

**EEE 102**

**Introduction to Digital Circuit Design**

Term Project Report

The Maze Runner

Yusuf Taha Sarı

22003405

Section 2

Instructor: Süleyman Serdar Kozat

**PURPOSE**

This project aims to create a simple, practical device with the knowledge gained in the digital circuit design course. Understanding the basics of creating new device and practicing at solving problems in both codding and hardware areas are main purposes of this project.

**DESIGN SPECIFICATIONS**

To create a device some components and devices are used;

• BASYS3 FPGA

• L298N Motor Driver

• 2 DC motors

• Robot Chassis and 3 wheels

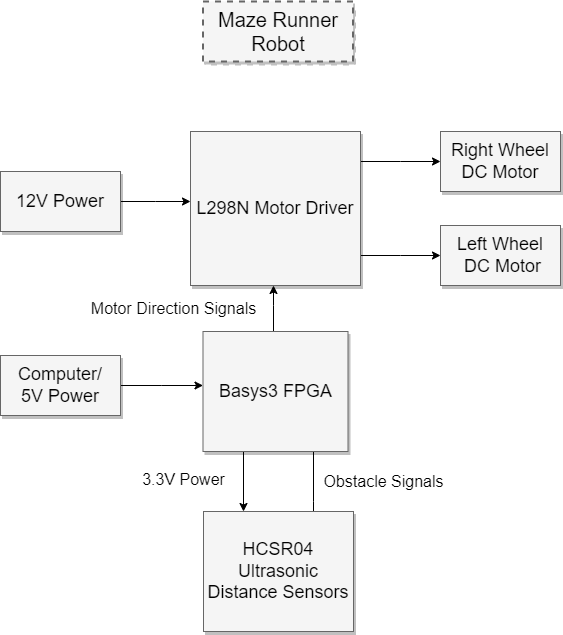
• 3 HCSR04 ultrasonic distance sensors

• Jumper cables

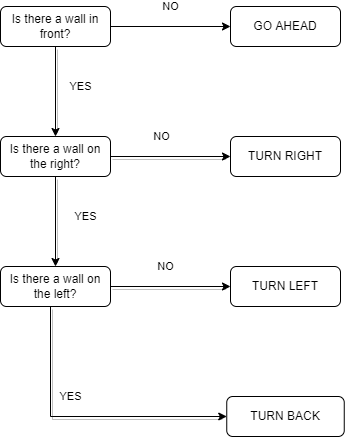
• 8 AA batteries

Maze runner is a robot which aims to solve puzzles and find a way to end of the road. There are 3 ultrasonic distance sensors on the front and front corners of the robot. They check whether there is a wall in front of them or not. They provide an output to Basys3 FPGA card according to their status. Basys3 gets that information and creates wheels’ directions signals. Motor driver gets both 12V power and wheel direction signals and powers DC motors accordingly.

**Schema Of Robot Design:**



**Algorithm For Wheel Directions:**

****

**METHODOLOGY**

4 different VHDL modules are used in this project.

• Motor Control (Main Module)

