Mehmet Bayık
21802166
EE-102 Section 3
Term Project – Car Safety System
31.05.2021

Car Safety System Term Project Report

Abstract/Objective:

In my project, I designed preventive safety system for car. Preventive safety system includes lane keep assist system (LKAS) and collision avoidance system (CAS). To create these systems, I used ultrasonic and infrared sensors. Also the car is controlled by IR remote.

Component List:

1. Basys 3 FPGA Board to implement project and simulating combinational circuits. I will use sensors via Basys 3.
2. 2 x Ultrasonic distance sensors to create collision avoidance system (CAS).
3. 2 x Infrared sensors to create lane keep assist system (LKAS).
4. Toy Car (4 x motors and wheels)
5. Powerbank to supply arduino, basys3, sensors.
6. 8 x Ni-Mh batteries pack to supply 2 x motor drivers and 4 x motors
7. 2 x Motor drivers
8. Arduino Uno to supply 5V to Basys3 and sensors.
9. Jumper cables and little breadboards
10. A white platform contains black lines and obstacle.

Design Specifications:

I used four module and one top module in this project. I created lane keep assist system with using two infrared sensors mounted on left and right side of car. When infrared sensors detect line, this sends signal to motor driver for 1.4 second and car turns left if line detected in right side or turns right if line detected in left side. In order to create obstacle avoidance system, I used two ultrasonic sensors. When one of them detects obstacle, car will stop and can only go backwards.

Also, I used two motor drivers to control and drive four motors individually. When car needs to turn left, two motors on the left side will not work and two motors on the right side will turn so car will turn left. Then, when detected line, car needs to turn more smooth. So, in this situation, three of motors work and one motor does not work so the car will turn more smooth.

Mehmet Bayık
21802166
EE-102 Section 3
Term Project – Car Safety System
31.05.2021

Design Methodology:

In this part I will discuss about schematics.

Top Module:

Top module is very clear and understandable thanks to modular design. This basically connects modules.

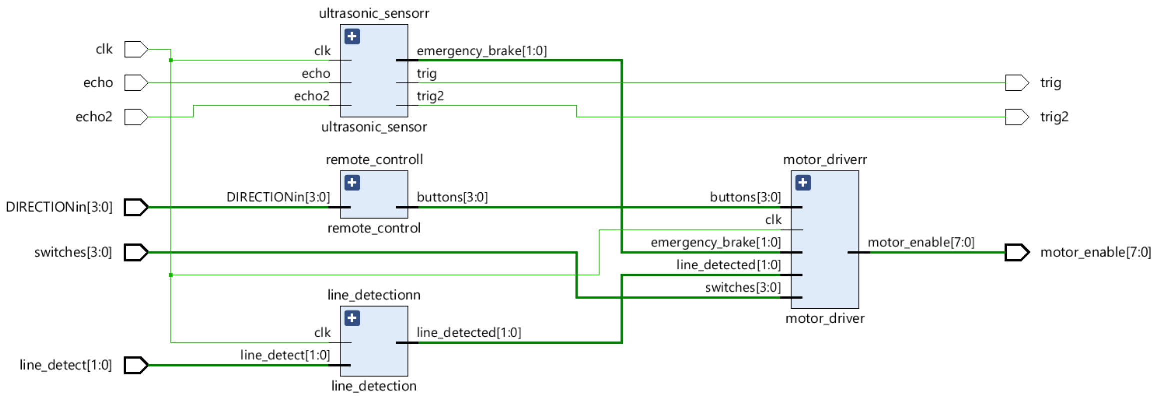


Figure 1: RTL Schematic of project

Ultrasonic Sensor:

In this module, I used two ultrasonic sensors. To use ultrasonic sensors, I send trig signal to both of them and counted time from start of trig signal to end of the echo signal. This is time of the travel. Then I divided by speed of sound and also divided by 2 because this will be double of distance. So I calculated distance and if it be less than 20 cm, car will stop (motors will not work).

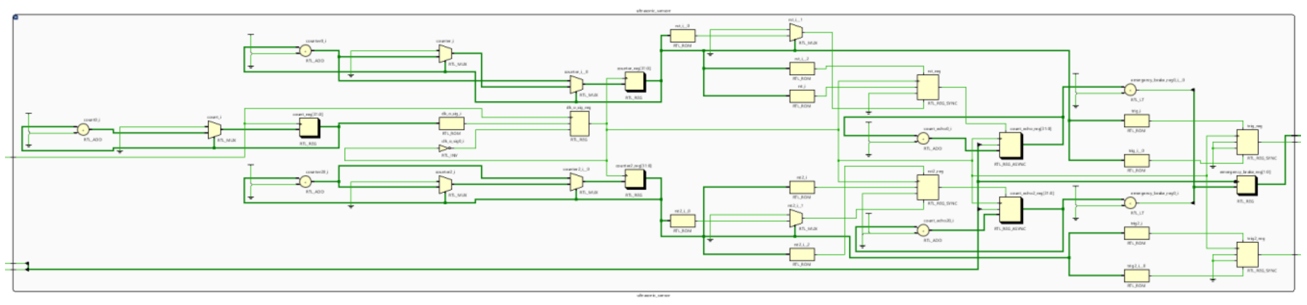


Figure 2: Schematic of ultrasonic sensor

Mehmet Bayık
21802166
EE-102 Section 3
Term Project – Car Safety System
31.05.2021

Motor Driver:

In this module, I used many loops to control motors with using motor drivers act upon signals comes from ultrasonic sensor, IR sensor and buttons we pressed on remote.

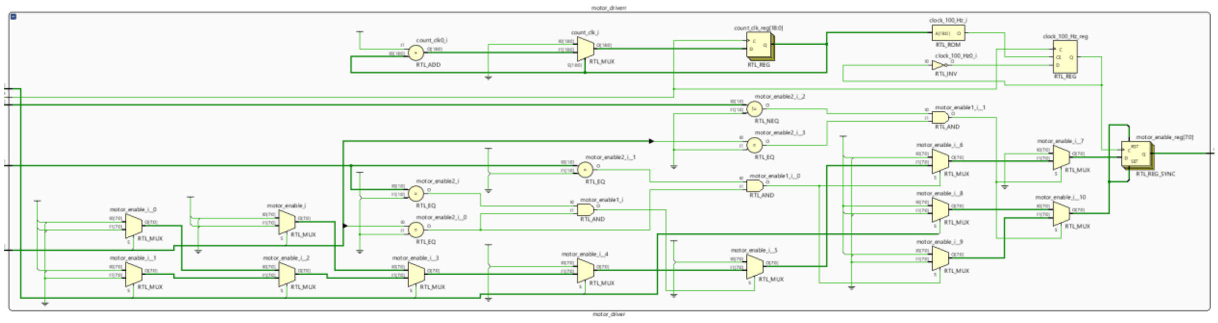


Figure 3: Schematic of Motor Driver

Remote Control:

This module is connects signals from coming to pmod connectors from arduino with other modules.

