EEE102 INTRODUCTION TO DIGITAL DESIGN

TERM PROJECT FINAL REPORT

PLANT-CARING SYSTEM

Name: Burak Gökkaya

Student ID: 22202209

Section: 001

Instructor: Volkan Kurşun

1.Purpose of the Project:

The aim of the project was to enhance digital design skills and VHDL. For that we are wanted to create an original project with using BASYS3 so that we can improve our abilities. Hence in this project, I wanted to provide proper conditions for a plant. Plant caring is hard thing because it is affected from a lot of factors. Most important things for plant are:

-Air Humidity

-Soil Humidity

-Temperature and

-Wind Condition.

Hence, I wanted to take ideal conditions for plant from the user and compare them with the environment conditions. If there is anormal situation about these conditions, my project will give the signal about it. For that I understand the mechanism behind the sensors and BASYS3.

2. DESIGN SPECIFICATION

For this project I used these components:

1. BASYS3 FPGA Board
2. DHT11 Temperature and Humidity Sensor
3. HCSR-04 Ultrasonic Distance Sensor
4. Soil Moisture Sensor
5. Power Supply
6. Breadboard
7. Bunch of Jumper Cables
8. Crocodile Cables
9. Arduino UNO

In this project, main component was BASYS3. BASYS3 is a digital board, and it is working with 3.3V. So that I tried to choose sensors which gives digital signals. DHT11 is a sensor which calculating air temperature and humidity. It gives digital signal and works with +5V. It measures 0-50°C temperature range and %20-%90 humidity range and it is working with single-wire serial interface, and it made my work harder. HCSR 04 is also working with +5V and it gives digital signal. Its ranging distance was 2-400 cm. It is sending sound wave and calculating its return time. Then it is calculating distance from d=v*t equation. Lastly, I used the Soil Moisture Sensor. It is operating at 3.3V to 5V and it gives analog and digital output. But using digital output was hard. There is potentiometer on the sensor, and you should adjust it for the threshold humidity value. I couldn't make

it and also it wasn't proper for my project. I wanted to changeable threshold value hence I decided to use its analog output. For that I used Arduino UNO for convert analog signal to digital signal (ADC).

On the other hand, I defined BASYS3 pins as inputs and option values. With help of these pins, users can decide the threshold values also I put the reset button, and it is resetting the system. Lastly, I decide the LEDs on the BASYS3 according to conditions.

3.METHODOLOGY

My sensors were working with 5V hence I decided to use extra power supply from outside. Hence, I used the power supply from laboratory.

Firstly, I started to make wind warning. For that I could use wind sensors, but they were expensive, and I thought I can figure out with another way. So, I decided to use HCSR04 and with this way I care about cost efficiency. I just wanted to take a signal for if there is an object in front of the HCSR-04. So that I designed a mechanism for this. For using HCSR-04, I read the datasheet of it. According to datasheet firstly I should set trigger high and then it sends the ultrasonic burst. When burst is sent ECHO pin is turn to high. This pin remains high until the burst hit the sensor again. While this process, a counter will count the time until ECHO goes low. Hence, we can calculate the distance from the object. We know speed of sound is 343 meters per second and we can write d=v*t. And if we want to calculate the distance with centimeter unit, this equation turns the d= time/58. With this way, we can use HCSR04 and calculate the distance. But 400 cm is very long distance for my project, and I decided the use threshold value at 30 cm. If there is an object in range 30 cm, wind warning will be high; else it will turn low.

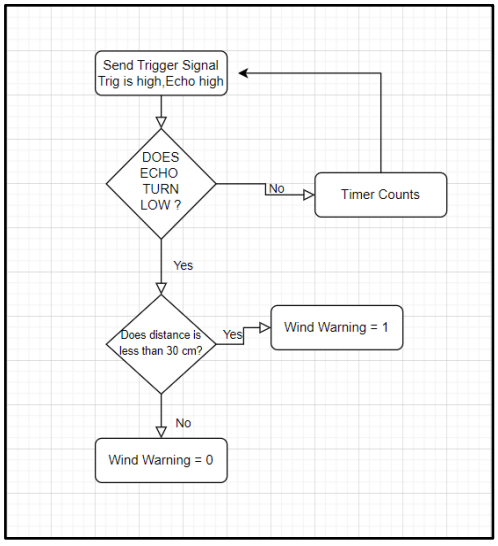


Figure 1: Working scheme of wind warning

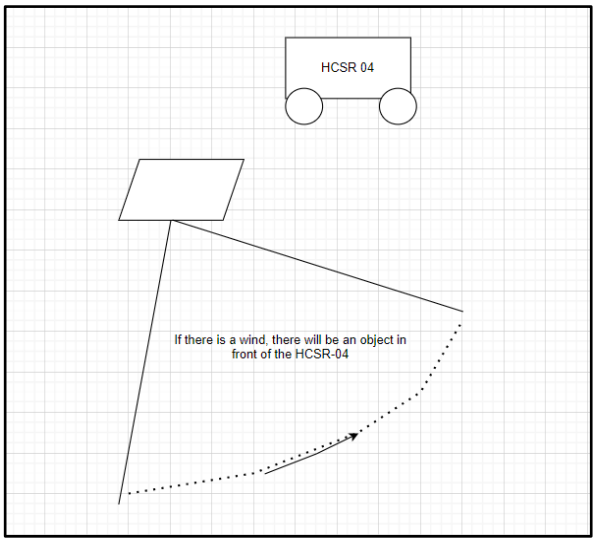


Figure 2: Wind warning mechanism

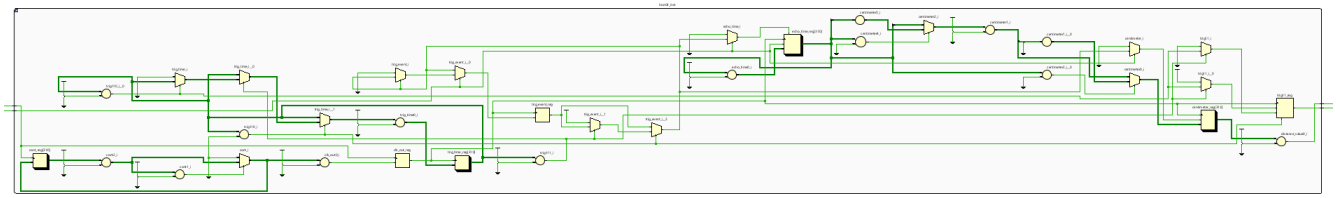


Figure 3: Elaborated Design of HC-SR04 Module

Then I started to design for DHT11. It was hardest part because it was working as single-wire interface so I should also send direction signal. At the beginning I couldn't understand this mechanism then after some research I found a paper which explain that mechanism with state transition diagram. I couldn't change a lot about state transition diagram because DHT11 work in a same way for everybody. But I added to measure the humidity part. After reading this report, datasheet started to make sense. To summarize, we send signal to start DHT11 and wait to complete the start signal. Once it is completed, DHT11 response it. DHT11 send 40-bit signal. First 8-bit integral RH data and then there is 8-bit decimal RH data, Then, there is 8-bit integral T data and 8-bit decimal T data. Also, we can check the data if it is true with last 8 bit. Hence at state 9; we are checking this. I used DHT11 for measuring temperature and air humidity then I displayed them on the Seven Segment Display for that firstly I turned 8-bit binary number to 2-bit hexadecimal number and then I turned hexadecimal number to decimal number. And then I displayed that on the SSD.

