engine must carry a **load** on the axle (belt, wheels, pulley, animatronic system, window or door to scale, etc., simple things such as propellers, clips, wires, links, sheets of paper, etc. are not allowed). The language in which it is implemented is defined by the teacher. Report the commented codes, simulations, operation photos with explanatory text of what happens.*

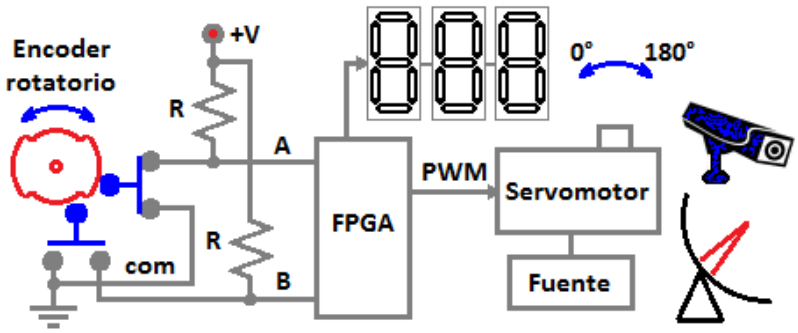

Figura 3.5. Diagrama a bloques para el control de posición del servo por PWM.

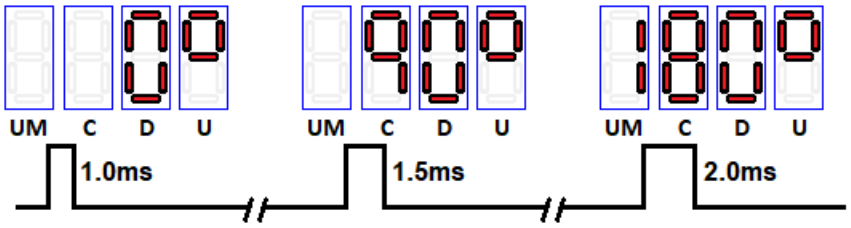

Figura 3.21. Ejemplo del valor del display para el servo con tope.

Challenge 2 (substitute the points 4 and 5).
Implement a 2DOF camera (laser, water weapon, etc.) positioner using 2 servomotors and a joystick (analog or digital). Report codes (HDL and UCF), photos and video.

6. Implementar: **(a)** un codificador de **teclado PS2 o USB** en leds y display 7seg o LCD (mínimo 30 teclas: números, letras, símbolos y funciones) como se muestra en el diagrama a bloques de la figura 3.6. Cuando se presiona el teclado, se muestran los caracteres en el display y en código binario (scan code) en los leds. **(b)** un sistema de control de posición de dos cámaras de seguridad de 2 GDL cada una, en el que las teclas de funciones permiten elegir cualquiera de las cámaras (F1 => cámara 1, F2 => cámara 2); una vez elegida la cámara con las teclas de flechas se permite controlar los movimientos arriba-abajo-izquierda-derecha (ver figuras 3.22 y 3.23). El lenguaje en el que se implementa lo define el profesor. Reportar los códigos comentados, simulaciones, fotos del funcionamiento con texto explicativo de lo que sucede. **Nota**: Los dos incisos se revisan juntos y este punto se puede entregar hasta por dos equipos.
*Implement: **(a)** a keyboard encoder PS2 or USB on LEDs and 7seg or LCD display (minimum 30 keys: numbers, letters, symbols and functions) as shown in the block diagram of figure 3.6. When the keyboard is pressed, the characters are shown on the display and in binary code (scan code) on the LEDs. **(b)** two camera security position control system of 2DOF each one, in which the function keys allow to choose any of the cameras (F1 => camera 1, F2 => camera 2); once the camera is chosen with the arrow keys, it is possible to control the movements up-down-left-right (see figures 3.22 y 3.23). The language in which it is implemented is defined by the teacher. Report the commented codes, simulations, operation photos with explanatory text of what happens. **Note**: the*

two subsections are delivered togheter and this point can be delivered by up to two teams.

