UPIITA-IPN **Disp. Log. Prog.**

Práctica 03: Aplicaciones de circuitos combinacionales y secuenciales.

García Morales José Manuel

Lemus Olivares Alexis

Morales Fernández Albert

Lazcano Malagón Christian Giancarlo

Grupo: 2MV30

Numero de Equipo: 10

RESUMEN

En este documento se presenta el uso de entradas analógicas, esto representado por servomotores, los cuales eran controlados por un elemento externo y al mismo tiempo por la FPGA.

ABSTRACT

This document presents the use of analog inputs, this represented by servomotors, which were controlled by an external element and at the same time by the FPGA.

RESUMÉ

Ce document présente l'utilisation d'entrées analogiques, représentée par des servomoteurs, qui étaient contrôlés par un élément externe et en même temps par la FPGA.

Disciplina: La disciplina es la clave del éxito.

Compromiso: Tengo el compromiso de superarme diariamente.

INTRODUCCIÓN

El Módulo KY-023 Sensor JoyStick es un dispositivo electromecánico consta de dos potenciómetros en un ángulo de 90 grados, lo que se requiere de 2 pines analógicos para realizar la interfaz con cualquier tarjeta de desarrollo: Arduino, ESP32, ESP8266, etc.

Este elemento te permite controlar y manejar determinados aparatos electrónicos. Normalmente se utilizan para proyectos robóticos en el cual se necesitan para la movilidad analógica de las articulaciones de un brazo robótico. El Módulo Joystick, es más utilizado para proyectos de robótica y control de dispositivos RF (Radio Frecuencia) NRF24L01.

ESPECIFICACIÓN Y CARACTERÍSTICAS

- Módulo: JoyStick biaxial XY (KY-023)
- Voltaje de alimentación: 3.3 V a 5V
- Salida: Analógica(X,Y) y Digital(Z)
- Número de potenciómetros: 2 de 10Kohm.
- Pulsador central normalmente abierto.
- Dimensiones: 40mm x 26mm x 32 mm
- Peso: 11 g

El módulo KY-023 cuenta con un posicionamiento de eje dado por valores entre 0 a 1023 y conforme lo movamos su valor ira variando y dando las coordenadas de cada eje, este valor lo podemos imprimir en el monitor serial en el IDE de Arduino.

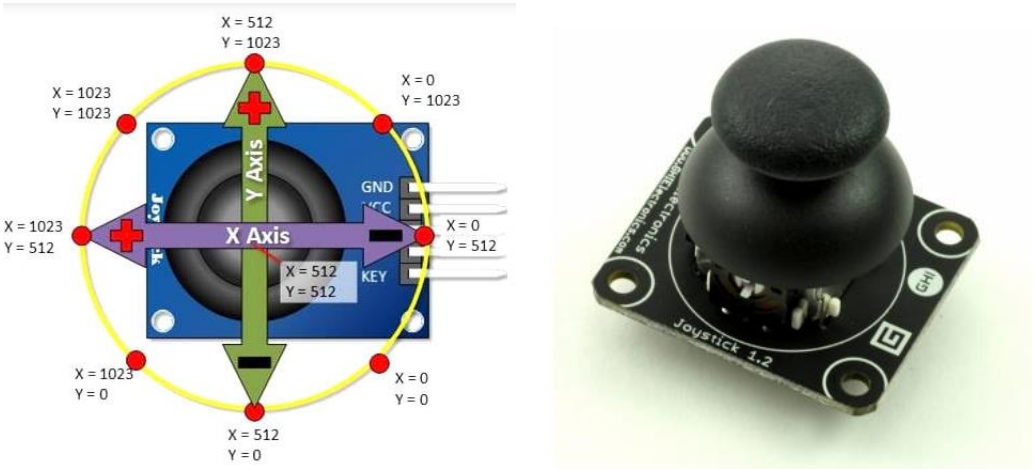


Fig. 1. Joystick

Es un pequeño actuador rotativo o bien motor que permite un control preciso en posición angular, este servomotor puede rotar de 0° hasta 180°, su voltaje de operación que va desde los 4.8 a 6 VDC. Este servo incluye 3 brazos y 3 tornillos, cuenta con un cable de hasta 25cm.

Este tipo de servomotores son utilizado en gran variedad de proyectos de electrónica, robótica, carros de control remoto, aeronaves y más. Funcionan con la mayoría de tarjetas electrónicas de control con microcontroladores, como por ejemplo las tarjetas de Arduino, Nodemcu, Esp32, Pic´s y Raspberry Pi y otras.

El servo SG90 tiene un conector universal tipo “S” que encaja perfectamente en la mayoría controladores de servos por ejemplo el Controlador PCA9685 16 o el Probador de Servo 3CH ECS CCPM.

Este tipo de servo es ideal para las primeras experiencias de aprendizaje y prácticas con servos, ya que sus requerimientos de energía son bastante bajos y se permite alimentarlo con la misma fuente de alimentación que el circuito de control. Por ejemplo, si se conecta a una tarjeta Arduino, se puede

alimentar durante las pruebas desde el puerto USB del PC sin mayor problema, pero te recomendamos usar una fuente de alimentación independiente para el servomotor.

ESPECIFICACIONES Y CARACTERISTICAS

- Modelo: SG90
- Color: Azul
- Tamaño: 22.8mm x 12.3mm x 22.5mm
- Peso: 13 g
- Grados / Angulo de Rotación Máximo: 0° a 180°
- Engranajes: Nylon
- Temperatura de trabajo: -30 a +60 Grados Celsius
- 7 microsegundos
- Voltaje de funcionamiento: 4.8VDC a 6VDC. Recomendado 5VDC
- Rojo =Alimentación (+)
- Café = Alimentación (–) o tierra
- Naranja= Señal PWM
- Rojo: VCC
- Línea naranja: entrada de pulso

Incluye:

- 1 x Micro Servo SG90 con cable de 25cm
- 3 x Brazos para Servo
- 3 x Tornillos

*Nota: El servomotor incorpora un controlador que sólo permite tener una rotación 0° hasta 180° si se manda el ancho de pulso PWM desde el microcontrolador. Si estas utilizando el probador de servos este estará limitado de 0° a 90°. Este servo no incorpora tope para definir en los engranes mecánicamente la rotación de 0° hasta 180° por lo que si gira de forma manual el servo tendrá una rotación continua sin tener tope.

Fig. 2. Servomotor

DESARROLLO

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 con 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

Se implemento un sistema de elevador, cuando subia en su totalidad la carga el sensor infrerojo que se encuentra en la parte superior detecta dicha craga, por lo que ya no es posible seguir subiendo y se emite una señal audible, se mueve el encoder en la direccion opuesta para que la carga baje, cuando esta se encuentra en el findo el sensor que se encuentra en la parte inferior realiza la misma funcion, ya no perimite que siga bajando y vuelve a emitir la señal audible