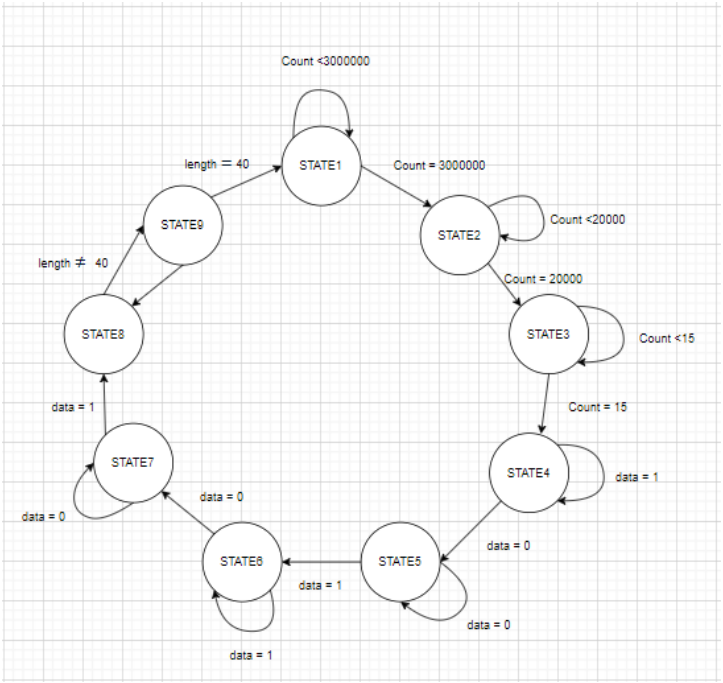


Figure 4: State Transition Diagram for DHT11

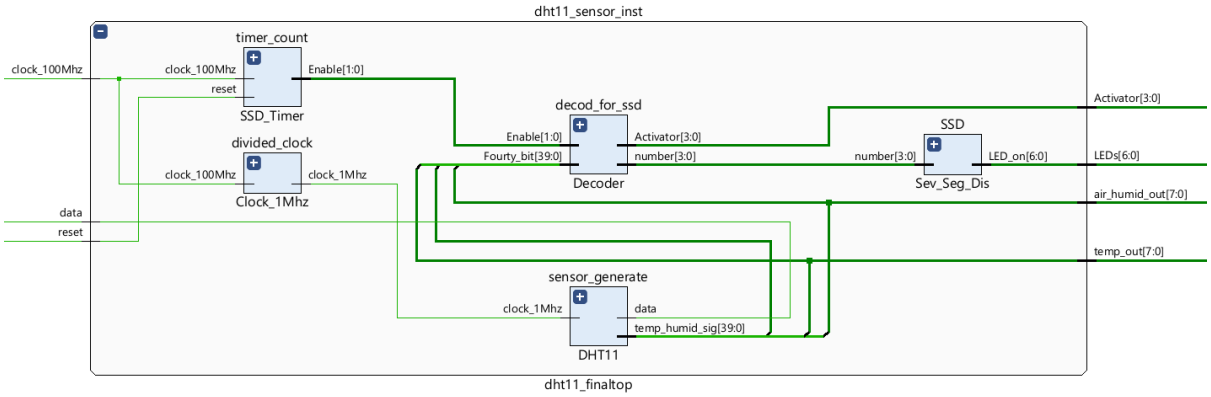


Figure 5: DHT11 Main Module

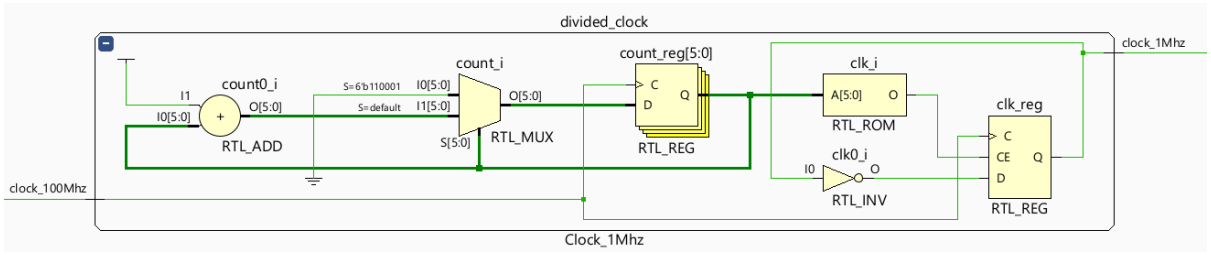


Figure 6: DHT11 Clock Divider

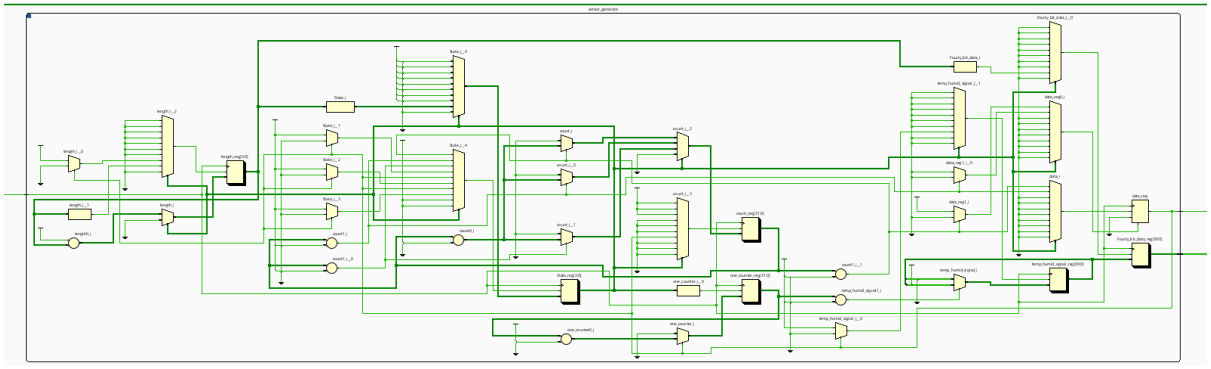


Figure 7: DHT11 Sensor Generator

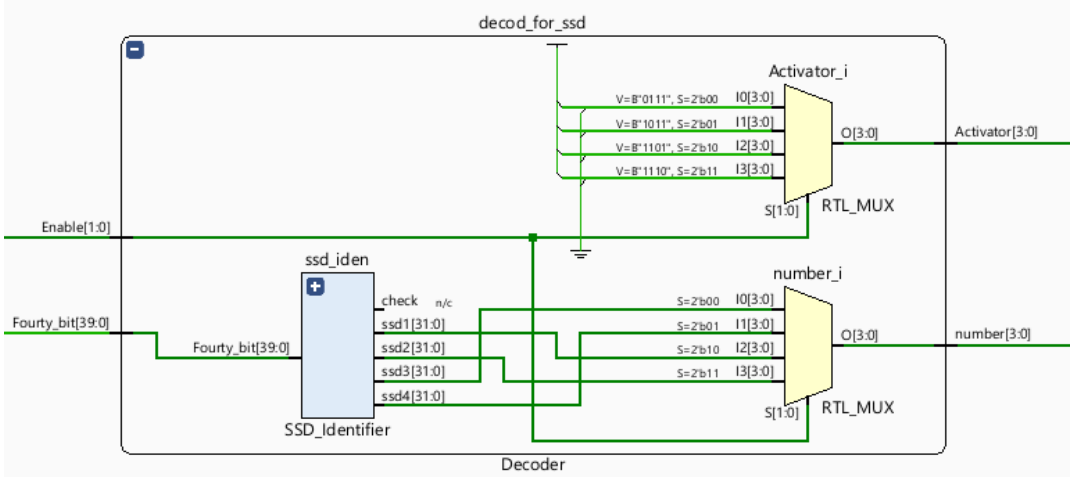


Figure 8: DHT11 Decoder for Seven Segment Display

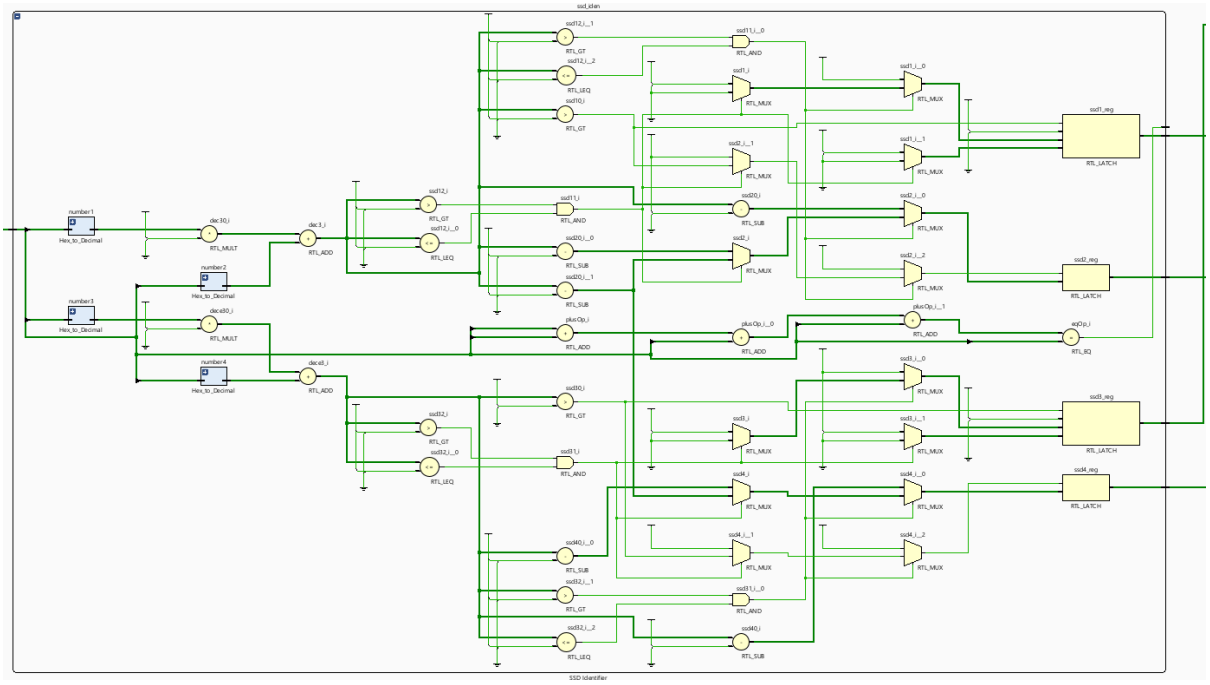


Figure 9: DHT11 Seven Segment Identificier

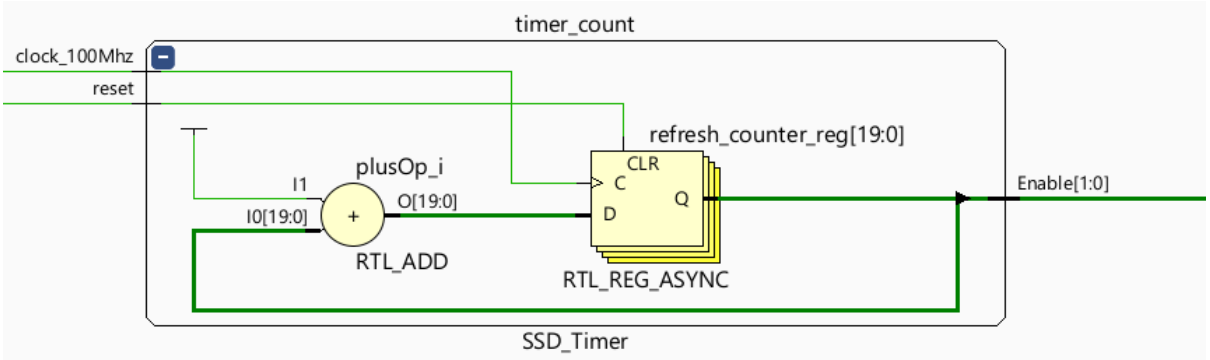


Figure 10:Timer Count

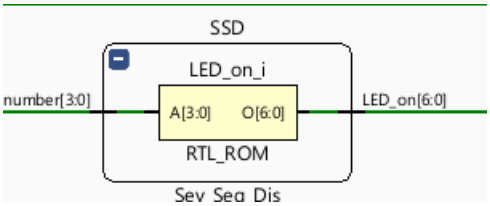


Figure 11: DHT11 SSD Display

At the third step, I took the values from soil moisture sensor. It was given analog hence I turned it to digital values with help of Arduino UNO. I decided the threshold values which are 700 and 500. And according to this it gives digital 1 or 0 signal. And there will be warning according to this signal.

Lastly I combine all of this sources under the top module. I defined option, sw_1 and reset parts hence users can interact with project. With help of option, user will choose which one he wants

to put threshold value for low humid, high humid, low temperature, high temperature. Then user will enter the inputs from the pins. If user want to reset these values, it can be done from reset button.

