EEE102: Introduction to Digital Circuit Design

Term Project Report

Solar Tracker

Elif Beray Sarıışık, 22003267, EE102 Section 02

Department of Electrical and Electronics Engineering, Bilkent University

YouTube Link: https://youtu.be/GA08dsI3h-s

Title: Solar Tracker

Purpose:

The purpose of the project is to sustain more efficient system for solar panels to capture sunlight more efficiently by turning its direction automatically to adjust an angle which the solar panel is orthogonal to the sunlight throughout the daytime. Additionally, BASYS3 FPGA board will be used to adjust the direction of the solar panel and visualize the angle between sun light and the direction of the solar panel on the 4-digit 7- segment display. The coding background of the project is mainly based on VHDL which will be also integrated with Arduino in order to utilize voltage regulation and to implement the light-dependent resistors on the circuit.

Design Specifications:

The components used for building the solar tracker:

- 2x SG90 9G servo motor
- 4x 5mm LDR
- Arduino Uno
- BASYS3 FPGA Board
- 4x 1k ohm resistor
- 8x 10k ohm resistor
- 8x 18k ohm resistor
- 2 breadboards
- Bunch of male-to-male jumper wires
- Bunch of crocodile cables
- The 3D printed mechanism

Firstly, the mechanisms for two servos and the LDR circuit are printed from the 3D printer. (see Fig. 1, 3) This mechanism for the LDR circuit is created such that it enables to determine the position of the light source properly. It divided in 4 segments to acquire 360 degree movement for the servos. One of the sides of the LDRs are connected to the Vcc of the Arduino Uno which is 5 V. The other side of the LDRs are connected to the 1k ohm resistor and resistors are grounded.

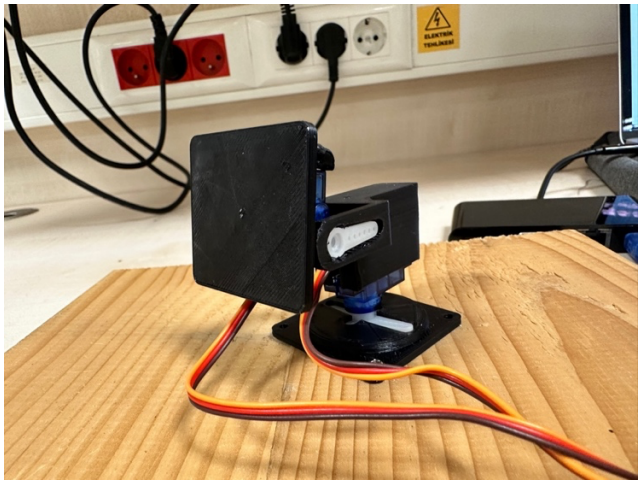

Fig. 1: The Mechanism for two Servo Motors

The side of LDRs which is attached with the resistors are assigned as the ports A0, A1, A2, A3 analog inputs of the Arduino Uno. (See Fig. 1) The voltage differences of LDRs under different conditions are observed by using the serial monitor on Arduino. The base of the voltage differences is determined according to the light of the lab environment. The values 120, 200, 300, and 500 are used for classifying the voltage differences of LDRs as 2-bit binary numbers. For example, the digital output for the voltage difference which is high than 500 is assigned as “11”.

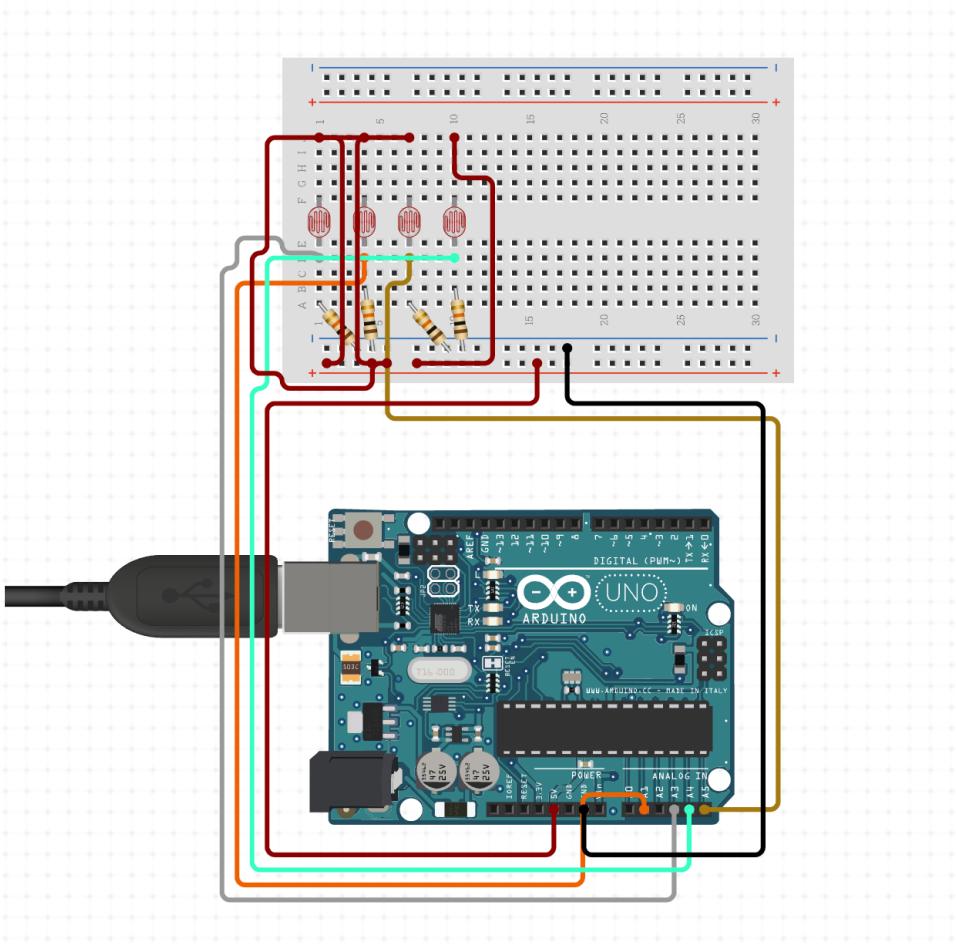

Fig. 2: Implementation of the LDR Circuit

After completing the adjustment for the LDRs, a voltage dividing circuit is created due to the fact that the PMOD ports of BASYS3 FPGA board do not enable the inputs which have higher voltage difference than 3.3V. Therefore, 8x 18k ohm and 10k ohm resistors are used to obtain the voltage difference 3.3V. After completing the voltage dividing circuit, 2-bits for each the LDR are connected to the PMOD ports of BASYS3 FPGA board. The ports A14, A16, A15 , and A17 are assigned for the horizontal movement of the servo mechanism. These are the 2-bit digital outputs which are located on left and right partA of the LDR mechanism. B15, C15, B16, and C16 are assigned for the vertical movement of the servo mechanism. These are the 2-bit digital outputs which are located on up and down part of the LDR mechanism. (See Fig. 2)