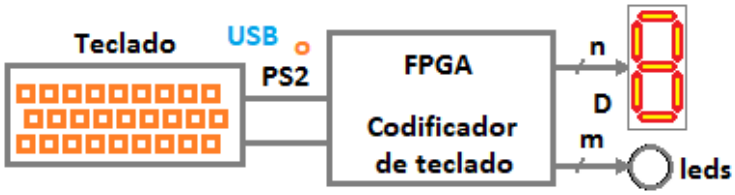

Figura 3.6. Diagrama a bloques para el control de posición del servo por PWM.

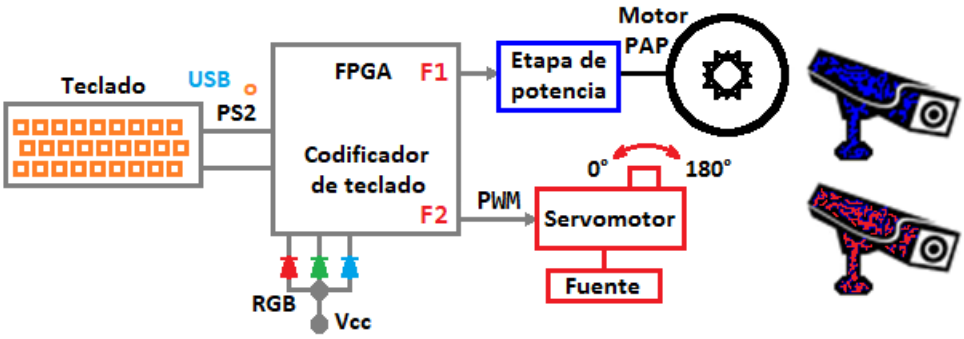

Figura 3.22. Diagrama a bloques del sistema de posición de 2 cámaras.

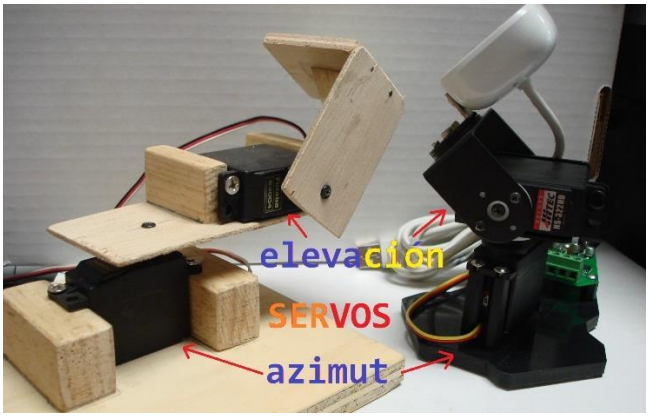

Figura 3.23. Mecanismos para el control de posición de 2 cámaras.

Challenge 3 (substitute point 6).
Implement a 2 cameras positioner controller with 2DOF each one, using 2 joysticks. Report codes (HDL and UCF), photos and video.

7. Realizar sus comentarios y conclusiones (incluir en por lo menos un párrafo Gracias "Gracias a esta práctica…"). Todos los códigos van en una sola columna con comentarios editados para que no brinquen de renglón y no van en imagen.
   Write your comments and conclutions. All the codes are written in one column with comments, only text, not images.

NOTA: Respetar la numeración de cada punto de este formato en el reporte escrito [**máximo 25 cuartillas** si se entrega impreso (Letra Times New Roman de 12ptos, interlineado sencillo)].

Proyectos opcionales utilizando PLDs y HDL.
Optional projects using PLDs and HDL.

Diseñar un sistema de control de posición de 10 cámaras de seguridad en el que las teclas de funciones permiten elegir cualquiera de las cámaras: F1 => cámara 1, F2 => cámara 2, etc.; una vez elegida la cámara con las teclas de flechas se permite controlar los movimientos arriba-abajo-izquierda-derecha. Es posible cambiar la selección de la cámara en cualquier momento.
Design a position control system of 10 security cameras in which the function keys allow to choose any of the cameras: F1 => camera 1, F2 => camera 2, etc.; once the camera has been selected with the arrow keys it is possible to control the movements up-down-left-right. You can change the camera selection at any time.