• Ultrasonic

• Trigger Generator

• Counter

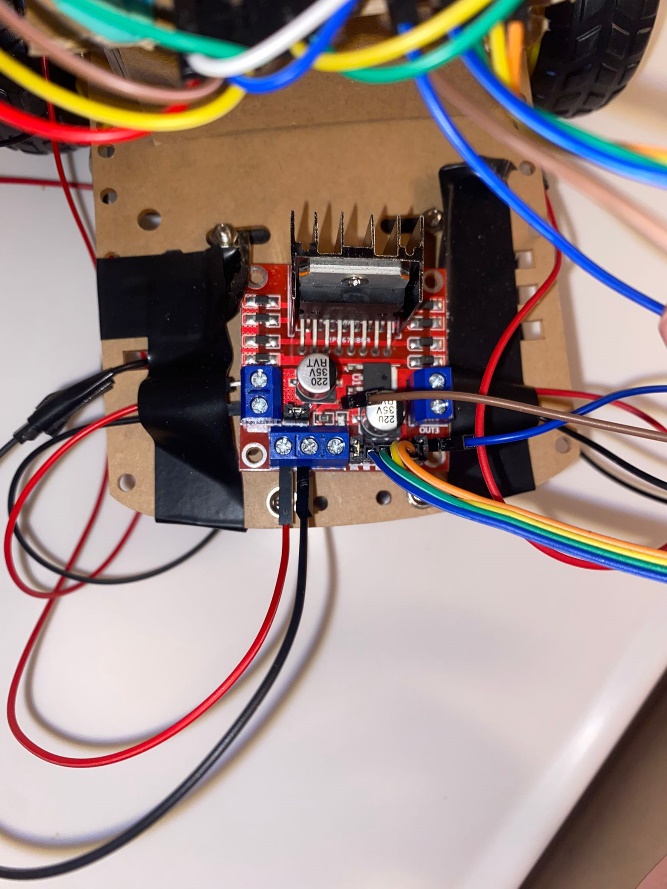
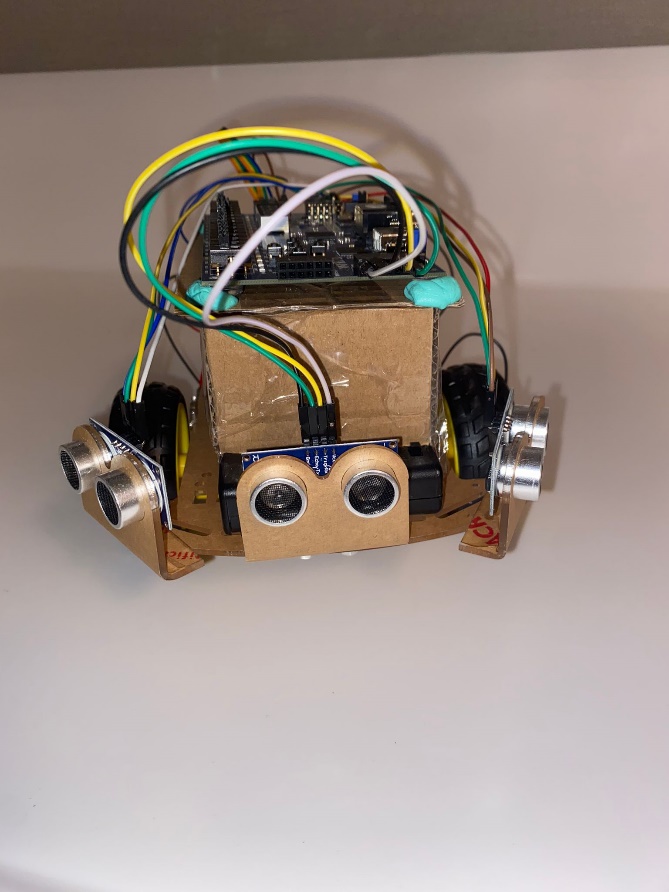
**Motor Control (Main Module):** In this module, outputs of ultrasonic module are used and 4 different wheel direction signals (2 for each) are created according to signals.

**Ultrasonic:** Ultrasonic module sends trigger which is generated at trigger generator module and sends output obstacle if there is an obstacle closer than 5 cm’s.

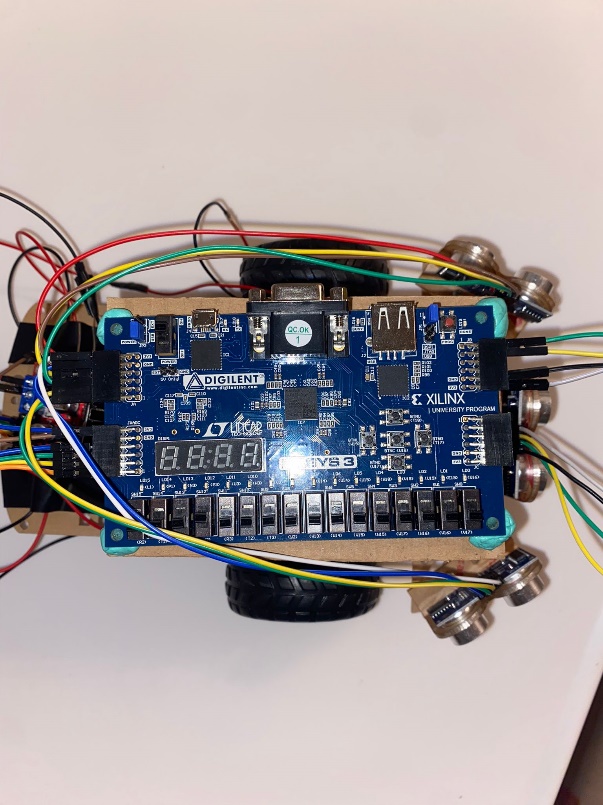
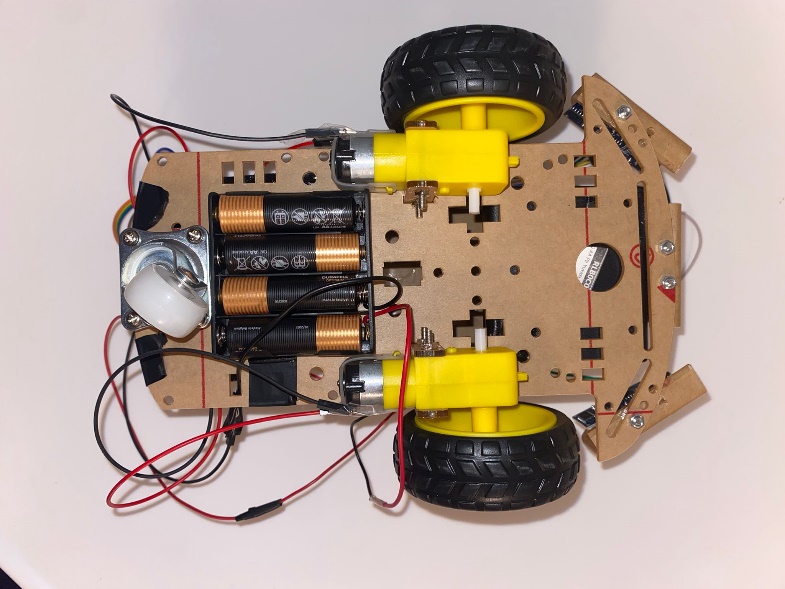
**Trigger Generator:** Creates trigger for HCSR04 sensor according to its data sheet.