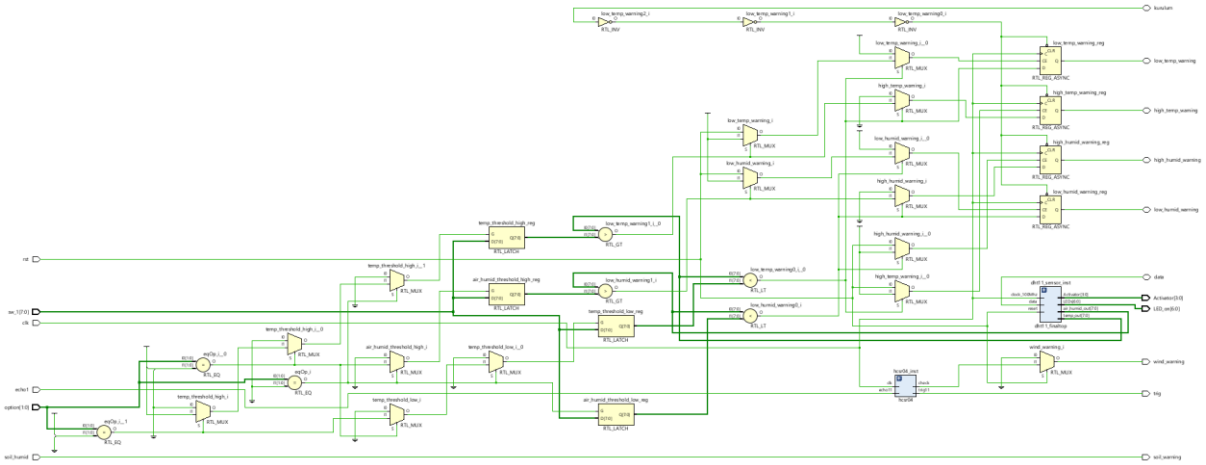


Figure 12: Elaborated Design

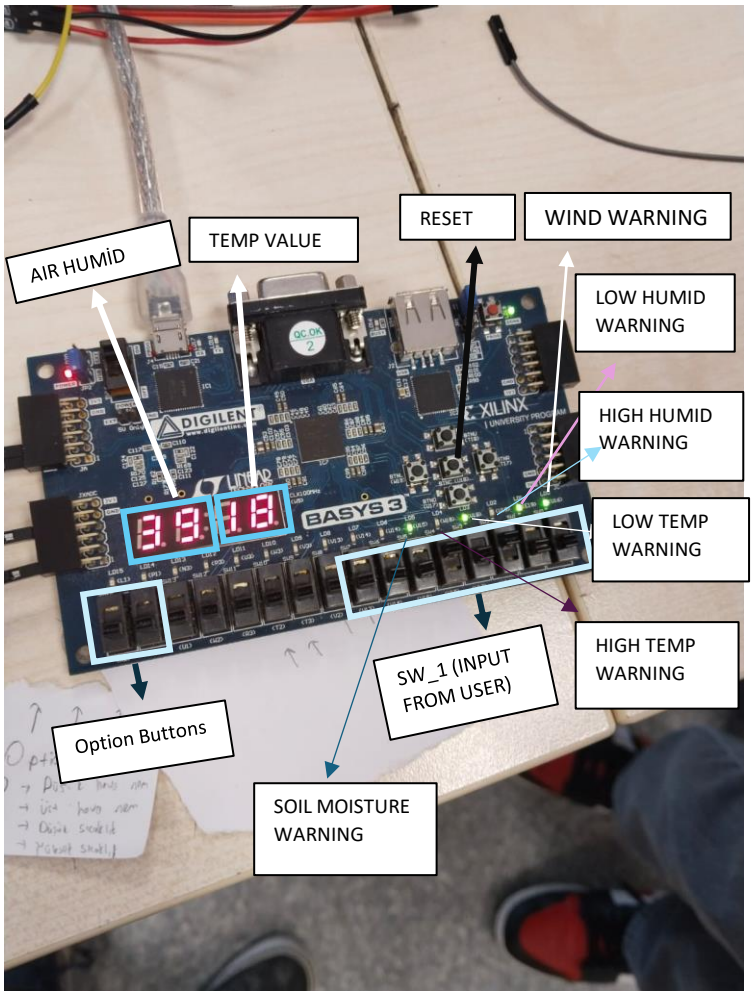


Figure 13: Working mechanism of project

4.CONCLUSION

As a result of this experiment, I learned more about BASYS3, especially ports for taking signals from sensors. Also, I learned there is some voltage problems while using sensors. A lot of sensors working with 5V, but BASYS 3 is working at 3.3V. Hence, I need to fix this problem. In addition, while reading datasheet of sensors I understood the timing mechanism better. I think time parts were hardest part because every sensor works with different frequency but also, they need to work under a top module, so I needed to fix this problem too. But at the end of the project, I managed to use them all working at the same time. Just I didn't use LCD screen which I mentioned in my project proposal as if I need it I will use it. But there wasn't a need. Seven Segment worked properly; hence it would be unnecessary. Also, I didn't mention about I am going to use soil moisture, but I added it in my project. So, there was a lot of sensors, and it made my work complicated. Finally, I used oscilloscope for looking the frequencies and voltages and it was unexpected thing for me, but it enhanced my oscilloscope knowledge. Hence I think, this project was hard, but it was very useful for learning digital systems.

REFERENCES

---

https://pdf1.alldatasheet.com/datasheet-pdf/view/1440068/ETC/DHT11.html

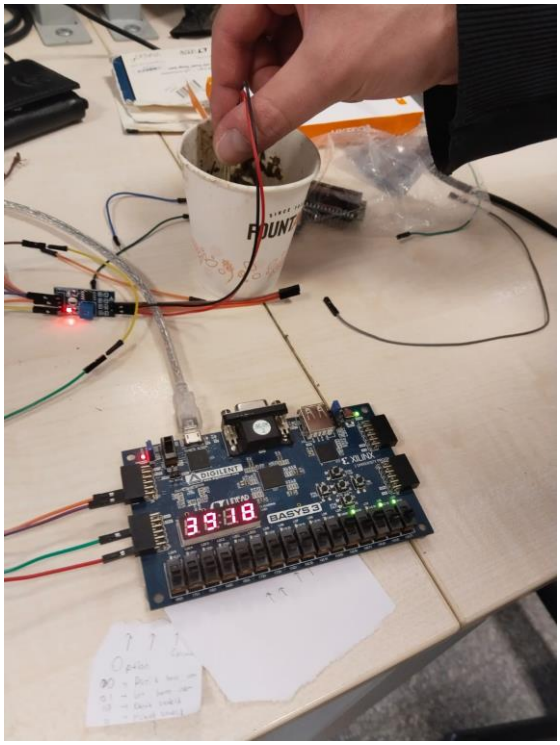
https://pdf1.alldatasheet.com/datasheet-pdf/view/1132204/ETC2/HCSR04.html

