EE 102 Term Project – Smart Parking Lot

Youtube:

https://youtu.be/k_M8dRj7Xz8

Objective:

The main objective of the project is to design a smart parking lot which executes certain operations itself with the help of several sensors and motors.

Methodology:

The first step of the project was to grasp the working principles of the sensors to make them work simultaneously. Initially, the gates of the parking lot were designed those of which were to be used for entrance and exit. A code for an ultrasonic HC-SR04 sensor was written and the same was done for a SG90 servomotor as well. The ultrasonic sensor was used to turn the servo 90 degrees as in the final version of the project and it worked. After that, a code for flame sensor was written as well and so were the codes for the two other ultrasonic sensors one of which was sending signals to the initial servo. When all the coding was completed, the project was implemented on the BASYS3 board using jumper wires and breadboard.

Design Specifications:

The project was designed in VHDL using several submodules and a main module. Each sub module of the main module was written for separate sensors and motors and the sub modules were used to command the motors to turn in the main module. The main module consisted of 5 inputs and 6 outputs as follows:

clk : The internal clock input of the Basys3

echo: The duration of the sound wave's echoing sent by the first sensor

echo1: The duration of the sound wave's echoing sent by the second sensor

echo2: The duration of the sound wave's echoing sent by the third sensor

flame_input: The digital input gathered from the flame sensor

trig: The triggering signal of the first sensor

triger: The triggering signal of the second sensor

trigers: The triggering signal of the third sensor

output3: The data sent to the LED from the third ultrasonic sensor

output1: The data sent to the first servo from the first and the second ultrasonic sensors

output2: The data sent to the second servo from the flame sensor

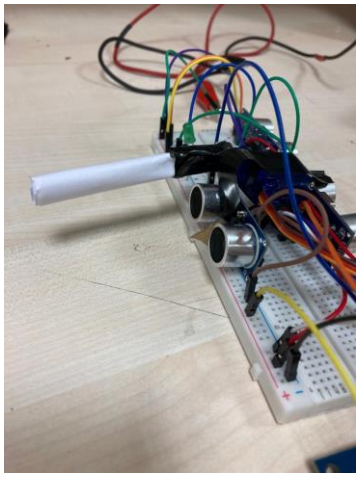

Figure 1: The gate and the first sensor

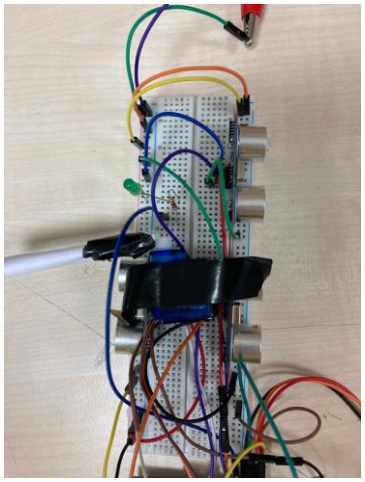

Figure 2: All of the ultrasonic sensors

The first ultrasonic sensor is responsible for opening the gate and the second is responsible for closing it if it detects that the parking space is full. The third sends signal to the LED if it detects that the driver occupies more space than he is supposed to. The flame sensor opens the emergency escape whenever it detects fire.

Results:

The design was successfully implemented on the BASYS3 board with the usage of several jumper wires and a breadboard. The outcome of the implementation can be seen via the youtube link.

Conclusion:

The design included several design principles such as PWM for servo and echolocation of the ultrasonic distance sensors. The flame sensor's working principle was relatively simple as it sent the logic signal 1 when it detected fire. To make the design less complicated, infrared sensors could be used instead of ultrasonic sensors which have the same working principle for the flame sensor except the fact that they send the logic signal 1 whenever they detect an object in front of them. However, ultrasonic sensors enabled us to adjust the distance range to detect the car's presence which was determined to be between 3-13 centimetres from the ultrasonic sensor. The distance range can be adjusted differently depending on the most optimal interval for the parking lot. Minor improvements can still be done. For instance, the

servos do not turn exactly 90 degrees which may create discomfort on the driver. Another thing that can be improved is the emergency escape's functionality. It would be more optimal if the parking lot can generate solutions to the problems other than fire.