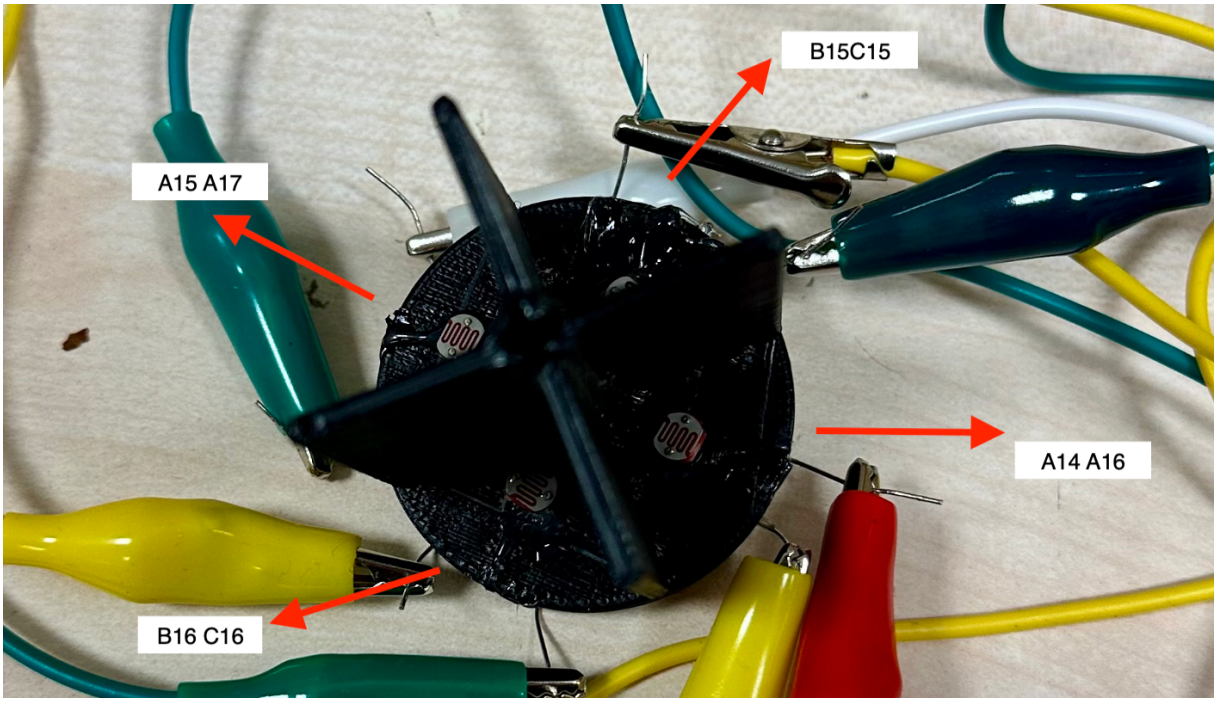

Fig. 3: The Port Configuration for the LDR Mechanism

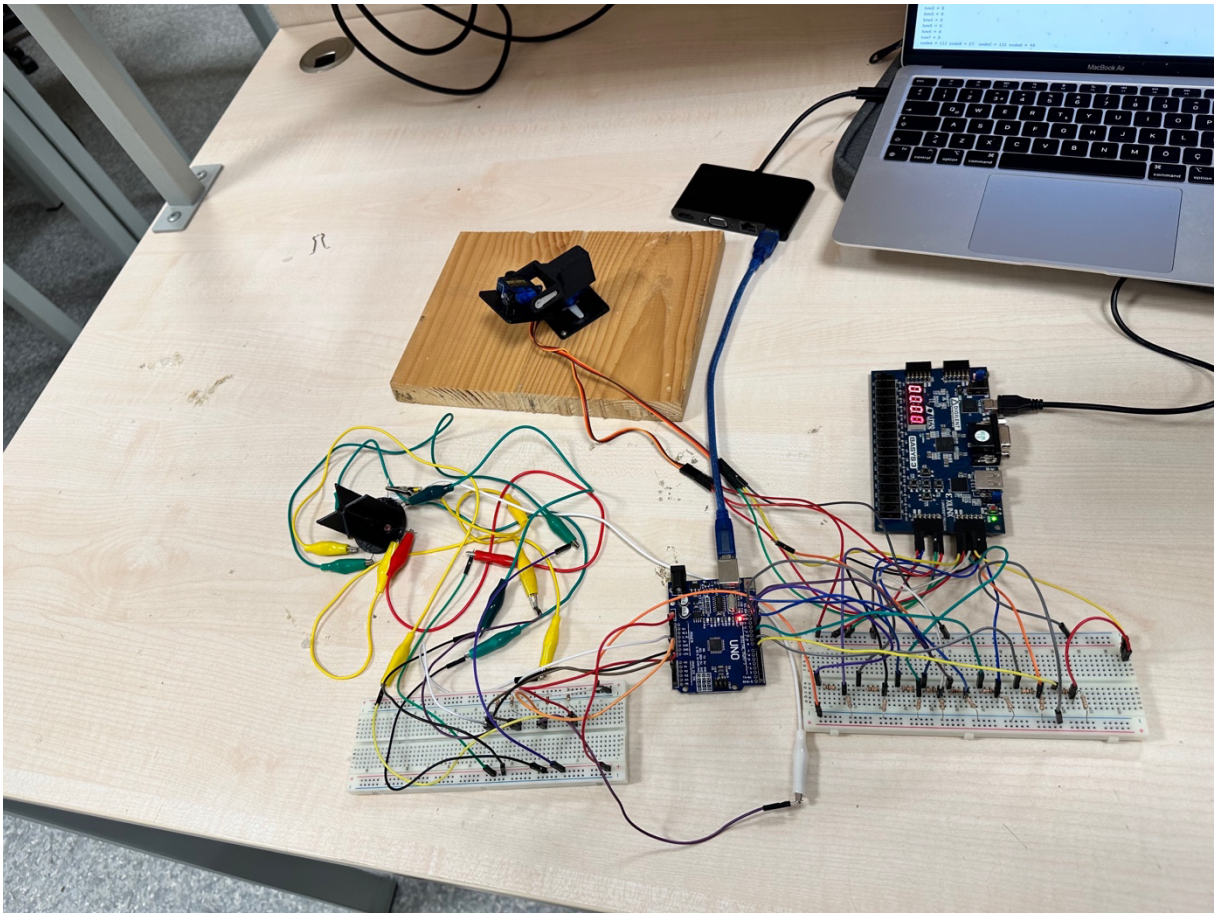

Fig. 4: Overall Circuit Implementation for the Project

After completing the mechanical part of the solar tracker (see Fig. 3), modules and submodules are created. The project consists of one Arduino module “project1.ino”, one top module “topmodule.vhd” and five submodules. (see Appendix B) The submodules are:

- leds.vhd
- counter.vhd
- servovertical.vhd
- servohorizontal.vhd
- topmodule_sevenseg.vhd

The submodule “leds.vhd” describes the assignment for each seven digit on seven segment digit. The submodule “counter.vhd” is created in order to utilize the clock frequency 50MHz. The submodules “servovertical.vhd” and “servohorizontal.vhd” are created in order to acquire proper rotation for the servos. The submodule “topmodule_sevenseg.vhd” is created in order to complete the concept of seven segment display via displaying intended numbers on seven segment display. The Arduino module “project1.ino” is created for both converting analog inputs to digital 2-bit outputs and comparing the voltage differences of each LDR so that assignment will be held in four different stages.

The inputs and outputs of the top module “topmodule.vhd” are:

```
clk : in std_logic
manual_rotate: in std_logic_vector(3 downto 0)
position : in std_logic_vector(3 downto 0)
vertical_pos : in std_logic_vector(3 downto 0)
led15, led14, led13, led12 : out std_logic
servo_h : out std_logic
servo_v : out std_logic
led_anode : out STD_LOGIC_VECTOR (3 downto 0)
led_cathode : out STD_LOGIC_VECTOR (6 downto 0)
```

Methodology:

The project comprises two phases which are a progress demo and a final demo. In the first phase of the project, the mechanical parts of the project will be implemented, such as printing the platform from a 3D printer and combining it with servos. One of the servos will help the panel to rotate 180° in the horizontal direction, whereas the other one will help to rotate 180° in the vertical direction. Therefore, this part will enable the solar panel to rotate 360° accordingly.