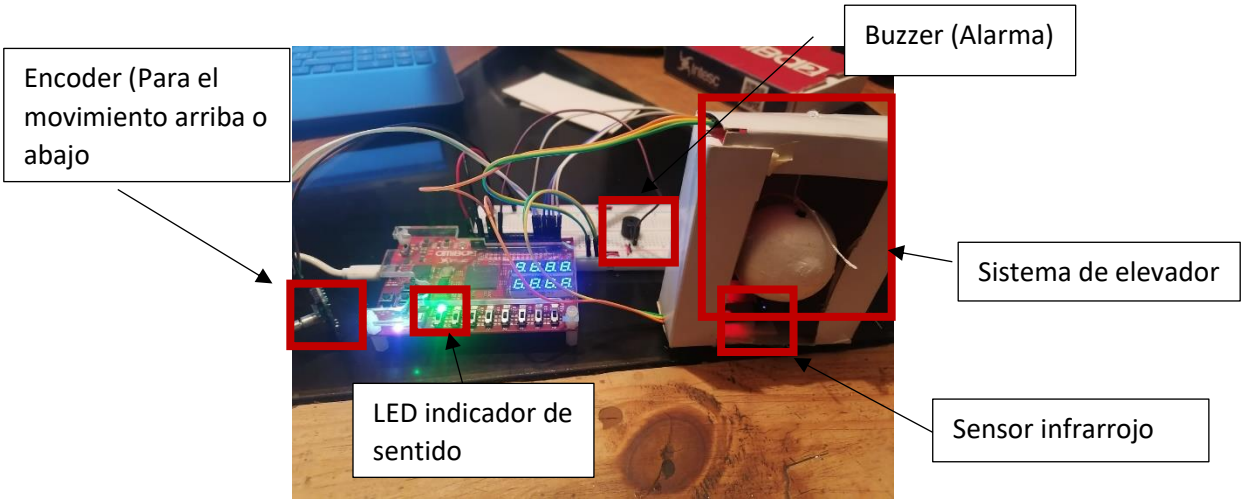


Fig. 3 Sistema de elevador controlado con un encoder y sensores limitantes

CODIGOS

```
--codigo para controlar los pasos de un motor PAP a traves de un encoder mecanico
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

ENTITY motor_encoder IS PORT(

CLK: in STD_LOGIC; -- reloj de 50MHz para la amiba2
S1,S2 : IN STD_LOGIC;
Entradas : in STD_LOGIC_VECTOR (5 downto 4); -- Primeras dos entradas de amiba 2
puertos,leds : OUT std_logic_vector (1 to 4); --leds testigos y salida al puerto para el motor,
representa el encdedido de las bobinas
LED: out STD_LOGIC_VECTOR (1 downto 0)  -- Led indicates the direction the shaft (encoder)
);
END motor_encoder;

architecture Behavioral of motor_encoder is
-- signals
   signal EncO : std_logic_vector (4 downto 0);
   signal AO, BO: std_logic;
begin

C0: entity work.Debouncer
port map (
clk=>clk,
Ain=>Entradas(4),
Bin=>Entradas(5),
Aout=> AO,
Bout=> BO);

C1: entity work.Encoder
port map (
clk=>clk,
A=>AO,
B=>BO,
puertos => puertos,
S1 => S1,
S2 => S2,
leds => leds,
LED=>LED
);
end Behavioral;
```

Debouncer.vhd

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

entity Debouncer is
Port (
clk : in STD_LOGIC;
-- signals from the pmod
Ain : in STD_LOGIC;
Bin : in STD_LOGIC;
-- debounced signals
Aout: out STD_LOGIC;
Bout: out STD_LOGIC
);
end Debouncer;

architecture Behavioral of Debouncer is
-- signals
signal sclk: std_logic_vector (6 downto 0);
signal sampledA, sampledB : std_logic;
begin

process(clk)
begin

if clk'event and clk = '1' then
sampledA <= Ain;
sampledB <= Bin;
-- clock is divided to 1MHz
-- samples every 1uS to check if the input is the same as the sample
-- if the signal is stable, the debouncer should output the signal
if sclk = "1100100" then
-- A
if sampledA = Ain then
Aout <= Ain;
end if;
-- B
if sampledB = Bin then
Bout <= Bin;
end if;
sclk <="0000000";
else
sclk <= sclk +1;
end if;
end if;
end process;
end Behavioral;
```

Encoder.vhd

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

entity Encoder is
Port (
    clk: in STD_LOGIC;
puertos,leds: OUT std_logic_vector (1 to 4); --leds testigos y salida al puerto para el motor,
representa el encdedido de las bobinas
    -- signals from the pmod
    A : in STD_LOGIC;
    B : in STD_LOGIC;
 S1, S2 : IN STD_LOGIC;
    -- position of the shaft
    -- direction indicator
    LED: out STD_LOGIC_VECTOR (1 downto 0)
);
end Encoder;

architecture Behavioral of Encoder is
-- FSM states and signals
type stateType is ( idle, R1, R2, R3, L1, L2, L3, add, sub);
signal curState, nextState: stateType;
signal conter: integer range 1 to 3:=3;
signal ciclo1,ciclo2: integer range 0 to 25:=0;
signal conta_1250us: integer range 1 to 55000:=1; -- pulso1 de 1250us@400Hz (0.25ms) 62500
signal SAL_400Hz: STD_LOGIC; -- reloj de 400Hz
signal sw1,sw2: std_logic_vector(3 downto 0):="0000";
signal sd1,sd2: std_logic:='0';
signal limite: integer:=15;
begin
   limite<=15; --lmite de pasos (por 4)

next_state: process (curState, A, B)
begin
curState <= nextState;
case curState is

    --detent position
    when idle =>  conter<=3;
        LED<= "00";
        if B = '0' then
            nextState <= R1; conter<=1;
        elsif A = '0' then
            nextState <= L1; conter<=2;
        else
             nextState <= idle;
        end if;
```

```
-- start of right cycle
--R1
when R1 =>  conter<=1;
   LED<= "01";
   if B='1' then
      nextState <= idle;
   elsif A = '0' then
      nextState <= R2;
   else
      nextState <= R1;
   end if;

--R2
when R2 =>  conter <=1;
   LED<= "01";
   if A ='1' then
      nextState <= R1;
   elsif B = '1' then
      nextState <= R3;
   else
      nextState <= R2;
   end if;

--R3
when R3 =>  conter <=1;
      LED<= "01";
   if B ='0' then
      nextState <= R2;
   elsif A = '1' then
      nextState <= add;
   else
      nextState <= R3;
   end if;

when add =>  conter <=1;
   LED<= "01";
   nextState <= idle;

   -- start of left cycle
   --L1
when L1 =>  conter <=2;
      LED<= "10";
   if A ='1' then
      nextState <= idle;
   elsif B = '0' then
      nextState <= L2;
   else
      nextState <= L1;
   end if;
```

```
      --L2
   when L2 =>  conter <=2;
         LED<= "10";
      if B ='1' then
         nextState <= L1;
      elsif A = '1' then
         nextState <= L3;
      else
         nextState <= L2;
      end if;

       --L3
   when L3 =>  conter <=2;
      LED<= "10";
      if A ='0' then
         nextState <= L2;
      elsif B = '1' then
         nextState <= sub;
      else
         nextState <= L3;
      end if;

   when sub =>  conter <=2;
      LED<= "10";
      nextState <= idle;

   when others =>
      LED<= "11";
      nextState <= idle;
end case;
end process;

puerto: process(CLK,conter) begin

      if rising_edge(CLK) then
         if (conta_1250us = 55000) then --cuenta 1250us (50MHz=62500)
               -- if (conta_1250us = 125000) then --cuenta 1250us (100MHz=125000)
               SAL_400Hz <= NOT(SAL_400Hz); --genera un barrido de 2.5ms
               conta_1250us <= 1;
         else
               conta_1250us <= conta_1250us + 1;
         end if;
      end if;

IF SAL_400Hz'EVENT and SAL_400Hz='1'  then
   if (conter = 1 or sd1= '1') and sd2 ='0' then
      if sw1 /= "0100" and ciclo1 /=limite AND S1 = '0'THEN sw1 <= sw1 + '1';
            case sw1 is
            when "0000" => leds <= "1100"; puertos <= "1100"; sd1 <='1';
```

```
            when "0001" => leds <= "0110"; puertos <= "0110"; sd1 <='1';
            when "0010" => leds <= "0011"; puertos <= "0011"; sd1 <='1';
            when others => leds <= "1001"; puertos <= "1001"; sd1 <='1'; ciclo1 <= ciclo1+1; sw1
<="0000"; --leds <= "0000"; puertos <= "0000";
            end case;
      end if;

   elsif (conter = 2 or sd2 ='1') and sd1 ='0'then
      if sw2 /= "0100" and ciclo2 /=limite AND S2 ='0' THEN sw2 <= sw2 + '1';
            case sw2 is
            when "0000" => leds <= "1001"; puertos <= "1001"; sd2 <='1';--1001
            when "0001" => leds <= "0011"; puertos <= "0011"; sd2 <='1';--0011
            when "0010" => leds <= "0110"; puertos <= "0110"; sd2 <='1';--0110
            when others => leds <= "1100"; puertos <= "1100"; sd2 <='1'; ciclo2 <= ciclo2+1; sw2
<= "0000"; sd2 <='1'; --leds <= "0000"; puertos <= "0000"; --1100
            end case;
      end if;
   else sw1 <="0000"; ciclo1 <=0; sw2 <="0000"; ciclo2 <=0;-- leds<="0000"; puertos<="0000";
   end if;

   if (ciclo1 = limite or ciclo2 = limite) then leds <= "0000"; puertos <= "0000"; sd1 <='0'; sd2
<='0'; end if;

end if;

   end process;

end Behavioral;
```

