DHT11 Seat Warmer

DHT11 Seat Warmer
29.12.2018
Şevki Gavrem Kulkuloğlu

• YouTube Link

https://www.youtube.com/watch?v=ZO7aof2uG_0

• Abstract

- I wanted to design a setup in which there is a seat wormer that works according to the temperature of the room/environment. To build this setup, I used a Basys3, a DHT11 (Temperature sensor), a level shifter(sn74lvc1t45), a transistor (bc 238) and a 5v relay.

• Project's Design Specification Plan

- An average room temperature is 25 degrees, so I determine the 25 degrees as my threshold temperature point. I will measure the temperature of the room via using DHT11 sensor. If my measurement is lower/colder than my threshold temperature, the seat warmer will start to warm up the seat. If my measurement is higher than threshold temperature, it will stop warming up the seat.

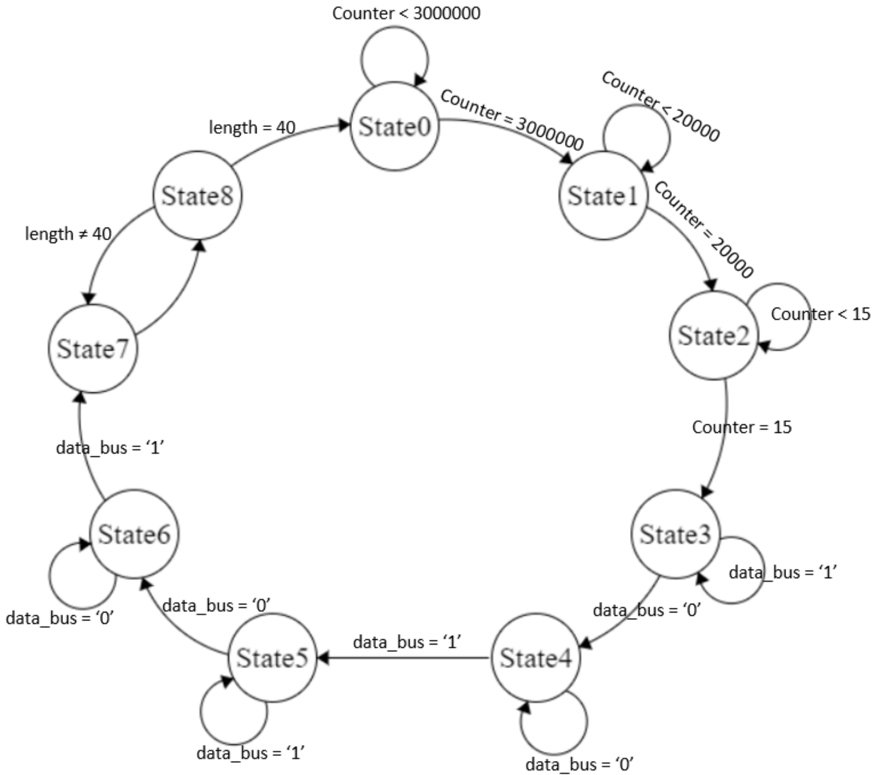
• Project's Design Methodology

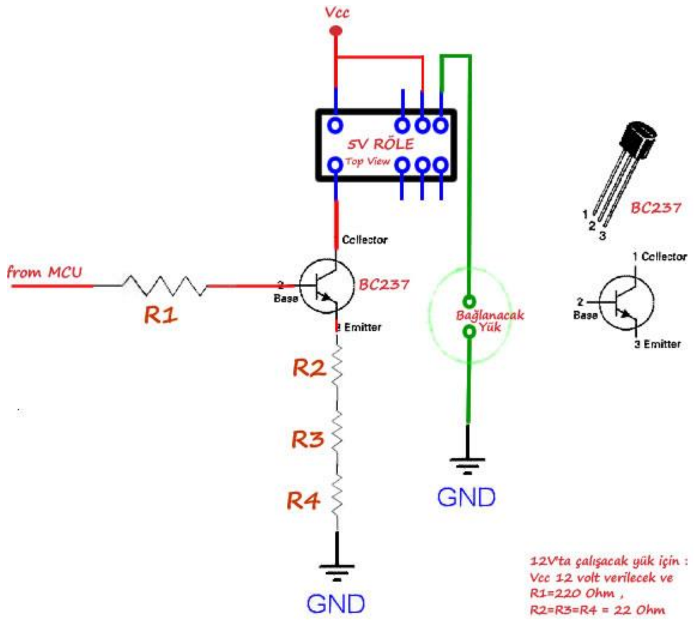
- The hardest part of my project was reading the data form DHT11 sensor. After some time, I realized that sensor was working with 5v and Basys3's pmod logic output was 3.3v, first I have used opamp to increase the 3.3v but it was just one directional. After that my consultation to my TA, I figure that need to use a level shifter. With my setup completed I just need to find my FSM design. I frequently changed my FSM design to read the sensor and finally I was able to read the sensor with nine state design.

- After I read check the data, I needed the control the seat_warmer but at first, I could not do that, after a brief research and talked with Onur abi I found a circuit design on

DHT11 Seat Warmer

our laboratory’s web page. After building that circuit I was finally able to control my seat warmer with Basys3’s logic output from pmod section.





○

DHT11 Seat Warmer

DHT11 Seat Warmer
29.12.2018
Şevki Gavrem Kulkuloğlu

• Results

- I Started my project with first understanding DHT11 temperature sensor. After I read the datasheet of it and some different FSM (Finite state machine) designs, I came up with an FSM design in which I would have 9 states. Nearly all of the states are working accordingly to sending/receiving data for some micro seconds, so First I made a Clock divider module in which I am using the 100Mhz Basys3 clock to obtain a 1Mhz clock. After that in order to wait while bus is released or send/ receive the data for some specific micro seconds I used a counter, I am counting in every rising edge of the 1Mhz clock. The Fist state is an unstable state, so in this step I am counting until 3000000 in order to wait 3 seconds after power on of the sensor. After I that I am passing the second state. In this state I need to awake the sensor with sending 20ms ‘0’ (low) on the bus (like I did in the first state, I have counted 20000 to get 20ms). I am passing the third state, in which I am sending high ‘1’ on bus for 15µs. After I awaked the sensor I am realizing the bus with ‘Z’ and passing the next state. In the fourth state I am checking the bus if it is low passing to next state. In the fifth state I need to receive low for 80 µs, so I am just checking the bus until it is high, when it is high, I am passing to next state. In the sixth state sensor should send 80 µs high ‘1’, so like previous state I am just checking the bus until it is low, when it is low, I am passing the next state. In the next state I am checking the bus, if it is high passing the next state. In the eighth state, I am going to get the data in a way that if sensor sending 50 µs low, 25 µs high it means ‘0’, if it is sending 50 µs low, 70 µs high it means ‘1’. So, in this State I am just counting the ‘1’ s and checking if it is lower or higher than low time (50 µs). After I obtain the ‘0’ or ‘1’ I am concatenating them to my Fourty_bit_signal form right side and while doing that I am counting evert bit I add the signal to know when I have the right 40-bit signal. After that in the ninth state I am checking the count I made in the previous state if it is 40, I am passing to first state and equaling Fourty_bit output with Fourty_bit_signal if it is not passing to previous state.

