Project Name: Rotating Obstacle Chaser
Oğuz Altan 21600966
Section: 01

EEE102 Introduction to Digital Design Course Project Report

YouTube video link:

https://youtu.be/qoW3Q4s6ehM

Abstract / Objective

Aim of this project is make a rotating obstacle chaser. I use a servo motor and ultrasonic sensor to measure the distance. The sensor is mounted on the top of motor, so it rotates due to motor's constant rotation. The distance detected by ultrasonic sensor goes to BASYS 3 FPGA. When sensor detects an object within a predetermined range by user, buzzer turns on and warn.

The Design Specification Plan

Components and Tools

- BASYS 3
- SG90 Servo Motor
- Arduino and prototype Shield as 5V Power Source
- Breadboard and Wires
- 5V Active Buzzer

Starting from servo motor, it is a motor that is used to move the tip of it to some predetermined degrees. It takes the information signal to what degrees it must go from BASYS 3 FPGA board.

Figure 1: Sg90 Servo Motor

In this project, it is used to move from 0 degrees to 180 degrees without any wait, namely it goes and comes back constantly. A technique called PWM (Pulse Width Modulation) is used to control it.

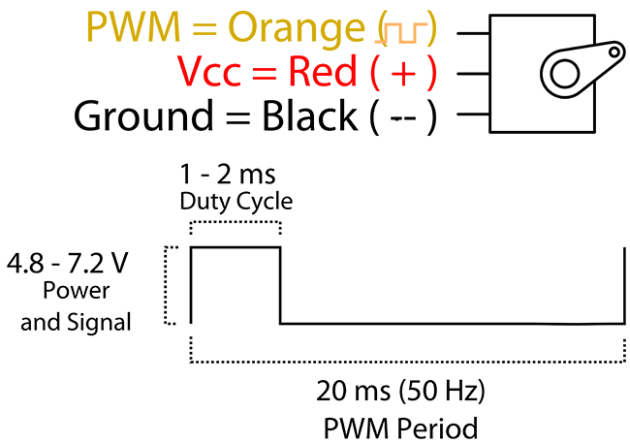


Figure 2: PWM for Sg90 Motor

As it can be seen from the figure 2, in every 20ms, a square wave signal is sent to motor to inform it to where it should go. To rotate it to -90 degrees, in this signal, where must be a high signal during 1 ms of these 20 ms signal and for +90 degrees this high should be 2 ms.

On the top of this motor, ultrasonic sensor is mounted. Ultrasonic sensor is used to measure the distance to some objects in front of it, using sound waves.

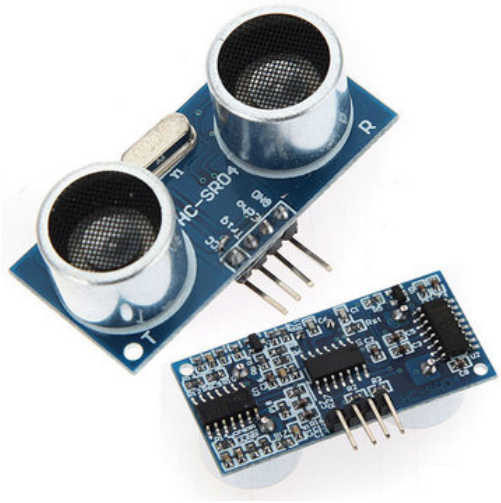


Figure 3: Hc-Sr04 Ultrasonic Sensor

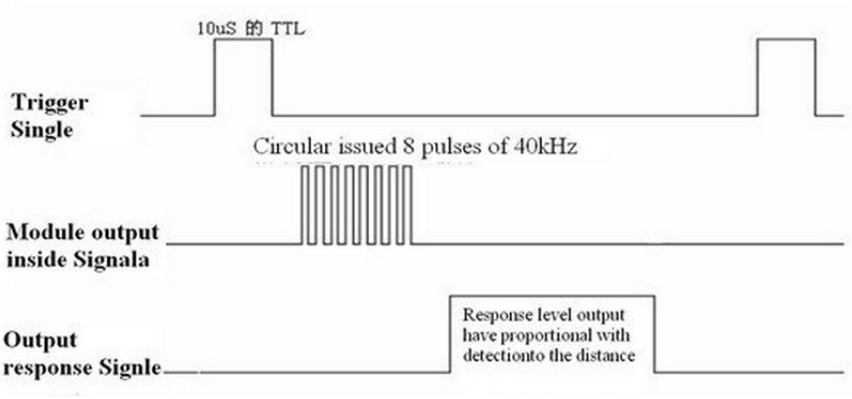


Figure 4: Timing Diagram for Ultrasonic Sensor

When it gets high signal to its trigger inputs, it emits 8 40 kHz sound waves and these waves bounce from the object in the front and comes back to sensor and it receives them. To measure the distance, the time between the receptions of the signals is used. The simple physics formula X = V.t is used here, as we can calculate time (t) and we know the speed of the soundwaves in a room.

