**VehicleWith Ultrasonic Eye**

by

1. EMRE BALSEVER & YUNUS ISSAHAK

June, 2015

**VehicleWith Ultrasonic Eye**

by

1. Emre BALSEVER & Yunus ISSAHAK

A Senior Design Final Report Submited to

the Department of Electrical and Electronics Engineering

of

Fatih University

in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Electrical and Electronics Engineering

June 2015

Istanbul, Turkey

# APPROVAL PAGE

This is to certify that I have read this final report and that in my opinion it is fully adequate, in scope and quality, as report for the course of EEE 498 Senior Design Project.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Dr. ONUR TOKER

Advisor

Examining Jury Members

.................................................. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.................................................. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.................................................. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.................................................. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.................................................. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

It is approved that this report has been written in compliance with the formatting rules laid down by Faculty of Engineering.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Assoc. Prof. Dr. LOKMAN ERZEN

Course Coordinator

June, 2015

# ABSTRACT

**VehicleWith Ultrasonic Eye**

AhmetEmre BALSEVER &Yunus ISSAHAK

Electrical and Electronics Engineering

June 2015

Advisor: Prof. Dr. ONUR TOKER

This Project designed for every car, buses or any vehicle..afpga will be used to control system. The user sets the distance between front car to follow it. Ultrasonic sensors will be used to calculate distance between two vehicle.

The user activate the system then system follow front car. The system read the distance then take last seven value and compare number. If last value bigger than first value it means front car is speeding up and fpga command to speed up. If last value smaller than first value it means front car is slowing down and controller command to slow down.

On this note, this is important, because nowadays many system become automated. But not enough interest in the automotive sector. We need to implement technological improvements in vehicles. There are many car accidents being everyday. This system facilitates driving car. Driving car will be easier.

TABLE OF CONTENTS

[APPROVAL PAGE i](#_Toc420929660)ii

[ABSTRACT iv](#_Toc420929661)

[LIST OF FIGURES vi](#_Toc420929663)

[INTRODUCTION 1](#_Toc420929664)

[1 –AUTONOMOS CAR 2](#_Toc420929665)

[1.1 EYES OF VEHICLES 3](#_Toc420929666)

[1.1.1 Radar 3](#_Toc420929667)

[1.1.2 Stereo Camera and Infrared Camera 4](#_Toc420929668)

[1.1.3 Lidar 5](#_Toc420929669)

[2 -FPGA AND ULTRASONIC 6](#_Toc420929674)

[2.1 FPGA 6](#_Toc420929675)

[2.2ULTRASONIC SENSOR 7](#_Toc420929675)

[3 – DESIGN AND BUILDING 9](#_Toc420929677)

[3.1 PROGRAMING CODE 9](#_Toc420929678)

[3.2MOUNTING SYSTEMS 23](#_Toc420929678)

[4- CONCLUSION 24](#_Toc420929684)

[5-REFERANCES 25](#_Toc420929685)

# 

# LIST OF FIGURES

[Figure 1 –Electronic Brain 1](#_Toc420928522)

[Figure 2 – Autonomous Car 2](#_Toc420928523)

[Figure 3 - Sensors 4](#_Toc420928524)

[Figure 4 - Cameras 4](#_Toc420928525)

[Figure 5 - Lidar Map 5](#_Toc420928526)

[Figure 6 - FPGA 6](#_Toc420928527)

[Figure 7 – Ultrsonic snsors 7](#_Toc420928528)

[Figure 8 – Our Prototype 23](#_Toc420928528)

INTRODUCTION

With the development of technology in everyday life are computer-controlled system expansion. An intelligent system is a machine with an embedded, Internet-connected computer that has the capacity to gather and analyze data and communicate with other systems. Requirements for an intelligent system include security, connectivity, the ability to adapt according to current data and the capacity for remote monitoring and management.

Essentially, an intelligent system is anything that contains a functional, although not usually [general-purpose](http://whatis.techtarget.com/definition/general-purpose-computer), computer with Internet connectivity.  An [embedded system](http://searchenterpriselinux.techtarget.com/definition/embedded-system)  may be powerful and capable of complex processing and data analysis, but it is usually specialized for tasks relevant to the host machine.