Moreover, a platform for the LDRs is printed from the 3D printer. The light dependent sensors are located such that it can differentiate the direction of the light source (see Fig.). This placement will enable the solar panel to turn in the reverse direction where two of four LDRs have bigger resistance implying darkness. Thus, the solar tracker will always follow the brightest light source by changing its position. The circuit for LDRs is implemented by combining with 4k ohm. Arduino UNO is used for both supplying power to the LDR circuit and taking the voltage differences of each LDRs as analog inputs. Arduino is also used for comparing the analog inputs and turning each analog inputs into 2-bit binary numbers for further implementation on VHDL. Since the PMOD ports of the BASYS3 FPGA board do not accept the inputs which have higher voltages than 3.3V, a voltage divider circuit is implemented by using 8x 18k ohm and 8x 10k ohm resistor. In order not to damage the FPGA board, voltage differences of each LDRs when their outputs equal to “11” are checked by using an oscilloscope probe.

After that, 8 digital outputs are assigned to the PMOD ports of the BASYS3 FPGA board as digital inputs. The control of the SG90 servo motors is studied from outside sources. By using the concept of pulse-width modulation, servomechanism is acknowledged and implemented on VHDL. The movement of the solar tracker is implemented as a sequential circuit. The horizontal movement of the solar tracker is controlled by the LDRs which are located such that it differentiates the position of the light source by comparing 4-bit binary numbers. The division of pulse-width modulation signal under 4 cases for horizontal movement slightly differs from the vertical one since due to the implementation of the mechanic system it cannot turn a large angle than 135 degrees. Therefore, two different submodules for the movement of the servos are implemented.

Additionally, three submodules are implemented for the 7-segment display on BASYS3 FPGA board. The horizontal angle between the light source and the initial position

of the solar tracker is displayed on the 7-segment display. Furthermore, the LEDs (L1, P1, E19, U16) are used for displaying both the horizontal direction and the significance of the light beam.

Results:

Thanks to the VIVADO software, the elaborated design of the project is attained (see Fig. 5). The elaborated design of the submodules can be found on the section Appendix A.

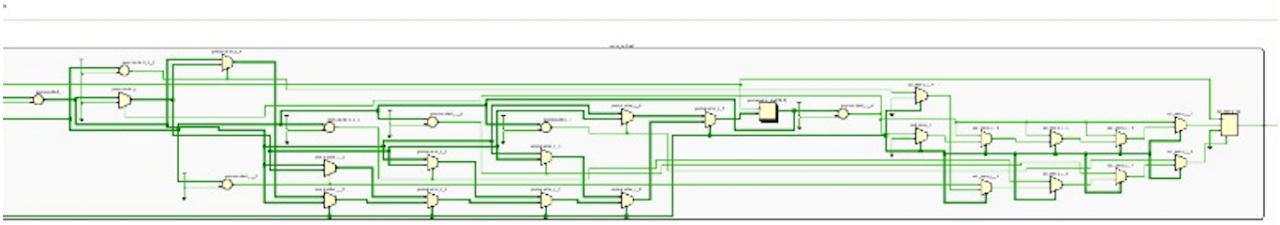

Fig. 5: Overall RTL Schematic of the Project

The solar tracker was able to position in 64 different ways. Some of the configurations are given below:

- When 3.3V PMOD input combination is “1100” for the horizontal movement, the solar tracker turns 90 degrees to the left and “90” is displayed on BASYS3 FPGA board 7-segment display. Additionally, two leftmost LEDs on the board are lightened in order to inform user that the location of the light source if fully left with respect to the initial position of the solar tracker. (see Fig.)

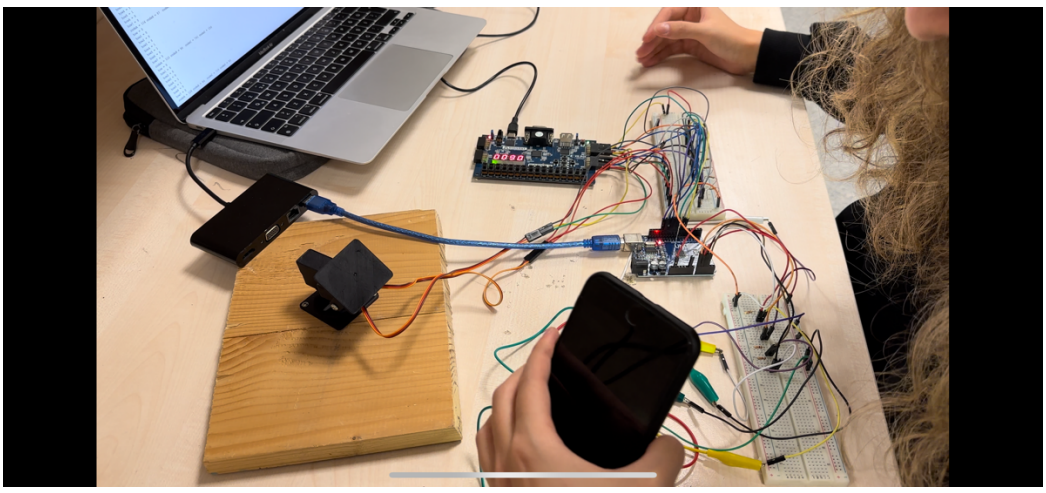

Fig. 6: Overall circuit when the PMOD input is “1100” for the horizontal movement

- When 3.3V PMOD input combination is “0011” for the horizontal movement, the solar tracker turns 90 degrees to the right and “90” is displayed on BASYS3 FPGA board 7-segment display. Additionally, two rightmost LEDs on the board are lightened

in order to inform user that the location of the light source if fully right with respect to the initial position of the solar tracker. (see Fig.)

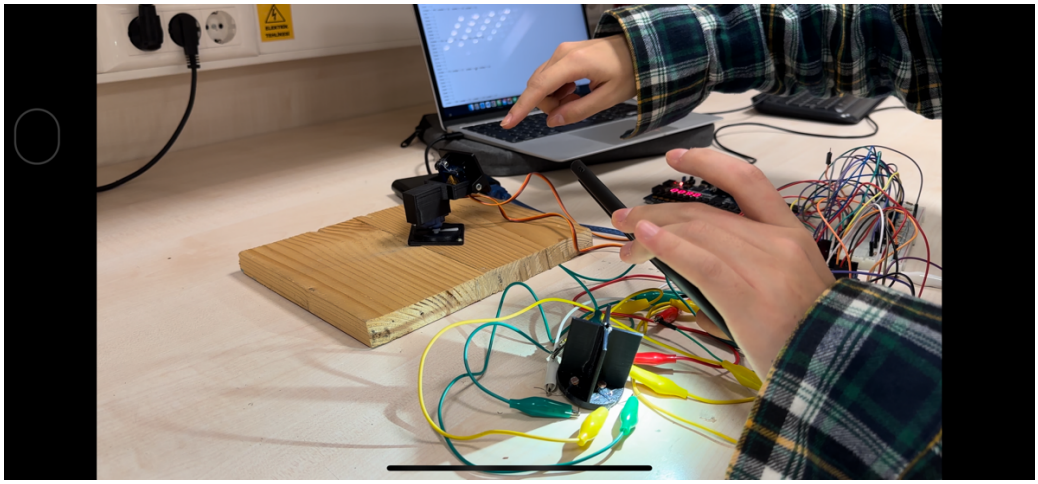

Fig. 7: Overall circuit when the PMOD input is “0011” for the horizontal movement

- When 3.3V PMOD input combination is “0011” for the vertical movement, the solar tracker turns 90 degrees downwards. Since solar tracker is created for 360 degrees movement, solar tracker also turns left due to left LDR also activated when the light comes from the southwest.