Finally, consider that the distance between the sensor and object is half of the way that sound waves takes as they go and bounce, come again to sensor.

$$V = 331.\sqrt[2]{1 + (T/273)}$$

This formula is used to measure the speed of soundwaves in a room with temperature T (degrees).

To warm the user, a 5V Active Buzzer is used. When ultrasonic sensor detects an objects withing a range determined by user, it sends a high signal to buzzer and buzzer turns on.

Figure 5: 5V Active Buzzer

Arduino is used to give 5V DC power to the circuit. Also a protoype shield and a breadboard is mounted to Arduino, all the external circuit is implemented on this breadboard.

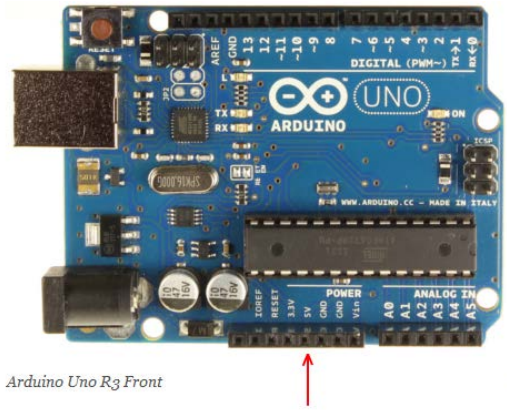


Figure 6: Arduino Uno and its power IO

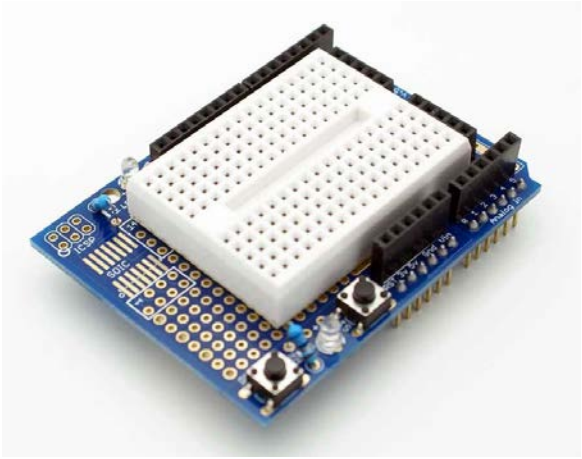


Figure 7: Arduino Uno Protoype Shield and a mini breadboard

Finally, BASYS 3 FPGA Board is the brain of this project, main control device that controls all of these external components and programmed using VHDL on Vivado program on computer. Inputs and outputs of BASYS 3 are used and also seven segment display on it is used.

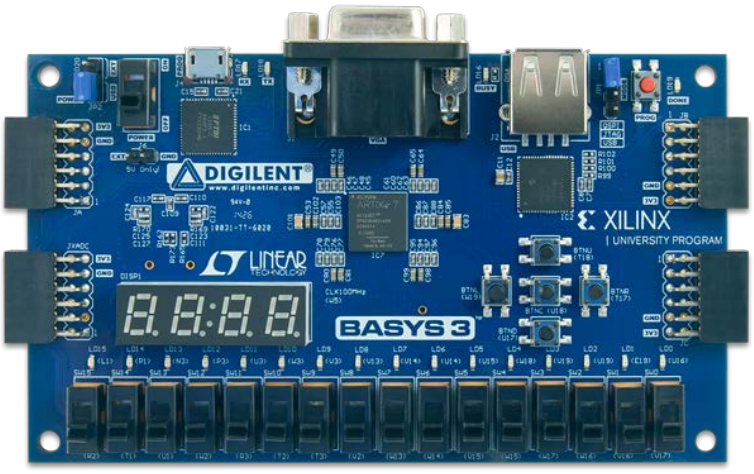


Figure 8: BASYS 3 FPGA Board

The range that is scanned by the ultrasonic sensor can be set by switches on the BASYS 3 board. For example, when the first switch is set to high, then the sensor scans for object within 10 cm. It is 20 cm for second swtich and 30 cm for third switch. The distance measured by sensor is always displayed on the seven segment display on the BASYS 3 board, so that user can see the distance.

