***Laboratorio 4 – Sensor Ultrasónico HC-SR0X***

Objetivos del laboratorio son conocer:

* Las interfaces del HC-SR0X hacia el FPGA
* Realizar un módulo reutilizable de VHDL para el HSCR04
* Validar el código en el test bench antes
* Implementar en Hardware

***Introducción***

Por el momento se ha visto solamente interfaces a circuitos digitales sencillos. No hemos abarcado en medida elementos sensores controlados. En este apartado podremos realizar la codificación en VHDL de un sensor de distancia, el HC-SR0X.

Este sensor es un elemento ultrasónico que usa la función de sonar para determinar la distancia a un objeto como lo realizan los delfines. El HC-SR0X ofrece una medición de no contacto a rangos de detección con alta precisión y mediciones estables en un empaquetado fácil de utilizar. Normalmente este dota de 2cm a 400 cm o 1 pulgada a 13 pies. Esta operación de este sensor no es afectada por la luz solar o material oscuro como los medidores de Sharp. El empaquetado comprende un transmisor ultrasónico y un receptor ultrasónico.



Figura 1. Método de Operación del HC-SR0X. Efecto de ecolocación.

Las características de este equipo, o por lo menos del HC-SR04 es:

Tabla 1. Rango de Operación del Sensor

|  |  |
| --- | --- |
| Descripción | Rango |
| Fuente de Alimentación | +5V DC |
| Cociente de Corriente | < 2mA |
| Corriente de Operación | 15mA |
| Ángulo de sensado | < 15° |
| Rango de Distancia | 2 cm a 400 cm / 1pie a 13pies |
| Resolución | 0.3 cm |
| Ángulo de Medición | 30° |
| Ancho de pulso de inicialización (trigger) | 10μS |
| Dimensiones | 45mm x 20mm x 15mm |

***Dispositivo Ultrasónico HC-SR0X***

El dispositivo es prácticamente sencillo. Consta de cuatro (4) terminales. Alimentación, señal de disparo (inicio de transmisión), eco y tierra.

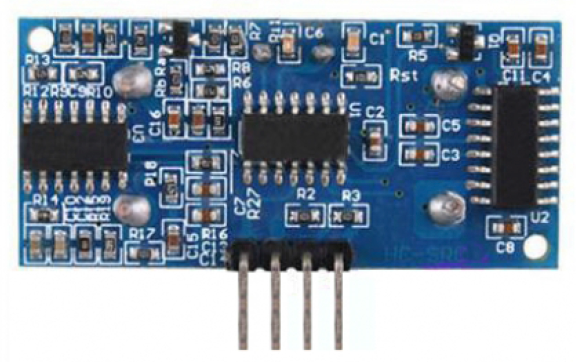
 

Figura 2. Vistas Frontal y Trasera del Sensor.

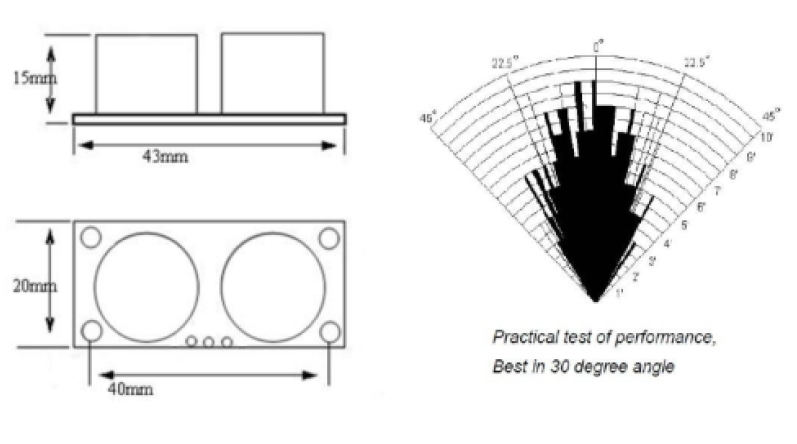


Figura 3. Dimensiones y ángulo de operación del sensor.