Diseñar un sistema de control de posición de tres cámaras de seguridad con tres joysticks, que permita los movimientos arriba-abajo-izquierda-derecha. Es posible controlar las tres cámaras en forma paralela. En vez de cámara se puede utilizar pintadoras, remachadoras, cortadoras, soldadoras, etc.
Design a three-position security camera control system with three joysticks, which allows the up-down-left-right movements. It is possible to control the three cameras in parallel. Instead of camera you can use painters, clinchers, cutters, welders, etc.

Implementar un controlador de velocidad de un motor de DC con o sin engranes al modificar el ancho de pulso de la señal de control utilizando un encoder.
Implement a DC motor speed controller with or without gears by modifying the pulse width of the control signal using an encoder.

Implementar un control de luz para lámparas leds al cambiar el PWM con un encoder, pero que al presionar el botón integrado en el encoder, se encienda a su máxima intensidad y al volverlo a presionar baje su intensidad.
Implement a light control for LED lamps when changing the PWM with an encoder, but pressing the integrated button in the encoder, will turn on at its maximum intensity and pressing it down again will reduce its intensity.

Implementar un sistema para detectar el nivel de llenado de dos tanques, tal que avise cuando el nivel de cualquiera de los tanques esté por debajo del 10% de llenado.
Implement a system to detect the level of filling of two tanks, such that warns when the level of any of the tanks is below the 10% of filling.

Implementar un sistema lanzapelotas que se controle en azimut y elevación con las flechas de un teclado PS2, lanzando la pelota cuando se presione enter.
Implement a ball-launcher system that is controlled in azimuth and elevation with the arrows of a PS2 keyboard, throwing the ball when press enter.

Implementar con HDL: un medidor de distancia utilizando un sensor de distancia ultrasónico SRF05, el cual al dispararse entrega una señal ECHO cuyo ancho varía con respecto a la distancia.
Implement with HDL: a distance meter using an ultrasonic distance sensor SRF05, which, when triggered, delivers an ECHO signal whose width varies with respect to the distance.