The Design Methodology

Servo motor has 3 connections, Vcc, ground and signal input. Vcc and ground areprovided by Arduino and implemented on breadboard. Signal input is connected to JA1 IO of BASYS 3. Ultrasonic sensor has 4 connections, Vcc, ground, trigger and echo. Vcc and ground are again conncted to breadboard. Trigger is connected to JB1 and echo is connected to JB2 IO of BASYS 3 board. With an plastic apparatus, sensor is mounted on the top of motor. This combinations is put inside a box, so that sensor is on the top of the box scannnig the area and motor is in the box. Buzzer's high is connected to JA2 of the BASYS 3 and its ground is connected to common ground on the breadboard. So when distance sensor detects some objects, it sends high signal to buzzer.

Finally, BASYS 3 and Arduino are connected to computer via USB.

Results

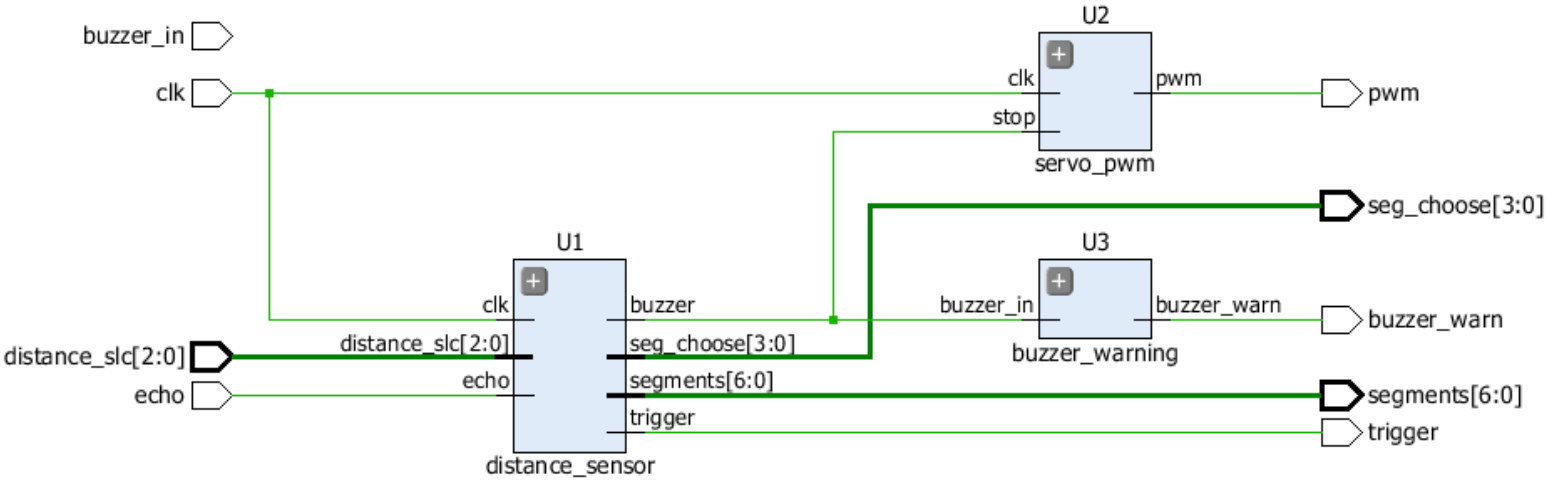


Figure 9: RTL Schematic

As it can be seen from the RTL Schematics, in the top module digitalsensor_topmodule, there are 3 modules and distance_sensor module contains sevensegment_decoder module. All of the inputs and outputs are shown.

Conclusion

This project is made to scan objects in an area and chase objects. Servo motor, ultrasonic sensor, buzzer, BASYS 3 FPGA board and other components are used for it. So, a mechanism is constructed. This allows user to detect objects in a range preset by user him/herself. Switches on the BASYS 3 are used.