Line Detection:

In this module, I used two infrared sensors output and created finite state machine to keep this signals for predetermined time (1.4 sec). Infrared sensors give digital output, so these signals can be used directly.

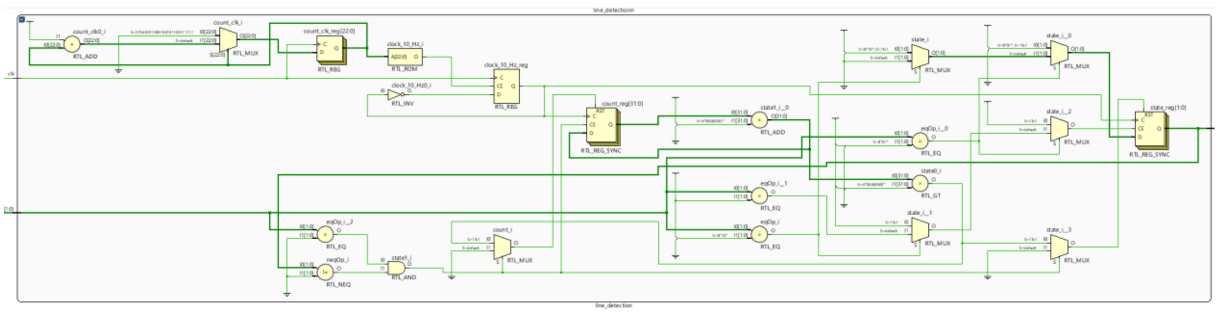


Figure 4: Schematic of Line Detection

Mehmet Bayık
21802166
EE-102 Section 3
Term Project – Car Safety System
31.05.2021

Results:

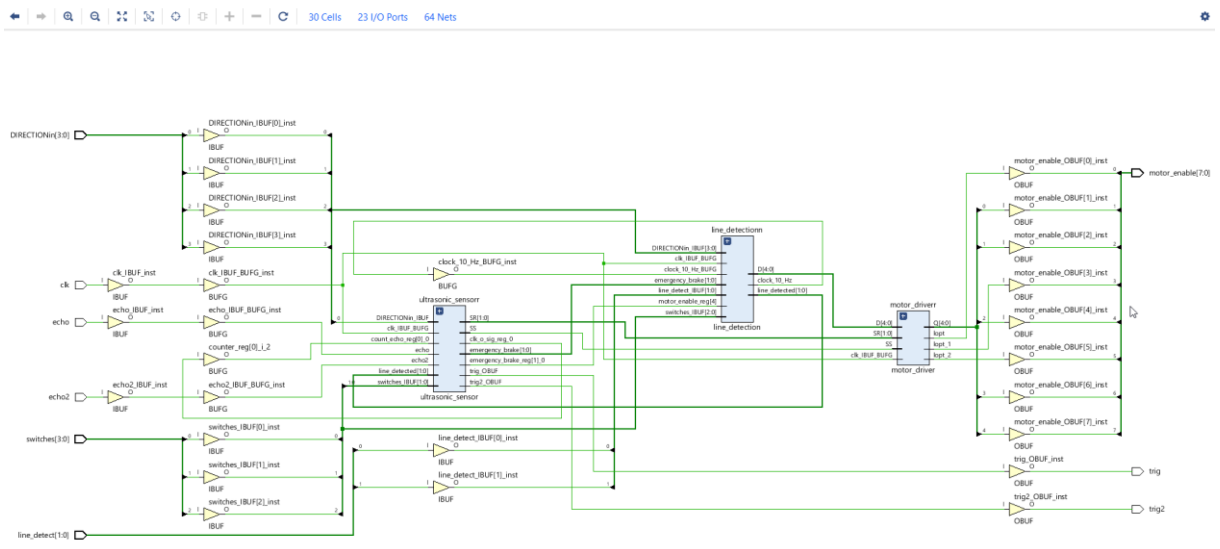
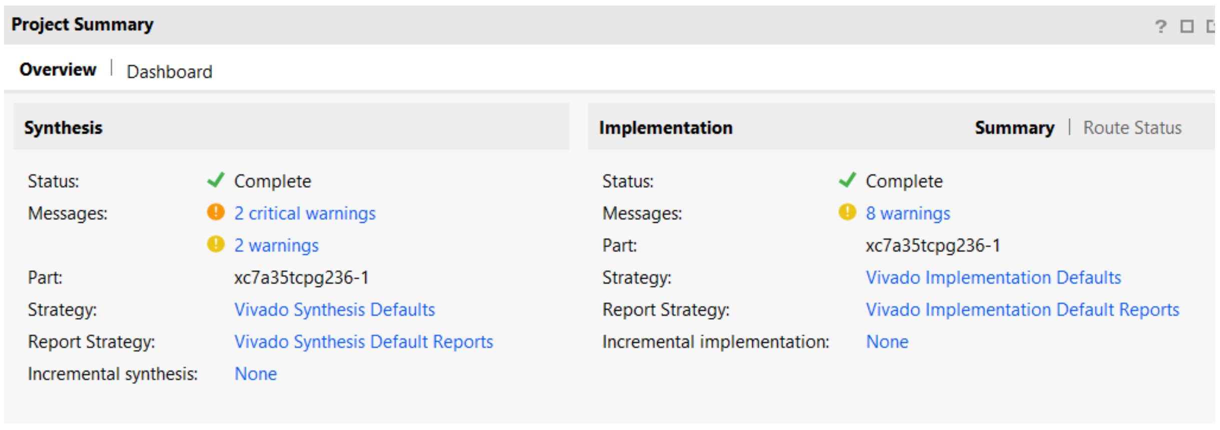