Appendix:

Park.lot_main:

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



use IEEE.NUMERIC_STD.ALL;




entity park_lot is
    Port ( clk : in STD_LOGIC;
           echo: in STD_LOGIC;
           echo1: in STD_LOGIC;
           echo2: in STD_LOGIC;

           flame_input: in STD_LOGIC;
           trig: out STD_LOGIC;
           triger: out STD_LOGIC;
           trigers: out STD_LOGIC;
          output3: out STD_LOGIC;
          output1 : out STD_LOGIC;
          output2: out STD_LOGIC);
```

```
end park_lot;

architecture Behavioral of park_lot is
component sensor is
   PORT( clk : in  STD_LOGIC;
         echo : in  STD_LOGIC;


  trig: out STD_LOGIC;

         out1 : out  std_logic_vector (6 downto 0));

end component;

component sensor2 is
    Port ( clk : in  STD_LOGIC;
         echo1 : in  STD_LOGIC;


  trig1: out STD_LOGIC;

         out2 : out  std_logic_vector (6 downto 0));
end component;

component sensor3 is
   Port ( clk : in  STD_LOGIC;
         echo1 : in  STD_LOGIC;


  trig1: out STD_LOGIC;
```

```
        out2 : out  std_logic_vector (6 downto 0));
end component;
component srvo is
    Port ( clk : in STD_LOGIC;
           reset : in STD_LOGIC:= '0';
           pos : in STD_LOGIC_VECTOR (6 downto 0);
            pos1 : in STD_LOGIC_VECTOR (6 downto 0);
            servo1: out STD_LOGIC;
           servo : out STD_LOGIC);
end component;
component flame_sensor is
    Port ( clk : in STD_LOGIC;
           in1 : in STD_LOGIC;
           out1 : out STD_LOGIC);
end component;

type state_type is(full, empty);
signal next_state: state_type;
signal reg_state: state_type:=empty;
signal reset : STD_LOGIC:= '0';
signal  pos : STD_LOGIC_VECTOR(6 downto 0);
signal  pos1 : STD_LOGIC_VECTOR(6 downto 0):= "0000000";
signal servo_runner: STD_LOGIC_VECTOR(6 downto 0);
signal servo_runner1: STD_LOGIC_VECTOR(6 downto 0);
signal servo_runner2: STD_LOGIC_VECTOR(6 downto 0):="0000000";
signal servo_runner3: STD_LOGIC_VECTOR(6 downto 0);

signal in1: std_logic;
```

```
signal out1: std_logic;



begin
U3: flame_sensor PORT MAP (clk=>clk, in1=>flame_input,out1=>out1);
process(pos1, out1)
begin
if out1 = '1' then
   pos1 <= "0100000";
elsif out1 = '0' then
   pos1 <= "0000000";
end if;
end process;

U1: sensor PORT MAP(clk => clk, echo => echo, trig =>trig,out1 => servo_runner);
U4: sensor2 PORT MAP(clk => clk, echo1 => echo1, trig1 =>triger,out2 => servo_runner1);
U5: sensor3 PORT MAP(clk => clk, echo1 => echo2, trig1 =>trigers,out2 => servo_runner3);



process(clk)
begin
if rising_edge(clk) then
   reg_state <= next_state;
end if;
end process;




process(reg_state, servo_runner1, servo_runner)
begin
```

```
    case reg_state is

    when empty =>
    if servo_runner1 <= "0000000" then
        servo_runner2 <= servo_runner;
        next_state <= empty;
    else
        servo_runner2 <= "0000000";
        next_state <= full;
    end if;
    when full =>
    if servo_runner1 <= "0000000" then
        next_state <= empty;
    else
        next_state <= full;
    end if;