Appendix

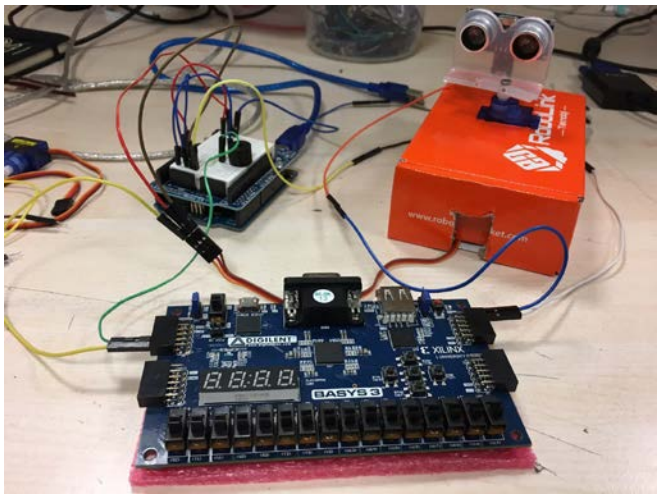


Figure 10: Project Setup Photo #1

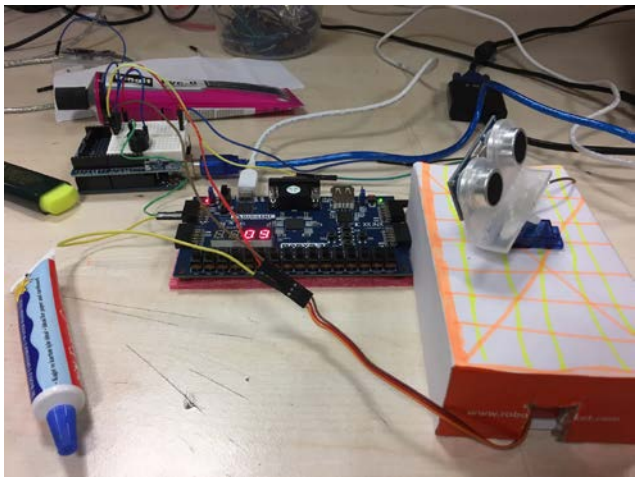


Figure 11: Figure 9: Project Setup Photo #2

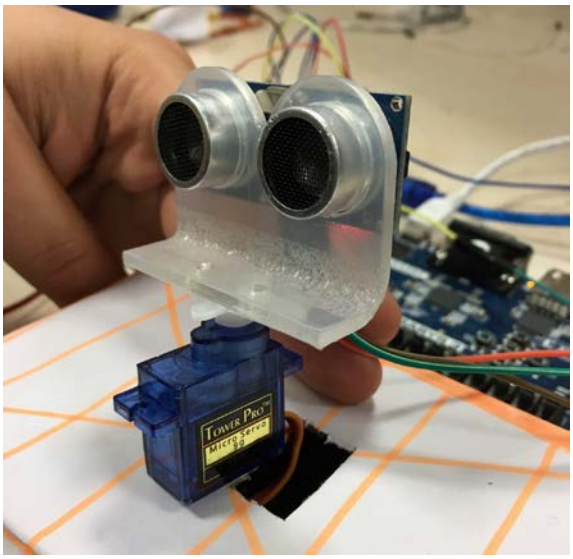


Figure 12: Servo Motor and Ultrasonic Sensor

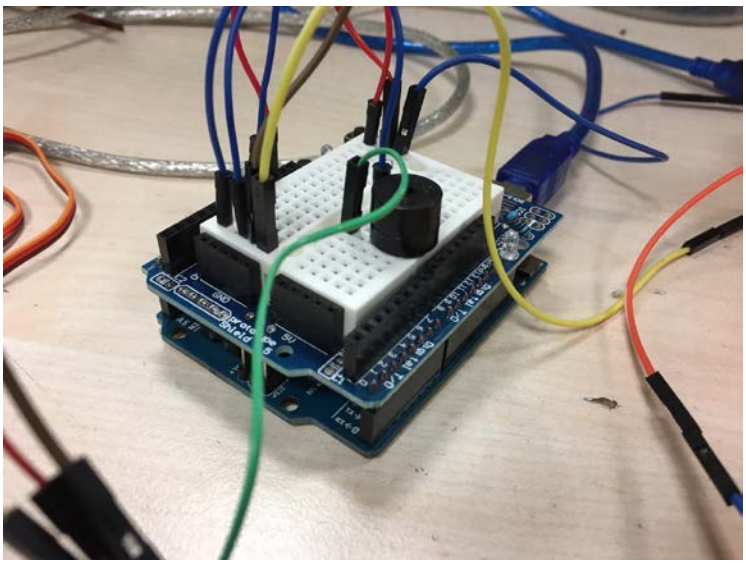


Figure 13: Arduino and Prototype Shield

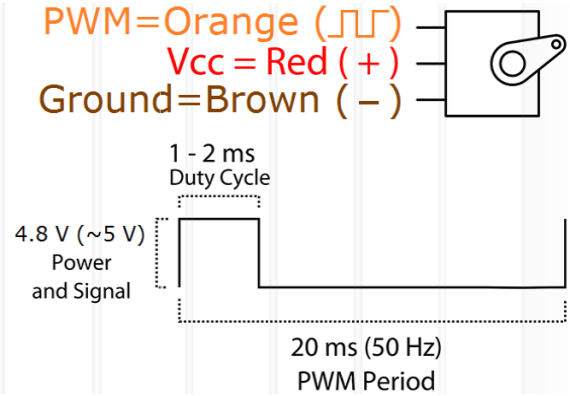
Data Sheets

SG90 9 g Micro Servo

Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but *smaller*. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

Specifications

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10 µs
- Temperature range: 0 ºC – 55 ºC



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.

Ultrasonic Ranging Module HC - SR04

☐ Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

☐ Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

Electric Parameter

Working Voltage DC 5 V
Working Current 15mA
Working Frequency 40Hz
Max Range 4m
Min Range 2cm
MeasuringAngle 15 degree
Trigger Input Signal 10uS TTL pulse
Echo Output Signal Input TTL lever signal and the range in proportion
Dimension 45*20*15mm

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.

☐ Attention:

♉ ☐ The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.

♉ ☐ When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

www.Elecfreaks.com