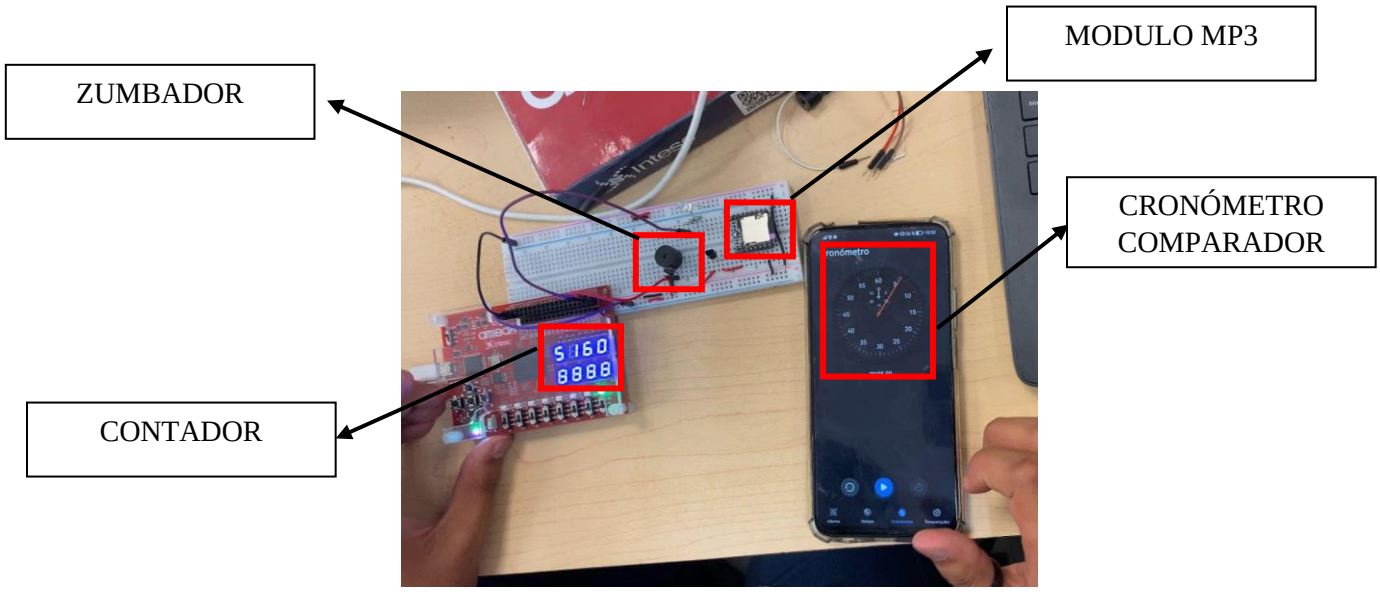
Restricciones.ucf

```
NET"LED(1)" LOC="N1";
NET"LED(0)" LOC="P1";
NET"CLK" LOC="E7";
NET"Entradas(5)" LOC="A7";
NET"Entradas(4)" LOC="B8";
net "S1" loc= "E4";
net "S2" loc= "F5";
NET"leds(1)" LOC="P11";
NET"leds(2)" LOC="P6";
NET"leds(3)" LOC="P8";
NET"leds(4)" LOC="P12";
NET"puertos(1)" LOC="A10";
NET"puertos(2)" LOC="C11";
NET"puertos(3)" LOC="B14";
NET"puertos(4)" LOC="B16";
```

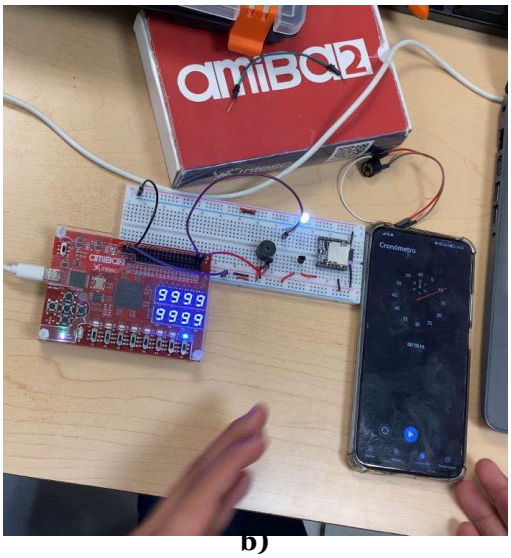
3. Implementar en TLD un contador de pulso en alto en ms con salida a display, el cual al presionar un botón (push) se visualiza el conteo en el display y cuando se suelta el botón se detiene en un valor, pero si se presiona otra vez continua el conteo. Se cuenta con un botón de reset para reiniciar 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 □ 3,333, 3,333 < G □ 6,666, 6,666 < B □ 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 □ 3,333, 3,333 < G □ 6,666, 6,666 < B □ 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.

En este punto se realizó un contador el cual cuando llegaba a 3.333 sonaba el zumbador, al igual que si pasaba por 6.666 y 9.999, en este último se activaba un archivo MP3, el cual estaba grabado en el módulo, con ayuda del cronometro, se verificaba el contador, cabe mencionar que en este último punto se debe tener en cuenta el error al iniciar y parar el cronometro.



a)



b)

Fig. 4. a), b), Comparador entre contador y cronómetro, con alamar por paso en 3.333, 6.666, 9.999.

CÓDIGOS

Contador.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
USE ieee.std_logic_unsigned.ALL;
-- Declaraciï¿½n de la entidad
ENTITY Contador IS
   GENERIC (freq : INTEGER := 15);
   PORT (
      UpDown : IN STD_LOGIC_VECTOR(1 DOWNTO 0); -- botones para subir y bajar los ms
      CLK : IN STD_LOGIC; -- reloj de 50MHz para la nexys 2 y 100MHz para nexys 3
      RESET : IN STD_LOGIC; -- reset
      SALED : OUT STD_LOGIC; -- salida del led testigo
      -- "abcdefgP"
      DISPLAY : OUT STD_LOGIC_VECTOR(7 DOWNTO 0); -- segmentos del display
      AN : OUT STD_LOGIC_VECTOR(3 DOWNTO 0); -- ï¿½nodos del display
      RGB: OUT STD_LOGIC_VECTOR(1 TO 3); -- salida a leds RGB
      BEEPtest : IN STD_LOGIC; -- botï¿½n de prueba para la salida a bocina
      BEEP : OUT STD_LOGIC; -- salida a bocina
  VOZ : OUT STD_LOGIC
   );
END Contador;
-- Declaraciï¿½n de la arquitectura
ARCHITECTURE Behavioral OF Contador IS
   -- Declaraciï¿½n de seï¿½ales de los divisores
   SIGNAL Conta_500us : INTEGER RANGE 1 TO 25_000 := 1; -- uso en el pulso de 1ms (pro.
divisor 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 (pro. divisor
ï¿½nodos)
   -- para la nexys 2 6250, si se usa la nexys 3 cambiar a 1250
   SIGNAL SAL_1ms, SAL_250us : STD_LOGIC := '0'; --igual que pulso y pulso1,
respectivamente
   -- Declaraciï¿½n de seï¿½ales de los contadores
   SIGNAL CONT : STD_LOGIC_VECTOR (15 DOWNTO 0) := (OTHERS => '0'); -- 16 bits
(proc. conteo)
   SIGNAL CONT2 : INTEGER := 0; -- cambia cont de 16 bits a entero (proc. conteo)
   -- Declaraciï¿½n de seï¿½ales de la asignaciï¿½n de U-D-C-UM
   SIGNAL P : STD_LOGIC_VECTOR (15 DOWNTO 0) := (OTHERS => '0'); -- asigna UNI,
DEC,CEN, MIL
   SIGNAL UNI, DEC, CEN, MIL : STD_LOGIC_VECTOR (3 DOWNTO 0) := (OTHERS =>
'0'); -- digitos unidades, decenas,
   -- centenas y unidad de millar
   -- Declaraciï¿½n de seï¿½ales de la multiplexaciï¿½n y asignaciï¿½n 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) := (OTHERS => '0'); -- sirve para
almacenar los valores del display
   -- Declaraciï¿½n de seï¿½ales de la base de tiempo
   SIGNAL PERIOD : INTEGER RANGE 0 TO 2499 := 0; -- periodo de 2.5 segundos (base de
tiempo en ms para el PWM)
   -- Declaraciï¿½n de seï¿½al para el BEEP
   SIGNAL sound : STD_LOGIC_VECTOR (freq DOWNTO 0) := (OTHERS => '0'); -- vector
para la generaciï¿½n del sonido beep
BEGIN
   ----------DIVISOR 1ms----------
   -- en este proceso se genera una seï¿½al "SAL_1ms" de 1ms de periodo para el conteo de ms
   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 alcanzï¿½ 0.5ms (nexys2)
            --if(Conta_500us = 50000) then -- pregunta si ya se alcanzï¿½ 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 generaciï¿½n de seï¿½al 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 llegï¿½ a cero mantiene el cero
               ELSE
                  CONT <= CONT - '1'; -- sino decrementa
               END IF;
            ELSIF UpDown = "10" THEN --incrementa
               IF (CONT >= "0010011100001111") THEN -- compara contra 9,999
                  -- (270F hex)
                  CONT <= CONT; -- si llego a 9999 mantiene 9999
               ELSE
                  CONT <= CONT + '1'; -- sino incrementa
               END IF;
            ELSE --cubre UpDown="00" y 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
      -- RGB
      IF CONT2 >= 0 AND CONT2 <= 3333 THEN
         RGB <= "100";