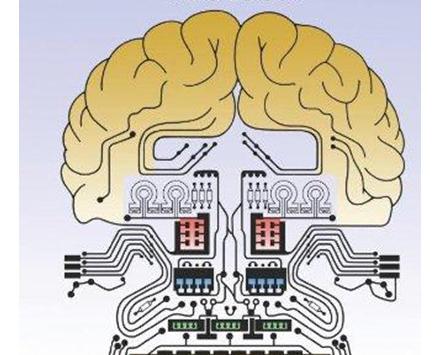


Figure 1-Electronic brain

# 1 - AUTONOMOS CAR

An **autonomous car**,[[1]](http://en.wikipedia.org/wiki/Autonomous_car#cite_note-1) also known as a **driverless car**,[[2]](http://en.wikipedia.org/wiki/Autonomous_car#cite_note-2) **self-driving car**[[3]](http://en.wikipedia.org/wiki/Autonomous_car#cite_note-3) and **robotic car**,[[4]](http://en.wikipedia.org/wiki/Autonomous_car#cite_note-thrun2010toward-4) is an [automated](http://en.wikipedia.org/wiki/Automation) or [autonomous vehicle](http://en.wikipedia.org/wiki/Vehicular_automation)capable of fulfilling the main transportation capabilities of a traditional car. As an autonomous vehicle, it is capable of sensing its environment and navigating without human input. Robotic cars exist mainly as prototypes and demonstration systems. As of 2014, the only self-driving vehicles that are commercially available are open-air shuttles for pedestrian zones that operate at 12.5 miles per hour (20.1 km/h).

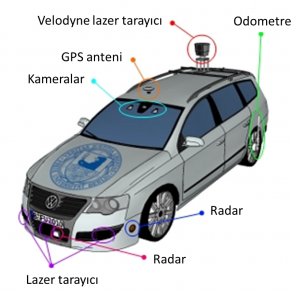
****

Figure 2–Autonomous Car

Autonomous vehicles sense their surroundings with such techniques as [radar](http://en.wikipedia.org/wiki/Radar), [lidar](http://en.wikipedia.org/wiki/Lidar), [GPS](http://en.wikipedia.org/wiki/GPS), and [computer vision](http://en.wikipedia.org/wiki/Computer_vision). Advanced control systems interpret sensory information to identify appropriate navigation paths, as well as obstacles and relevant signage.[[6]](http://en.wikipedia.org/wiki/Autonomous_car#cite_note-6)[[7]](http://en.wikipedia.org/wiki/Autonomous_car#cite_note-7) By definition, autonomous vehicles are capable of [updating their maps based on sensory input](http://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping), allowing the vehicles to keep track of their position even when conditions change or when they enter uncharted environments.

## EYES OF VEHICLES

Autonomous system need some equipmentsto understand what's going on around, to identify bodies. We could call them eyes of computer.

### Lazer, Radar

**Radar**is an object-detection system that uses [radio waves](http://en.wikipedia.org/wiki/Radio_wave) to determine the range, altitude, direction, or speed of objects. Page 2 Pulse-Doppler Method for Automotive Radar December 2013 Altera Corporation Implementing Digital Processing for Automotive Radar Using SoCs Pulse-Doppler Method for Automotive Radar Many radar systems employ pulse-Doppler methods, in which the transmitter operates for a short duration, then the system switches to receive mode until the next transmit pulse. The pulse-Doppler radar sends successive pulses at specific intervals or a pulse repetition interval (PRI). As the radar returns, the reflections are processed coherently to extract range and relative motion of detected objects. More sophisticated processing methods, such as space time adaptive radar (STAP) further process radar returns and extract target data even when heavily obscured by ground clutter or background returns surrounding the object(s) of interest. f For further information on radar basics, pulse-Doppler radar, STAP radar, and SAR radar, refer to EETimes’ tutorial on “Radar Basics.” In automotive radar, the range can be as short as a few meters to as much as a few hundred meters. For a range of 2 m, the round-trip transit time of the radar pulse is 13 ns. This short range requires that the transmitter and receiver operate simultaneously, which requires separate antennas. The pulse-Doppler radar sends a pulse periodically, and the ratio of the time the transmitter is active to the total time elapsed is the duty cycle. Since duty cycles are typically small, this ratio limits the total transmit power. The power, in turn limits the range of detection. Achieving a 1- to 2 m range resolution also requires a sample rate on the order of 100 MSPS or more, and the ability to digitally process the data in both range and Doppler dimensions. This high sample rate can increase the cost of the radar system. Alternatively, a radar method known as continuous wave frequency modulated (CWFM) can be employed. CWFM does not sent out pulses and then monitor the returns, or radar echo.