***Modo de Operación***

Para realizar las mediciónes de este módulo necesitamos realizar los siguientes pasos:

* Primeramente enviar un pulso alto de +5V por alrededor de unos 10μs.
* Cuando esto ocurra, el sensor comenzará a transmitir una secuencia de 8 ciclos ultrasónicos a rango de 40kHz y esperará para la reflexión de la ráfaga ultrasónica.
* Cuando el sensor detecte todo lo requerido para medir la distancia enviará por el pin de eco un retardo en proporción a la distancia.
* Para obtener la distancia se debe medir el pulso de encendido del pin de eco

La ecuación que carácteriza la medición del sensor es:

O puede utilizar la velocidad del sonido que es 340 m/s y realizar los cálculos, así la ecuación de distancia será el doble de la manejada.

El diagrama de tiempo se muestra a continuación



Figura 4. Diagrama de Tiempo del sensor.

Debajo se da una idea de implementación del código en VHDL y del Test Bench. ***Haga las modificaciones para conseguir validez del sensor. Al conectar al FPGA conecte todos los cables y utilice una resistencia de 120 ohms entre ECHO y la terminal del FPGA.***

***Implementación de Código de Entidad en VHDL***

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

ENTITY HCSR0X IS

GENERIC(INPUT\_CLK : INTEGER := 1000000; --- input clock in Hz

DIVIDER : INTEGER := 10; --- how much pulses for 1 us

MAX\_CM\_DIST : INTEGER := 500 --- CM = MAX\_CM\_DIST/100

);

PORT(CLK : IN STD\_LOGIC;

ECHO : IN STD\_LOGIC;

START\_TRIG : IN STD\_LOGIC;

TRIG : OUT STD\_LOGIC;

DISTANCE : OUT INTEGER

);

END HCSR0X;

ARCHITECTURE Bhvl OF HCSR0X IS

CONSTANT MHZ\_1 : INTEGER := 1000000;

CONSTANT CNTR\_MAX\_1US : INTEGER := INPUT\_CLK/(MHZ\_1 \* DIVIDER);

CONSTANT CNTR\_MAX\_10US : INTEGER := 10;

CONSTANT CNTR\_MAX\_100US : INTEGER := 100;

SIGNAL CNTR\_1US : INTEGER := 0 ;

SIGNAL CNTR\_10US : INTEGER := 0 ;

SIGNAL CNTR\_100US : INTEGER := 0 ;

SIGNAL PULSE\_WIDTH : INTEGER := 0 ;

SIGNAL CNTR\_1US\_FLAG : STD\_LOGIC := '0';

SIGNAL CNTR\_10US\_FLAG : STD\_LOGIC := '0';

SIGNAL CNTR\_100US\_FLAG : STD\_LOGIC := '0';

SIGNAL FLAG : STD\_LOGIC := '0';

BEGIN

--- Internal Signal 1us

PFLAG:

PROCESS(START\_TRIG)

BEGIN

IF RISING\_EDGE(START\_TRIG) THEN

FLAG <= '1';

END IF;

END PROCESS;

--- Internal Signal 1us

CNTR1US:

PROCESS(CLK)

BEGIN

IF RISING\_EDGE(CLK) AND FLAG = '1' THEN

CNTR\_1US <= CNTR\_1US + 1;

CNTR\_1US\_FLAG <= '0';

IF CNTR\_1US = CNTR\_MAX\_1US THEN

CNTR\_1US <= 0;

CNTR\_1US\_FLAG <= '1';

END IF;

END IF;

END PROCESS;

--- Internal Signal 10us

CNTR10US:

PROCESS(CNTR\_1US\_FLAG)

BEGIN

IF RISING\_EDGE(CNTR\_1US\_FLAG) AND FLAG = '1' THEN

TRIG <= '1';