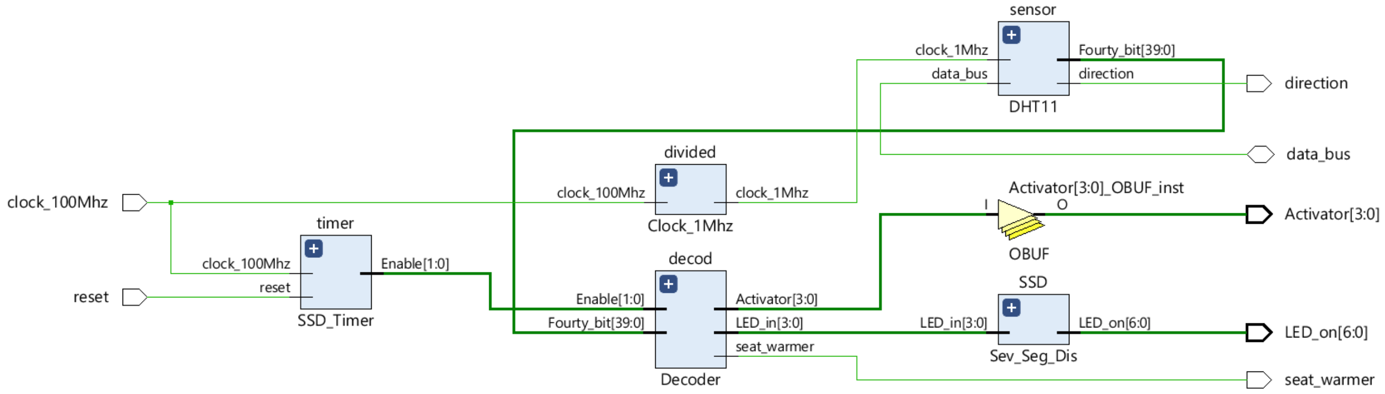
DHT11 Seat Warmer

DHT11 Seat Warmer
29.12.2018
Şevki Gavrem Kulkuloğlu

- After I got my Fourty_bit data form DHT11 module, it is entering the Decoder module as input. In that module Fourty_bit directly entering the SSD_Identifier module component. In SSD_Identifier module, first I am diving 40-bit to 5 8-bits and adding first 4 8-bits and checking the equality to last 8-bits, if it is data is correct so check= 1 if not it is wrong check is equal to 0. After that I need the temperature data so I am taking the 2 4 bits from 23-20 and 19-16 and sending them to Hex_to_Decimal component module to conversion. In Hex_to_Decimal module I am converting hex values to decimals and giving these as outputs to SSD_Identifier. After SSD_Identifier is received these values I will add them up and determining the two digit of the temperature which will be displayed on the seven segment and also checking the temperature for the seat warmer in here, if its low then 25-degree seat_warmer = 1, if not seat_warmer = 0. After decoder has received these 4 values it is determining the Seven segment display.
- In top module I have combine all of my modules.