https://components101.com/modules/soil-moisture-sensor-module

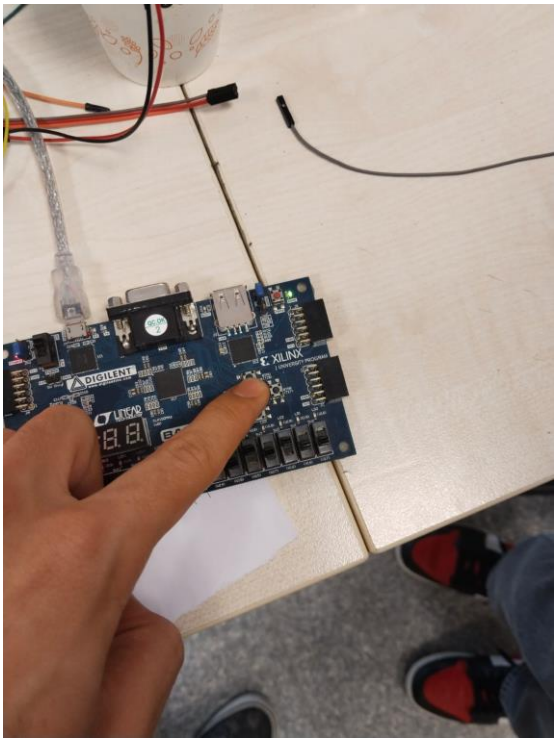
https://github.com/Gavrem/seat_warmer_dht11/blob/main/Project%20Final%20Report.pd

APPENDIX

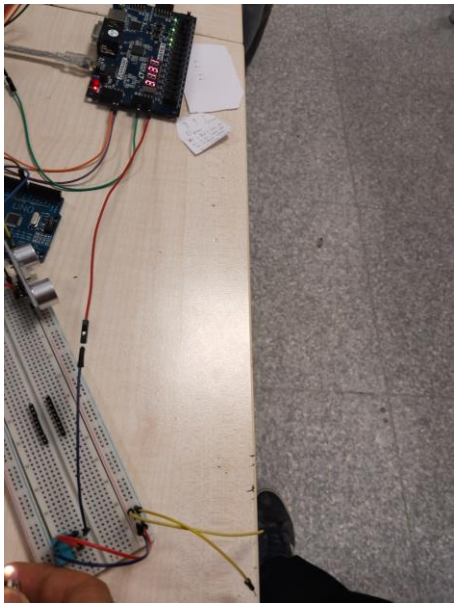
PHOTOS



Appendix 1: Soil Moisture Sensor



Appendix 2: Reset Button Effect



Appendix 3: Temperature while there is a lighter (31°C)

VHDL CODES

MAIN TOP CODE

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use IEEE.std_logic_unsigned.all;
entity top is
   port(
      clk : in std_logic;
      sw_1 : in unsigned(7 downto 0);
      rst : in std_logic;
      data : inout std_logic;
      echo1:in std_logic;
      trig : out std_logic;
      option : in std_logic_vector(1 downto 0);
      wind_warning : out STD_LOGIC;
      low_humid_warning,high_humid_warning,low_temp_warning,high_temp_warning :inout std_logic;
      Activator : out std_logic_vector (3 downto 0);
      LED_on : out std_logic_vector (6 downto 0);
      soil_humid : in std_logic;
      soil_warning: out std_logic;
      kurulum : inout std_logic
   );
end top;

architecture behavior of top is
   component dht11_finaltop is
         port(
      clock_100Mhz : in std_logic;
      reset : in std_logic;
      Activator : out std_logic_vector (3 downto 0);
      LEDs : out std_logic_vector (6 downto 0);
      data : inout std_logic;
```

```
        air_humid_out : inout std_logic_vector(7 downto 0);
        temp_out : inout std_logic_vector(7 downto 0)
            );
end component;
    component hcsr04 is
        port(
            clk: in std_logic;
            echo11: in std_logic;
            trig11: out std_logic;
            check : out std_logic
        );
    end component;
    signal lcd_message: std_logic_vector(2 downto 0);
    signal a: std_logic;
    signal clk1,clk2: std_logic;
    signal harf1,harf2,harf3,harf4 : std_logic_vector(6 downto 0);
    signal light: STD_LOGIC_VECTOR (6 downto 0) := (others => '0');
    signal refrCounter: STD_LOGIC_VECTOR (17 downto 0):= (others => '0');
    signal LEDActivater: std_logic_vector(1 downto 0):= (others => '0');
    signal temp_threshold_low :  unsigned(7 downto 0); -- Sıcaklık alt sınırı
    signal temp_threshold_high : unsigned(7 downto 0); -- Sıcaklık üst sınırı
    signal air_humid_threshold_low :  unsigned(7 downto 0); -- Nem alt sınırı
    signal air_humid_threshold_high :  unsigned(7 downto 0); -- Nem üst sınırı
    signal dht11 : std_logic_vector(39 downto 0);
    signal temp_out, air_humid_out : unsigned(7 downto 0);
begin
clk1 <= clk;
clk2 <= clk;
    dht11_sensor_inst : dht11_finaltop
        port map(
            unsigned(temp_out) => temp_out,
            unsigned(air_humid_out) => air_humid_out,
            data => data,
            clock_100Mhz => clk1,
            reset => rst,
            Activator => Activator,
```

```
        Leds => Led_on
    );
 hcsr04_inst : hcsr04
    port map(
        clk => clk2,
        echo11 => echo1,
        trig11 => trig,
        check => a
       );
temp_out <=  unsigned(dht11(23 downto 16));
air_humid_out <= unsigned(dht11(39 downto 32));
process(option, sw_1)
begin
    if  option = "00" then
        air_humid_threshold_low <= sw_1;
    elsif option = "01" then
        air_humid_threshold_high <= sw_1;
    elsif option = "10" then
        temp_threshold_low <= sw_1;
    else
        temp_threshold_high <= sw_1;
    end if;
end process;
process(rst,temp_out,air_humid_out,kurulum)
begin
    soil_warning <= soil_humid;
    wind_warning <= a;
    if rst = '1' then
        lcd_message <= "000";
        low_temp_warning <= '0';
        high_temp_warning <= '0';
        low_humid_warning <= '0';
        high_humid_warning <= '0';
        wind_warning <= '0';
        air_humid_threshold_low <= "00000000";
        temp_humid_threshold_low <= "00000000";
```