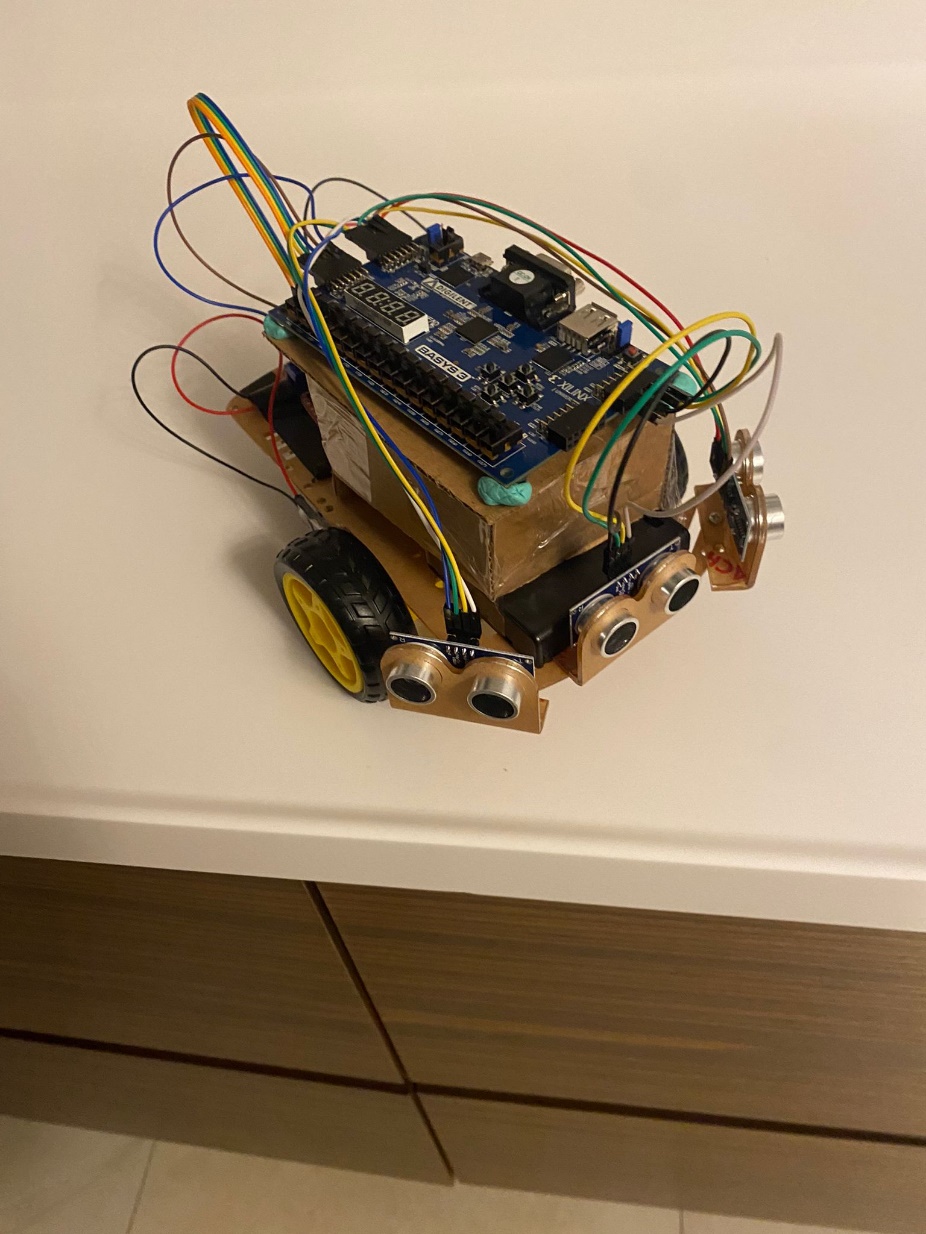
**Counter:** Counts time to find the distance.