My RTL Schematics

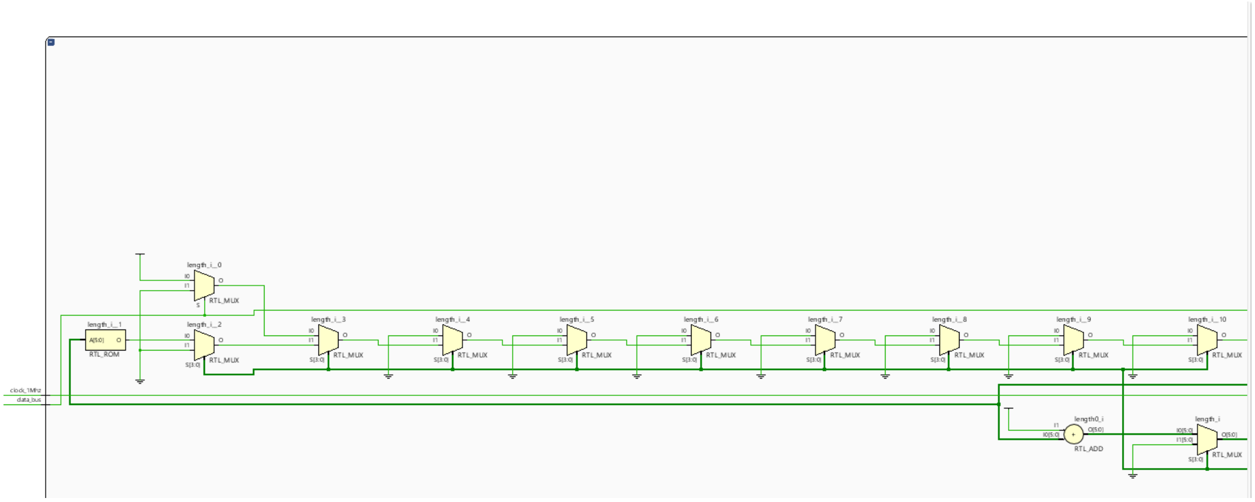
TOP Module RTL

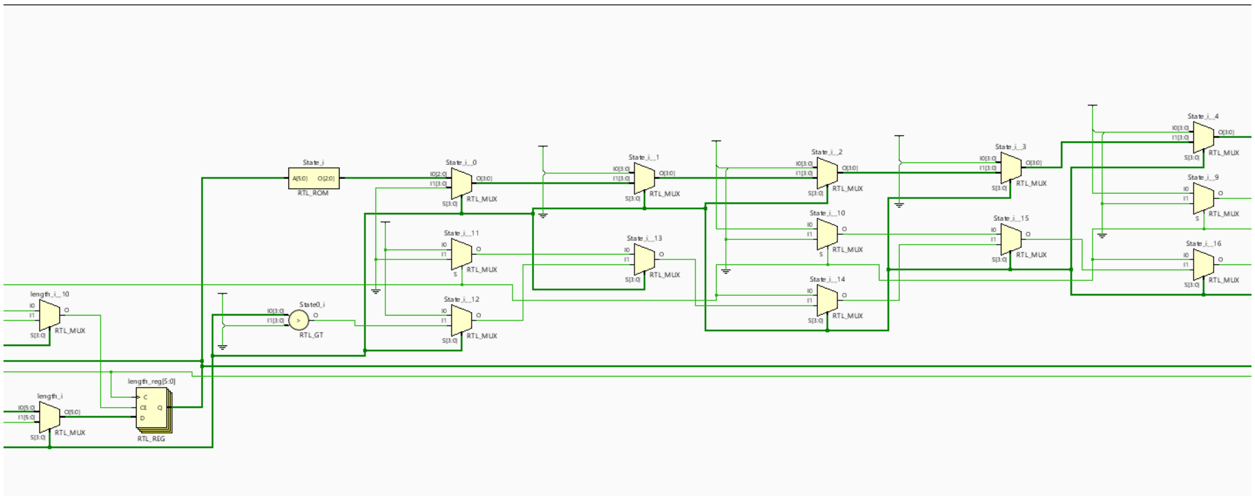


DHT11 Module

DHT11 Seat Warmer

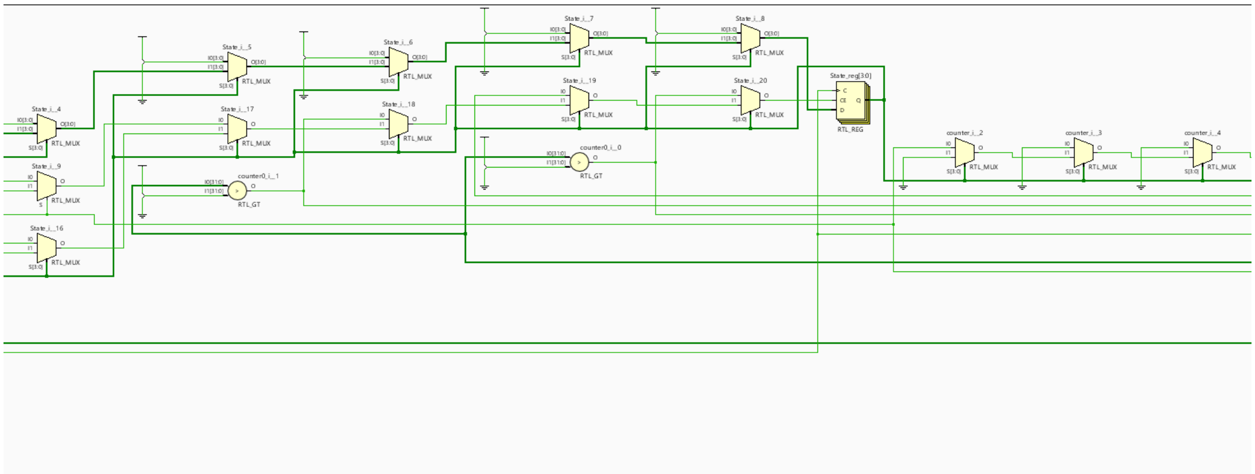
DHT11 Seat Warmer
29.12.2018
Şevki Gavrem Kulkuloğlu