```
            air_humid_threshold_high <= "11111111";
            temp_humid_threshold_high <= "11111111";
        end if;
        if kurulum = '1' then
            if rising_edge(clk) then
                if temp_out < temp_threshold_low then
                    low_temp_warning <= '1';
                elsif temp_out > temp_threshold_high then
                    high_temp_warning <= '1';
                else
                    high_temp_warning <= '0';
                    low_temp_warning <= '0';
                end if;
                ----------
                if air_humid_out < air_humid_threshold_low then
                    low_humid_warning <= '1';
                elsif air_humid_out > air_humid_threshold_high then
                    high_humid_warning <= '1';
                else
                    high_humid_warning <= '0';
                    low_humid_warning <= '0';
                end if;
            end if;
        else
            high_humid_warning <= '0';
            low_humid_warning <= '0';
            high_temp_warning <= '0';
            low_temp_warning <= '0';
        end if;
    end process;
end Behavior;
```

HCSR 04

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
```

```
entity hcsr04 is
port( clk : in std_logic;
  echo11 : in std_logic;
  trig11 : out std_logic;
  check : out std_logic);
end entity;

architecture Behavioral of hcsr04 is
signal sw_int : integer range 0 to 255 := 0;
signal centimeter : integer :=0 ;
signal clk_1MHz : std_logic := '0';
signal clk_out      : std_logic := '0';
signal distance_value: std_logic := '0';
begin

process(clk) -- 100MHz / 100 = 1MHz clock
variable cont: integer:=0;
begin
    if rising_edge(clk) then
        cont := cont + 1;
        if cont = 100 then
          cont:=0;
        end if;

        if cont < 50 then
          clk_out <= '1';
        else
          clk_out <= '0';
        end if;
    end if;
end process;

clk_1MHz <= clk_out;

process(clk_1MHz)
```

```
variable trig_time,echo_time: integer:=0;
variable trig_event :std_logic:='0';
begin
if rising_edge(clk_1MHz) then
if (trig_time = 0) then
trig11 <= '1';
  elsif (trig_time = 10) then          -- trig time
trig11 <= '0';
trig_event := '1';
  elsif (trig_time = 100000) then    -- wait trig
trig_time := 0;
trig11 <= '1';
end if;

trig_time := trig_time+1;

if (echo11 = '1') then
echo_time := echo_time + 1;
elsif (echo11 = '0' and trig_event = '1')then
centimeter <= echo_time / 58;
      echo_time := 0 ;
      trig_event := '0';
  end if;
end if;
sw_int <= centimeter;
if sw_int < 30 then
   distance_value <= '1';
else
   distance_value <= '0';
end if;
check <= distance_value;
end process;
end Behavioral;
```

DHT11 TOP

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

```
entity dht11_finaltop is
Port ( clock_100Mhz : in std_logic;
reset : in std_logic;
Activator : out std_logic_vector (3 downto 0);
LEDs : out std_logic_vector (6 downto 0);
data : inout std_logic;
air_humid_out : inout std_logic_vector(7 downto 0);
temp_out : inout std_logic_vector(7 downto 0)
);
end dht11_finaltop;
architecture Behavioral of dht11_finaltop is
signal Enable : std_logic_vector(1 downto 0);
signal Led : integer;
signal clock1Mhz : std_logic ;
signal temp_humid_sig : std_logic_vector( 39 downto 0);
component DHT11 is
Port(
   clock_1Mhz: in std_logic;
   data: inout std_logic;
   temp_humid_sig: out std_logic_vector(39 downto 0)
   );
end component;
component Decoder is
PORT(
   Enable : in std_logic_vector(1 downto 0);
   Fourty_bit : in STD_LOGIC_VECTOR( 39 downto 0);
   Activator : out std_logic_vector(3 downto 0):="0000";
   number : out integer);
end component;
component Sev_Seg_Dis is
PORT
(number : in integer;
LED_on : out STD_LOGIC_VECTOR (6 downto 0));
end component;
component SSD_Timer is
PORT
```

```
(clock_100Mhz : in STD_LOGIC;
reset : in STD_LOGIC;
Enable: out std_logic_vector(1 downto 0));
end component;
component Clock_1Mhz is
Port(
clock_100Mhz : in std_logic;
clock_1Mhz : out std_logic
);
end component;
begin
sensor_generate : DHT11 port map (clock1Mhz,data,temp_humid_sig);
clock_frequency_change: Clock_1Mhz Port Map (clock_100Mhz, clock1Mhz);
timer_count : SSD_Timer PORT MAP (clock_100Mhz,reset,Enable);
decod_for_ssd : Decoder PORT MAP (Enable,temp_humid_sig,Activator,Led);
SSD : Sev_Seg_Dis PORT MAP (Led,LEDs);
air_humid_out <= temp_humid_sig(39 downto 32);
temp_out <= temp_humid_sig(23 downto 16);
end Behavioral;
```

DHT11_SENSOR_GENERATE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity DHT11 is
    Port (
        clock_1Mhz : in std_logic;
        data : inout std_logic;
        temp_humid_sig : out std_logic_vector(39 downto 0)
    );
end DHT11;
architecture Behavioral of DHT11 is
    signal State : std_logic_vector(3 downto 0) := "0000";
    signal direction: std_logic:= '0';
    signal one_counter : integer := 0;
    signal count : integer := 0;
    signal length : integer range 0 to 40 := 0;
```

```
    signal temp_humid_signal : std_logic_vector(39 downto 0) := (others => '0');
    signal Fourty_bit_data : std_logic_vector(39 downto 0) := (others => '0');
begin
    process(clock_1Mhz)
    begin
        if rising_edge(clock_1Mhz) then
            -- Durumu Kontrol Et --
            case State is
                when "0000" =>
                    count <= count + 1;
                    if count > 3000000 then
                        State <= "0001";
                        data <= '0';
                        direction <= '1';
                        count <= 0;
                    end if;

                when "0001" =>
                    count <= count + 1;
                    if count > 20000 then
                        data <= 'Z';
                        direction <= '0';
                        count <= 0;
                        State <= "0010";
                    end if;

                when "0010" =>
                    count <= count + 1;
                    if count > 15 then
                        count <= 0;
                        State <= "0011";
                    end if;

                when "0011" =>
                    if data = '0' then
                        State <= "0100";
```

```
    end if;

when "0100" =>
    if data = '1' then
        State <= "0101";
    end if;

when "0101" =>
    if data = '0' then
        State <= "0110";
    end if;

when "0110" =>
    if data = '1' then
        State <= "0111";
        count <= 0;
    end if;

when "0111" =>
    one_counter <= one_counter + 1;
    if data = '0' then
        if one_counter < 50 then
            temp_humid_signal <= temp_humid_signal(38 downto 0) & '0';
        else
            temp_humid_signal <= temp_humid_signal(38 downto 0) & '1';
        end if;
        length <= length + 1;
        one_counter <= 0;
        State <= "1000";
    end if;

when "1000" =>
    if length = 40 then
        Fourty_bit_data <= temp_humid_signal;
        State <= "0000";
        length <= 0;
```