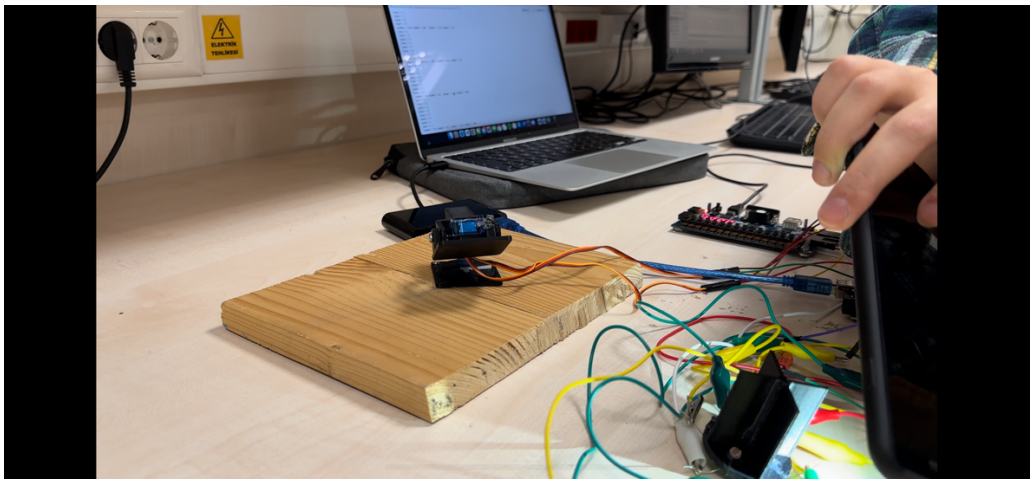

Fig. 8: Overall circuit when the PMOD input is “0011” for the vertical movement

- When 3.3V PMOD input combination is “1100” for the vertical movement, the solar tracker turns 90 degrees upwards. Since solar tracker is created for 360 degrees movement, solar tracker also turns right due to left LDR also activated when the light comes from the northeast.

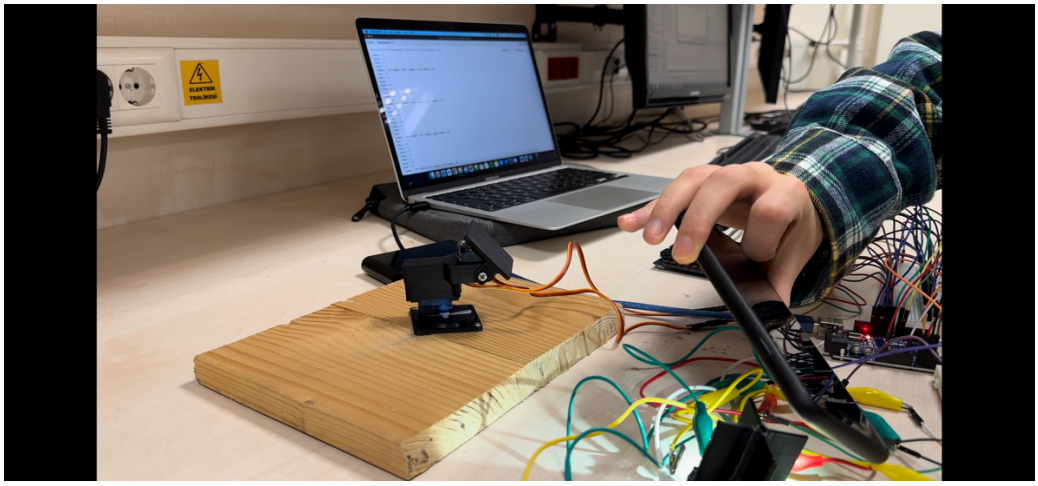

Fig. 9: Overall circuit when the PMOD input is “1100” for the vertical movement

Seven segment display on BASYS3 FPGA board is utilized for displaying the horizontal angle between the initial position of the solar tracker and the light source. Two leftmost and rightmost LEDs are used in order to inform user the position of the light source and the magnificence of the light. Fig. 10, 11, 12 are some examples from different input combinations.

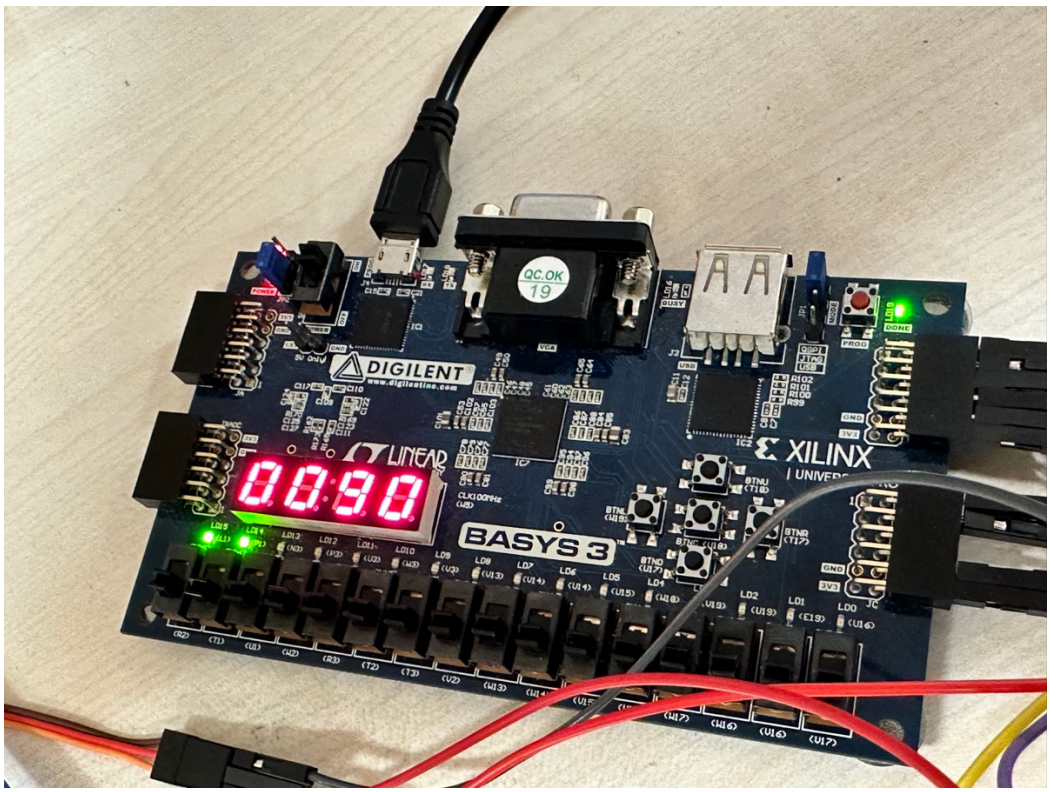

Fig. 10: BASYS3 board when the input is “1100”