    end case;
end process;
```

```
U2: srvo PORT MAP(clk => clk, reset => reset, pos => servo_runner2, pos1 => pos1, servo =>
output1, servo1 => output2);

output3 <= servo_runner3(5);

end Behavioral;
```

Srvo:

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



entity srvo is
    Port ( clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           pos : in STD_LOGIC_VECTOR (6 downto 0);
            pos1 : in STD_LOGIC_VECTOR (6 downto 0);
            servo1: out STD_LOGIC;
           servo : out STD_LOGIC);
end srvo;

architecture Behavioral of srvo is

component freq_div is
     Port ( clk : in STD_LOGIC;
           out_clk : out STD_LOGIC;
           reset : in STD_LOGIC);
end component;

component pwm is
   Port ( clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           pos : in STD_LOGIC_VECTOR (6 downto 0);
           pos1: in STD_LOGIC_VECTOR(6 downto 0);
           servo1: out STD_LOGIC;
           servo : out STD_LOGIC);
end component;
```

```
signal clk_out: std_logic := '0';
begin

U1: freq_div PORT MAP( clk => clk, reset => reset, out_clk => clk_out);
U2:  pwm PORT MAP( clk => clk_out, reset=>reset, pos => pos, pos1 => pos1, servo1 => servo1,
servo => servo);

end Behavioral;
```

Freq_div:

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity freq_div is
    Port ( clk : in STD_LOGIC;
           out_clk : out STD_LOGIC;
           reset : in STD_LOGIC);
end freq_div;

architecture Behavioral of freq_div is
signal new_clk: std_logic;
```

```
signal divider: unsigned(10 downto 0):="00000000000";
begin

process(reset, clk)
begin
   if reset = '1' then
      new_clk <= '0';
      divider <= "00000000000";
   elsif (rising_edge(clk)) then
      if divider = "11000011000" then
         new_clk <= not(new_clk);
         divider <= "00000000000";
      else
         divider <= divider + 1;
      end if;
   end if;
end process;

out_clk <= new_clk;



end Behavioral;
```

Pwm:

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

-- Uncomment the following library declaration if using
```

```
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity pwm is
    Port ( clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           pos : in STD_LOGIC_VECTOR (6 downto 0);
           pos1: in STD_LOGIC_VECTOR(6 downto 0);
           servo1: out STD_LOGIC;
           servo : out STD_LOGIC);
end pwm;

architecture Behavioral of pwm is

signal period_det: unsigned(10 downto 0);
signal pwm_s: unsigned(7 downto 0);
signal pwm_s1: unsigned(7 downto 0);
begin

    pwm_s <= unsigned('0' & pos) + 32;
    pwm_s1 <= unsigned('0' & pos1) + 32;
    process(clk,reset)
    begin
    if reset = '1' then
```

```
        period_det <= (others => '0');
    elsif rising_edge(clk) then
        if period_det = 1279 then
            period_det <= (others => '0');
        else
            period_det <= period_det + 1;
        end if;
    end if;
end process;

servo <= '1' when (period_det < pwm_s) else '0';
servo1 <= '1' when (period_det < pwm_s1) else '0';

end Behavioral;
```

Flame_sensor:

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
```

```
entity flame_sensor is
    Port ( clk : in STD_LOGIC;
           in1 : in STD_LOGIC;
           out1 : out STD_LOGIC);
end flame_sensor;

architecture Behavioral of flame_sensor is

begin
process(clk)
begin
if rising_edge(clk) then
    out1 <= in1;
end if;
end process;