Revisar el siguiente Código e indicar para que sirve:
Review the next code and say what is it for:

```
--------------------------------------------------------------------------------
-- Contador ascendente y descendente, con 2 reset asíncronos y salida pwm a led.

Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

                -- Declaración de la entidad
entity Contador is
Port (UpDown: in STD_LOGIC_VECTOR(1 DOWNTO 0); -- botones subir y bajar ms
        CLK: in STD_LOGIC; -- reloj de 50MHz-nexys 2 y 100MHz-nexys 3
        RESET: in STD_LOGIC; -- reset
        SALED: OUT STD_LOGIC; -- salida del led testigo
        DISPLAY: out STD_LOGIC_VECTOR(7 DOWNTO 0); -- segmentos "abcdefgP"
        AN: out STD_LOGIC_VECTOR(3 DOWNTO 0); -- ánodos del display
        RGB: out STD_LOGIC_VECTOR(1 TO 3)); -- salida a leds RGB para el rango
end Contador;

                -- Declaración de la arquitectura
architecture Behavioral of Contador is

                -- Declaración de señales de los divisores
signal Conta_500us: integer range 1 to 25_000:=1; -- uso en el pulso de 1ms
                                -- para la nexys 2 25000, si se usa la nexys 3 cambiar a 50000.
signal contadors: integer range 1 to 6_250:=1; --  pulso1 de 0.25ms
                                -- para la nexys 2 6250, si se usa la nexys 3 cambiar a 12500
signal SAL_1ms,SAL_250us: std_logic; --igual que pulso y pulso1, respectivamente

                -- Declaración de señales de los contadores
signal CONT: std_logic_vector (15 DOWNTO 0):=(others=>'0'); -- 16 bits (proc. conteo)
SIGNAL CONT2: INTEGER; -- cambia cont de 16 bits a entero (proc. conteo)
```

```
                -- Declaración de señales de la asignación de U-D-C-UM
signal P: std_logic_vector (15 DOWNTO 0); -- asigna UNI, DEC, CEN, MIL
signal UNI,DEC,CEN,MIL: std_logic_vector (3 DOWNTO 0); -- digitos unidades, decenas,
                                                        -- centenas y unidad de millar

        -- Declaración de señales de la multiplexación y asignación de U-D-C-UM al display
signal SEL: std_logic_vector (1 downto 0):="00"; -- selector de barrido
signal D: std_logic_vector (3 downto 0); -- sirve para almacenar los valores del display

                -- Declaración de señales de la base de tiempo
SIGNAL PERIOD: INTEGER RANGE 0 TO 2499:=0; -- periodo de 2.5 segundos @ PWM

BEGIN

----------------------DIVISOR 1ms-----------------------------------
-- en este proceso se genera una señal "SAL_1ms" de 1ms de periodo
PROCESS(reset, CLK)
begin
        if reset ='1' then
                Conta_500us <= 1; -- se reinicializa
        elsif(CLK'event and CLK='1') THEN
                if(Conta_500us = 25_000) then -- pregunta si ya se alcanzó 0.5ms (nexys2)
                --if(Conta_500us = 50000) then -- pregunta si ya se alcanzó 0.5ms (nexys3)
                        SAL_1ms <= not SAL_1ms; --se genera 0.5ms en bajo y 0.5ms en alto
                        Conta_500us <= 1; --reinicia el Contador a 1
                else
                        Conta_500us <= Conta_500us + 1;
                end if;
        end if;
end process; --termina el proceso de generación de señal 1ms


--------------------------------CONTEO-----------------------------------
-- con Up se incrementa y Down decrementa en el rango 0<CONT<9999
PROCESS(RESET, SAL_1ms, UpDown, CONT)
begin
        if RESET='1' then
                CONT<=(others=>'0');
        else
                if(SAL_1ms'EVENT and SAL_1ms='1') then -- reloj SAL de 1ms
                        if UpDown="01" then -- decrementa (Down)
                                if(CONT=x"0000") then -- compara contra 0
                                        CONT<=CONT; -- si llegó a cero mantiene el cero
                                else
                                        CONT<=CONT-'1'; -- sino decrementa