**RESULT**

** **

(Image 1: L298N Motor Driver) (Image 2: HCSR04 Ultrasonic Distance Sensors)

  (Image 3 Basys3 FPGA Card) (Image 4: DC motors and AA batteries)



(Image 5: The Maze Runner)

**Youtube Video Link:** <https://youtu.be/WEcnTylsJEs>

**APPENDICES**

**Motor Control (Main Module):**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.numeric\_std.all;

entity motor\_control is

Port ( clk, sw0, sw15, sw14, pulse\_F, pulse\_R, pulse\_L : in STD\_LOGIC;

pwm1, pwm2, pwm3, pwm4 : out STD\_LOGIC;

trigger\_F, trigger\_R, trigger\_L : inout STD\_LOGIC);

end motor\_control;

architecture Behavioral of motor\_control is

signal count: integer range 0 to 50000;

signal speed\_1: integer := 0;

signal obstacle\_F: std\_logic;

signal obstacle\_R: std\_logic;

signal obstacle\_L: std\_logic;

signal sonuc\_F : std\_logic;

signal sonuc\_R : std\_logic;

signal sonuc\_L : std\_logic;

component ultrasonic is

port(

clk: in std\_logic;

pulsee: in std\_logic; -- echo

trigOut:inout std\_logic; -- trigger out

obs:inout std\_logic;

sonucc:out std\_logic);

end component;

begin

a: ultrasonic

port map (clk=>clk,

pulsee => pulse\_F,

trigOut => trigger\_F,

obs => obstacle\_F,

sonucc => sonuc\_F);

b: ultrasonic

port map (clk=>clk,

pulsee => pulse\_R,

trigOut => trigger\_R,

obs => obstacle\_R,

sonucc => sonuc\_R);

c:ultrasonic

port map (clk=>clk,

pulsee => pulse\_L,

trigOut => trigger\_L,

obs => obstacle\_L,

sonucc => sonuc\_L);

process(clk, sw0, obstacle\_F, obstacle\_R, obstacle\_L, speed\_1)

begin

if (sw0='1') then

speed\_1 <= 12000;

else

speed\_1 <= 0;

end if;

if(rising\_edge(clk)) then

count <= count+1;

if(count = 49999) then

count <= 0;

end if;

end if;

if(count < speed\_1 ) then

--pwm1 = sol tekerlek kırmız

--pwm2 = sol tekerlek siyah

--pwm4 = sağ tekerlek kırmız

--pwm3 = sağ tekerlek siyah

if(sonuc\_F = '0' and sonuc\_R = '0'and sonuc\_L ='0') then

pwm1 <= '1';

pwm2 <= '0';

pwm3 <= '1';

pwm4 <= '0';

elsif(sonuc\_F = '0' and sonuc\_R = '0'and sonuc\_L ='1') then

pwm1 <= '1';

pwm2 <= '0';

pwm3 <= '1';

pwm4 <= '0';

elsif(sonuc\_F = '0' and sonuc\_R = '1'and sonuc\_L ='0') then

pwm1 <= '0';

pwm2 <= '1';

pwm3 <= '0';

pwm4 <= '1';

elsif(sonuc\_F = '0' and sonuc\_R = '1'and sonuc\_L ='1') then

pwm1 <= '0';

pwm2 <= '1';

pwm3 <= '0';

pwm4 <= '1';

elsif(sonuc\_F = '1' and sonuc\_R = '0'and sonuc\_L ='0') then

pwm1 <= '0';

pwm2 <= '1';

pwm3 <= '1';

pwm4 <= '0';

elsif(sonuc\_F = '1' and sonuc\_R = '0'and sonuc\_L ='1') then

pwm1 <= '1';

pwm2 <= '0';

pwm3 <= '1';

pwm4 <= '0';

elsif(sonuc\_F = '1' and sonuc\_R = '1'and sonuc\_L ='0') then

pwm1 <= '0';

pwm2 <= '1';

pwm3 <= '0';

pwm4 <= '1';

else

pwm1 <= '0';

pwm2 <= '1';

pwm3 <= '1';

pwm4 <= '0';

end if;

else

pwm1 <= '0';

pwm2 <= '0';

pwm3 <= '0';

pwm4 <= '0';

end if;

end process;

end Behavioral;