      ELSIF CONT2 > 3333 AND CONT2 <= 6666 THEN
         RGB <= "010";
      ELSIF CONT2 > 6666 AND CONT2 <= 9999 THEN
         RGB <= "001";
      ELSE
         RGB <= "000";
      END IF;
   END PROCESS;
   ----------------------------sound2BEEP-----------------------------
   -- BEEP <= sound(15)
   PROCESS (clk, sound(freq), BEEPtest, CONT2)
   BEGIN
      -- RGB
      IF (CLK'event AND CLK = '1') THEN
         sound <= sound + '1';
      END IF;
      -- BEEP
      IF (CONT2 >= 3325 AND CONT2 <= 3341) OR (CONT2 >= 6658 AND CONT2 <= 6674)
OR (CONT2 >= 9800 AND CONT2 <= 9899) THEN
         BEEP <= sound(freq);
      ELSE
         BEEP <= '0';
      END IF;
```

```
  IF (CONT2 >= 9900) THEN
VOZ <= '1';
ELSE
VOZ <= '0';
END IF;
   END PROCESS;

   --------------------PWM CONTROLADO POR CONTADOR DE ms/4-----------------------
   PROCESS (RESET, SAL_1ms, PERIOD)
   BEGIN
      IF (RESET = '1' OR PERIOD >= 2_499) 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
            -- if (PERIOD <= CONT) then SALED <='1'; -- Salida a led testigo
         ELSE
            SALED <= '0';
         END IF;
      END IF;
   END PROCESS; --fin del proceso PWM

   -    PROCESS (CONT)
      VARIABLE UM_C_D_U : STD_LOGIC_VECTOR(29 DOWNTO 0);
      --30 bits para separar las U.Millar-Centenas-Decenas-Unidades
   BEGIN
      --ciclo de inicializaciï¿½n
      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 asignaciï¿½n 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 ï¿½NODOS------------------
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 ï¿½nodos
--------------------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
```

Restricciones.ucf

```
//DISPLAYS
NET "DISPLAY(7)" LOC = "P16"; // a
NET "DISPLAY(6)" LOC = "P15"; // b
NET "DISPLAY(5)" LOC = "T15"; // c
NET "DISPLAY(4)" LOC = "T14"; // d
NET "DISPLAY(3)" LOC = "T13"; // e
NET "DISPLAY(2)" LOC = "R16"; // f
NET "DISPLAY(1)" LOC = "R15"; // g
NET "DISPLAY(0)" LOC = "R14"; // P
//ANODOS
NET "AN(3)" LOC = "E15"; // AN0
NET "AN(2)" LOC = "C15"; // AN1
NET "AN(1)" LOC = "D11"; // AN2
NET "AN(0)" LOC = "E11"; // AN3

//UpDown,RESET--PUSH BUTTON
NET "UpDown(1)" LOC = "J3"; // btn0 SUBE
#NET "UpDown(0)" LOC = "H3"; // btn1 BAJA
NET "RESET" LOC = "L1"; // btn2 RST
NET "BEEPtest" LOC = "M2"; // btn3 beep test
//CLK reloj 50MHz
NET "CLK" LOC = "E7";
//SALED salida a led testigo
NET "SALED" LOC = "D8"; // LD0
//RGB salida a leds de rango
NET "RGB(1)" LOC = "P12"; // JA4
NET "RGB(2)" LOC = "N12"; // JB4
NET "RGB(3)" LOC = "T12"; // JC4