Figure 5: Implementation Schematic of Project



Project Summary

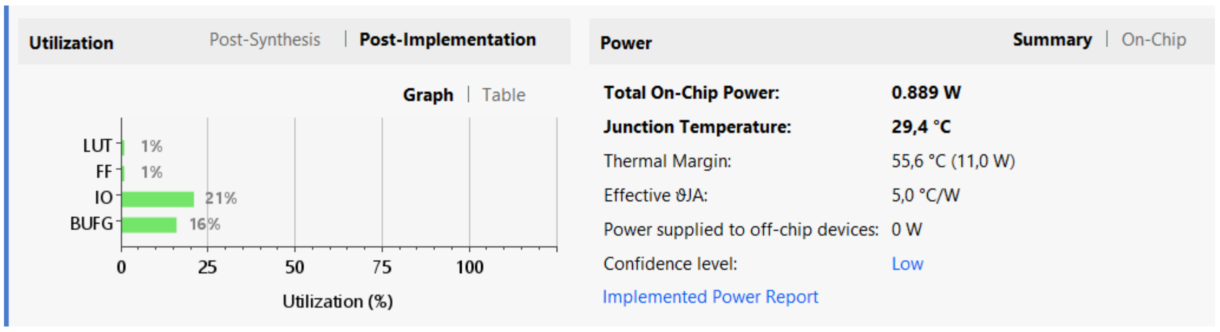
Overview | Dashboard

Synthesis	
Status:	✓ Complete
Messages:	2 critical warnings
	2 warnings
Part:	xc7a35tcpg236-1
Strategy:	Vivado Synthesis Defaults
Report Strategy:	Vivado Synthesis Default Reports
Incremental synthesis:	None

Implementation	**Summary** | Route Status
Status:	✓ Complete
Messages:	8 warnings
Part:	xc7a35tcpg236-1
Strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Implementation Default Reports
Incremental implementation:	None

Figure 6: Project Summary

Two critical warnings caused by clocks used in ultrasonic sensor module . It is not affecting project. Other warnings are not important.



Power — **Summary** | On-Chip

Total On-Chip Power:	**0.889 W**
Junction Temperature:	**29,4 °C**
Thermal Margin:	55,6 °C (11,0 W)
Effective ϑJA:	5,0 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

Figure 6: Project Post-Implementation Utilization

Mehmet Bayık
21802166
EE-102 Section 3
Term Project – Car Safety System
31.05.2021

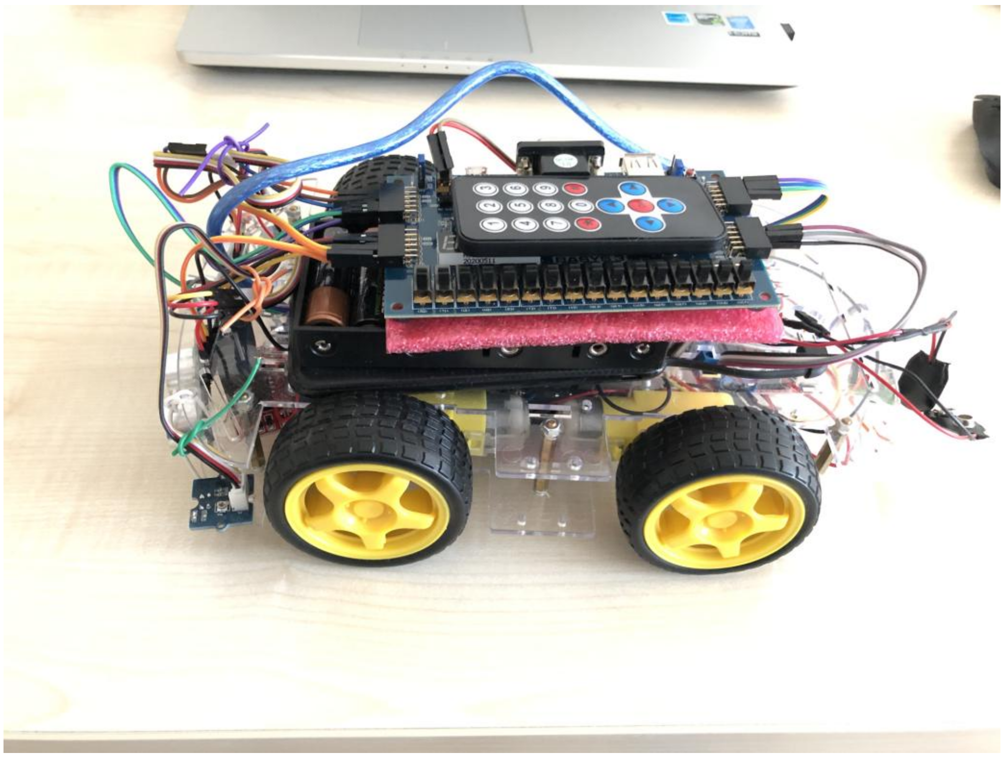


Figure 7: Last setup of car

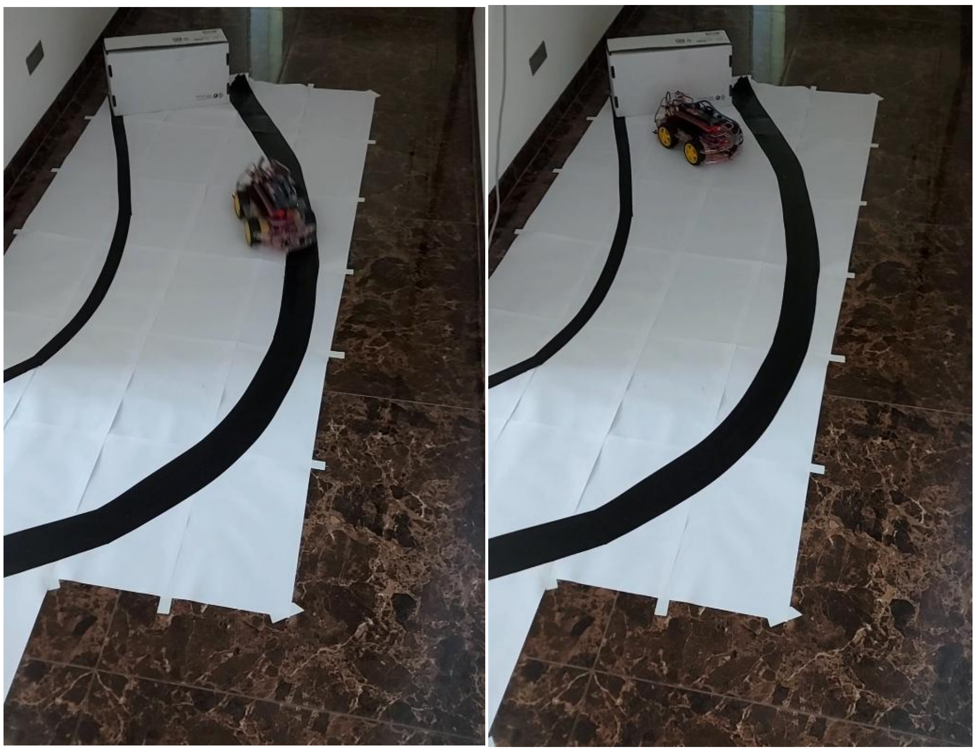


Figure 8: Working car in platform

Mehmet Bayık
21802166
EE-102 Section 3
Term Project – Car Safety System
31.05.2021