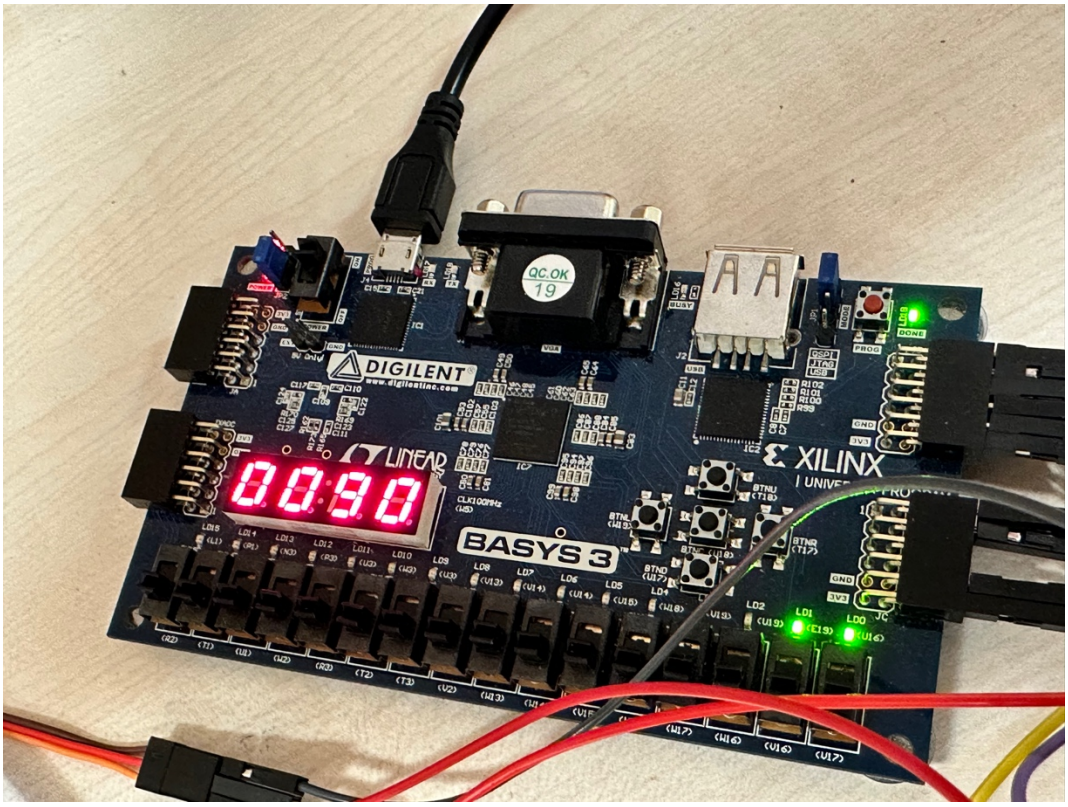

Fig. 11: BASYS3 board when the input is “0011”

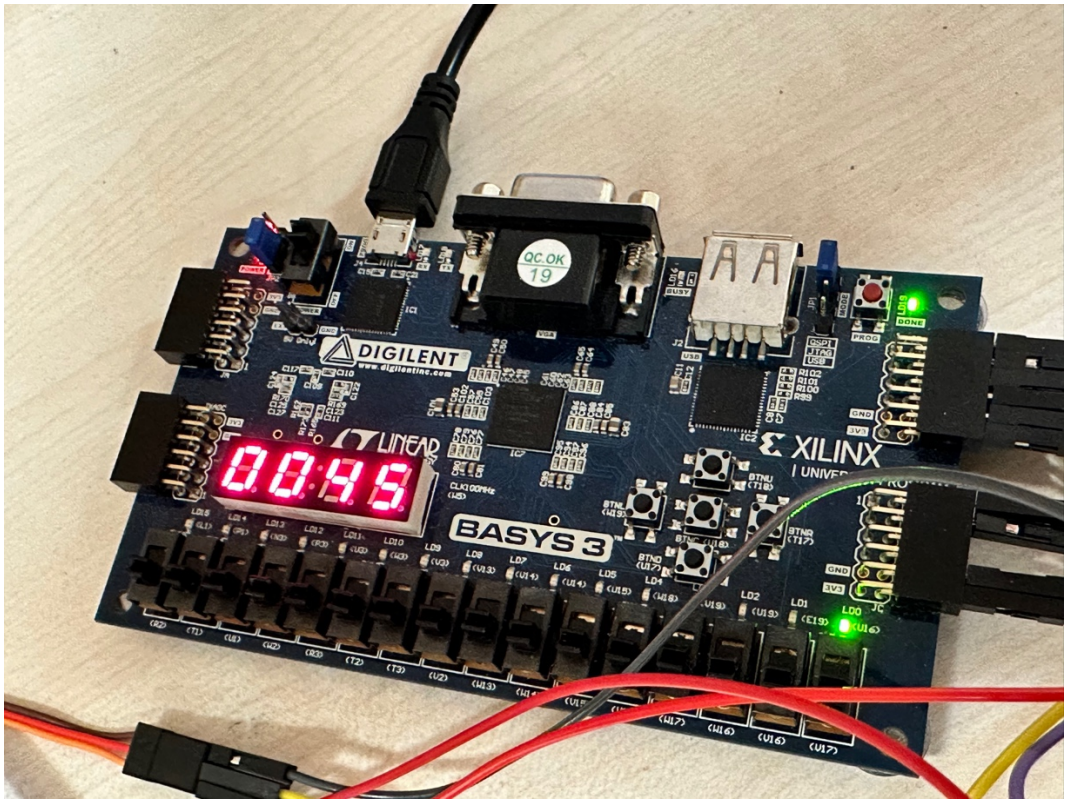

Fig. 12: BASYS3 board when the input is “0010”

Conclusion:

The purpose of the project is to amplify the efficiency of the solar panels by implementing a solar tracker system which tracks the position of the sun and turns such that the solar tracker is orthogonal to the light source. A mechanism is printed from the 3D printer to acquire proper 360-degree movement for the servos. In order to detect the position of the light source, four light-dependent resistors are used. BASYS3 FPGA board will be used to adjust the direction of the solar panel and visualize the angle between sun light and the direction of the solar panel on the 4-digit seven segment display. The project's primary programming language is VHDL, which will also be combined with Arduino in order to construct light-dependent resistors on the circuit and utilize voltage regulation. (see Appendix B) A voltage divider circuit is implemented since the PMOD pins of BASYS3 do not accept the voltage difference higher than 3.3V.

Since the light adjustment of the solar tracker is made according to the lab environment, the solar tracker is rotated by using a flashlight. The sensitivity of the solar tracker can be improved by dividing the corresponding digital outputs of each LDRs more bits which creates more PMOD connection and more complicated circuit. Due to solar tracker processes two different boards concurrently, there is a slight delay. Overall, the aim of this project is fulfilled. The solar tracker successfully tracks the position of the light source and the horizontal angle between the light source and the solar tracker is displayed on seven segment display. Furthermore, the LEDs are used to inform user about the any change in the position or magnificence of the light source.

References:

Fbuenonet. “Mini Pan Tilt- Servo G9.” *Thingiverse,* 2015.

https://www.thingiverse.com/thing:708819

GreatScottLab. “DIY Miniature Solar Tracker.” *Instructables,* 2018.

https://www.instructables.com/DIY-Miniature-Solar-Tracker/

Y. Shehu, M. Irshaidat and M. Soufian, "A FPGA Implementation of a Dual-Axis Solar Tracking System," *2019 12th International Conference on Developments in eSystems Engineering (DeSE)*, 2019, pp. 970-974, doi: 10.1109/DeSE.2019.00180.