```
                else
                    State <= "0101";
                end if;

            when others =>
                State <= "0000";
        end case;
    end if;
  end process;
  temp_humid_sig <= Fourty_bit_data;
end Behavioral;
```

CLOCK_DIVIDER

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

use IEEE.NUMERIC_STD.ALL;
entity Clock_1Mhz is
Port (
clock_100Mhz : in std_logic;
clock_1Mhz : out std_logic
);
end Clock_1Mhz;
architecture Behavioral of Clock_1Mhz is
signal clk : std_logic:= '0';
signal count: integer range 0 to 49:= 0;
begin
process(clock_100Mhz)
begin
if rising_edge(clock_100Mhz) then
if count = 49 then
count <= 0;
clk <= not clk;
else
count <= count + 1;
end if;
end if;
```

```
end process;
clock_1Mhz <= clk;
end Behavioral;
```

SSD_TIMER

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;
use IEEE.NUMERIC_STD.ALL;
entity SSD_Timer is
Port ( clock_100Mhz : in STD_LOGIC;
reset : in STD_LOGIC;
Enable: out std_logic_vector(1 downto 0));
end SSD_Timer;
architecture Behavioral of SSD_Timer is
signal refresh_counter: STD_LOGIC_VECTOR (19 downto 0):= "00000000000000000000";
begin
process(clock_100Mhz,reset)
begin
if(reset='1') then
refresh_counter <= (others => '0');
elsif(rising_edge(clock_100Mhz)) then
refresh_counter <= refresh_counter + 1;
end if;
end process;

Enable <= refresh_counter(19 downto 18);
end Behavioral;
```

DECODER_FOR_SSD

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

entity Decoder is
    Port (
        Enable : in std_logic_vector(1 downto 0);
```

```
        Fourty_bit : in STD_LOGIC_VECTOR(39 downto 0);
        Activator : out std_logic_vector(3 downto 0) := "0000";
        number : out integer range 0 to 10
    );
end Decoder;

architecture Behavioral of Decoder is
    signal check, ssd1, ssd2,ssd3,ssd4 : integer;

    component SSD_Identifier is
        Port (
            Fourty_bit : in STD_LOGIC_VECTOR(39 downto 0);
            check : out integer range 0 to 1;
            ssd2 : out integer;
            ssd1 : out integer;
            ssd3: out integer;
            ssd4: out integer
        );
    end component;

begin
    ssd_iden : SSD_Identifier port map(Fourty_bit, check, ssd2, ssd1,ssd3,ssd4);

    process(Enable, Fourty_bit)
    begin
        case Enable is
            when "00" =>
                Activator <= "0111";
                number <= ssd3;
            when "01" =>
                Activator <= "1011";
                number <= ssd4;
            when "10" =>
                Activator <= "1101";
                number <= ssd1;
            when "11" =>
```

```
            Activator <= "1110";
            number <= ssd2;
        when others =>
            Activator <= "0000";
            number <= 0;
    end case;
  end process;

end Behavioral;
```

SSD IDENTIFIER

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;
entity SSD_Identifier is
Port ( Fourty_bit : in STD_LOGIC_VECTOR( 39 downto 0);
check :out integer range 0 to 1;
ssd1 :out integer;
ssd2 :out integer;
ssd3 :out integer;
ssd4 :out integer );
end SSD_Identifier;
architecture Behavioral of SSD_Identifier is
signal temp1 : std_logic_vector( 3 downto 0 );
signal temp2 : std_logic_vector( 3 downto 0 );
signal humid1 : std_logic_vector( 3 downto 0 );
signal humid2 : std_logic_vector( 3 downto 0 );
signal ctrl : std_logic_vector( 7 downto 0 );
signal ctrl1 : std_logic_vector( 7 downto 0 );
signal ctrl2 : std_logic_vector( 7 downto 0 );
signal ctrl3 : std_logic_vector( 7 downto 0 );
signal ctrl4 : std_logic_vector( 7 downto 0 );
signal dec1,dec2,dec3,dece1,dece2,dece3 : integer range 0 to 30;
component Hex_to_Decimal is
Port ( four_bit : in STD_LOGIC_VECTOR (3 downto 0);
decimal : out integer range 0 to 15 );
end component;
```

```
begin
ctrl1 <= Fourty_bit( 39 downto 32);
ctrl2 <= Fourty_bit( 31 downto 24);

ctrl3 <= Fourty_bit( 23 downto 16);
ctrl4 <= Fourty_bit( 15 downto 8);
ctrl <= Fourty_bit( 7 downto 0);
humid1 <=  Fourty_bit(39 downto 36);
humid2 <=  Fourty_bit(39 downto 36);
temp1 <= Fourty_bit(23 downto 20);
temp2 <= Fourty_bit(19 downto 16);
number1: Hex_to_Decimal port map(temp1,dec1);
number2: Hex_to_Decimal port map(temp2,dec2);
number3: Hex_to_Decimal port map(humid1,dece1);
number4: Hex_to_Decimal port map(humid2,dece2);
dec3 <= (dec1*16) + dec2;
dece3 <= (dece1*16)+dece2;
process (dec3)
begin
if dec3 > 9 and dec3 <= 19 then
ssd1<= 1;
ssd2<= dec3 - 10;
elsif dec3 > 19 and dec3 <= 29 then
ssd1<= 2;
ssd2<= dec3 - 20;
elsif dec3 > 29 and dec3 <= 39 then
ssd1<= 3;
ssd2<= dec3 - 30;
end if;
end process;
process (dec3)
begin
if dece3 > 9 and dece3 <= 19 then
ssd3<= 1;
ssd4<= dece3 - 10;
elsif dece3 > 19 and dece3 <= 29 then
```

```
ssd3<= 2;
ssd4<= dece3 - 20;
elsif dece3 > 29 and dece3 <= 39 then
ssd3<= 3;
ssd4<= dec3 - 30;
end if;
end process;
check <= 1 when (ctrl1 + ctrl2 +ctrl3 +ctrl4 ) = ctrl else 0;
end Behavioral;
```

HEX TO DECIMAL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Hex_to_Decimal is
Port (four_bit: in STD_LOGIC_VECTOR (3 downto 0);
decimal: out integer range 0 to 15);
end Hex_to_Decimal;
architecture Behavioral of Hex_to_Decimal is
begin
process(four_bit)
begin
case four_bit is
when "0000" => decimal <= 0;
when "0001" => decimal <= 1;
when "0010" => decimal <= 2;
when "0011" => decimal <= 3;
when "0100" => decimal <= 4;
when "0101" => decimal <= 5;
when "0110" => decimal <= 6;
when "0111" => decimal <= 7;
when "1000" => decimal <= 8;
when "1001" => decimal <= 9;
when "1010" => decimal <= 10;
when "1011" => decimal <= 11;
when "1100" => decimal <= 12;
when "1101" => decimal <= 13;
when "1110" => decimal <= 14;
```

```
when "1111" => decimal <= 15;
when others => decimal <= 0;
end case;
end process;
end Behavioral;
```