Conclusion:

In this project, I created remote controlled toy car with preventive safety systems. Checking datasheets of all equipments, voltage requirements, output signal types was challenging. Some sensors need 5V so I needed to use arduino to supply 5V since Basys3 can only supply 3.3V. Also, motors need more voltage than other equipments. Firstly I used 9V battery but these 9V batteries has little energy inside near 150 mAh. It drains away in 15 minutes when I used 9V alkaline battery as main battery source. So, I splitted power sources: powerbank and batteries. I powered arduino with powerbank and powered all with 5V output of arduino other equipments including Basys3 except motor drivers. Other power source was 8 x 2600 mAh Ni-Mh batteries. I connected these batteries to motor drivers. I used common ground between 5V and 9.6V because if I do not use common ground, sensors was not working as expected.

Coding part was very educatory. I learned a lot of things while writing codes of these project. I had many hard times. For example while testing ultrasonic sensor, I was connected sensor trig input to wrong pmod connector and until I realized this, I did many changes in code. To use infrared remote controller, I tried so many times but I did not achieved to decode nec protocol in VHDL. So, I write arduino code with using arduino library to decode remote control and I take outputs from arduino.

In conclusion, this project was challenging but educative. I learned how to search for VHDL since there is not many VHDL sources in the internet. Writing codes in VHDL is very different from other languages because VHDL is low level descriptive language. Many things are undoable in VHDL. For example I coincide multiple driven errors because changing signals in multiple process's is not allowed in VHDL. But VHDL has many advantages too. For example it is very fast thanks to not using microprocessor but logic gates.

Appendices:

Top Module:

```
library IEEE;
```

Mehmet Bayık
21802166
EE-102 Section 3
Term Project – Car Safety System
31.05.2021

```
use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;

use IEEE.std_logic_unsigned.all;


entity car_safety is

        Port(

                clk: in std_logic;

                echo:in std_logic;

     trig:out std_logic:='1';

                echo2:in std_logic;

     trig2:out std_logic:='1';

                line_detect: in std_logic_vector(1 downto 0);

                DIRECTIONin : in std_logic_vector(3 downto 0);

                motor_enable: out std_logic_vector(7 downto 0);

                switches: in std_logic_vector(3 downto 0)


                );

end car_safety;


architecture Behavioral of car_safety is


component remote_control is

Port(

        DIRECTIONin : in std_logic_vector(3 downto 0);
```

```
   buttons : out std_logic_vector(3 downto 0)


          );

end component;


component motor_driver is

Port(

          clk: in std_logic;

   buttons: in std_logic_vector(3 downto 0);

   motor_enable: out std_logic_vector(7 downto 0);

          line_detected: in std_logic_vector(1 downto 0);

          emergency_brake:in std_logic_vector(1 downto 0);

          switches: in std_logic_vector(3 downto 0)


          );

end component;


component line_detection is

Port(

   clk: in std_logic;

          line_detect: in std_logic_vector(1 downto 0);

   line_detected: out std_logic_vector(1 downto 0)


          );
```

```
end component;


component ultrasonic_sensor is

Port(

        clk: in std_logic;

   echo:in std_logic;

   trig:out std_logic:='1';

        echo2:in std_logic;

   trig2:out std_logic:='1';

   emergency_brake:out std_logic_vector(1 downto 0)


        );

end component;


signal buttons : std_logic_vector(3 downto 0);

signal line_detected: std_logic_vector(1 downto 0);

signal emergency_brake: std_logic_vector(1 downto 0);


begin


remote_controll: remote_control port map ( DIRECTIONin => DIRECTIONin, buttons =>
buttons);

motor_driverr: motor_driver port map ( clk => clk, buttons => buttons, motor_enable =>
motor_enable, line_detected => line_detected, emergency_brake => emergency_brake,
switches => switches);
```

```
line_detectionn: line_detection port map ( clk => clk, line_detect => line_detect,
line_detected => line_detected);

ultrasonic_sensorr: ultrasonic_sensor port map ( clk => clk, echo => echo, trig => trig, echo2
=> echo2, trig2 => trig2, emergency_brake => emergency_brake);


end Behavioral;
```

Motor Driver Module:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;



entity motor_driver is

   Port(

                  clk: in std_logic;

      buttons: in std_logic_vector(3 downto 0);

      motor_enable: out std_logic_vector(7 downto 0);

                  line_detected: in std_logic_vector(1 downto 0);

                  emergency_brake:in std_logic_vector(1 downto 0);

                  switches: in std_logic_vector(3 downto 0)



      );

end motor_driver;
```

```
architecture Behavioral of motor_driver is

signal clock_100_Hz: std_logic;

begin



clk_div : process (clk) -- 100MHz to 100Hz, 50% duty cycle, clock divider

variable count_clk : integer range 0 to 499999 := 0;  -- (100.000.000/100)/2 - 1

begin

   if (rising_edge(clk)) then

            if (count_clk = 499999) then

               clock_100_Hz <= not clock_100_Hz; -- to obtain a 50% duty cycle

               count_clk := 0;

            else

               count_clk := count_clk + 1;

                   end if;

          end if;

end process clk_div;



process(clock_100_Hz)

variable split_second: integer := 0;

  begin

  if rising_edge(clock_100_Hz) then

          if emergency_brake(1 downto 0) /= "00" and switches(1) = '0' then --stop

            if buttons(0) = '1' then --backward

                            motor_enable(7 downto 6) <= "10";
```

```
                        motor_enable(5 downto 4) <= "10";