```

```
                end if;
            elsIF UpDown="10" then --incrementa
                if(CONT >= "0010011100001111") then -- compara vs 9,999
                                                    -- (270F hex)
                    CONT<=CONT; -- si llego a 9999 mantiene 9999
                else
                    CONT<=CONT+'1'; -- sino incrementa
                end if;
            else --cubre UpDown="00" e UpDown="11"
                CONT<=CONT;
            end if;
        end if;
    end if;

    CONT2<=CONV_INTEGER(CONT); --se convierte CONT a entero

end process;


---------------------------------RGB-----------------------------------
-- 0< R <3333          3333< G <6666              6666< B <9999
PROCESS(CONT2)
begin
    if    CONT2>=0    and CONT2 <= 3333 then RGB <="100"; -- 0< R <3333
    elsif CONT2>3333 and CONT2 <= 6666 then RGB <="010"; -- 3333< G <6666
    elsif CONT2>6666 and CONT2 <= 9999 then RGB <="001"; -- 6666< B <9999
    else                                    RGB <="000"; -- apaga todos
    end if;
end process;


--------------------PWM CONTROLADO POR CONTADOR DE ms/4-----------------------

PROCESS (RESET, SAL_1ms, PERIOD)
BEGIN
    if (RESET='1' OR PERIOD >= 2499) then --9999="0010011100001111"
        PERIOD <= 0; --(others=>'0');
    elsif (SAL_1ms'EVENT and SAL_1ms='1') then -- reloj SAL_1ms de 1ms
        PERIOD <= PERIOD + 1;
        if (PERIOD <= CONT2/4) then SALED <='1'; -- Salida a led testigo t/4
--      if (PERIOD <= CONT) then SALED <='1'; -- Salida a led testigo
        else                     SALED <='0';
        end if;
    end if;
end PROCESS; --fin del proceso PWM
```

```
----------CONVERTIR DE BIN A BCD------------------
-- Este proceso contiene un algoritmo recorre y suma 3 para convertir un número binario a
-- bcd, que se manda a los displays.
-- El algoritmo consiste en desplazar (shift) el vector inicial (en binario) el número de veces
-- según sea el número de bits. Cuando alguno de los bloques de 4 bits (U-D-C-UM) sea igual
-- o mayor a 5 (por eso el >4) se le debe sumar 3 a ese bloque, después se continua
-- desplazando hasta que cualquier otro bloque cumpla con esa condición y se le sumen 3.
-- Inicialmente se rota 3 veces porque es el número mínimo de bits que debe tener para que
-- sea igual o mayor a 5.
-- Finalmente se asigna a otro vector, el vector ya convertido, que cuenta con 4 bloques para
-- las 4 cifras de 4 bits cada una.

PROCESS(CONT)
        VARIABLE UM_C_D_U:STD_LOGIC_VECTOR(29 DOWNTO 0);
 --30 bits para separar las Unidad_de_Millar-Centenas-Decenas-Unidades
BEGIN

--ciclo de inicialización
  FOR I IN 0 TO 29 LOOP --
    UM_C_D_U(I):='0'; -- se inicializa con 0
  END LOOP;

  UM_C_D_U(13 DOWNTO 0):=CONT(13 downto 0); --contador de 14 bits
 -- UM_C_D_U(17 DOWNTO 4):=CONT(13 downto 0); --contador de 14 bits, carga desde
                                                -- el shift4
--ciclo de asignación UM-C-D-U
  FOR I IN 0 TO 13 LOOP
--  FOR I IN 0 TO 9 LOOP -- si carga desde shift4 solo hace 10 veces el ciclo shift add
-- los siguientes condicionantes comparan (>=5) y suman 3
    IF UM_C_D_U(17 DOWNTO 14) > 4 THEN                  -- U
      UM_C_D_U(17 DOWNTO 14):= UM_C_D_U(17 DOWNTO 14)+3;
    END IF;

    IF UM_C_D_U(21 DOWNTO 18) > 4 THEN                  -- D
      UM_C_D_U(21 DOWNTO 18):= UM_C_D_U(21 DOWNTO 18)+3;
    END IF;

    IF UM_C_D_U(25 DOWNTO 22) > 4 THEN                  -- C
      UM_C_D_U(25 DOWNTO 22):= UM_C_D_U(25 DOWNTO 22)+3;
    END IF;

    IF UM_C_D_U(29 DOWNTO 26) > 4 THEN                  -- UM
      UM_C_D_U(29 DOWNTO 26):= UM_C_D_U(29 DOWNTO 26)+3;
    END IF;
          -- realiza el corrimiento
          UM_C_D_U(29 DOWNTO 1):= UM_C_D_U(28 DOWNTO 0);
```

```
 END LOOP;

  P<=UM_C_D_U(29 DOWNTO 14); -- guarda en P y en seguida se separan UM-C-D-U