end Behavioral;
```

Sensor:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity sensor is
    Port ( clk : in  STD_LOGIC;
           echo : in  STD_LOGIC;
```

```
   trig: out STD_LOGIC;

         out1 : out  std_logic_vector (6 downto 0));



end sensor;

architecture Behavioral of sensor is

COMPONENT trigger
PORT(
clk : IN std_logic;
trigger : OUT std_logic);



END COMPONENT;

COMPONENT time_counter
PORT(
clk : in  STD_LOGIC;
         reset : in  STD_LOGIC;

         enable : in  STD_LOGIC;

         cnt_output : out  unsigned(19 downto 0));

END COMPONENT;

COMPONENT dist_calc
```

```
PORT(
   echo_count : in unsigned(19 downto 0);

   distance : out  unsigned(3 downto 0));

END COMPONENT;

COMPONENT motor_runner
PORT ( distance_in : in  STD_LOGIC_VECTOR (3 downto 0);

        display_out : out  STD_LOGIC_VECTOR (6 downto 0));

END COMPONENT;

signal trig1: std_logic;

signal cnt1 : unsigned(19 downto 0);
signal e_count : unsigned(19 downto 0);

signal distance_indicator : unsigned(3 downto 0);




begin

U1: trigger PORT MAP(
clk,
trig1);
```

```
trig <= trig1;

U2: time_counter PORT MAP(
clk,
trig1,
echo,
cnt1);

process(echo) begin
if falling_edge(echo) then
e_count <= cnt1;
end if;


end process;

U3: dist_calc PORT MAP(
e_count,

distance_indicator

);

U4: motor_runner PORT MAP(
std_logic_vector(distance_indicator),

out1
);
```

```
end Behavioral;
```

Sensor2:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity sensor2 is
    Port ( clk : in  STD_LOGIC;
           echo1 : in  STD_LOGIC;


   trig1: out STD_LOGIC;

           out2 : out  std_logic_vector (6 downto 0));


end sensor2;

architecture Behavioral of sensor2 is

COMPONENT sub1
PORT(
clk : IN std_logic;
trigger1 : OUT std_logic);
```

```
END COMPONENT;

COMPONENT sub2
PORT(
clk : in  STD_LOGIC;
        reset1 : in  STD_LOGIC;

        enable1 : in  STD_LOGIC;

        cnt_output1 : out  unsigned(19 downto 0));

END COMPONENT;

COMPONENT sub3
PORT(
   echo_count1 : in unsigned(19 downto 0);

   distance1 : out  unsigned(3 downto 0));

END COMPONENT;

COMPONENT sub4
PORT ( distance_in1 : in  STD_LOGIC_VECTOR (3 downto 0);

        display_out1 : out  STD_LOGIC_VECTOR (6 downto 0));

END COMPONENT;
```

```
signal trige: std_logic;

signal cnt2 : unsigned(19 downto 0);
signal e_count1 : unsigned(19 downto 0);

signal distance_indicator1 : unsigned(3 downto 0);




begin

U1: sub1 PORT MAP(
clk,
trige);
trig1 <= trige;

U2: sub2 PORT MAP(
clk,
trige,
echo1,
cnt2);

process(echo1) begin
if falling_edge(echo1) then
e_count1 <= cnt2;
end if;
```

```
end process;

U3: sub3 PORT MAP(
e_count1,

distance_indicator1

);

U4: sub4 PORT MAP(
std_logic_vector(distance_indicator1),

out2
);
end Behavioral;
```

Sensor3:

```
 library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity sensor3 is
    Port ( clk : in  STD_LOGIC;
           echo1 : in  STD_LOGIC;


   trig1: out STD_LOGIC;
```

```
        out2 : out  std_logic_vector (6 downto 0));


end sensor3;

architecture Behavioral of sensor3 is

COMPONENT sensor3sub1
PORT(
clk : IN std_logic;
trigger1 : OUT std_logic);


END COMPONENT;

COMPONENT sensor3sub2
PORT(
clk : in  STD_LOGIC;
        reset1 : in  STD_LOGIC;

        enable1 : in  STD_LOGIC;

        cnt_output1 : out  unsigned(19 downto 0));

END COMPONENT;

COMPONENT sensor3sub3
PORT(
   echo_count1 : in unsigned(19 downto 0);
```

```
    distance1 : out  unsigned(3 downto 0));

END COMPONENT;