Appendix A

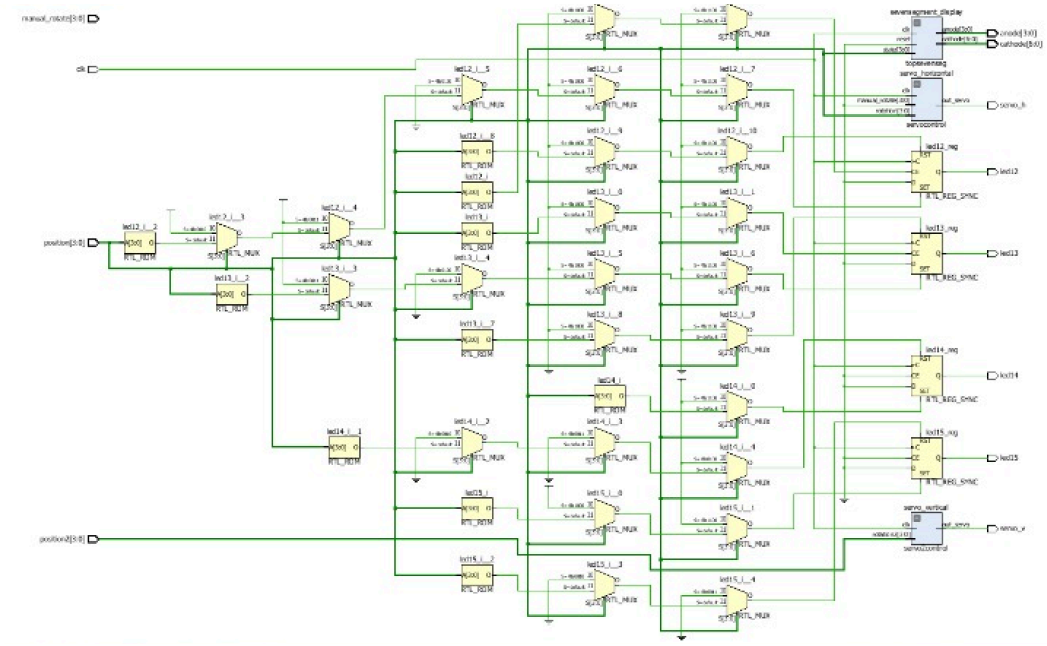

Fig. 13: The RTL Schematic of the submodules “leds.vhd” and “counter.vhd”