//BEEP salida a bocina
NET "BEEP" LOC = "B16"; // JA10
NET "VOZ" LOC = "B14"; // JA10
```

Desafío 2 (sustituye los puntos 4 y 5). Implemente un posicionador de cámara 2DOF (láser, arma de agua, etc.) utilizando 2 servomotores y un joystick (analógico o digital). Reportar códigos (HDL y UCF), fotos y video.

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.

En este ejercicio con ayuda del joystick cuyas salidas analógicas fueron digitalizadas mediante un circuito con OPAMP TL084, se debía mandar un voltaje de entrada a la FPGA de 3.3V, lo que se logró con optoacoplador, entonces, se mandan los movimientos al mecanismo de servomotores, el cual tiene una carga (lámpara), cabe mencionar que en el circuito se cuenta con unos LED bicolor que señalan la dirección de movimiento del joystick, al igual que se mueve la carga.

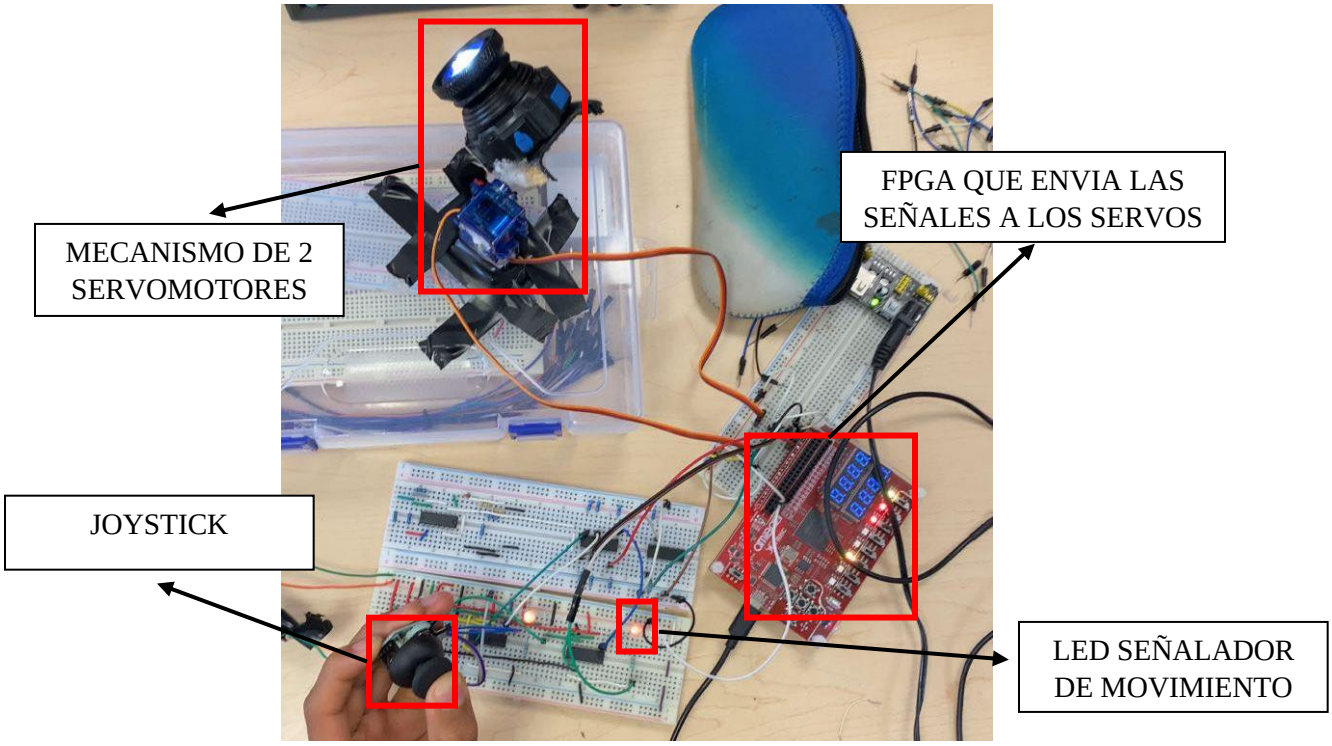
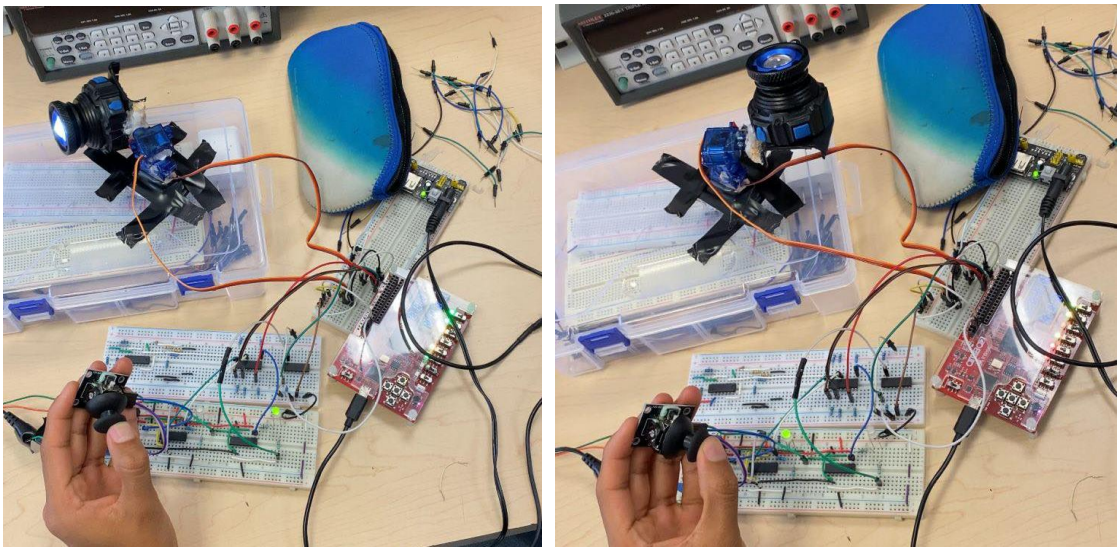


Fig. 5. Posicionador de cámara 2DOF



a) **b)**

Fig. 6. Movimiento del joystick con respuesta en servomotores

CÓDIGOS

```
LIBRARY IEEE;
USE IEEE.numeric_std.ALL;
USE ieee.std_logic_arith.ALL;
USE ieee.std_logic_unsigned.ALL;

ENTITY TDOF IS
   GENERIC (
      Max : NATURAL := 500000
      --min : INTEGER := 18332; --valor mÃ-nimo del contador para el tiempo en alto
      --max : INTEGER := 121686; --valor mÃ¡ximo del contador para el tiempo en alto
      --inc : INTEGER := 3334 -- incremento para el tiempo en alto
   );
   PORT (
      --Encoder
      CLK : IN STD_LOGIC; --reloj 50 Mhz
      XRin : IN STD_LOGIC; --XRin Xright
      XLin : IN STD_LOGIC; --XLin Xleft
      YUin : IN STD_LOGIC; --Yup
      YDin : IN STD_LOGIC; --YDown
      servomotor1, servomotor2 : OUT STD_LOGIC; --Salida a pwm para controlar posicion de
servos

      LedLim1, LedLim2 : OUT STD_LOGIC_VECTOR(5 DOWNTO 0); --Salida a leds
indicadores del limite actual
      reset : IN STD_LOGIC --Boton de reset asincrono

   );
END ENTITY TDOF;
```

```
ARCHITECTURE Behavioral OF TDOF IS
   --signals
   --s
   CONSTANT limMinS1 : INTEGER :=50000;
   CONSTANT limMaxS1 : INTEGER :=100000;

   CONSTANT limMinS2 : INTEGER :=24500;
   CONSTANT limMaxS2 : INTEGER :=120000;

   CONSTANT NoDivS1 : INTEGER :=70;
   CONSTANT NoDivS2 : INTEGER :=60;

   Signal IncS1 : INTEGER;
   Signal IncS2 : INTEGER;


   SIGNAL PWM_Count1 : INTEGER RANGE 1 TO Max;--500000;
   SIGNAL PWM_Count2 : INTEGER RANGE 1 TO Max;--500000;

   ---------------------------------------------------------------------------
   CONSTANT lim_deb : INTEGER := 8_999_999;

   ---------------------------------------------------------------------------
   --Encoder
   SIGNAL sclk : INTEGER RANGE 0 TO lim_deb := 0;--STD_LOGIC_VECTOR (8
DOWNTO 0);
   SIGNAL sampledXR, sampledXL : STD_LOGIC;
   SIGNAL XRout, XLout : STD_LOGIC;
   SIGNAL sampledYU, sampledYD : STD_LOGIC;
   SIGNAL YUout, YDout : STD_LOGIC;
   SIGNAL lim1, lim2 : STD_LOGIC_VECTOR(5 DOWNTO 0) := "000000";
   SIGNAL limInt1, limInt2 : INTEGER RANGE 0 TO 59 := 0;
   SIGNAL q, Xs, Ys : STD_LOGIC;

   SIGNAL conta_1250us : INTEGER RANGE 1 TO 5000000 := 1; -- pulso1 de 1250us@400Hz
(0.25ms) 62500
   SIGNAL SAL_400Hz : STD_LOGIC; -- reloj de 400Hz

BEGIN
   ---------------------------------------------------------------------------
   IncS1 <= (limMaxS1-limMinS1)/NoDivS1;
   IncS2 <= (limMaxS2-limMinS2)/NoDivS2;


   ---------------------------------------------------------------------------
   GeneradorPWM1 : PROCESS (clk)
   BEGIN
   ---------------------------------------------------------------------------
   -- Lim = (180/3)=60
   -- min = 24100
   -- max = 110k
```

```
   -- inc = (24100-110000)/60=1432
   ---------------------------------------------------------------------------
      IF rising_edge(clk) THEN
         PWM_Count1 <= PWM_Count1 + 1;
      END IF;

      IF PWM_Count1 <= limMinS1 + (limInt1 * IncS1) THEN --min+(Lim*incremento)
incremento = 7666
         servomotor1 <= '1';

      ELSE
         servomotor1 <= '0';


      END IF;

   END PROCESS GeneradorPWM1;
   ---------------------------------------------------------------------------
   ---------------------------------------------------------------------------
   GeneradorPWM2 : PROCESS (clk)
   BEGIN

      IF rising_edge(clk) THEN
         PWM_Count2 <= PWM_Count2 + 1;
      END IF;

      IF PWM_Count2 <= limMinS2 + (limInt2 * IncS2) THEN --min+(Lim*incremento)
incremento = 7666
         servomotor2 <= '1';

      ELSE
         servomotor2 <= '0';

      END IF;

   END PROCESS GeneradorPWM2;
   ---------------------------------------------------------------------------
   ControlLim : PROCESS (clk)
   BEGIN
      IF (clk'event AND clk = '1') THEN-- si no existe un reset y el cambio de clk=1
         q <= XRout;-- funcionamiento normal del ffd
      END IF;