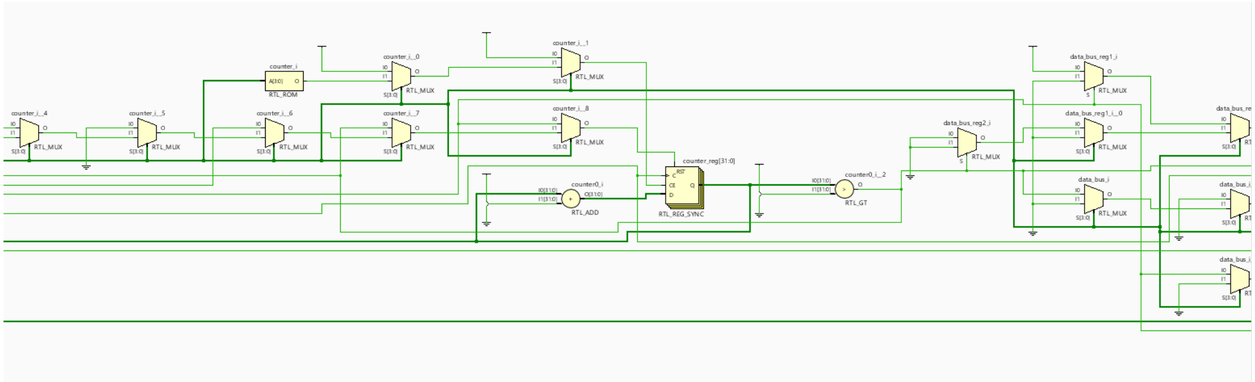


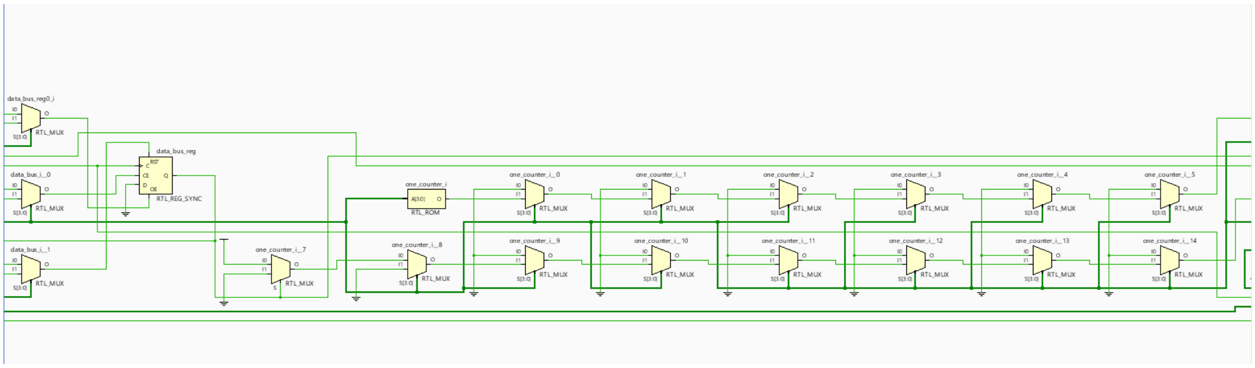


DHT11 Seat Warmer

DHT11 Seat Warmer
29.12.2018
Şevki Gavrem Kulkuloğlu

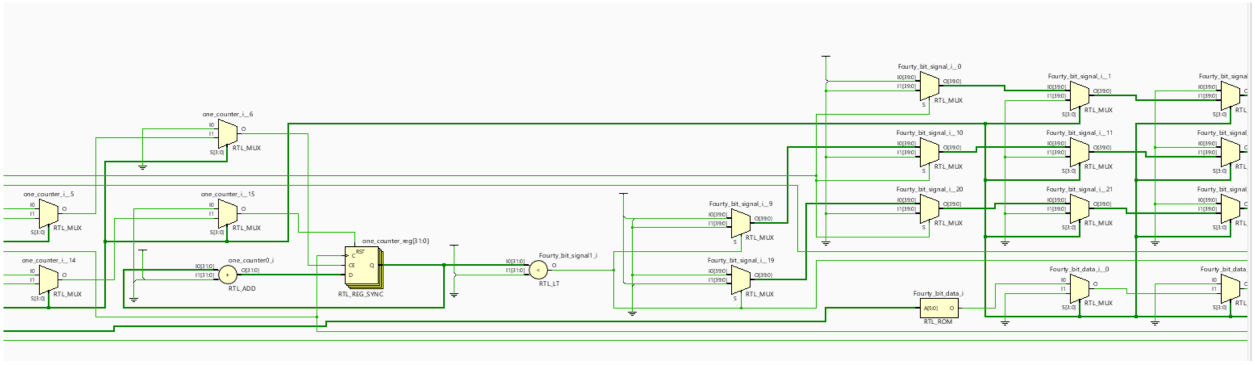


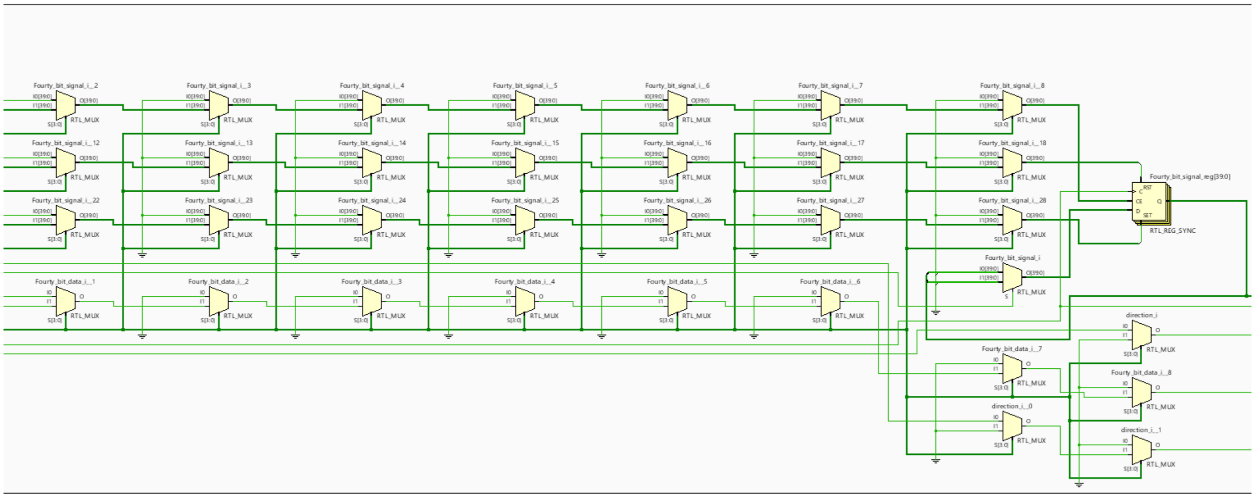


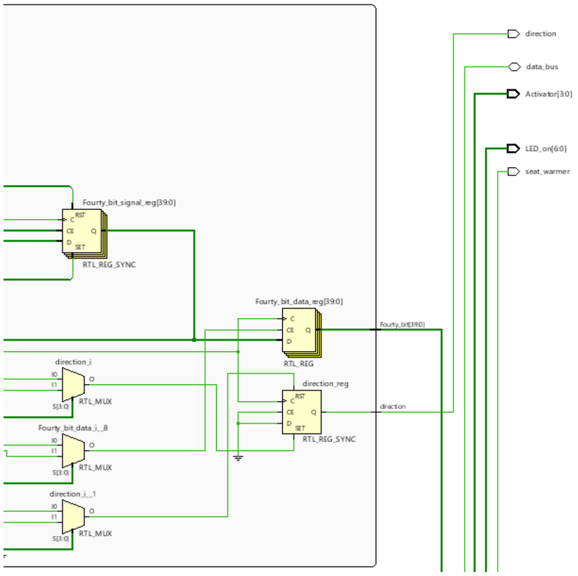


DHT11 Seat Warmer

DHT11 Seat Warmer
29.12.2018
Şevki Gavrem Kulkuloğlu



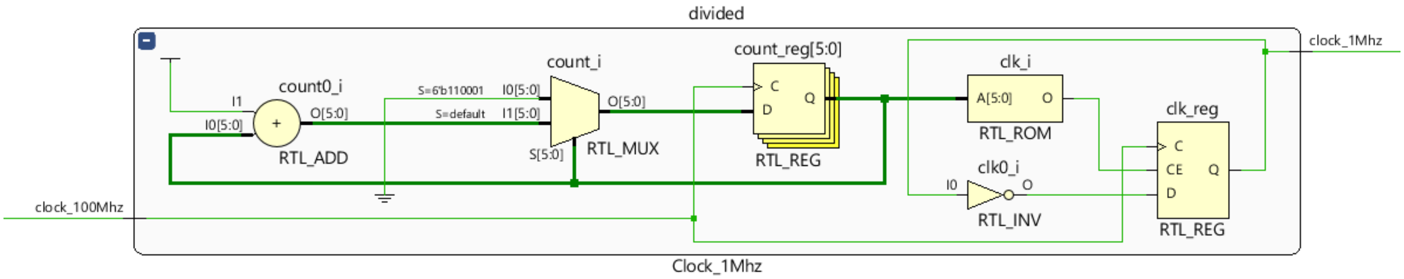




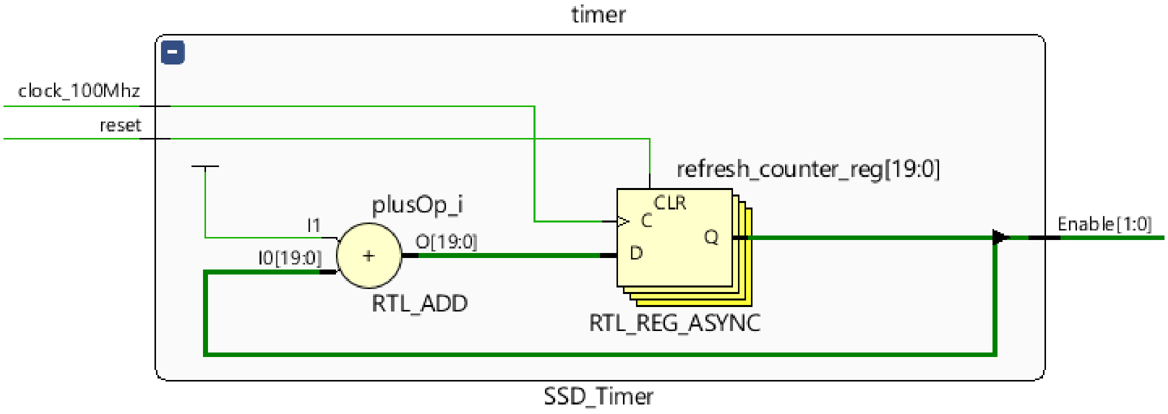
DHT11 Seat Warmer

DHT11 Seat Warmer
29.12.2018
Şevki Gavrem Kulkuloğlu

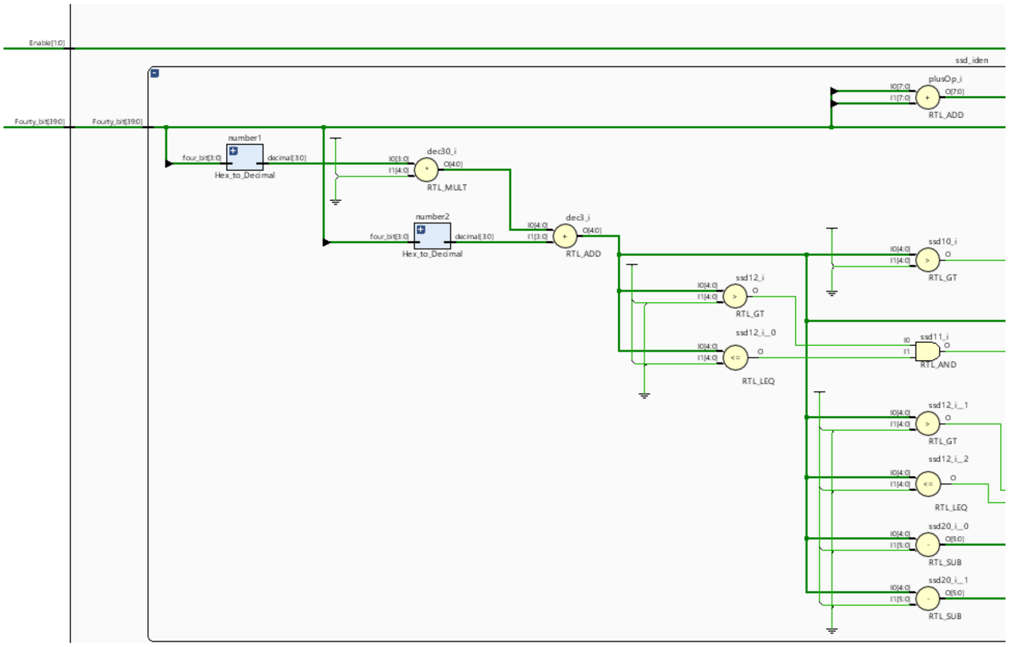
Clock_1Mhz module

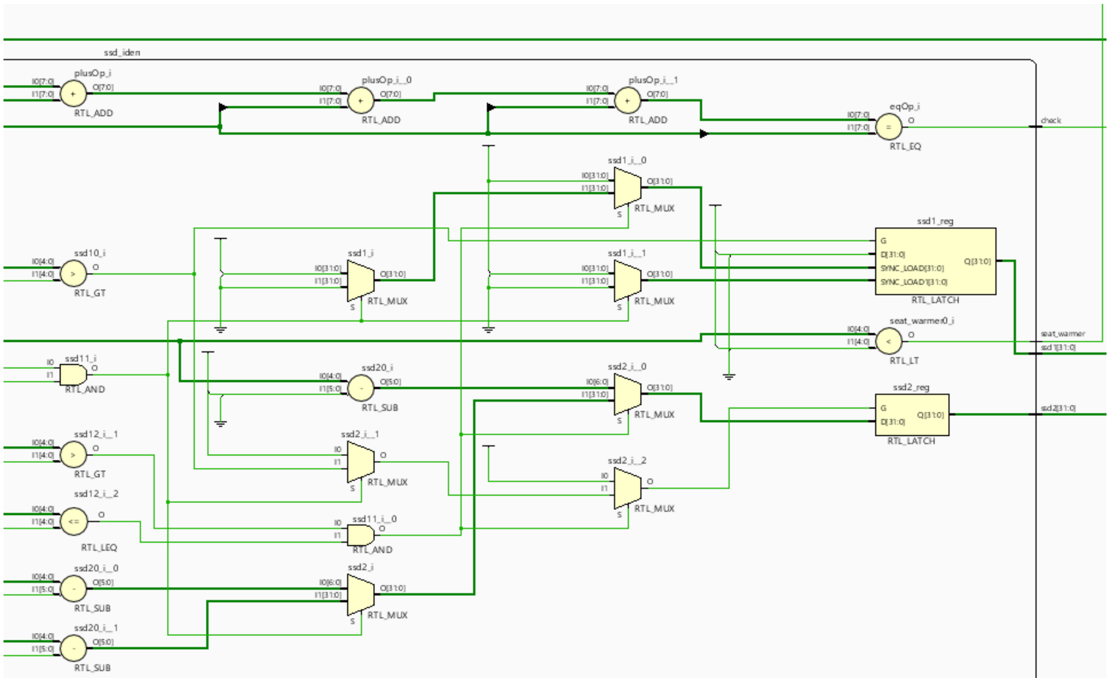


SSD_Timer Module

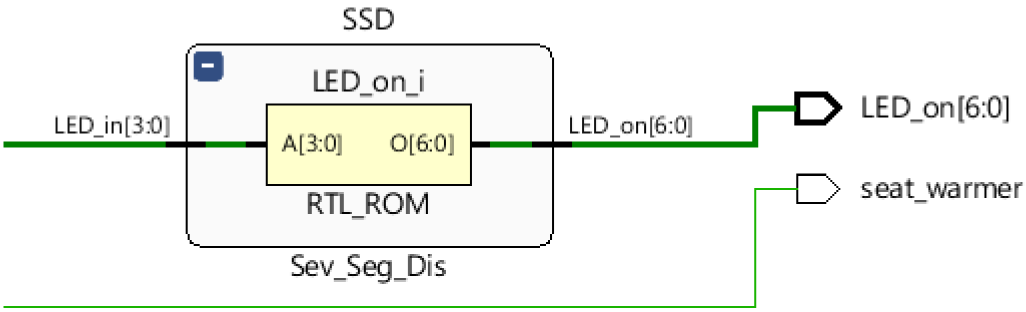


Decoder module





Sev_Seg_Dis module



• Conclusion

- During the process of my project I have learnt more about Basys3 boards, especially its pmod sections. In my project's process I have burnt 2 DHT11 sensor and a level shifter, So the max. voltage that material can take is very important. I have learnt how to use digital sensors, how important every single information on the datasheet and how much I have to learn every one of them. I have figured out that slaves and masters are communicating on a data line and that line's voltage can be detect by both of them, in my case to obtain that scenario, I have used a level shifter. I have some problems while trying to control my seat warmer because it was working 5V and pmod output was 3.3V. My TA suggested me to use relay, I builder the setup but it was not working due to the 3.3v's insufficiency. After a brief research and I talked with Onur abi I built the setup on our laboratories page and basically add a transistor to my circuit. After that it started to work. While the time I was trying to read the data, I have used the oscilloscope frequently and learn how to used it more specific ways.

- ## References

- Role devresi. (n.d.). Retrieved from http://eelab.bilkent.edu.tr/TR/ee212destek.html

- # Appendix

- Top Module

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



entity TopModule is
   Port ( clock_100Mhz : in std_logic;