COMPONENT sensor3sub4
PORT ( distance_in1 : in  STD_LOGIC_VECTOR (3 downto 0);

        display_out1 : out  STD_LOGIC_VECTOR (6 downto 0));

END COMPONENT;

signal trige: std_logic;

signal cnt2 : unsigned(19 downto 0);
signal e_count1 : unsigned(19 downto 0);

signal distance_indicator1 : unsigned(3 downto 0);




begin

U1: sensor3sub1 PORT MAP(
clk,
trige);
trig1 <= trige;
```

```
U2: sensor3sub2 PORT MAP(
clk,
trige,
echo1,
cnt2);

process(echo1) begin
if falling_edge(echo1) then
e_count1 <= cnt2;
end if;



end process;

U3: sensor3sub3 PORT MAP(
e_count1,

distance_indicator1


);

U4: sensor3sub4 PORT MAP(
std_logic_vector(distance_indicator1),

out2
);
end Behavioral;
```

Trigger:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity trigger is

    Port ( clk : in  STD_LOGIC;

           trigger : out  STD_LOGIC);
end trigger;

architecture Behavioral of trigger is

signal tick: unsigned(19 downto 0) := (others =>'1');
constant nclks: integer := 10000000;
begin
  process (clk) begin
   if rising_edge(clk) then
    if tick < nclks-1 then
     tick <= tick + 1;
    else
     tick <= (others => '0');
     end if;
   end if;
  end process;
trigger <= '1' when (tick < 1000) else '0';

end behavioral;
```

Time_counter:

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity time_counter is

    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;

           enable : in  STD_LOGIC;

           cnt_output : out  unsigned(19 downto 0));

end time_counter;

architecture Behavioral of time_counter is

signal rythm: unsigned(19 downto 0);

begin

process (reset, clk, enable)
begin

if reset = '1' then
rythm <= (others => '0');
elsif rising_edge(clk) then
if enable = '1' then
rythm <= rythm + 1;
```

```
end if;
end if;

end process;

cnt_output <= rythm;



end Behavioral;
```

Dist_calc:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;



entity dist_calc is
port(
echo_count : in unsigned(19 downto 0);

distance : out  unsigned(3 downto 0));
end dist_calc;

architecture Behavioral of dist_calc is

begin

Distance <="0000" when (echo_count < 5800) else --0cm
  "0001" when (echo_count > 5800 and echo_count < 17400) else
  "0010" when (echo_count > 17400 and echo_count < 29000) else
```

```
 "0011" when (echo_count > 29000 and echo_count < 43500) else --5-7.5cm
 "0100" when (echo_count > 43500  and echo_count < 55100) else --7.5-9.5cm
 "0101" when (echo_count > 55100  and echo_count < 60900) else --9.5-10.5cm
 "0110" when (echo_count > 60900 and echo_count < 66700) else --10.5-11.5cm
 "0111" when (echo_count > 66700 and echo_count < 75400) else --11.5-13cm
 "1000" when (echo_count > 75400 and echo_count < 81200) else --13-14cm
 "1001" when (echo_count > 81200 and echo_count < 92800) else --14-16cm
 "1010" when (echo_count > 92800 and echo_count < 98600) else --16-17cm
 "1011" when (echo_count > 98600 and echo_count < 104400) else --17-18cm;
 "1100" when (echo_count > 104400 and echo_count < 110200) else --18-19cm;
 "1101" when (echo_count > 110200 and echo_count <116000) else --19-20cm;
 "1110" when (echo_count > 116000 and echo_count < 127400) else --20-22cm;
 "1111";--more than 22cm

end Behavioral;
```

Motor_runner:

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



entity motor_runner is
   Port ( distance_in : in  STD_LOGIC_VECTOR(3 downto 0);

          display_out : out  STD_LOGIC_VECTOR (6 downto 0));

end motor_runner;

architecture Behavioral of motor_runner is
```

```
begin

display_out <="0000000" when distance_in = "0000" else
  "0100000" when distance_in = "0001" else
  "0100000" when distance_in = "0010" else
  "0100000" when distance_in = "0011" else
  "0100000" when distance_in = "0100" else
  "0100000" when distance_in = "0101" else
  "0100000" when distance_in = "0110" else
  "0100000" when distance_in = "0111" else
  "0100000" when distance_in = "1000" else
  "0000000";



end Behavioral;
```

Sub1:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity sub1 is

    Port ( clk : in  STD_LOGIC;

           trigger1 : out  STD_LOGIC);
end sub1;

architecture Behavioral of sub1 is
```

```
signal tick: unsigned(19 downto 0) := (others =>'1');
constant nclks: integer := 10000000;
begin
  process (clk) begin
   if rising_edge(clk) then
    if tick < nclks-1 then
     tick <= tick + 1;
    else
     tick <= (others => '0');
     end if;
   end if;
  end process;
trigger1 <= '1' when tick < 1000 else '0';

end behavioral;
```

Sub2:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity sub2 is

    Port ( clk : in  STD_LOGIC;
           reset1 : in  STD_LOGIC;

           enable1 : in  STD_LOGIC;

           cnt_output1 : out  unsigned(19 downto 0));
```

```
end sub2;

architecture Behavioral of sub2 is

signal rythm1: unsigned(19 downto 0);

begin

process (reset1, clk, enable1)
begin

if reset1 = '1' then
rythm1 <= (others => '0');
elsif rising_edge(clk) then
if enable1 = '1' then
rythm1 <= rythm1 + 1;

end if;
end if;

end process;

cnt_output1 <= rythm1;



end Behavioral;
```

Sub3:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
```

```
entity sub3 is
port(
echo_count1 : in unsigned(19 downto 0);

distance1 : out  unsigned(3 downto 0));
end sub3;

architecture Behavioral of sub3 is

begin

Distance1 <="0000" when (echo_count1 < 5800) else --0cm
  "0001" when (echo_count1 > 5800 and echo_count1 < 17400) else
  "0010" when (echo_count1 > 17400 and echo_count1 < 29000) else
  "0011" when (echo_count1 > 29000 and echo_count1 < 43500) else --5-7.5cm
  "0100" when (echo_count1 > 43500  and echo_count1 < 55100) else --7.5-9.5cm
  "0101" when (echo_count1 > 55100  and echo_count1 < 60900) else --9.5-10.5cm
  "0110" when (echo_count1 > 60900 and echo_count1 < 66700) else --10.5-11.5cm
  "0111" when (echo_count1 > 66700 and echo_count1 < 75400) else --11.5-13cm
  "1000" when (echo_count1 > 75400 and echo_count1 < 81200) else --13-14cm
  "1001" when (echo_count1 > 81200 and echo_count1 < 92800) else --14-16cm
  "1010" when (echo_count1 > 92800 and echo_count1 < 98600) else --16-17cm
  "1011" when (echo_count1 > 98600 and echo_count1 < 104400) else --17-18cm;
  "1100" when (echo_count1 > 104400 and echo_count1 < 110200) else --18-19cm;
  "1101" when (echo_count1 > 110200 and echo_count1 <116000) else --19-20cm;
  "1110" when (echo_count1 > 116000 and echo_count1 < 127400) else --20-22cm;
  "1111";--more than 22cm
```

```
end Behavioral;
```

Sub4:

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


entity sub4 is
    Port ( distance_in1 : in  STD_LOGIC_VECTOR(3 downto 0);

           display_out1 : out  STD_LOGIC_VECTOR (6 downto 0));

end sub4;