                        motor_enable(3 downto 2) <= "10";

                        motor_enable(1 downto 0) <= "01";

        else

     motor_enable(7 downto 6) <= "00";

     motor_enable(5 downto 4) <= "00";

     motor_enable(3 downto 2) <= "00";

     motor_enable(1 downto 0) <= "00";

end if;

      else

          if line_detected = "01" and switches(0) = '0'  then --turn left

             motor_enable(7 downto 6) <= "01";

    motor_enable(5 downto 4) <= "01";

    motor_enable(3 downto 2) <= "00";

    motor_enable(1 downto 0) <= "10";


          elsif line_detected = "10" and switches(0) = '0' then --turn right

             motor_enable(7 downto 6) <= "01";

    motor_enable(5 downto 4) <= "01";

    motor_enable(3 downto 2) <= "01";

    motor_enable(1 downto 0) <= "00";


          else

     if buttons(3) = '1' then --forward
        motor_enable(7 downto 6) <= "01";

        motor_enable(5 downto 4) <= "01";

        motor_enable(3 downto 2) <= "01";

        motor_enable(1 downto 0) <= "10";


     elsif buttons(2) = '1' then --left

        if switches(2) = '1' then

        motor_enable(7 downto 6) <= "01";

        motor_enable(5 downto 4) <= "01";

        motor_enable(3 downto 2) <= "00";

        motor_enable(1 downto 0) <= "10";

        else

        motor_enable(7 downto 6) <= "00";

        motor_enable(5 downto 4) <= "01";

        motor_enable(3 downto 2) <= "00";

        motor_enable(1 downto 0) <= "10";

        end if;


     elsif buttons(1) = '1' then --right

        if switches(2) = '1' then

        motor_enable(7 downto 6) <= "01";

        motor_enable(5 downto 4) <= "01";

        motor_enable(3 downto 2) <= "01";

        motor_enable(1 downto 0) <= "00";
```

```
            else

            motor_enable(7 downto 6) <= "01";

            motor_enable(5 downto 4) <= "00";

            motor_enable(3 downto 2) <= "01";

            motor_enable(1 downto 0) <= "00";

            end if;


          elsif buttons(0) = '1' then --backward

            motor_enable(7 downto 6) <= "10";

            motor_enable(5 downto 4) <= "10";

            motor_enable(3 downto 2) <= "10";

            motor_enable(1 downto 0) <= "01";


          else                                          -- stop

            motor_enable(7 downto 6) <= "00";

            motor_enable(5 downto 4) <= "00";

            motor_enable(3 downto 2) <= "00";

            motor_enable(1 downto 0) <= "00";


          end if;

                 end if;

           end if;

  end if;

end process;
end Behavioral;
```

Remote Control Module:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity remote_control is

   Port (

      DIRECTIONin : in std_logic_vector(3 downto 0);

      buttons : out std_logic_vector(3 downto 0)



      );

end remote_control;


architecture Behavioral of remote_control is


begin


buttons <= DIRECTIONin;


end Behavioral;
```

Line Detection Module:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;

use IEEE.std_logic_unsigned.all;
```

Mehmet Bayık
21802166
EE-102 Section 3
Term Project – Car Safety System
31.05.2021

```
entity line_detection is
   Port(
       clk: in std_logic;
       line_detect: in std_logic_vector(1 downto 0);
       line_detected: out std_logic_vector(1 downto 0)
       );
end line_detection;

architecture Behavioral of line_detection is

signal clock_10_Hz: std_logic;
signal state: std_logic_vector(1 downto 0);

begin

clk_div : process (clk) -- 100MHz to 10Hz, 50% duty cycle, clock divider
variable count_clk : integer range 0 to 4999999 := 0;  -- (100.000.000/10)/2 - 1
begin
   if (rising_edge(clk)) then
            if (count_clk = 4999999) then
               clock_10_Hz <= not clock_10_Hz; -- to obtain a 50% duty cycle
               count_clk := 0;
            else
```

```
                count_clk := count_clk + 1;
                   end if;
            end if;
end process clk_div;


process(clock_10_Hz)
variable count: integer:=0;
begin
   if rising_edge(clock_10_Hz) then
      if line_detect = "01" then
         state <= "01";
      elsif line_detect = "10" then
         state <= "10";
      elsif line_detect = "11" then
         state <= "00";
      else
         state <= state;
      end if;


      if line_detect = "00" and state /= "00" then
         count := count +1;
         if count > 14 then
            state <= "00";
            count := 0;
```

```
          end if;

        end if;


    end if;

end process;


line_detected <= state;


end Behavioral;
```

Ultrasonic Sensor Module:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;


entity ultrasonic_sensor is

    Port (

        clk: in std_logic;

        echo:in std_logic;

        trig:out std_logic:='1';

                        echo2:in std_logic;

        trig2:out std_logic:='1';

        emergency_brake:out std_logic_vector(1 downto 0)

        );

end ultrasonic_sensor;
```

Mehmet Bayık
21802166
EE-102 Section 3
Term Project – Car Safety System
31.05.2021

```
architecture Behavioral of ultrasonic_sensor is

signal count:integer;

signal clk_o_sig,rst,rst2:std_logic:='0';

signal counter,count_echo:integer:=0;

signal counter2,count_echo2:integer:=0;


begin

process(clk) --1 MHz clock divider

   begin

      if rising_edge(clk) then

         if (count=49) then

         clk_o_sig <= not clk_o_sig;

         count<=0;

         else

         count <=count+1;

         end if;

      end if;

end process;


process (clk_o_sig)

   begin


   if rising_edge(clk_o_sig) then
```

```
        if counter=10 then

        trig <='0';

        counter <= counter+1;

        elsif counter=60000 then

        counter <=0;

        trig <='1';

        rst <='1';

        else

        rst <='0';

        counter <= counter+1;

        end if;

    end if;

end process;


process (echo,clk_o_sig,rst)

    begin

    if rst='1' then

    count_echo <=0;

    elsif rising_edge(clk_o_sig) then

        if echo = '1' then

        count_echo <=count_echo+1;

        end if;

    end if;

end process;
```

```
process(echo)

   begin

   if falling_edge (echo) then

      if count_echo < 1161 then --290x3 +1 = 871 = 15 cm

      emergency_brake(1) <= '1';

      else

      emergency_brake(1) <= '0';

      end if;

   end if;

end process;





process (clk_o_sig)

   begin


   if rising_edge(clk_o_sig) then

      if counter2=10 then

      trig2 <='0';

      counter2 <= counter2+1;

      elsif counter2=60000 then

      counter2 <=0;
```

```
        trig2 <='1';

        rst2 <='1';

        else

        rst2 <='0';

        counter2 <= counter2+1;

        end if;

    end if;

end process;


process (echo2,clk_o_sig,rst2)

    begin

    if rst2='1' then

    count_echo2 <=0;

    elsif rising_edge(clk_o_sig) then

        if echo2 = '1' then

        count_echo2 <=count_echo2+1;

        end if;

    end if;

end process;


process(echo2)

    begin

    if falling_edge (echo2) then

        if count_echo2 < 1161 then --290x3 +1 = 871 = 15 cm
        emergency_brake(0) <= '1';
        else
        emergency_brake(0) <= '0';
        end if;
    end if;
end process;
end Behavioral;
```