         reset        : in std_logic;
         Activator    : out std_logic_vector (3 downto 0);
         LED_on       : out std_logic_vector (6 downto 0);
         direction    : out std_logic;
         data_bus     : inout std_logic;
         seat_warmer   : out std_logic);
end TopModule;

architecture Behavioral of TopModule is
signal Enable_1       : std_logic_vector(1 downto 0);
signal Led_in         : integer;
signal clock1Mhz      : std_logic  ;
signal Fourty_bit_sig : std_logic_vector( 39 downto 0);

component DHT11 is
       Port (
              clock_1Mhz: in std_logic;
```

```
                data_bus: inout std_logic;
                direction: out std_logic;
                Fourty_bit: 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";
          LED_in     : out integer;
          seat_warmer   : out std_logic);
 end component;

component Sev_Seg_Dis is
PORT
         (LED_in : 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 : DHT11 port map (clock1Mhz,data_bus,direction,Fourty_bit_sig);
divided: Clock_1Mhz Port Map (clock_100Mhz, clock1Mhz);
timer   : SSD_Timer PORT MAP (clock_100Mhz,reset,Enable_1);
decod   : Decoder PORT MAP (Enable_1,Fourty_bit_sig,Activator,Led_in,seat_warmer);
SSD    : Sev_Seg_Dis PORT MAP (Led_in,LED_on);

end Behavioral;
```

- ## DHT11 Module

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

entity DHT11 is
     Port (  clock_1Mhz: in std_logic;
             data_bus  : inout std_logic;
             direction : out std_logic;
             Fourty_bit: out std_logic_vector(39 downto 0));
end DHT11;

architecture Behavioral of DHT11 is

------Signals-------
```

DHT11 Seat Warmer

```
signal State     : std_logic_vector(3 downto 0 ):="0000";
signal one_counter: integer:= 0;
signal counter    : integer:= 0;
signal length     : integer range 0 to 40:= 0;
signal Fourty_bit_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
           ----Check state----
          if State = "0000"  then
           counter <= counter + 1;
           if(counter > 3000000) then
            State <= "0001";