**Ultrasonic:**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.numeric\_std;

entity ultrasonic is

port(

clk: in std\_logic;

pulsee: in std\_logic; -- echo

trigOut:inout std\_logic; -- trigger out

sonucc:out std\_logic;

obs : inout std\_logic);

end entity;

architecture behaviour of ultrasonic is

component counter is

generic(n : positive :=10);

port( clkk: in std\_logic;

en : in std\_logic;

res : in std\_logic; -- active low

counter\_output: out std\_logic\_vector(n-1 downto 0));

end component;

component trigger\_generator is

port( clk: in std\_logic;

triggg : out std\_logic);

end component;

--signal triggerOut: std\_logic;

--signal distanceOut:std\_logic(21 downto 0);

signal pulse\_width: std\_logic\_vector(21 downto 0);

signal trigg:std\_logic;

signal triggnot: std\_logic;

signal obstacle\_reg: STD\_LOGIC\_VECTOR( 69 downto 0):="0000000000000000000000000000000000000000000000000000000000000000000000";

begin

triggnot <= not(trigg);

counter\_echo\_pulse :

counter generic map(22)

port map(clkk => clk,

en =>pulsee,

res => triggnot,

counter\_output => pulse\_width);

trigger\_gen :

trigger\_generator port map(clk,trigg);

obstacle\_det: process(pulse\_width)

begin

if(pulse\_width < 55000) then

obs <= '1';

--pwm\_r <='1';

--pwm\_b <='0';

else

obs <= '0';

--pwm\_r <='0';

--pwm\_b <='1';

end if;

end process;

process(clk)

begin

if clk'event and clk ='1' then

obstacle\_reg<= obstacle\_reg(68 downto 0) & obs ;

end if;

if (obstacle\_reg = "1111111111111111111111111111111111111111111111111111111111111111111111") then

sonucc <='0';

else

sonucc <='1';

end if;

end process;

trigOut <= trigg;

end architecture;

**Counter:**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity counter is

generic(n : positive :=10);

port( clkk: in std\_logic;

en : in std\_logic;

res : in std\_logic; -- active low

counter\_output: out std\_logic\_vector(n-1 downto 0));

end entity;

architecture behavioural of counter is

signal count : std\_logic\_vector(n-1 downto 0);

begin

process(clkk,res)

begin

if(res = '0') then

count <= (others=>'0');

elsif(clkk'event and clkk='1') then

if(en = '1') then

count <= count+1;

end if;

end if;

end process;

counter\_output <= count;

end architecture;

**Trigger Generator:**

library ieee;

use ieee.std\_logic\_1164.all;

entity trigger\_generator is

port( clk: in std\_logic;

triggg : out std\_logic);

end entity;

architecture behaviour of trigger\_generator is

component counter is

generic(n : positive :=10);

port( clkk: in std\_logic;

en : in std\_logic;

res : in std\_logic; -- active low

counter\_output: out std\_logic\_vector(n-1 downto 0));

end component;

signal resetCounter:std\_logic;

signal outputCounter: std\_logic\_vector(23 downto 0);

begin

trigger\_gen:counter generic map(24) port map(clk,'1',resetCounter,outputCounter);

process(clk)

constant ms\_100:std\_logic\_vector(23 downto 0):="010011000100101101000000";--20ns/100ms

-- constant ms100And20us: std\_logic\_vector(23 downto 0):="010011000100111100100110";

constant ms\_100And20us: std\_logic\_vector(23 downto 0):="010011000100110100110011";--20ns/(100ms+20us)

begin

if(outputCounter > ms\_100 and outputCounter < ms\_100And20us) then

trigg <= '1';

else

trigg <='0';

end if;

if(outputCounter = ms\_100and20us or outputCounter="XXXXXXXXXXXXXXXXXXXXXXXX") then

resetCounter <= '0';

else

resetCounter <= '1';

end if;

end process;

end architecture;