      Xs <= XRout OR XLout;
      Ys <= YDout OR YUout;

      IF RESET = '0' THEN
         IF rising_edge(SAL_400Hz) then
            IF (Xs = '1') THEN
               IF (XRin = '1' AND lim1 < "111100" AND limInt1 < 60) THEN --60
```

```
            lim1 <= lim1 + '1';
            limInt1 <= limInt1 + 1;
          ELSIF (XLin = '1' AND lim1 >= "000001" AND limInt1 >= 1) THEN
            lim1 <= lim1 - '1';
            limInt1 <= limInt1 - 1;
          END IF;
        END IF;
        IF (Ys = '1') THEN
          IF (YUin = '1' AND lim2 < "111100" AND limInt2 < 60) THEN --60
            lim2 <= lim2 + '1';
            limInt2 <= limInt2 + 1;
          ELSIF (YDin = '1' AND lim2 >= "000001" AND limInt2 >= 1) THEN
            lim2 <= lim2 - '1';
            limInt2 <= limInt2 - 1;
          END IF;
        END IF;
      END IF;
    ELSE
      lim1 <= "000000";
      limInt1 <= 0;
      lim2 <= "000000";
      limInt2 <= 0;
    END IF;

    LedLim1 <= lim1;
    LedLim2 <= lim2;

  END PROCESS;
  ---------------------------------------------------------------------------
  ---------------------------------------------------------------------------
  debouncer_botones : PROCESS (clk, XRin, XLin) BEGIN

    IF clk'event AND clk = '1' THEN
      sampledXR <= XRin;
      sampledXL <= XLin;
      sampledYU <= YUin;
      sampledYD <= YDin;

      -- clock is divided to 1MHz
      -- samples every 1uS to check if the input is the same as the sample
      -- if the signal is stable, the debouncer should output the signal
      IF sclk = lim_deb THEN

        -- XRout
        IF sampledXR = XRin THEN
          XRout <= XRin;
        END IF;
        --XLout
        IF sampledXL = XLin THEN
          XLout <= XLin;
        END IF;
```

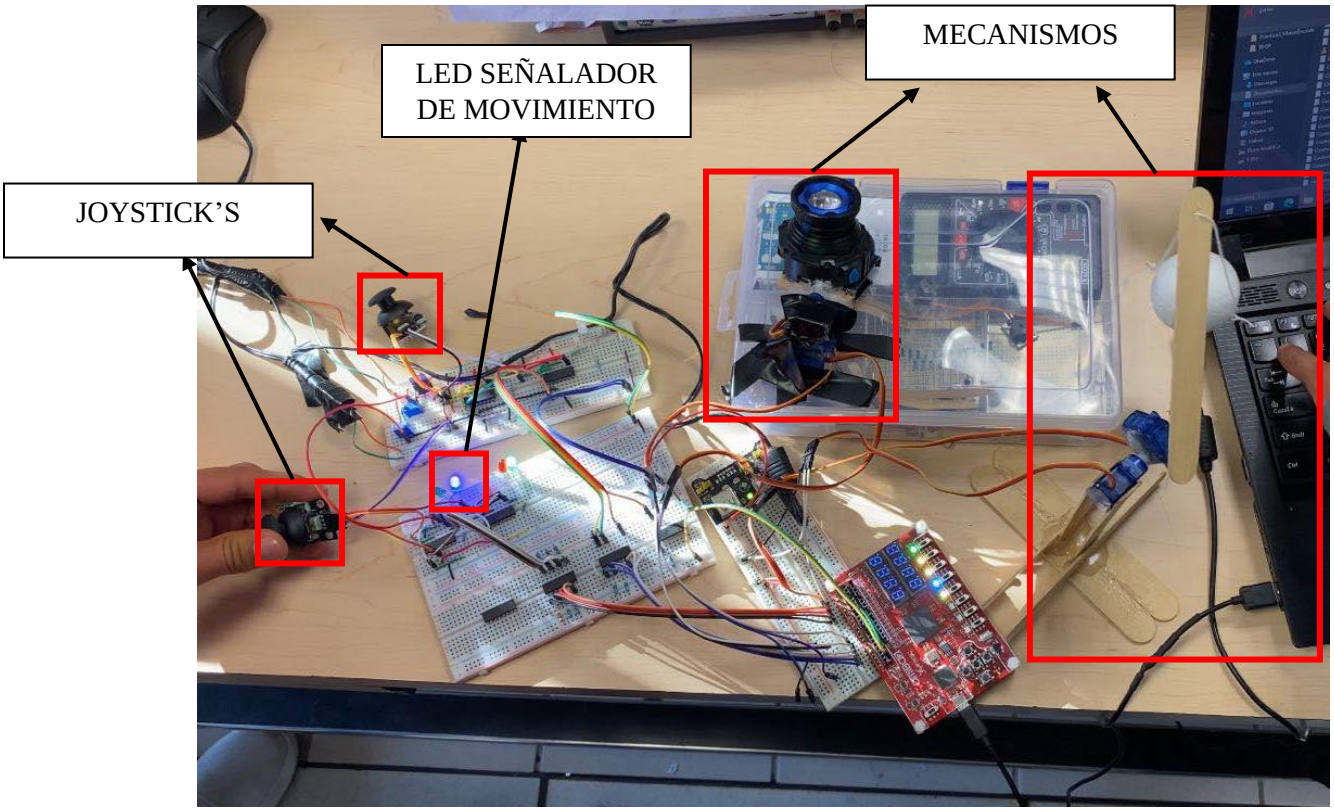
```
            -- YUout
            IF sampledYU = YUin THEN
               YUout <= YUin;
            END IF;
            --YDout
            IF sampledYD = YDin THEN
               YDout <= YDin;
            END IF;

            sclk <= 0;
         ELSE
            sclk <= sclk + 1;
         END IF;
      END IF;
   END PROCESS;
   ---------------------------------------------------------------------------

   CLK_400 : PROCESS (CLK) BEGIN
      IF (rising_edge(CLK)) THEN
         IF (conta_1250us = 2000000) THEN --cuenta 1250ms (50MHz=62500) 62500*20us =
1.25ms 1/(2*1.25ms)=400Hz
            SAL_400Hz <= NOT(SAL_400Hz); --genera un barrido de 2.5ms
            conta_1250us <= 1;
         ELSE
            conta_1250us <= conta_1250us + 1;
         END IF;
      END IF;
   END PROCESS;