 END PROCESS;

--UNIDADES
UNI<=P(3 DOWNTO 0);
--DECENAS
DEC<=P(7 DOWNTO 4);
--CENTENAS
CEN<=P(11 DOWNTO 8);
--MILLARES
MIL<=P(15 DOWNTO 12);


--------------------DIVISOR ÁNODOS--------------------

process (CLK) begin
       if rising_edge(CLK) then
          if (contadors = 6250) then --cuenta 0.125ms (50MHz=6250)
--          if (contadors = 12500) then --cuenta 0.125ms (100MHz=12500)
             SAL_250us <= NOT(SAL_250us); --genera un barrido de 0.25ms
             contadors <= 1;
          else
             contadors <= contadors+1;
          end if;
       end if;
   end process; -- fin del proceso Divisor Ánodos


--------------------MULTIPLEXOR----------------------

PROCESS(SAL_250us, sel, UNI, DEC,CEN, MIL)
BEGIN
        IF SAL_250us'EVENT and SAL_250us='1' THEN SEL <= SEL+'1';

        CASE(SEL) IS
        when "00" => AN <="0111";D <= UNI;          -- UNIDADES
        when "01" => AN <="1011";D <= DEC;          -- DECENAS
        when "10" => AN <="1101";D <= CEN;          -- CENTENAS
        when "11" => AN <="1110";D <= MIL;          -- UNIDAD DE MILLAR
        when OTHERS=>AN <="1110";     D <= MIL;     -- UNIDAD DE MILLAR
        END CASE;

        end if;
```

```
END PROCESS; -- fin del proceso Multiplexor


--------------------DISPLAY---------------------

PROCESS(D)
Begin
        case(D) is                      --      abcdefgP
        WHEN "0000" => DISPLAY <= "00000011"; --0
        WHEN "0001" => DISPLAY <= "10011111"; --1
        WHEN "0010" => DISPLAY <= "00100101"; --2
        WHEN "0011" => DISPLAY <= "00001101"; --3
        WHEN "0100" => DISPLAY <= "10011001"; --4
        WHEN "0101" => DISPLAY <= "01001001"; --5
        WHEN "0110" => DISPLAY <= "01000001"; --6
        WHEN "0111" => DISPLAY <= "00011111"; --7
        WHEN "1000" => DISPLAY <= "00000001"; --8
        WHEN "1001" => DISPLAY <= "00001001"; --9
        WHEN OTHERS => DISPLAY <= "11111111"; --apagado
        END CASE;
END PROCESS; -- fin del proceso Display

-----------------------------------------------

end Behavioral; -- fin de la arquitectura


//////////////////////////////////////////////////////////////////////////////////
//   Archivo de restricciones de usuario (UCF) Nexys 2      //
//////////////////////////////////////////////////////////////////////////////////
//CLK reloj 50MHz
NET "CLK" LOC = "B8";

//DISPLAYS
NET "DISPLAY(7)" LOC = "L18";           // a
NET "DISPLAY(6)" LOC = "F18";           // b
NET "DISPLAY(5)" LOC = "D17";           // c
NET "DISPLAY(4)" LOC = "D16";           // d
NET "DISPLAY(3)" LOC = "G14";           // e
NET "DISPLAY(2)" LOC = "J17";           // f
NET "DISPLAY(1)" LOC = "H14";           // g
NET "DISPLAY(0)" LOC = "C17";           // P

//ANODOS
NET "AN(3)" LOC = "F17";                // AN0
NET "AN(2)" LOC = "H17";                // AN1
NET "AN(1)" LOC = "C18";                // AN2
```

```
NET "AN(0)" LOC = "F15";                // AN3

//UpDown,RESET--PUSH BUTTON
NET "UpDown(1)" LOC = "B18";            // btn0 SUBE
NET "UpDown(0)" LOC = "D18";            // btn1 BAJA
NET "RESET" LOC = "H13”;                // btn2 RST

//SALED salida a led testigo
NET "SALED" LOC = "J14";                // LD0

//RGB salida a leds de rango
NET "RGB(1)" LOC = "M15";               // JA4
NET "RGB(2)" LOC = "T17";               // JB4
NET "RGB(3)" LOC = "H16";               // JC4

#Nexys2 Pmod Connector Pin Assignments
#Pmod JA              Pmod JB              Pmod JC              Pmod JD
#JA1:L15  JA7:K13     JB1:M13  JB7:P17     JC1:G15  JC7:H15     JD1:J13  JD7:K14
#JA2:K12  JA8:L16     JB2:R18  JB8:R16     JC2:J16  JC8:F14     JD2:M18  JD8:K15
#JA3:L17  JA9:M14     JB3:R15  JB9:T18     JC3:G13  JC9:G16     JD3: N18  JD9:J15
#JA4:M15 JA10:M16     JB4:T17  JB10:U18    JC4:H16  JC10:J12    JD4:P18  JD10:J14
```