SEVEN SEGMENT DISPLAY

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Sev_Seg_Dis is
Port (number: in integer range 0 to 10;
LED_on: out STD_LOGIC_VECTOR (6 downto 0));
end Sev_Seg_Dis;
architecture Behavioral of Sev_Seg_Dis is
begin
process(number)
begin
case number is

when 0 => LED_on <= "0000001"; -- "0"
when 1 => LED_on <= "1001111"; -- "1"
when 2 => LED_on <= "0010010"; -- "2"
when 3 => LED_on <= "0000110"; -- "3"
when 4 => LED_on <= "1001100"; -- "4"
when 5 => LED_on <= "0100100"; -- "5"
when 6 => LED_on <= "0100000"; -- "6"
when 7 => LED_on <= "0001111"; -- "7"
when 8 => LED_on <= "0000000"; -- "8"
when 9 => LED_on <= "0000100"; -- "9"
when others => LED_on <= "0000001";
end case;
end process;
end Behavioral;
```

CONSTRAINT

```
#clock
set_property PACKAGE_PIN W5 [get_ports clk]
```

```
    set_property IOSTANDARD LVCMOS33 [get_ports clk]
 #dht11_data
set_property PACKAGE_PIN J3 [get_ports data]
    set_property IOSTANDARD LVCMOS33 [get_ports data]
  #ssd
set_property PACKAGE_PIN W7 [get_ports {LED_on[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED_on[6]}]
set_property PACKAGE_PIN W6 [get_ports {LED_on[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED_on[5]}]
set_property PACKAGE_PIN U8 [get_ports {LED_on[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED_on[4]}]
set_property PACKAGE_PIN V8 [get_ports {LED_on[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED_on[3]}]
set_property PACKAGE_PIN U5 [get_ports {LED_on[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED_on[2]}]
set_property PACKAGE_PIN V5 [get_ports {LED_on[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED_on[1]}]
set_property PACKAGE_PIN U7 [get_ports {LED_on[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED_on[0]}]
set_property PACKAGE_PIN U2 [get_ports {Activator[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Activator[0]}]
set_property PACKAGE_PIN U4 [get_ports {Activator[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Activator[1]}]
set_property PACKAGE_PIN V4 [get_ports {Activator[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Activator[2]}]
set_property PACKAGE_PIN W4 [get_ports {Activator[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Activator[3]}]

# Switches for sw_1
set_property PACKAGE_PIN V17 [get_ports {sw_1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw_1[0]}]
set_property PACKAGE_PIN V16 [get_ports {sw_1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw_1[1]}]
set_property PACKAGE_PIN W16 [get_ports {sw_1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw_1[2]}]
set_property PACKAGE_PIN W17 [get_ports {sw_1[3]}]
```

set_property IOSTANDARD LVCMOS33 [get_ports {sw_1[3]}]

set_property PACKAGE_PIN W15 [get_ports {sw_1[4]}]

set_property IOSTANDARD LVCMOS33 [get_ports {sw_1[4]}]

set_property PACKAGE_PIN V15 [get_ports {sw_1[5]}]

set_property IOSTANDARD LVCMOS33 [get_ports {sw_1[5]}]

set_property PACKAGE_PIN W14 [get_ports {sw_1[6]}]

set_property IOSTANDARD LVCMOS33 [get_ports {sw_1[6]}]

set_property PACKAGE_PIN W13 [get_ports {sw_1[7]}]

set_property IOSTANDARD LVCMOS33 [get_ports {sw_1[7]}]

#Switches for option

set_property PACKAGE_PIN T1 [get_ports {option[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {option[0]}]

set_property PACKAGE_PIN R2 [get_ports {option[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {option[1]}]

set_property PACKAGE_PIN U1 [get_ports {kurulum}]

set_property IOSTANDARD LVCMOS33 [get_ports {kurulum}]

#Button

set_property PACKAGE_PIN U18 [get_ports {rst}]

set_property IOSTANDARD LVCMOS33 [get_ports {rst}]

 # LEDs

set_property PACKAGE_PIN U16 [get_ports {wind_warning}]

set_property IOSTANDARD LVCMOS33 [get_ports {wind_warning}]

set_property PACKAGE_PIN E19 [get_ports {low_humid_warning}]

set_property IOSTANDARD LVCMOS33 [get_ports {low_humid_warning}]

set_property PACKAGE_PIN U19 [get_ports {high_humid_warning}]

set_property IOSTANDARD LVCMOS33 [get_ports {high_humid_warning}]

set_property PACKAGE_PIN V19 [get_ports {low_temp_warning}]

set_property IOSTANDARD LVCMOS33 [get_ports {low_temp_warning}]

set_property PACKAGE_PIN W18 [get_ports {high_temp_warning}]

set_property IOSTANDARD LVCMOS33 [get_ports {high_temp_warning}]

#HCSR

set_property PACKAGE_PIN J1 [get_ports echo1]

set_property IOSTANDARD LVCMOS33 [get_ports echo1]

set_property PACKAGE_PIN L2 [get_ports trig]

```
set_property IOSTANDARD LVCMOS33 [get_ports trig]
#SOIL
set_property PACKAGE_PIN N2 [get_ports soil_humid]
set_property IOSTANDARD LVCMOS33 [get_ports soil_humid]
set_property PACKAGE_PIN U15 [get_ports soil_warning]
set_property IOSTANDARD LVCMOS33 [get_ports soil_warning]
```

ARDUINO CODE

```
#define toprakIslak 500
#define toprakKuru 750
#define sensorGuc 3.3
#define sensorPin A0
#define digitalCikisPin 8
void setup() {
  pinMode(sensorGuc, OUTPUT);
  pinMode(digitalCikisPin, OUTPUT);
  digitalWrite(sensorGuc, LOW);
  Serial.begin(9600);
}

void loop() {
  int nem = sensordenOku();
  Serial.print("Analog Çıkış: ");
  Serial.println(nem);
  if (nem < toprakIslak) {
    Serial.println("Durum: Toprak çok ıslak");
    dijitalCikis(1);

  } else if (nem >= toprakIslak && nem < toprakKuru) {
    Serial.println("Durum: Toprak nem oranı ideal");
    dijitalCikis(0);



  } else {
```

```
    Serial.println("Durum: Toprak çok kuru - sulama zamanı!");
    dijitalCikis(1);

  }

  delay(1000);
 Serial.println();
}
int sensordenOku() {
  digitalWrite(sensorGuc, HIGH);  // Sensörü AÇIK konuma getirin
  delay(10);
  int deger = analogRead(sensorPin);
  digitalWrite(sensorGuc, LOW);
  return deger;
}
 void dijitalCikis(int deger) {
  digitalWrite(digitalCikisPin, deger);
}
```