Constraints:

```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
    set_property IOSTANDARD LVCMOS33 [get_ports clk]

set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets echo_IBUF]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets echo2_IBUF]

#Pmod Header JXAC
#ir sensor sol (sol alt pin)
set_property PACKAGE_PIN N1 [get_ports line_detect[1]]
    set_property IOSTANDARD LVCMOS33 [get_ports line_detect[1]]
#sol üst iki
set_property PACKAGE_PIN L3 [get_ports echo]
    set_property IOSTANDARD LVCMOS33 [get_ports echo]
set_property PACKAGE_PIN J3 [get_ports trig]
```

```
    set_property IOSTANDARD LVCMOS33 [get_ports trig]
set_property PACKAGE_PIN M3 [get_ports echo2]
    set_property IOSTANDARD LVCMOS33 [get_ports echo2]
set_property PACKAGE_PIN K3 [get_ports trig2]
    set_property IOSTANDARD LVCMOS33 [get_ports trig2]



#Pmod Header JB
#üst dörtlü 8 to sol ,11 to sag
set_property PACKAGE_PIN A14 [get_ports DIRECTIONin[0]]
    set_property IOSTANDARD LVCMOS33 [get_ports DIRECTIONin[0]]
set_property PACKAGE_PIN A16 [get_ports DIRECTIONin[1]]
    set_property IOSTANDARD LVCMOS33 [get_ports DIRECTIONin[1]]
set_property PACKAGE_PIN B15 [get_ports DIRECTIONin[2]]
    set_property IOSTANDARD LVCMOS33 [get_ports DIRECTIONin[2]]
set_property PACKAGE_PIN B16 [get_ports DIRECTIONin[3]]
    set_property IOSTANDARD LVCMOS33 [get_ports DIRECTIONin[3]]

#Pmod Header JA
#ir sensor sag (sol alt pin)
set_property PACKAGE_PIN G3 [get_ports line_detect[0]]
    set_property IOSTANDARD LVCMOS33 [get_ports line_detect[0]]

#sag dörtlü
```

```
set_property PACKAGE_PIN L2 [get_ports motor_enable[7]]

   set_property IOSTANDARD LVCMOS33 [get_ports motor_enable[7]]

set_property PACKAGE_PIN J1 [get_ports motor_enable[6]]

   set_property IOSTANDARD LVCMOS33 [get_ports motor_enable[6]]

set_property PACKAGE_PIN K2 [get_ports motor_enable[5]]

   set_property IOSTANDARD LVCMOS33 [get_ports motor_enable[5]]

set_property PACKAGE_PIN H1 [get_ports motor_enable[4]]

   set_property IOSTANDARD LVCMOS33 [get_ports motor_enable[4]]


#Pmod Header JC

#sol dörtlü

set_property PACKAGE_PIN P18 [get_ports motor_enable[3]]

   set_property IOSTANDARD LVCMOS33 [get_ports motor_enable[3]]

set_property PACKAGE_PIN N17 [get_ports motor_enable[2]]

   set_property IOSTANDARD LVCMOS33 [get_ports motor_enable[2]]

set_property PACKAGE_PIN P17 [get_ports motor_enable[1]]

   set_property IOSTANDARD LVCMOS33 [get_ports motor_enable[1]]

set_property PACKAGE_PIN R18 [get_ports motor_enable[0]]

   set_property IOSTANDARD LVCMOS33 [get_ports motor_enable[0]]


#Switches

#lane keep assist

set_property PACKAGE_PIN V17 [get_ports switches[0]]

   set_property IOSTANDARD LVCMOS33 [get_ports switches[0]]
```

```
#collision avoidance

set_property PACKAGE_PIN V16 [get_ports switches[1]]

   set_property IOSTANDARD LVCMOS33 [get_ports switches[1]]

#forward turn

set_property PACKAGE_PIN W16 [get_ports switches[2]]

   set_property IOSTANDARD LVCMOS33 [get_ports switches[2]]

#empty for leds

set_property PACKAGE_PIN W17 [get_ports switches[3]]

   set_property IOSTANDARD LVCMOS33 [get_ports switches[3]]
```