END Behavioral;
```

Link del video: https://youtube.com/shorts/Yzh_ec2D7Hw

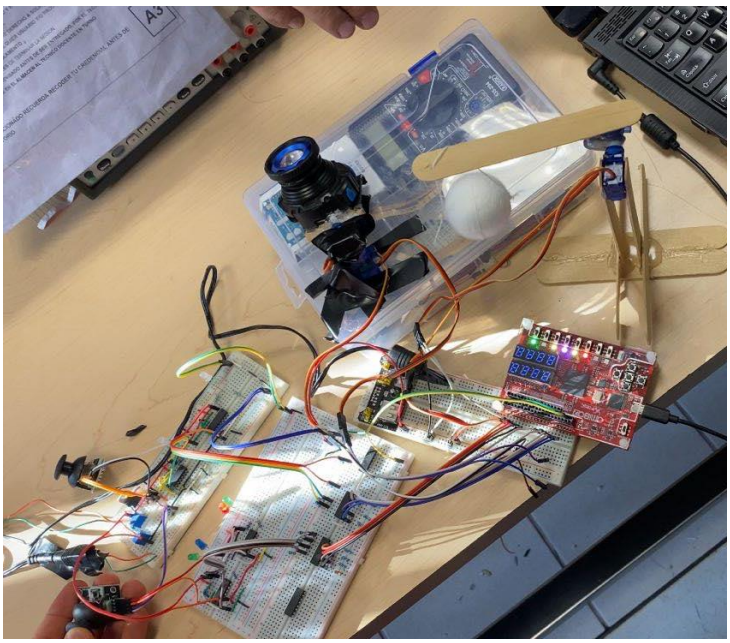
Desafío 3 (sustituir el punto 6). Implementar un controlador de posicionamiento de 2 cámaras con 2DOF cada una, utilizando 2 joysticks. Reportar códigos (HDL y UCF), fotos y video.

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.

Para este punto se duplicó el reto dos, es decir el circuito para digitalizar la señal del joystick se realizó para los dos que se necesitaban, al igual que la programación, realizando un sistema de carga con una lámpara que simulaba la cámara, y el segundo par de servos se pusieron como un sistema de grúa con bola de demolición.



a)



b)

Fig. 7. Posicionamiento de 2 cámaras con 2DOF con 2 servomotores cada uno.

CÓDIGOS

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
USE ieee.std_logic_arith.ALL;
USE ieee.std_logic_unsigned.ALL;

ENTITY TDOF IS
   GENERIC (
      Max : NATURAL := 500000
      --min : INTEGER := 18332; --valor mÃ-nimo del contador para el tiempo en alto
      --max : INTEGER := 121686; --valor mÃ¡ximo del contador para el tiempo en alto
      --inc : INTEGER := 3334 -- incremento para el tiempo en alto
   );
   PORT (
      --Encoder
      CLK : IN STD_LOGIC; --reloj 50 Mhz

      XRin1 : IN STD_LOGIC; --XRin1 Xright
      XLin1 : IN STD_LOGIC; --XLin1 Xleft
      YUin1 : IN STD_LOGIC; --Yup
      YDin1 : IN STD_LOGIC; --YDown
      servomotor1, servomotor2 : OUT STD_LOGIC; --Salida a pwm para controlar posicion de
servos

      XRin2 : IN STD_LOGIC; --XRin1 Xright
      XLin2 : IN STD_LOGIC; --XLin1 Xleft
      YUin2 : IN STD_LOGIC; --Yup
      YDin2 : IN STD_LOGIC; --YDown
      servomotor11, servomotor22 : OUT STD_LOGIC; --Salida a pwm para controlar posicion
de servos

      LedLim1, LedLim2 : OUT STD_LOGIC_VECTOR(5 DOWNTO 0); --Salida a leds
indicadores del limite actual
      LedLim11, LedLim22 : OUT STD_LOGIC_VECTOR(5 DOWNTO 0); --Salida a leds
indicadores del limite actual

      reset : IN STD_LOGIC --Boton de reset asincrono

   );
END ENTITY TDOF;

ARCHITECTURE Behavioral OF TDOF IS
   --ssalida 1
   CONSTANT limMinS1 : INTEGER := 50000;
   CONSTANT limMaxS1 : INTEGER := 100000;

   CONSTANT limMinS2 : INTEGER := 24500;
   CONSTANT limMaxS2 : INTEGER := 120000;
```

```
    CONSTANT NoDivS1 : INTEGER := 60;
    CONSTANT NoDivS2 : INTEGER := 60;

    SIGNAL IncS1 : INTEGER;
    SIGNAL IncS2 : INTEGER;

    SIGNAL PWM_Count1 : INTEGER RANGE 1 TO Max;--500000;
    SIGNAL PWM_Count2 : INTEGER RANGE 1 TO Max;--500000;
    ---------------------------------------------------------------------------
    --Salida 2
    CONSTANT limMinS11 : INTEGER := 50000;
    CONSTANT limMaxS11 : INTEGER := 100000;

    CONSTANT limMinS22 : INTEGER := 24500;
    CONSTANT limMaxS22 : INTEGER := 120000;

    CONSTANT NoDivS11 : INTEGER := 60;
    CONSTANT NoDivS22 : INTEGER := 60;

    SIGNAL IncS11 : INTEGER;
    SIGNAL IncS22 : INTEGER;

    SIGNAL PWM_Count11 : INTEGER RANGE 1 TO Max;--500000;
    SIGNAL PWM_Count22 : INTEGER RANGE 1 TO Max;--500000;

    ---------------------------------------------------------------------------
    CONSTANT lim_deb : INTEGER := 8_999_999;

    ---------------------------------------------------------------------------
    --Encoder
    SIGNAL sclk : INTEGER RANGE 0 TO lim_deb := 0;--STD_LOGIC_VECTOR (8
DOWNTO 0);
    --SALIDA 1
    SIGNAL sampledXR1, sampledXL1 : STD_LOGIC;
    SIGNAL XRout1, XLout1 : STD_LOGIC;
    SIGNAL sampledYU1, sampledYD1 : STD_LOGIC;
    SIGNAL YUout1, YDout1 : STD_LOGIC;

    SIGNAL lim1Salida1, lim2Salida1 : STD_LOGIC_VECTOR(5 DOWNTO 0) := "000000";
    SIGNAL limInt1Salida1, limInt2Salida1 : INTEGER RANGE 0 TO 59 := 0;
    SIGNAL Xs1, Ys1 : STD_LOGIC;

    --SALIDA 2

    SIGNAL sampledXR2, sampledXL2 : STD_LOGIC;
    SIGNAL XRout2, XLout2 : STD_LOGIC;
    SIGNAL sampledYU2, sampledYD2 : STD_LOGIC;
    SIGNAL YUout2, YDout2 : STD_LOGIC;

    SIGNAL lim1Salida2, lim2Salida2 : STD_LOGIC_VECTOR(5 DOWNTO 0) := "000000";
```

```
   SIGNAL limInt1Salida2, limInt2Salida2 : INTEGER RANGE 0 TO 59 := 0;
   SIGNAL Xs2, Ys2 : STD_LOGIC;

   SIGNAL conta_1250us : INTEGER RANGE 1 TO 5000000 := 1; -- pulso1 de 1250us@400Hz
(0.25ms) 62500
   SIGNAL SAL_400Hz : STD_LOGIC; -- reloj de 400Hz

BEGIN
   ---------------------------------------------------------------------------
   --salida 1
   IncS1 <= (limMaxS1 - limMinS1)/NoDivS1;
   IncS2 <= (limMaxS2 - limMinS2)/NoDivS2;
   --salida 2
   IncS11 <= (limMaxS11 - limMinS11)/NoDivS11;
   IncS22 <= (limMaxS22 - limMinS22)/NoDivS22;

   ---------------------------------------------------------------------------
   GeneradorPWM1 : PROCESS (clk)
   BEGIN
      ---------------------------------------------------------------------------
      -- Lim = (180/3)=60
      -- min = 24100
      -- max = 110k
      -- inc = (24100-110000)/60=1432
      ---------------------------------------------------------------------------
      --SALIDA 1
      IF rising_edge(clk) THEN
         PWM_Count1 <= PWM_Count1 + 1;
         PWM_Count11 <= PWM_Count11 + 1;
      END IF;
      IF PWM_Count1 <= limMinS1 + (limInt1Salida1 * IncS1) THEN --min+(Lim*incremento)
incremento = 7666
         servomotor1 <= '1';
      ELSE
         servomotor1 <= '0';
END IF;
      --SALIDA 2
      IF PWM_Count11 <= limMinS11 + (limInt1Salida2 * IncS11) THEN --
min+(Lim*incremento) incremento = 7666
         servomotor11 <= '1';
      ELSE
         servomotor11 <= '0';
      END IF;

   END PROCESS GeneradorPWM1;
   ---------------------------------------------------------------------------
   ---------------------------------------------------------------------------
   GeneradorPWM2 : PROCESS (clk)
   BEGIN
      --SALIDA 1
      IF rising_edge(clk) THEN
```

```
        PWM_Count2 <= PWM_Count2 + 1;
        PWM_Count22 <= PWM_Count22 + 1;
      END IF;
      IF PWM_Count2 <= limMinS2 + (limInt2Salida1 * IncS2) THEN --min+(Lim*incremento)
incremento = 7666
        servomotor2 <= '1';
      ELSE
        servomotor2 <= '0';
      END IF;
      --SALIDA
      IF PWM_Count22 <= limMinS22 + (limInt2Salida2 * IncS22) THEN --
min+(Lim*incremento) incremento = 7666
        servomotor22 <= '1';
      ELSE
        servomotor22 <= '0';
      END IF;

   END PROCESS GeneradorPWM2;
   -------------------------------------------------------------------------------
   ControlLim : PROCESS (clk)
   BEGIN

      Xs1 <= XRout1 OR XLout1;
      Ys1 <= YDout1 OR YUout1;

      Xs2 <= XRout2 OR XLout2;
      Ys2 <= YDout2 OR YUout2;

      IF RESET = '0' THEN
        IF rising_edge(SAL_400Hz) THEN
          IF (Xs1 = '1') THEN
            IF (XRin1 = '1' AND lim1Salida1 < "111100" AND limInt1Salida1 < 60) THEN --
60
              lim1Salida1 <= lim1Salida1 + '1';
              limInt1Salida1 <= limInt1Salida1 + 1;
            ELSIF (XLin1 = '1' AND lim1Salida1 >= "000001" AND limInt1Salida1 >= 1)
THEN
              lim1Salida1 <= lim1Salida1 - '1';
              limInt1Salida1 <= limInt1Salida1 - 1;
            END IF;
          END IF;
          IF (Ys1 = '1') THEN
            IF (YUin1 = '1' AND lim2Salida1 < "111100" AND limInt2Salida1 < 60) THEN --
60
              lim2Salida1 <= lim2Salida1 + '1';
              limInt2Salida1 <= limInt2Salida1 + 1;
            ELSIF (YDin1 ='1' AND lim2Salida1 >= "000001" AND limInt2Salida1 >= 1)
THEN
              lim2Salida1 <= lim2Salida1 - '1';
              limInt2Salida1 <= limInt2Salida1 - 1;
            END IF;
```

```
            END IF;
        END IF;
        --salida 2

        IF (Xs2 = '1') THEN
            IF (XRin2 = '1' AND lim1Salida2 < "111100" AND limInt1Salida2 < 60) THEN --60
                lim1Salida2 <= lim1Salida2 + '1';
                limInt1Salida2 <= limInt1Salida2 + 1;
            ELSIF (XLin1 = '1' AND lim1Salida2 >= "000001" AND limInt1Salida2 >= 1) THEN
                lim1Salida2 <= lim1Salida2 - '1';
                limInt1Salida2 <= limInt1Salida2 - 1;
            END IF;
        END IF;
        IF (Ys2 = '1') THEN
            IF (YUin2 = '1' AND lim2Salida2 < "111100" AND limInt2Salida2 < 60) THEN --60
                lim2Salida2 <= lim2Salida2 + '1';
                limInt2Salida2 <= limInt2Salida2 + 1;
            ELSIF (YDin1 ='1' AND lim2Salida2 >= "000001" AND limInt2Salida2 >= 1) THEN
                lim2Salida2 <= lim2Salida2 - '1';
                limInt2Salida2 <= limInt2Salida2 - 1;
            END IF;
        END IF;

    ELSE
        lim1Salida1 <= "000000";
        limInt1Salida1 <= 0;
        lim2Salida1 <= "000000";
        limInt2Salida1 <= 0;

        lim1Salida2 <= "000000";
        limInt1Salida2 <= 0;
        lim2Salida2 <= "000000";
        limInt2Salida2 <= 0;

    END IF;

    LedLim1 <= lim1Salida1;
    LedLim2 <= lim2Salida1;

    LedLim11 <= lim1Salida2;
    LedLim22 <= lim2Salida2;

END PROCESS;
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
debouncer_botones : PROCESS (clk, XRin1, XLin1) BEGIN

    IF clk'event AND clk = '1' THEN
        sampledXR1 <= XRin1;
        sampledXL1 <= XLin1;
```

```
sampledYU1 <= YUin1;
sampledYD1 <= YDin1;

sampledXR2 <= XRin2;
sampledXL2 <= XLin2;
sampledYU2 <= YUin2;
sampledYD2 <= YDin2;

-- clock is divided to 1MHz
-- samples every 1uS to check if the input is the same as the sample
-- if the signal is stable, the debouncer should output the signal
IF sclk = lim_deb THEN

   -- XRout1
   IF sampledXR1 = XRin1 THEN
      XRout1 <= XRin1;
   END IF;
   --XLout1
   IF sampledXL1 = XLin1 THEN
      XLout1 <= XLin1;
   END IF;
   -- YUout1
   IF sampledYU1 = YUin1 THEN
      YUout1 <= YUin1;
   END IF;
   --YDout1
   IF sampledYD1 =
      YDin1 THEN
      YDout1 <= YDin1;
   END IF;

   -- XRout2
   IF sampledXR2 = XRin2 THEN
      XRout2 <= XRin2;
   END IF;
   --XLout2
   IF sampledXL2 = XLin2 THEN
      XLout2 <= XLin2;
   END IF;
   -- YUout2
   IF sampledYU2 = YUin2 THEN
      YUout2 <= YUin2;
   END IF;
   --YDout2
   IF sampledYD2 = YDin2 THEN
      YDout2 <= YDin2;
   END IF;


   sclk <= 0;
```

```
        ELSE
          sclk <= sclk + 1;
        END IF;
      END IF;
  END PROCESS;
  ----------------------------------------------------------------------------

  CLK_400 : PROCESS (CLK) BEGIN
    IF (rising_edge(CLK)) THEN
      IF (conta_1250us = 2000000) THEN --cuenta 1250ms (50MHz=62500) 62500*20us =
1.25ms 1/(2*1.25ms)=400Hz
        SAL_400Hz <= NOT(SAL_400Hz); --genera un barrido de 2.5ms
        conta_1250us <= 1;
      ELSE
        conta_1250us <= conta_1250us + 1;
      END IF;
    END IF;
  END PROCESS;

END Behavioral;
```