VHDL codes

Top Module

Digitalsonar_topmodule

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

entity digitalsonar_topmodule is
    Port ( clk  : in  STD_LOGIC;
    trigger     : out STD_LOGIC;
    echo        : in  STD_LOGIC;
    seg_choose     : out STD_LOGIC_VECTOR (3 downto 0);
    segments    : out STD_LOGIC_VECTOR (6 downto 0);
```

```
        pwm : out std_logic;
        distance_slc : in STD_LOGIC_VECTOR (2 downto 0) := "000"
        );
end digitalsonar_topmodule;

architecture Behavioral of digitalsonar_topmodule is

signal buzzerstop: std_logic;

COMPONENT distance_sensor
    PORT(clk        : in  STD_LOGIC;
        trigger     : out STD_LOGIC;
        echo        : in  STD_LOGIC;
        seg_choose     : out STD_LOGIC_VECTOR (3 downto 0);
        segments    : out STD_LOGIC_VECTOR (6 downto 0);
        buzzer : out STD_LOGIC;
        distance_slc : in STD_LOGIC_VECTOR (2 downto 0) := "000"
        );
END COMPONENT;

COMPONENT servo_pwm
    PORT(clk : in  STD_LOGIC;
        stop : in  STD_LOGIC;
        pwm : out STD_LOGIC
        );
END COMPONENT;
```

```
begin

  U1: distance_sensor PORT MAP ( clk => clk, trigger => trigger, echo => echo, seg_choose =>
seg_choose,

   segments => segments, buzzer => buzzerstop,  distance_slc => distance_slc);

  U2: servo_pwm PORT MAP( clk => clk, stop => buzzerstop, pwm => pwm);