Arduino Code for IR Remote:

```
#include <IRremote.h>


#define MAX_TIME 250 // max ms between codes


long lastPressTime = 0;

int state = 0;


int IR_RECEIVE_PIN = 7;




int FORWARD = 8;

int LEFT = 9;

int RIGHT = 10;
int BACKWARD = 11;

byte same;




void setup()

{

  pinMode(FORWARD, OUTPUT);

  pinMode(LEFT, OUTPUT);

  pinMode(RIGHT, OUTPUT);

  pinMode(BACKWARD, OUTPUT);

  Serial.begin(9600);

  IrReceiver.begin(IR_RECEIVE_PIN, ENABLE_LED_FEEDBACK,
USE_DEFAULT_FEEDBACK_LED_PIN);

}

void loop() {


  if (IrReceiver.decode())

  {

    IrReceiver.printIRResultMinimal(&Serial);

    Serial.println();


      if (IrReceiver.decodedIRData.command == 0x18)

      {

        digitalWrite(LEFT, 0);

        digitalWrite(RIGHT, 0);
```

```
 digitalWrite(BACKWARD, 0);

 if (state == 0)

 {

 digitalWrite(FORWARD, 0);

 state = 1;  // Button pressed, so set state to HIGH


 }

 digitalWrite(FORWARD, 1);

 lastPressTime = millis();

}

if (IrReceiver.decodedIRData.command == 0x8)

{

 digitalWrite(FORWARD, 0);

 digitalWrite(RIGHT, 0);

 digitalWrite(BACKWARD, 0);

 if (state == 0)

 {

 digitalWrite(LEFT, 0);

 state = 1;  // Button pressed, so set state to HIGH


 }

 digitalWrite(LEFT, 1);

 lastPressTime = millis();

}
```

```
if (IrReceiver.decodedIRData.command == 0x5A)

{

 digitalWrite(FORWARD, 0);

 digitalWrite(LEFT, 0);

 digitalWrite(BACKWARD, 0);


 if (state == 0)

 {

 digitalWrite(RIGHT, 0);

 state = 1;  // Button pressed, so set state to HIGH


 }

 digitalWrite(RIGHT, 1);

 lastPressTime = millis();

}


if (IrReceiver.decodedIRData.command == 0x52)

{

 digitalWrite(FORWARD, 0);

 digitalWrite(LEFT, 0);

 digitalWrite(RIGHT, 0);

 if (state == 0)

 {

 digitalWrite(BACKWARD, 0);
```

```
      state = 1;  // Button pressed, so set state to HIGH


      }

      digitalWrite(BACKWARD, 1);

      lastPressTime = millis();

    }

  }


IrReceiver.resume();

if (state == 1 && millis() - lastPressTime > MAX_TIME)

  {

    state = 0; // Haven't heard from the button for a while, so not pressed

    digitalWrite(FORWARD, 0);

    digitalWrite(LEFT, 0);

    digitalWrite(RIGHT, 0);

    digitalWrite(BACKWARD, 0);

  }

}
```

Mehmet Bayık
21802166
EE-102 Section 3
Term Project – Car Safety System
31.05.2021

Photos From Project Process:

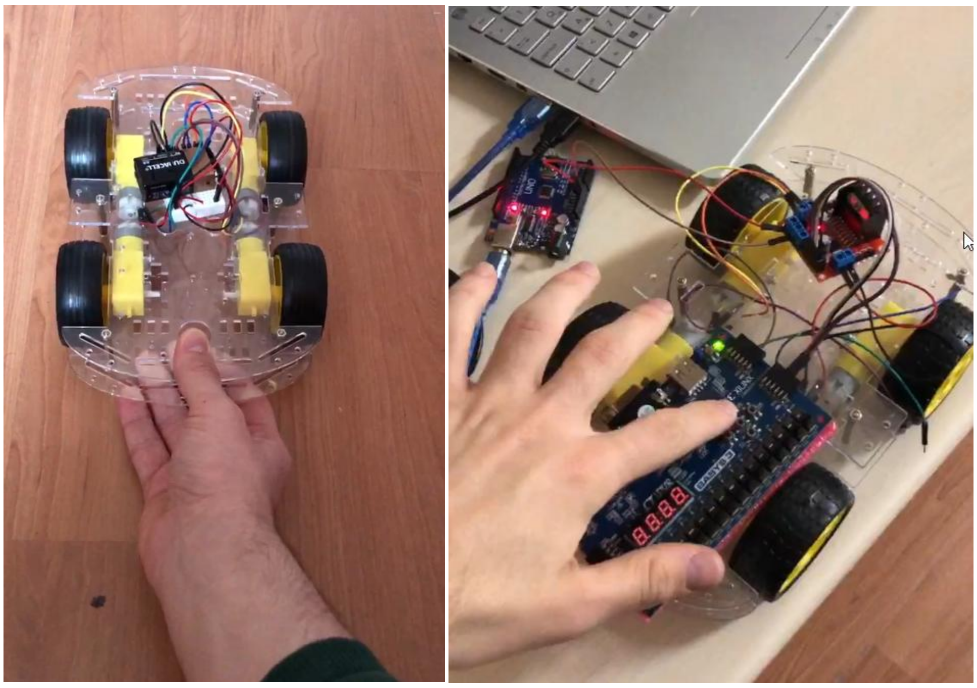


Figure 9: Motor tests

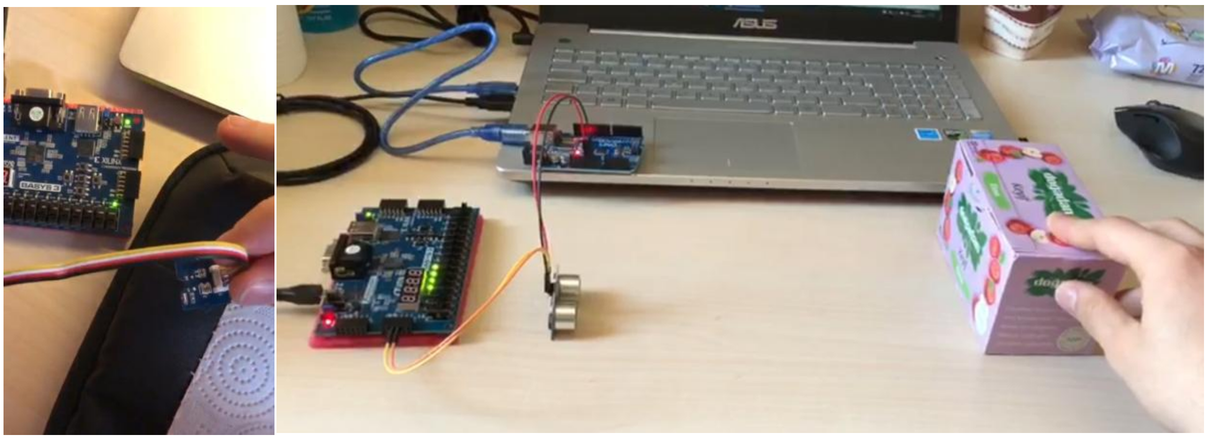


Figure 10: Sensor tests

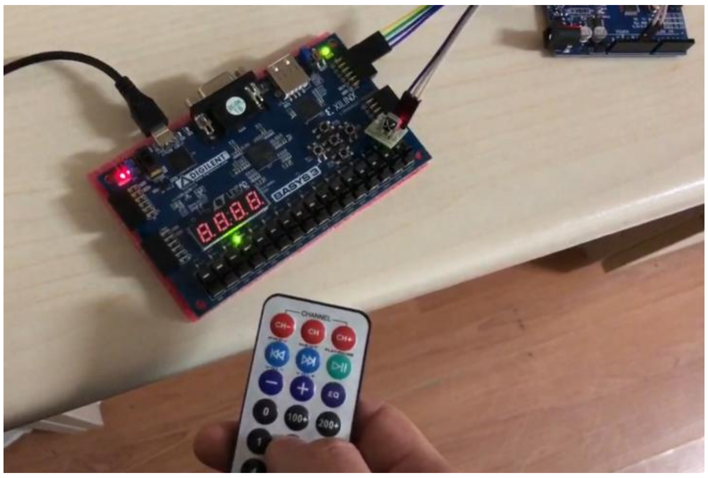


Figure 11: IR remote test

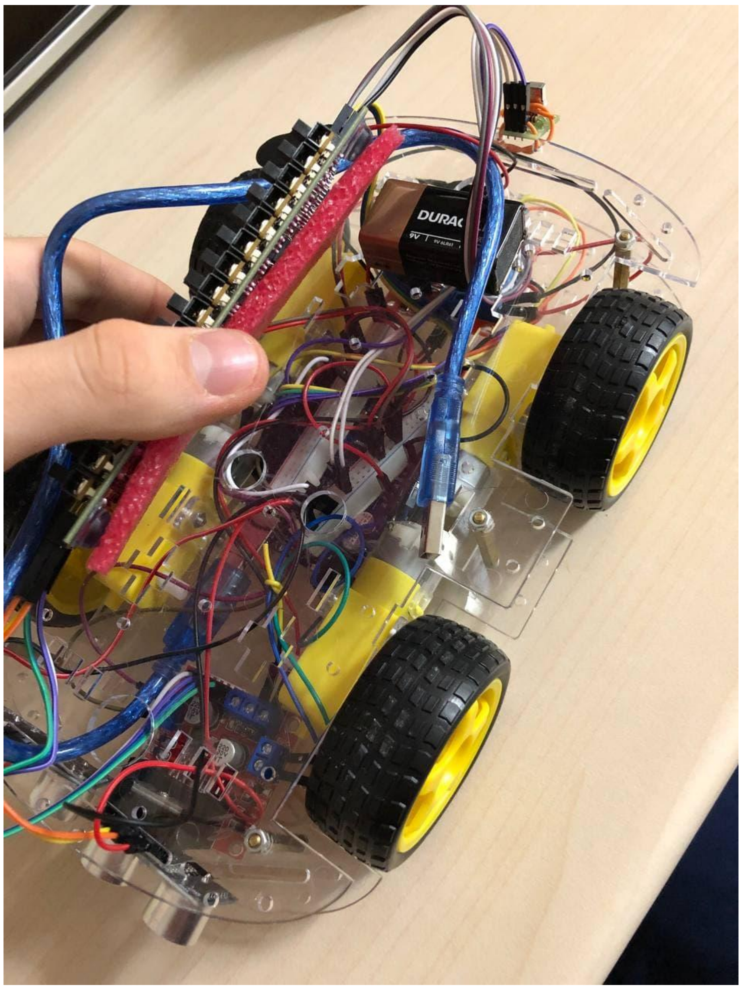


Figure 11: Cable connections

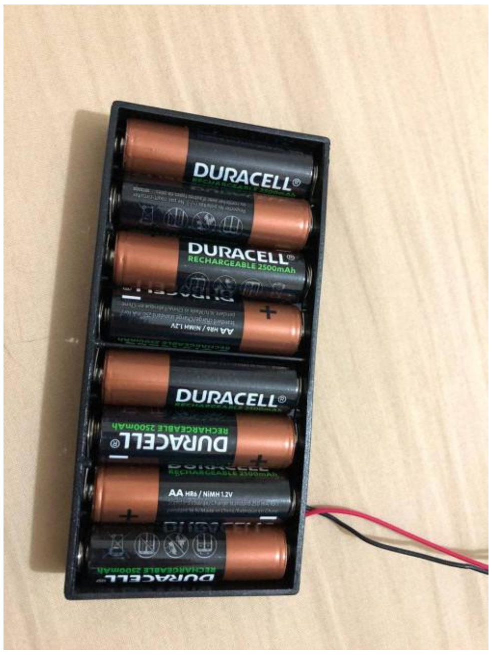


Figure 12: 8 x Ni-Mh batteries pack

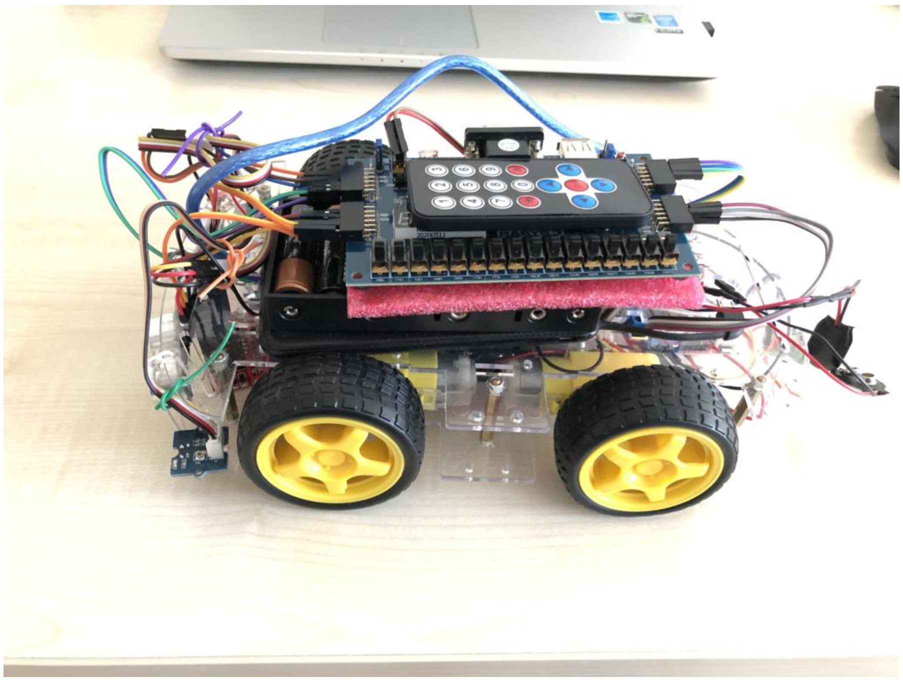


Figure 13: Last version of car