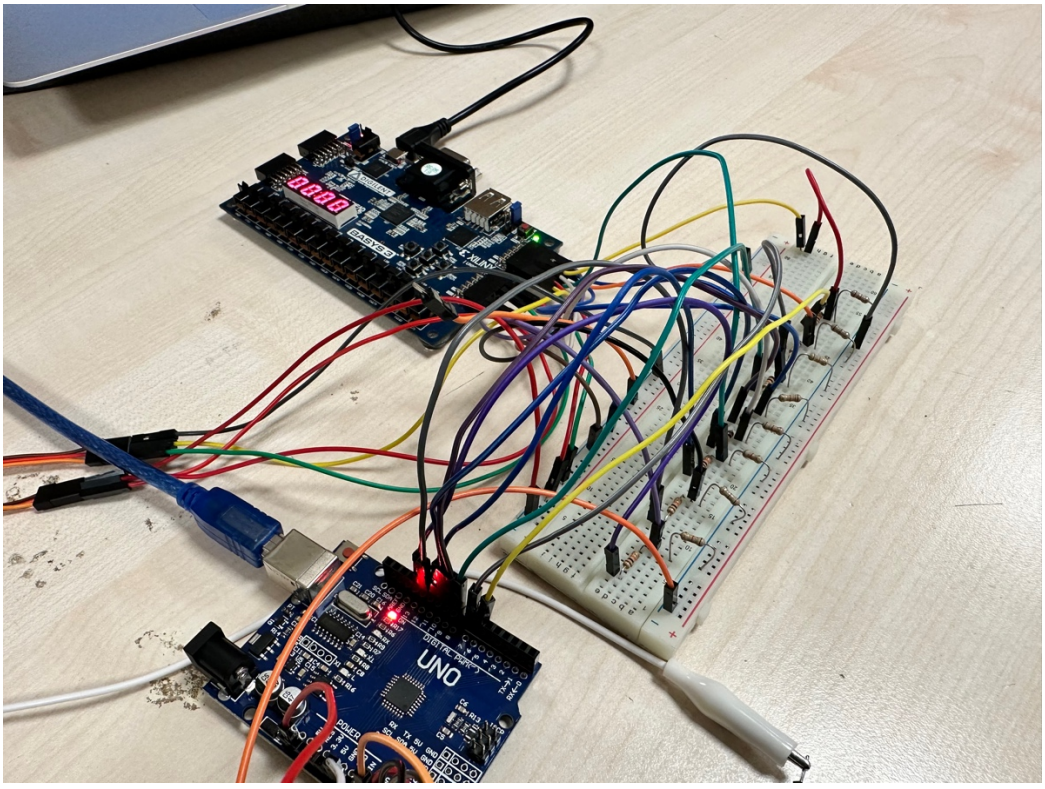

Fig. 14: The Voltage Divider Circuit

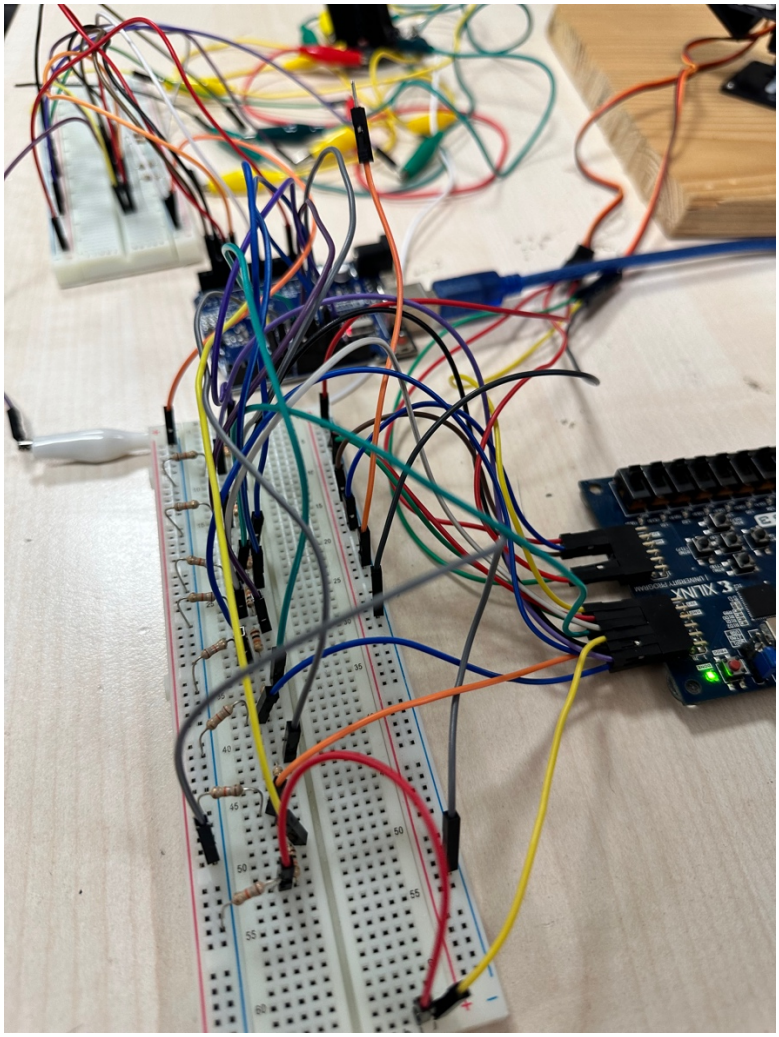

Fig. 15: The Pin Configuration between the Voltage Divider Circuit and BASYS3

Appendix B

project1.ino

```
void setup() {

  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(6, OUTPUT);
  Serial.begin(9600);

}

void loop() {

  int rightldr = analogRead(A2);
  int leftldr = analogRead(A3);
  int upperldr = analogRead(A0);
  int downldr = analogRead(A1);
  Serial.print("  rightldr = "); Serial.print(rightldr);
  Serial.print(" leftldr = "); Serial.print(leftldr);
  Serial.print("  upperldr = "); Serial.print(upperldr);
  Serial.print(" downldr = "); Serial.println(downldr);
  delay(1500);
  digitalWrite(13,LOW);
  digitalWrite(12,LOW);
  digitalWrite(11,LOW);
  digitalWrite(10,LOW);
  digitalWrite(9,LOW);
  digitalWrite(8,LOW);
  digitalWrite(7,LOW);
  digitalWrite(6,LOW);


  if(rightldr > 500){
    digitalWrite(13,HIGH);
    digitalWrite(12,HIGH);
  }
  else if(rightldr > 300){
    digitalWrite(13,HIGH);
    digitalWrite(12,LOW);
  }
  else if(rightldr > 200){
    digitalWrite(13,LOW);
    digitalWrite(12,HIGH);
  }
```

```
else if(rightldr > 120){
  digitalWrite(13,LOW);
  digitalWrite(12,LOW);
}

if(leftldr > 500){
  digitalWrite(11,HIGH);
  digitalWrite(10,HIGH);
}
else if(leftldr > 300){
  digitalWrite(11,HIGH);
  digitalWrite(10,LOW);
}
else if(leftldr > 200){
  digitalWrite(11,LOW);
  digitalWrite(10,HIGH);
}
else if(leftldr > 120){
  digitalWrite(11,LOW);
  digitalWrite(10,LOW);
}

if(upperldr > 500){
  digitalWrite(9,HIGH);
  digitalWrite(8,HIGH);
}
else if(upperldr > 300){
  digitalWrite(9,HIGH);
  digitalWrite(8,LOW);
}
else if(upperldr > 200){
  digitalWrite(9,LOW);
  digitalWrite(8,HIGH);
}
else if(upperldr > 120){
  digitalWrite(9,LOW);
  digitalWrite(8,LOW);
}

if(downldr > 500){
  digitalWrite(7,HIGH);
  digitalWrite(6,HIGH);
}
else if(downldr > 300){
  digitalWrite(7,HIGH);
  digitalWrite(6,LOW);
}
else if(downldr > 200){
  digitalWrite(7,LOW);
  digitalWrite(6,HIGH);
}
```

```
  else if(downldr > 120){
    digitalWrite(7,LOW);
    digitalWrite(6,LOW);
  }

  Serial.print("  The corresponding number is11 = ");
  Serial.println(digitalRead(13));
  Serial.print("  The corresponding number is12 = ");
Serial.println(digitalRead(12));
  Serial.print("  The corresponding number is21 = ");
Serial.println(digitalRead(11));
  Serial.print("  The corresponding number is22 = ");
Serial.println(digitalRead(10));
  Serial.print("  The corresponding number is31 = ");
Serial.println(digitalRead(9));
  Serial.print("  The corresponding number is33 = ");
Serial.println(digitalRead(8));
  Serial.print("  The corresponding number is41 = ");
Serial.println(digitalRead(7));
  Serial.print("  The corresponding number is44 = ");
Serial.println(digitalRead(6));
  delay(50);
```

topmodule.vhd

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


entity topmodule is
    Port(
    clk : in std_logic;
     manual_rotate: in std_logic_vector(3 downto 0);
        position : in std_logic_vector(3 downto 0);
         vertical_pos : in std_logic_vector(3 downto 0);
         led15, led14, led13, led12 : out std_logic;
        servo_h : out std_logic;
        servo_v : out std_logic;
        led_anode : out STD_LOGIC_VECTOR (3 downto 0);
         led_cathode : out STD_LOGIC_VECTOR (6 downto 0)

    );
end topmodule;

architecture Behavioral of topmodule is

signal res: std_logic;
```

```
signal state: std_logic_vector(3 downto 0);
component servo2control is
Port (clk: in std_logic;
     rotate_state2: in std_logic_vector(3 downto 0);
     serv_motor_output: out std_logic);
end component;

component servocontrol is
Port (clk: in std_logic;

     rotate_state: in std_logic_vector(3 downto 0);
     serv_motor_output: out std_logic);
end component;

Component topmodule_sevenseg is
   Port( clk : in STD_LOGIC;
 res : in STD_LOGIC;
 state: in STD_LOGIC_vector (3 downto 0);
 led_anode : out STD_LOGIC_VECTOR (3 downto 0);
 led_cathode : out STD_LOGIC_VECTOR (6 downto 0));
end component;

begin

servo_vertical: servovertical Port Map(clk => clk, rotate_state2 => vertical_pos,
serv_motor_output => servo_v);
servo_horizontal: servohorizontal Port Map(clk => clk, rotate_state => horizontal_pos,
serv_motor_output => servo_h);
sevensegment_display: topmodule_sevenseg Port Map( clk =>clk,res=> res, state=>position,
led_anode=>led_anode, led_cathode=>led_cathode);
process(clk ) begin
   if rising_edge(clk) then
      if (position = "1100") then
         led15 <= '1';
         led14 <= '1';
      elsif (position = "1000") then
                  led15 <= '1';
                  led14 <= '1';
      elsif (position = "0100") then
                  led15 <= '1';
                  led14 <= '0';
      elsif (position = "0011") then
               led13 <= '1';
               led12 <= '1';
      elsif (position = "0010") then
            led13 <= '1';
            led12 <= '1';
      elsif (position = "0001") then
           led13 <= '0';
           led12 <= '1';
```

```
        else
            led13 <= '0';
            led12 <= '0';
            led14 <= '0';
            led15 <= '0';
        end if;
    end if;
end process;

end Behavioral;
```

servohorizontal.vhd

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


entity servohorizontal is
Port (clk: in std_logic;

      rotate_state: in std_logic_vector(3 downto 0);
      manual_rotate: in std_logic_vector(3 downto 0);
      serv_motor_output: out std_logic);
end servohorizontal;

architecture Behavioral of servohorizontal is

signal number_pwm: integer:=0;

begin

process(rotate_state,clk,number_pwm) begin
if rising_edge(clk) then
number_pwm<=number_pwm+1;

    if rotate_state="1100" then

        if number_pwm<60000 then
            serv_motor_output<='1';
        elsif number_pwm<2000000 then
            serv_motor_output<='0';
        else
            number_pwm<=0;
        end if;

    elsif rotate_state ="1000" then

        if number_pwm<85000 then
            serv_motor_output<='1';
        elsif number_pwm<2000000 then
```

```
            serv_motor_output<='0';
        else
            number_pwm<=0;
        end if;

        elsif rotate_state ="0100" then

                if number_pwm<105000 then
                    serv_motor_output<='1';
                elsif number_pwm<2000000 then
                    serv_motor_output<='0';
                else
                    number_pwm<=0;
                end if;
        elsif rotate_state ="0001" then

                        if number_pwm<140000 then
                            serv_motor_output<='1';
                        elsif number_pwm<2000000 then
                            serv_motor_output<='0';
                        else
                            number_pwm<=0;
                        end if;

    elsif rotate_state ="0010" then

        if number_pwm<180000 then
            serv_motor_output<='1';
        elsif number_pwm<2000000 then
            serv_motor_output<='0';
        else
            number_pwm<=0;
        end if;

    elsif rotate_state ="0011" then

        if number_pwm<205000 then
            serv_motor_output<='1';
        elsif number_pwm<2000000 then
            serv_motor_output<='0';
        else
            number_pwm<=0;
        end if;
--else
--        if manual_rotate(0) = '1' then
--            if number_pwm<140000 then
--                                serv_motor_output<='1';
--        elsif manual_rotate(1) = '1' then
--                if number_pwm<180000 then
--                                 serv_motor_output<='1';
```

```
--      elsif manual_rotate(2) = '1' then
--            if number_pwm<90000 then
--                              serv_motor_output<='1';
--       elsif manual_rotate(3) = '1' then
--            if number_pwm<70000 then
--                  serv_motor_output<='1';
--       end if;
----    elsif rotate_state ="011" then

----       if number_pwm<160000 then
----          serv_motor_output<='1';
----       elsif number_pwm<2000000 then
----          serv_motor_output<='0';
----       else
----          number_pwm<=0;
----       end if;

----    elsif rotate_state ="010" then

----       if number_pwm<175000 then
----          serv_motor_output<='1';
----       elsif number_pwm<2000000 then
----          serv_motor_output<='0';
----       else
----          number_pwm<=0;
----       end if;

----    elsif rotate_state ="001" then

----       if number_pwm<190000 then
----          serv_motor_output<='1';
----       elsif number_pwm<2000000 then
----          serv_motor_output<='0';
----       else
----          number_pwm<=0;
----       end if;

----    elsif  rotate_state ="000" then

----       if number_pwm<200000 then
----          serv_motor_output<='1';
----       elsif number_pwm<2000000 then
----          serv_motor_output<='0';
----       else
----          number_pwm<=0;
----       end if;


end if;
```

```
end if;
end process;

end Behavioral;
```