CNTR\_10US <= CNTR\_10US + 1;

CNTR\_10US\_FLAG <= '0';

IF CNTR\_10US = CNTR\_MAX\_10US THEN

CNTR\_10US <= 0;

CNTR\_10US\_FLAG <= '1';

END IF;

END IF;

END PROCESS;

--- Internal Signal 10us

CNTR100US:

PROCESS(CNTR\_10US\_FLAG)

BEGIN

IF RISING\_EDGE(CNTR\_10US\_FLAG) AND FLAG = '1' THEN

CNTR\_100US <= CNTR\_100US + 1;

CNTR\_100US\_FLAG <= '0';

IF CNTR\_100US = CNTR\_MAX\_100US THEN

CNTR\_100US <= 0;

CNTR\_100US\_FLAG <= '1';

TRIG <= '0';

FLAG <= '0';

END IF;

END IF;

END PROCESS;

-- Measure Time of Signal

MEASURE:

PROCESS(CLK, CNTR\_1US\_FLAG)

BEGIN

IF RISING\_EDGE(CLK) AND RISING\_EDGE(CNTR\_1US\_FLAG) THEN

IF ECHO = '1' THEN

PULSE\_WIDTH <= PULSE\_WIDTH + 1;

ELSE

PULSE\_WIDTH <= 0;

END IF;

END IF;

END PROCESS;

DISTANCE <= PULSE\_WIDTH/58; -- Distance in cm

END Bhvl;

***Implementación del Test Bench***

-- Definicion de librería

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

-- Creamos una entidad de prueba

entity hcsr0xTB is

end hcsr0xTB;

-- Inicio de la arquitectura del test bench

architecture testbench of hcsr0xTB is

-- Definimos el componente

component HCSR0X is

GENERIC(INPUT\_CLK : INTEGER; --- input clock in Hz

DIVIDER : INTEGER; --- how much pulses for 1 us

MAX\_CM\_DIST : INTEGER --- CM = MAX\_CM\_DIST/100

);

end component HCSR0X;

-- Creación de señales

constant INPUT\_CLK : INTEGER := 100000000;

constant DIVIDER : INTEGER := 10;

constant MAX\_CM\_DIST : INTEGER := 500;

constant T : TIME := 1 us;

signal CLK : STD\_LOGIC := '0';

signal ECHO : STD\_LOGIC := '0';

signal START\_TRIG : STD\_LOGIC := '1';

signal TRIG : STD\_LOGIC := '0';

signal DISTANCE : INTEGER := 0;

-- Inicio del test bench

begin

-- Mapeo del PWM\_LED

UUT: entity work.hcsr0x(Bhvl)

generic map(INPUT\_CLK => INPUT\_CLK, DIVIDER => DIVIDER, MAX\_CM\_DIST => MAX\_CM\_DIST)

port map(CLK => CLK, ECHO => ECHO, START\_TRIG => START\_TRIG, TRIG => TRIG, DISTANCE => DISTANCE);

-- Datos de prueba

CLKDATA: process

begin

Clk <= not Clk;

wait for T/2;

end process CLKDATA;

MEA: process

begin

START\_TRIG <= '1';

wait for 10\*T;

START\_TRIG <= '0';

wait for 100\*T;

ECHO <= '1';

wait for T;

ECHO <= '0';

wait for 10\*T;

end process MEA;

end testbench;

***Evaluación del Laboratorio 3***

25% - Realizar el laboratorio superior, buscar los cambios que harán que el laboratorio funcione apropiadamente

* Realizar Código en VHDL
* Realizar Test Bench

25% - Agregar bit de salida llamado Data Valid de 2ms y registrar la salida de distancia en LEDs

* Data Valid que indica que la medición fue correcta
* 4 leds que regitran si estamos entre 1, 2, 3 o 4 cm

50% - Implementar en el FPGA