architecture Behavioral of sub4 is

begin

display_out1 <="0000000" when distance_in1 = "0000" else
  "0100000" when distance_in1 = "0001" else
  "0100000" when distance_in1 = "0010" else
  "0100000" when distance_in1 = "0011" else
  "0100000" when distance_in1 = "0100" else
  "0100000" when distance_in1 = "0101" else
  "0100000" when distance_in1 = "0110" else
  "0100000" when distance_in1 = "0111" else
  "0100000" when distance_in1 = "1000" else
  "0000000";
end behavioral;
```

Sensor3sub1:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity sensor3sub1 is

    Port ( clk : in  STD_LOGIC;

           trigger1 : out  STD_LOGIC);
end sensor3sub1;

architecture Behavioral of sensor3sub1 is

signal tick: unsigned(19 downto 0) := (others =>'1');
constant nclks: integer := 10000000;
begin
  process (clk) begin
   if rising_edge(clk) then
    if tick < nclks-1 then
     tick <= tick + 1;
    else
     tick <= (others => '0');
     end if;
   end if;
  end process;
trigger1 <= '1' when tick < 1000 else '0';

end behavioral;
```

Sensor3sub2:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity sensor3sub2 is

    Port ( clk : in  STD_LOGIC;
           reset1 : in  STD_LOGIC;

           enable1 : in  STD_LOGIC;

           cnt_output1 : out  unsigned(19 downto 0));

end sensor3sub2;

architecture Behavioral of sensor3sub2 is

signal rythm1: unsigned(19 downto 0);

begin

process (reset1, clk, enable1)
begin

if reset1 = '1' then
rythm1 <= (others => '0');
elsif rising_edge(clk) then
if enable1 = '1' then
rythm1 <= rythm1 + 1;
```

```
end if;
end if;

end process;

cnt_output1 <= rythm1;


end Behavioral;
```

Sensor3sub3:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;


entity sensor3sub3 is
port(
echo_count1 : in unsigned(19 downto 0);

distance1 : out  unsigned(3 downto 0));
end sensor3sub3;

architecture Behavioral of sensor3sub3 is

begin

Distance1 <="0000" when (echo_count1 < 5800) else --0cm
  "0001" when (echo_count1 > 5800 and echo_count1 < 17400) else
  "0010" when (echo_count1 > 17400 and echo_count1 < 29000) else
  "0011" when (echo_count1 > 29000 and echo_count1 < 43500) else --5-7.5cm
```

```
 "0100" when (echo_count1 > 43500  and echo_count1 < 55100) else --7.5-9.5cm
 "0101" when (echo_count1 > 55100  and echo_count1 < 60900) else --9.5-10.5cm
 "0110" when (echo_count1 > 60900 and echo_count1 < 66700) else --10.5-11.5cm
 "0111" when (echo_count1 > 66700 and echo_count1 < 75400) else --11.5-13cm
 "1000" when (echo_count1 > 75400 and echo_count1 < 81200) else --13-14cm
 "1001" when (echo_count1 > 81200 and echo_count1 < 92800) else --14-16cm
 "1010" when (echo_count1 > 92800 and echo_count1 < 98600) else --16-17cm
 "1011" when (echo_count1 > 98600 and echo_count1 < 104400) else --17-18cm;
 "1100" when (echo_count1 > 104400 and echo_count1 < 110200) else --18-19cm;
 "1101" when (echo_count1 > 110200 and echo_count1 <116000) else --19-20cm;
 "1110" when (echo_count1 > 116000 and echo_count1 < 127400) else --20-22cm;
 "1111";--more than 22cm

end Behavioral;
```

Sensor3sub4:

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



entity sensor3sub4 is
   Port ( distance_in1 : in  STD_LOGIC_VECTOR(3 downto 0);

          display_out1 : out  STD_LOGIC_VECTOR (6 downto 0));

end sensor3sub4;

architecture Behavioral of sensor3sub4 is

begin
```

```
display_out1 <="0000000" when distance_in1 = "0000" else
  "0100000" when distance_in1 = "0001" else
  "0100000" when distance_in1 = "0010" else
  "0100000" when distance_in1 = "0011" else
  "0100000" when distance_in1 = "0100" else
  "0100000" when distance_in1 = "0101" else
  "0100000" when distance_in1 = "0110" else
  "0100000" when distance_in1 = "0111" else
  "0100000" when distance_in1 = "1000" else
  "0000000";
end behavioral;
```