servovertical.vhd

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


entity servovertical is
Port (
   clk: in std_logic;
     rotate_state2: in std_logic_vector(3 downto 0);
     serv_motor_output: out std_logic);
end servovertical;

architecture Behavioral of servovertical is

signal number_pwm: integer:=0;

begin

process(rotate_state2,clk,number_pwm) begin
if rising_edge(clk) then
number_pwm<=number_pwm+1;

   if rotate_state2="1100" then

      if number_pwm<98000 then
         serv_motor_output<='1';
      elsif number_pwm<2000000 then
         serv_motor_output<='0';
      else
         number_pwm<=0;
      end if;

   elsif rotate_state2 ="1000" then

      if number_pwm<120000 then
         serv_motor_output<='1';
      elsif number_pwm<2000000 then
         serv_motor_output<='0';
      else
         number_pwm<=0;
      end if;
```

```
    elsif rotate_state2 ="0001" then

        if number_pwm<200000 then
           serv_motor_output<='1';
        elsif number_pwm<2000000 then
           serv_motor_output<='0';
        else
           number_pwm<=0;
        end if;
    elsif rotate_state2 ="0010" then

              if number_pwm<220000 then
                 serv_motor_output<='1';
              elsif number_pwm<2000000 then
                 serv_motor_output<='0';
              else
                 number_pwm<=0;
              end if;
    elsif rotate_state2 ="0011" then

        if number_pwm<240000 then
           serv_motor_output<='1';
        elsif number_pwm<2000000 then
           serv_motor_output<='0';
        else
           number_pwm<=0;
        end if;

--    elsif rotate_state ="011" then

--        if number_pwm<160000 then
--           serv_motor_output<='1';
--        elsif number_pwm<2000000 then
--           serv_motor_output<='0';
--        else
--           number_pwm<=0;
--        end if;

--    elsif rotate_state ="010" then

--        if number_pwm<175000 then
--           serv_motor_output<='1';
--        elsif number_pwm<2000000 then
--           serv_motor_output<='0';
--        else
--           number_pwm<=0;
--        end if;

--    elsif rotate_state ="001" then
```

```
--       if number_pwm<190000 then
--          serv_motor_output<='1';
--       elsif number_pwm<2000000 then
--          serv_motor_output<='0';
--       else
--          number_pwm<=0;
--       end if;

--    elsif  rotate_state ="000" then

--       if number_pwm<200000 then
--          serv_motor_output<='1';
--       elsif number_pwm<2000000 then
--          serv_motor_output<='0';
--       else
--          number_pwm<=0;
--       end if;

   end if;

end if;
end process;

end Behavioral;
```

topmodule_sevenseg.vhd

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




entity topmodule_sevenseg is
 Port ( clk : in STD_LOGIC;
 res : in STD_LOGIC;
 state: in STD_LOGIC_vector(3 downto 0);
 led_anode : out STD_LOGIC_VECTOR (3 downto 0);
 led_cathode : out STD_LOGIC_VECTOR (6 downto 0));
end topmodule_sevenseg;
architecture rtl of topmodule_sevenseg is
signal phase_control: STD_LOGIC_VECTOR (1 downto 0);
signal sec_control: STD_LOGIC;
begin
clock_control: entity work.counter(rtlclock)
 port map(clk => clk,
 res => res,
 phase => phase_control,
 sec => sec_control);
```

```
led_display: entity work.leds(rtl_display)
 port map(clk => clk,
 res => res,
 phase => phase_control,
 sec => sec_control,
 state => state,
 led_anode => led_anode,
 led_cathode => led_cathode);
end rtl;
```

counter.vhd

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



entity counter is
Port ( clk : in STD_LOGIC;
res : in STD_LOGIC;
phase : out STD_LOGIC_VECTOR (1 downto 0);
sec : out STD_LOGIC );
end counter;

architecture rtlclock of counter is
signal state_count: STD_LOGIC_VECTOR (27 downto 0);
signal phasor: STD_LOGIC_VECTOR (1 downto 0);
begin
process(clk,res) begin
if res = '1' then
state_count <= (others => '0');
else
if rising_edge(clk) then
state_count <= state_count + '1';
if state_count =x"5F5E0FF" then
state_count <= (others => '0');
end if;
end if;
end if;
end process;
-------------------------------------------------------------------
sec <= '1' when state_count =x"5F5E0FF" else '0';
phase <= state_count(19 downto 18);
-------------------------------------------------------------------
end rtlclock;
```

leds.vhd

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

entity leds is
 Port (clk:in STD_LOGIC;
 res: in STD_LOGIC;
 phase: in STD_LOGIC_VECTOR (1 downto 0);
 sec: in STD_LOGIC;
 state: in std_logic_vector (3 downto 0);
 led_anode : out STD_LOGIC_VECTOR (3 downto 0);
 led_cathode: out STD_LOGIC_VECTOR (6 downto 0));
end leds;
----------------------------------------------------------------------
architecture rtl_display of leds is
signal led_number: STD_LOGIC_VECTOR (15 downto 0);
signal led_select: STD_LOGIC_VECTOR (3 downto 0);
begin
----------------------------------------------------------------------
 process(phase, led_number) begin
 case phase is
 when "00" => led_anode <= "0111";
 led_select <= led_number(15 downto 12);
 when "01" => led_anode <= "1011";
 led_select <= led_number(11 downto 8);
 when "10" => led_anode <= "1101";
 led_select <= led_number(7 downto 4);
 when "11" => led_anode <= "1110";
 led_select <= led_number(3 downto 0);
 when others => led_anode <= "0000";

 end case;
 end process;
----------------------------------------------------------------------
 process(clk,res) begin
 if res = '1' then
 led_number <= (others => '0');
 elsif rising_edge(clk) then
 if state = "1100" then --  90
 led_number <= "0000000010010000";
 elsif state = "1000" then -- 45
  led_number <= "0000000001000101";
  elsif state = "0100" then --
  led_number <= "0000010001010000";
  elsif state = "0011" then -- 120
  led_number <= "0000000010010000";
  elsif state = "0010" then -- 120
  led_number <= "0000000001000101";
  elsif state = "0001" then -- 120
    led_number <= "0000000001000101";
```

```
 else
 led_number <= "0000000000000000";



 end if;
 end if;
 end process;
 ----------------------------------------------------------------------
 process(led_select) begin
 case led_select is --1234567
 when "0000" => led_cathode <= "0000001"; -- "0"
 when "0001" => led_cathode <= "1001111"; -- "1"
 when "0010" => led_cathode <= "0010010"; -- "2"
 when "0011" => led_cathode <= "0000110"; -- "3"
 when "0100" => led_cathode <= "1001100"; -- "4"
 when "0101" => led_cathode <= "0100100"; -- "5"
 when "0110" => led_cathode <= "0100000"; -- "6"
 when "0111" => led_cathode <= "0001111"; -- "7"
 when "1000" => led_cathode <= "0000000"; -- "8"
 when "1001" => led_cathode <= "0000100"; -- "9"
 when others => led_cathode <= "1111111";
 end case;
 end process;
----------------------------------------------------------------------
end rtl_display;
```