end Behavioral;
```

distance_sensor

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;


entity distance_sensor is

    Port ( clk        : in  STD_LOGIC;

           trigger     : out STD_LOGIC;

           echo       : in  STD_LOGIC;

           seg_choose     : out STD_LOGIC_VECTOR (3 downto 0);

           segments    : out STD_LOGIC_VECTOR (6 downto 0);

           buzzer : out STD_LOGIC := '0';

           distance_slc : in STD_LOGIC_VECTOR (2 downto 0) := "000"

           );


end distance_sensor;
```

```
architecture Behavioral of distance_sensor is

    signal count            : unsigned(16 downto 0) := (others => '0');
    signal cm       : unsigned(15 downto 0) := (others => '0');
    signal cm_unit : unsigned(3 downto 0)  := (others => '0');
    signal cm_decimal  : unsigned(3 downto 0)  := (others => '0');
    signal out_unit      : unsigned(3 downto 0)  := (others => '0');
    signal out_decimal       : unsigned(3 downto 0)  := (others => '0');
    signal digit            : unsigned(3 downto 0)  := (others => '0');
    signal echo_last    : std_logic := '0';
    signal echo_current      : std_logic := '0';
    signal echo_not_current   : std_logic := '0';
    signal waitt          : std_logic := '0';
    signal segment_counter     : unsigned(15 downto 0) := (others => '0');

    COMPONENT sevensegment_decoder
    Port ( digit : in unsigned (3 downto 0);
           segments : out STD_LOGIC_VECTOR (6 downto 0)
          );
    END COMPONENT;

    begin
     decoder : sevensegment_decoder PORT MAP (digit => digit,
     segments => segments);

    seven_seg: process(clk)
```

```
        begin
          if rising_edge(clk) then

          if segment_counter(segment_counter'high) = '1' then
                  digit <= out_unit;
                  seg_choose   <= "1110";
          else
                  digit <= out_decimal;
                  seg_choose   <= "1101";
          end if;
              segment_counter <= segment_counter +1;
           end if;
        end process;

process(clk)
   begin
      if rising_edge(clk) then

        if waitt = '0' then

          if count = 1000 then  -- 10us trigger
            trigger    <= '0';
            waitt    <= '1';
            count      <= (others => '0');
          else
            trigger <= '1';
```

```
      count <= count+1;
    end if;

elsif echo_last = '0' and echo_current = '1' then
    count <= (others => '0');
    cm  <= (others => '0');
    cm_unit <= (others => '0');
    cm_decimal  <= (others => '0');

elsif echo_last = '1' and echo_current = '0' then
    out_unit <= cm_unit;
    out_decimal <= cm_decimal;

    if distance_slc = "001" then
      if (cm_decimal < "0001") then
       buzzer <= '1';

      else
       buzzer <= '0';
      end if;

     elsif distance_slc = "010" then
       if (cm_decimal <"0010") then
        buzzer <= '1';

       else
```

```
          buzzer <= '0';
        end if;

      elsif distance_slc = "100" then
        if (cm_decimal <"0011") then
         buzzer <= '1';

        else
         buzzer <= '0';
        end if;

      else
       buzzer <= '0';
      end if;

elsif count = 5799 then --5800-1 distance = time/58

    if cm_unit = 9 then
        cm_unit <= (others => '0');
        cm_decimal  <= cm_decimal + 1;
    else
        cm_unit <= cm_unit + 1;
    end if;

    cm <= cm + 1;
    count <= (others => '0');
```

```
            if cm = 3448 then
               waitt <= '0';
            end if;

         else
            count <= count + 1;

         end if;

         echo_last  <= echo_current;
         echo_current <= echo_not_current;
         echo_not_current <= echo;

      end if;

   end process;
    end Behavioral;
```

servo_pwm

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

   entity servo_pwm is

   Generic(
```

```
--100 MHz = 10 ns, our PWM period is 20 ms
count_max : integer := 2000000; --  20 ms/(100MHz = 10 ns)
duty_max : integer := 240000; -- maximum duty cycle / count_max == 2 ms / 10 ns
duty_min : integer := 60000; -- minimum duty cycle / count_max == 1 ms/ 10ns
duty_delta : integer := 1000 -- sets the speed
);

Port( clk : in STD_LOGIC;
stop : in STD_LOGIC;
pwm : out std_logic
);
end servo_pwm;

architecture Behavioral of servo_pwm is

signal counter: integer range 0 to count_max := 0;
signal duty : integer range duty_min to duty_max := duty_min;

begin

prescaler: process(all)

variable direction_up : boolean := true;
begin
if rising_edge(clk) then
```

```
  if counter < count_max then
  counter <= counter + 1;

  else
    if direction_up then
      if duty < duty_max and stop = '0' then
       duty <= duty + duty_delta;
      else
       direction_up := false;
      end if;
    else
      if duty > duty_min and stop = '0' then
       duty <= duty - duty_delta;
      else
       direction_up := true;
      end if;
    end if;
   counter <= 0;
  end if;
 end if;
end process;

 pwm_s: process(all)
 begin

  if counter < duty and stop = '0' then
```

```
    pwm <= '1';
   else
```

sevensegment_decoder

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

entity sevensegment_decoder is
Port ( digit : in STD_LOGIC_VECTOR (3 downto 0);
    segments : out STD_LOGIC_VECTOR (6 downto 0));
end sevensegment_decoder;

architecture Behavioral of sevensegment_decoder is

begin
    process(digit)
    begin

    case digit is
        when "0001" => segments <= "1111001";
        when "0010" => segments <= "0100100";
        when "0011" => segments <= "0110000";
        when "0100" => segments <= "0011001";
        when "0101" => segments <= "0010010";
        when "0110" => segments <= "0000010";
        when "0111" => segments <= "1111000";
```

```
            when "1000" => segments <= "0000000";
            when "1001" => segments <= "0010000";
            when "1010" => segments <= "0001000";
            when "1011" => segments <= "0000011";
            when "1100" => segments <= "1000110";
            when "1101" => segments <= "0100001";
            when "1110" => segments <= "0000110";
            when "1111" => segments <= "0001110";
            when others => segments <= "1000000";
        end case;
    end process;
end behavioral;
```