            data_bus <= '0';
            direction <= '1';
            counter <= 0;
           end if;
           ----Check state----
          elsif State = "0001"  then
            counter <= counter + 1;
           if(counter > 20000) then
            data_bus <= 'Z';
            direction <= '0';
            counter <= 0;
            State <= "0010";
           end if;
         ----Check state----
         elsif State = "0010"  then
           counter <= counter + 1;
           if(counter > 15) then
           counter <= 0;
            State <= "0011";
           end if;
          ----Check state----
         elsif State = "0011"  then
           if(data_bus = '0') then
            State <= "0100";
           end if;
           ----Check state----
         elsif State = "0100"  then
            if(data_bus = '1') then
            State <= "0101";
           end if;
          ----Check state----
        elsif State = "0101"  then
            if(data_bus = '0') then
            State <= "0110";
           end if;
           ----Check state----
        elsif State = "0110"  then
           if(data_bus = '1') then
            State <= "0111";
            counter <= 0;
           end if;
          ----Check state----
         elsif State = "0111"  then
           one_counter <= one_counter + 1;
```

```
          if(data_bus = '0') then
           if(one_counter < 50) then
            Fourty_bit_signal <= Fourty_bit_signal(38 DOWNTO 0) & '0';
           else
            Fourty_bit_signal <= Fourty_bit_signal(38 DOWNTO 0) & '1';
           end if;
           length <= length + 1;
           one_counter <= 0;
           State <= "1000";
          end if;
           ----Check state----
           elsif State = "1000"  then
          if(length = 40) then
           Fourty_bit_data <= Fourty_bit_signal;
           State <= "0000";
           length <= 0;
          else
           State <= "0101";
          end if;
          elsif State > "1000" then
          State <= "0000";
     end if;
    end if;
end process;
Fourty_bit <= Fourty_bit_data;
end Behavioral;
```