Link del video: https://youtube.com/shorts/uVJzHW6Rz-k

CONCLUSIONES

GARCÍA MORALES JOSÉ MANUEL

Gracias a esta práctica se pudo comprender de la mejor manera el cómo funcionan el acondicionamiento de una señal, esto visto en la materia de sensores, ya que para poder enviar las señales que moverían al servomotor, se tenía que hacer uso de salidas analógicas, las cuales ya debían de estar acondicionadas a el voltaje de entrada de la FPGA, para poder realizar los movientes del joystick hacu el servomotor. Otra de las cosas que me pareció importante fue el uso de sensores ópticos, los cuales limitaban el giro de un motor a pasos, dicho experimento nos serviría mucho en la implementación de un animatron.

LEMUS OLIVARES ALEXIS

En esta práctica se usaron salidas analógicas, que fueron digitalizadas mediante circuitos con OPAMP, ya acondicionadas con optoacoplador, para ser ingresadas a la fpga y generar el movimiento de servomotores.

Así mismo se retomó el movimiento del motor a pasos unipolar, en este caso con Encoder, aplicando la acción de dos sensores que limitaban el movimiento del motor en ciertos puntos.

Finalmente, la creación de un contador

MORALES FERNÁNDEZ ALBERT

Para el desarrollo de esta práctica aplicamos conocimientos previos de la unidad de aprendizaje de circuitos lógicos como son el control de motores a pasos, circuitos combinacionales o secuenciales, así como de electrónica para desarrollar el circuito que realiza la función convertidor analógico digital con ayuda de amplificadores operacionales para recibir los valores del joystick, además el uso de sensores infrarrojos para colocar límites al movimiento de un motor a pasos unipolar controlado por un encoder.

LAZCANO MALAGON CHRISTIAN GIANCARLO

Gracias a esta practica se pudo aplicar el movimiento limitado de un motor a pasos unipolar con ayuda de dos sensores, además con los servomotores se pudo realizar un circuito útil que puede ser utilizado en la industria como ensamblador, el contador puede ayudar a tomar el tiempo de alguna secuencia que se este realizando y de esta manera alertar cuando haya terminado

Con todo esto se pudo comprender como funcionan los circuitos secuencias así como también el acondicionamiento de una señal para su posterior uso

BIBLIOGRAFÍA

- https://uelectronics.com/producto/servomotor-sg90-rc-9g/
- 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 octubre-2022, disponible en http://www.boletin.upiita.ipn.mx/index.php/ciencia/216 Prácticas de Laboratorio de Dispositivos Lógicos Programables .37 cyt-numero-34/122-control-de-un-servomotor-con-un-coder-mecanico-rotatorio-utilizando-vhdl-y-la-nexys-ii