buzzer_warning

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

entity buzzer_warning is
 Port( buzzer_in : in std_logic;
       buzzer_warn: out std_logic
       );
end buzzer_warning;

architecture Behavioral of buzzer_warning is

begin
```

```
buzzer_warn <= buzzer_in;


end Behavioral;
```

master.xdc (constraints)

```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
                set_property IOSTANDARD LVCMOS33 [get_ports clk]
                create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports
clk]


## Switches
set_property PACKAGE_PIN V17 [get_ports {distance_slc[0]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {distance_slc[0]}]
set_property PACKAGE_PIN V16 [get_ports {distance_slc[1]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {distance_slc[1]}]
set_property PACKAGE_PIN W16 [get_ports {distance_slc[2]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {distance_slc[2]}]
#set_property PACKAGE_PIN W17 [get_ports {pos[1]}]
#               set_property IOSTANDARD LVCMOS33 [get_ports {pos[1]}]
#set_property PACKAGE_PIN W15 [get_ports {pos[2]}]
#               set_property IOSTANDARD LVCMOS33 [get_ports {pos[2]}]
#set_property PACKAGE_PIN V15 [get_ports {pos[3]}]
#               set_property IOSTANDARD LVCMOS33 [get_ports {pos[3]}]
#set_property PACKAGE_PIN W14 [get_ports {pos[4]}]
#               set_property IOSTANDARD LVCMOS33 [get_ports {pos[4]}]
```

```
#set_property PACKAGE_PIN W13 [get_ports {pos[5]}]
#                   set_property IOSTANDARD LVCMOS33 [get_ports[5]}]
#set_property PACKAGE_PIN V2 [get_ports {pos[6]}]
#                   set_property IOSTANDARD LVCMOS33 [get_ports[6]}]
#set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
                    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
#set_property PACKAGE_PIN T2 [get_ports {sw[10]}]
                    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
#set_property PACKAGE_PIN R3 [get_ports {sw[11]}]
                    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]
#set_property PACKAGE_PIN W2 [get_ports {sw[12]}]
                    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
#set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
                    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
#set_property PACKAGE_PIN T1 [get_ports {sw[14]}]
                    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[14]}]
#set_property PACKAGE_PIN R2 [get_ports {sw[15]}]
                    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[15]}]


## LEDs
#set_property PACKAGE_PIN U16 [get_ports {led[0]}]
                    #set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
#set_property PACKAGE_PIN E19 [get_ports {led[1]}]
                    #set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
#set_property PACKAGE_PIN U19 [get_ports {led[2]}]
                    #set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
```

```
#set_property PACKAGE_PIN V19 [get_ports {led[3]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
#set_property PACKAGE_PIN W18 [get_ports {led[4]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
#set_property PACKAGE_PIN U15 [get_ports {led[5]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
#set_property PACKAGE_PIN U14 [get_ports {led[6]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
#set_property PACKAGE_PIN V14 [get_ports {led[7]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
#set_property PACKAGE_PIN V13 [get_ports {led[8]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {led[8]}]
#set_property PACKAGE_PIN V3 [get_ports {led[9]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {led[9]}]
#set_property PACKAGE_PIN W3 [get_ports {led[10]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {led[10]}]
#set_property PACKAGE_PIN U3 [get_ports {led[11]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {led[11]}]
#set_property PACKAGE_PIN P3 [get_ports {led[12]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {led[12]}]
#set_property PACKAGE_PIN N3 [get_ports {led[13]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {led[13]}]
#set_property PACKAGE_PIN P1 [get_ports {led[14]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {led[14]}]
#set_property PACKAGE_PIN L1 [get_ports {led[15]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {led[15]}]
```