- **Clock_1Mhz Module**

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

DHT11 Seat Warmer

```
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;
```

DHT11 Seat Warmer

DHT11 Seat Warmer
29.12.2018
Şevki Gavrem Kulkuloğlu

- SSD_Timer Clock

```
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 Module**

```
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";
          LED_in     : out integer range 0 to 10;
          seat_warmer   : out std_logic);
end Decoder;

architecture Behavioral of Decoder is

signal  check, ssd1, ssd2 : 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;
          seat_warmer   : out std_logic );
end component;
begin

ssd_iden : SSD_Identifier port map(Fourty_bit,check, ssd2, ssd1,seat_warmer );
```

```
process(Enable)
begin
   case Enable is
   when "00" =>
      Activator <= "0111";
      LED_in <= check;
   when "01" =>
      Activator <= "1011";
      LED_in <=  0 ;
   when "10" =>
      Activator <= "1101";
      LED_in <= ssd1;
   when "11" =>
      Activator <= "1110";
      LED_in <= ssd2;
    when others =>
     Activator <= "0000";
        LED_in <=  0;
   end case;
end process;


end Behavioral;
```

- **SSD_Identifier Module**

```
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;
          seat_warmer   : out std_logic );
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 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 : 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);
```

DHT11 Seat Warmer

```
ctrl3 <= Fourty_bit( 23 downto 16);
ctrl4 <= Fourty_bit( 15 downto 8);
ctrl <= Fourty_bit( 7 downto 0);

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);
dec3 <= (dec1*16) + dec2;

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;
   if dec3 < 25 then
   seat_warmer <= '1';
   else
   seat_warmer <= '0';
   end if;
end process;
```

```
check <= 1 when (ctrl1 + ctrl2 +ctrl3 +ctrl4 ) = ctrl else 0;


end Behavioral;
```

- **Hex_to_Decimal_module**

```
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; -- "0"
   when "0001" => decimal <= 1; -- "1"
   when "0010" => decimal <= 2; -- "2"
   when "0011" => decimal <= 3; -- "3"
   when "0100" => decimal <= 4; -- "4"
   when "0101" => decimal <= 5; -- "5"
   when "0110" => decimal <= 6; -- "6"
   when "0111" => decimal <= 7; -- "7"
   when "1000" => decimal <= 8; -- "8"
   when "1001" => decimal <= 9; -- "9"
```

```
  when "1010" => decimal <= 10; -- a
  when "1011" => decimal <= 11; -- b
  when "1100" => decimal <= 12; -- C
  when "1101" => decimal <= 13; -- d
  when "1110" => decimal <= 14; -- E
  when "1111" => decimal <= 15; -- F
   when others => decimal <= 0;
  end case;
end process;


end Behavioral;
```

- **Sev_Seg_Dis Module**

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

entity Sev_Seg_Dis is
  Port (LED_in: 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(LED_in)
begin
  case LED_in 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;
```

- **Constrain**

```
set_property PACKAGE_PIN W5 [get_ports clock_100Mhz]

set_property IOSTANDARD LVCMOS33 [get_ports clock_100Mhz]

set_property PACKAGE_PIN V2 [get_ports reset]

set_property IOSTANDARD LVCMOS33 [get_ports reset]

set_property PACKAGE_PIN J3 [get_ports data_bus]

 set_property IOSTANDARD LVCMOS33 [get_ports data_bus]

set_property PACKAGE_PIN L3 [get_ports direction]

 set_property IOSTANDARD LVCMOS33 [get_ports direction]

set_property PACKAGE_PIN M2 [get_ports seat_warmer]

 set_property IOSTANDARD LVCMOS33 [get_ports seat_warmer]



set_property PACKAGE_PIN W7 [get_ports {LED_on[6]}]

set_property IOSTANDARD LVCMOS33 [get_ports {LED_on[6]}]
```