"Imprime la parte UPIITA a tu trabajo, busca información en la red y compártela con tus compañeros".

“Intercambia opiniones y experiencias con tus compañeros de grupo, acerca del conocimiento que has adquirido después de haber finalizado la práctica.”

Referencias.
References.

Referencias consultadas en septiembre -2021:
Teclado de computadora, http://picmania.garcia-cuervo.net/proyectos_teclado_ps2.htm
Código y diagrama de envío, http://www.pyroelectro.com/tutorials/keybd/software.html

Juan Antonio Jaramillo Gómez, Mirna Salmerón Guzmán, Rafael Santiago Godoy, CONTROL DE POSICIÓN DE UN SERVOMOTOR UTILIZANDO LA TARJETA DE DESARROLLO NEXYS 2 Y VHDL, boletín UPIITA número 29, consultado en septiembre-2021, disponible en http://www.boletin.upiita.ipn.mx/index.php/ciencia/211-cyt-numero-29/214-control-de-posicion-de-un-servomotor-utilizando-la-tarjeta-de-desarrollo-nexys-2-y-vhdl

Juan Antonio Jaramillo Gómez, Mirna Salmerón Guzmán, Rafael Santiago Godoy, CONTROL DE UN SERVOMOTOR CON UN ENCODER MECÁNICO ROTATORIO UTILIZANDO VHDL Y LA NEXYS II, boletín UPIITA número 34, consultado en septiembre-2021, disponible en http://www.boletin.upiita.ipn.mx/index.php/ciencia/216-

cyt-numero-34/122-control-de-un-servomotor-con-un-coder-mecanico-rotatorio-utilizando-vhdl-y-la-nexys-ii

Juan Antonio Jaramillo Gómez, Mirna Salmerón Guzmán, Brahim El Filali, Método de Conversión de Binario a BCD con VHDL, boletín UPIITA número 63, consultado en septiembre -2021, disponible en http://www.boletin.upiita.ipn.mx/index.php/ciencia/735-cyt-numero-63/1434-metodo-de-conversion-de-binario-a-bcd-con-vhdl

PWM, consultado en septiembre-2021, disponible en https://www.fpga4student.com/2017/08/verilog-code-for-pwm-generator.html

Librería sensor ultrasónico HC-SR04, consultado en septiembre-2021, disponible en https://intesc.mx/libreria-sensor-ultrasonico-hc-sr04/?v=0b98720dcb2c

A continuación, se presenta una tabla para el registro de los puntos entregados en el laboratorio de la práctica 3.
Next, a table is presented for the recording of the points delivered in the laboratory of the practice 3.

Tabla de registro para el trabajo de laboratorio (TL3).
Laboratory work register (LW3).

Laboratorio de Dispositivos Lógicos Programables

Lab DLP			Punto 1	Punto 2	Punto 3	Punto 4	Punto 5	Punto 6	**Práctica 3 Aplicaciones de circuitos combinacionales y secuenciales.**			
Nombre	mesa	Empresa	ALU	PAP encoder	TLD 9999ms	Vel-servo	Pos-servo	PS2-USB	Com	TL3	R3	P3
	1											
	1											
	1											
	2											
	2											
	2											
	3											
	3											
	3											
Fecha									6hr. 4clas	6pts	6pts	
Challenge 1. 1, 3												
Challenge 2. 4, 5.												
Challenge 3. 6												

Challenge 1. Implement US dist. Meter (SRF-05) 0-400cm, 7-seg, 0 to 30cm an alarm.
Challenge 2. Implement a 2DOF camera positioner, 2 servos and joystick.
Challenge 3. Implement 2 cameras position control, 2DOF each one using 2 joystick.

1 Aritméticas A+B, A-B, AxB, A/B; Op. Lógicas not A, A and B, A or B, A xor B, encode
1 Op. Arithmetic A + B, A-B, AxB, A / B; Op. Logic not A, A and B, A or B, A xor B, enc

2 Motor PAP con encoder rotatorio mecánico y magnético u óptico.
2 PAP motor with mechanical and magnetic or optical rotary encoder.

3 TLD Cnt hasta 9999ms c/beep&RGB@rango, reset y MSG.
3 TLD Cnt up to 9999ms w/beep&RGB@range, reset and MSG.

4 Ctrl. de velocidad y sentido de giro en servomotor sin tope con encoder, salida a display.
4 Speed/direction servomotor ctrl without stop, display.

5 Ctrl. posición de servomotor con tope (0°-180°) con encoder, salida a display.
5 servomotor position cntrl (0°-180°) with stop, display output.

6 Codificador de teclado PS2. Control de posición de 2 cámaras de 2GDL usando flechas del teclado.
6 PS2 keyboard encoder. 2 cameras position control of 2DOF with keyboard's arrows.

Página dejada en blanco

Page left in blank