```
#7 segment display
set_property PACKAGE_PIN W7 [get_ports {segments[0]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {segments[0]}]
set_property PACKAGE_PIN W6 [get_ports {segments[1]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {segments[1]}]
set_property PACKAGE_PIN U8 [get_ports {segments[2]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {segments[2]}]
set_property PACKAGE_PIN V8 [get_ports {segments[3]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {segments[3]}]
set_property PACKAGE_PIN U5 [get_ports {segments[4]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {segments[4]}]
set_property PACKAGE_PIN V5 [get_ports {segments[5]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {segments[5]}]
set_property PACKAGE_PIN U7 [get_ports {segments[6]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {segments[6]}]

set_property PACKAGE_PIN V7 [get_ports dp]
                set_property IOSTANDARD LVCMOS33 [get_ports dp]

set_property PACKAGE_PIN U2 [get_ports {seg_choose[0]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {seg_choose[0]}]
set_property PACKAGE_PIN U4 [get_ports {seg_choose[1]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {seg_choose[1]}]
set_property PACKAGE_PIN V4 [get_ports {seg_choose[2]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {seg_choose[2]}]
```

```
set_property PACKAGE_PIN W4 [get_ports {seg_choose[3]}]
                set_property IOSTANDARD LVCMOS33 [get_ports {seg_choose[3]}]


#Pmod Header JA
#Sch name = JA1
set_property PACKAGE_PIN J1 [get_ports {pwm}]
                set_property IOSTANDARD LVCMOS33 [get_ports {pwm}]
#Sch name = JA2
set_property PACKAGE_PIN L2 [get_ports {buzzer}]
                set_property IOSTANDARD LVCMOS33 [get_ports {buzzer}]
##Sch name = JA3
#set_property PACKAGE_PIN J2 [get_ports {JA[2]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JA[2]}]
##Sch name = JA4
#set_property PACKAGE_PIN G2 [get_ports {JA[3]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JA[3]}]
##Sch name = JA7
#set_property PACKAGE_PIN H1 [get_ports {JA[4]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JA[4]}]
##Sch name = JA8
#set_property PACKAGE_PIN K2 [get_ports {JA[5]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JA[5]}]
##Sch name = JA9
#set_property PACKAGE_PIN H2 [get_ports {JA[6]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JA[6]}]
##Sch name = JA10
```

```
#set_property PACKAGE_PIN G3 [get_ports {JA[7]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JA[7]}]


##Pmod Header JB
#Sch name = JB1
set_property PACKAGE_PIN A14 [get_ports {trigger}]
                set_property IOSTANDARD LVCMOS33 [get_ports {trigger}]
#Sch name = JB2
set_property PACKAGE_PIN A16 [get_ports {echo}]
                set_property IOSTANDARD LVCMOS33 [get_ports {echo}]
##Sch name = JB3
#set_property PACKAGE_PIN B15 [get_ports {JB[2]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JB[2]}]
##Sch name = JB4
#set_property PACKAGE_PIN B16 [get_ports {JB[3]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JB[3]}]
##Sch name = JB7
#set_property PACKAGE_PIN A15 [get_ports {JB[4]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JB[4]}]
##Sch name = JB8
#set_property PACKAGE_PIN A17 [get_ports {JB[5]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JB[5]}]
##Sch name = JB9
#set_property PACKAGE_PIN C15 [get_ports {JB[6]}]
```

```
                #set_property IOSTANDARD LVCMOS33 [get_ports {JB[6]}]
##Sch name = JB10
#set_property PACKAGE_PIN C16 [get_ports {JB[7]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JB[7]}]



##Pmod Header JC
##Sch name = JC1
#set_property PACKAGE_PIN K17 [get_ports {JC[0]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JC[0]}]
##Sch name = JC2
#set_property PACKAGE_PIN M18 [get_ports {JC[1]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JC[1]}]
##Sch name = JC3
#set_property PACKAGE_PIN N17 [get_ports {JC[2]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JC[2]}]
##Sch name = JC4
#set_property PACKAGE_PIN P18 [get_ports {JC[3]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JC[3]}]
##Sch name = JC7
#set_property PACKAGE_PIN L17 [get_ports {JC[4]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JC[4]}]
##Sch name = JC8
#set_property PACKAGE_PIN M19 [get_ports {JC[5]}]
                #set_property IOSTANDARD LVCMOS33 [get_ports {JC[5]}]
```

##Sch name = JC9

#set_property PACKAGE_PIN P17 [get_ports {JC[6]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {JC[6]}]

##Sch name = JC10

#set_property PACKAGE_PIN R18 [get_ports {JC[7]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {JC[7]}]