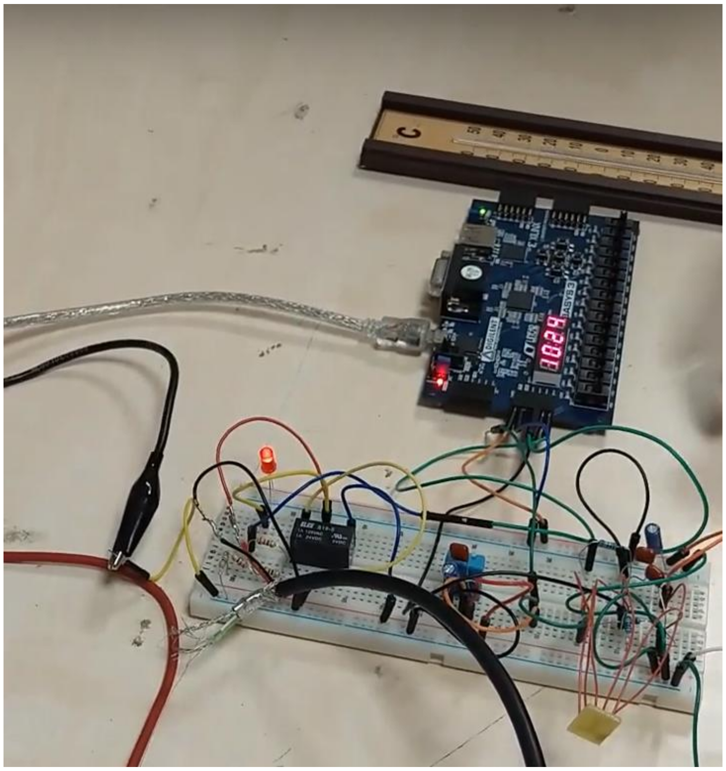
DHT11 Seat Warmer

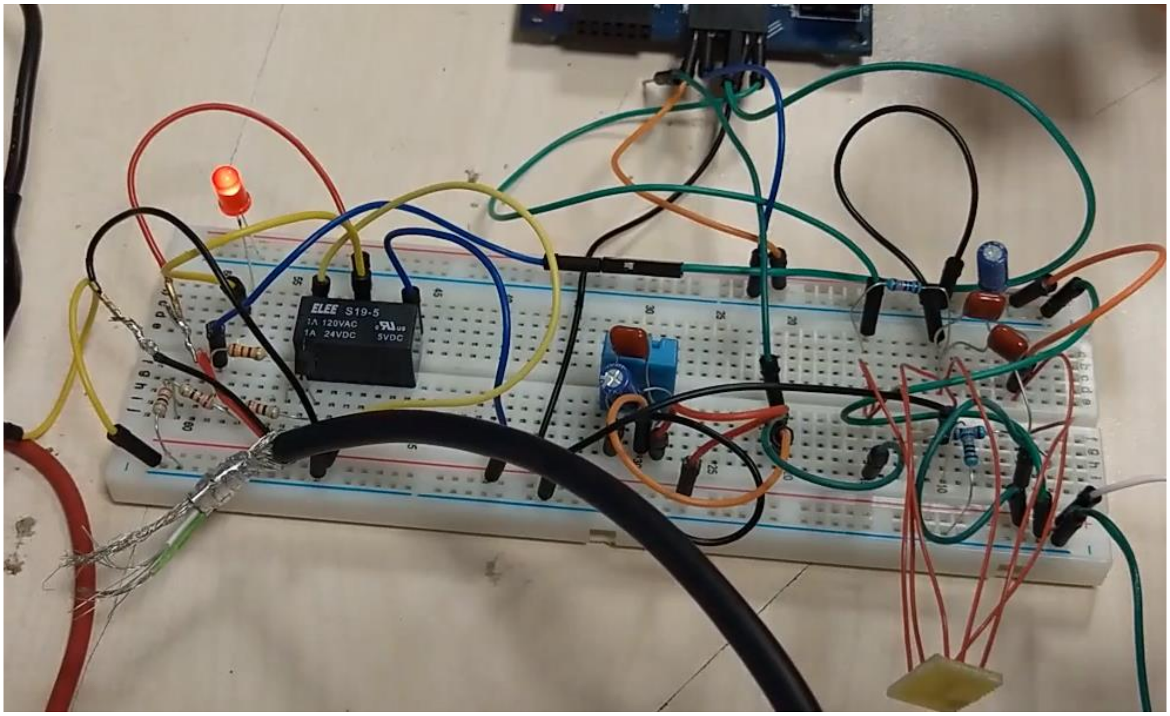
```
set_property PACKAGE_PIN W6 [get_ports {LED_on[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED_on[5]}]
set_property PACKAGE_PIN U8 [get_ports {LED_on[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED_on[4]}]
set_property PACKAGE_PIN V8 [get_ports {LED_on[3]}]
set_property IOSTANDARD LVCMOS33 [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]}]
```

DHT11 Seat Warmer

DHT11 Seat Warmer
29.12.2018
Şevki Gavrem Kulkuloğlu

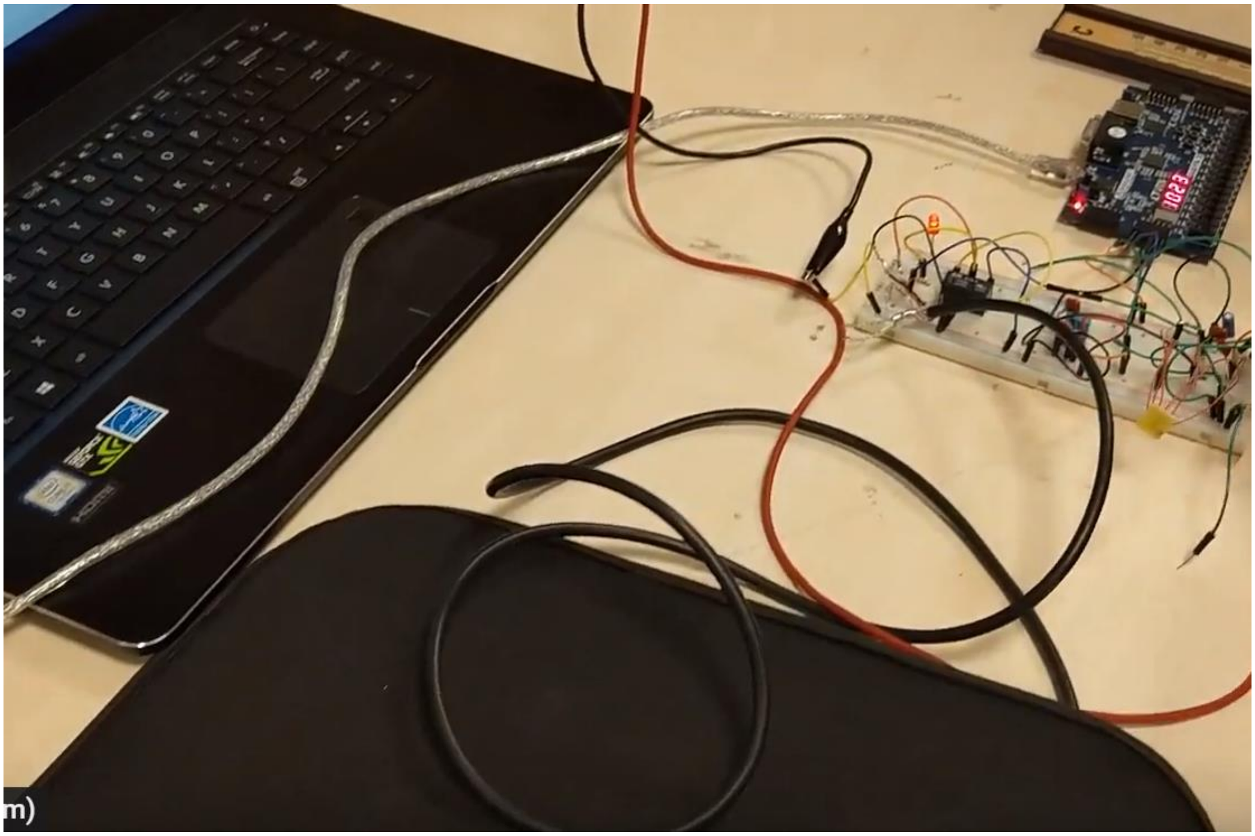
- **Photos**





DHT11 Seat Warmer

DHT11 Seat Warmer
29.12.2018
Şevki Gavrem Kulkuloğlu



- **Datasheets**
- https://akizukidenshi.com/download/ds/aosong/DHT11.pdf
- http://www.ti.com/lit/ds/symlink/sn74lvc1t45.pdf
- http://pdf1.alldatasheet.com/datasheet-pdf/view/50723/FAIRCHILD/BC238.html
-