Figure 3-Sensors

### 1.1.2 Stereo Camera and Infrared Camera

A **stereo camera** is a type of[camera](http://en.wikipedia.org/wiki/Camera) with two or more lenses with a separate [image sensor](http://en.wikipedia.org/wiki/Image_sensor) or film frame for each lens. This allows the camera to simulate human [binocular vision](http://en.wikipedia.org/wiki/Binocular_vision), and therefore gives it the ability to capture three-dimensional images, a process known as[stereo photography](http://en.wikipedia.org/wiki/Stereo_photography).Stereo cameras may be used for making [stereoviews](http://en.wikipedia.org/wiki/Stereoscopy) and 3D pictures for movies, or for [range imaging](http://en.wikipedia.org/wiki/Range_imaging#Stereo_triangulation).

**Infrared** (**IR**) is invisible radiant energy, [electromagnetic radiation](http://en.wikipedia.org/wiki/Electromagnetic_radiation) with longer[wavelengths](http://en.wikipedia.org/wiki/Wavelength) than those of[visible light](http://en.wikipedia.org/wiki/Light), extending from the nominal [red](http://en.wikipedia.org/wiki/Red) edge of the [visible spectrum](http://en.wikipedia.org/wiki/Visible_spectrum) at 700 [nanometers](http://en.wikipedia.org/wiki/Nanometre)([frequency](http://en.wikipedia.org/wiki/Frequency_spectrum) 430 [THz](http://en.wikipedia.org/wiki/THz)) to 1 mm (300 [GHz](http://en.wikipedia.org/wiki/GHz))[[1]](http://en.wikipedia.org/wiki/Infrared#cite_note-1) (although people can see infrared up to at least 1050 nm in experiments[[2]](http://en.wikipedia.org/wiki/Infrared#cite_note-Sliney1976-2)[[3]](http://en.wikipedia.org/wiki/Infrared#cite_note-LynchLivingston2001-3)[[4]](http://en.wikipedia.org/wiki/Infrared#cite_note-Dash2009-4)[[5]](http://en.wikipedia.org/wiki/Infrared#cite_note-Saidman1933-5)). Most of the [thermal radiation](http://en.wikipedia.org/wiki/Thermal_radiation) emitted by objects near room temperature is infrared.

### C:\Users\Caan\Desktop\project raport\flir-t-series.pngC:\Users\Caan\Desktop\project raport\stereo.jpg

Figure 4- Cameras

### 

### 1.1.3 Lidar

**Lidar** (also written **LIDAR**,**LiDAR** or **LADAR**) is a [remote sensing](http://en.wikipedia.org/wiki/Remote_sensing) technology that measures distance by illuminating a target with a [laser](http://en.wikipedia.org/wiki/Laser)and analyzing the reflected light. Although thought by some to be an acronym of Light Detection And Ranging,[[1]](http://en.wikipedia.org/wiki/Lidar#cite_note-NOAA-1) the term lidar was actually created as a[portmanteau](http://en.wikipedia.org/wiki/Portmanteau) of "[light](http://en.wikipedia.org/wiki/Light)" and "[radar](http://en.wikipedia.org/wiki/Radar)".[[2]](http://en.wikipedia.org/wiki/Lidar#cite_note-Oxford-2)[[3]](http://en.wikipedia.org/wiki/Lidar#cite_note-James_Ring_p._672-3-3) Lidar is popularly used as a technology to make high-resolution maps.

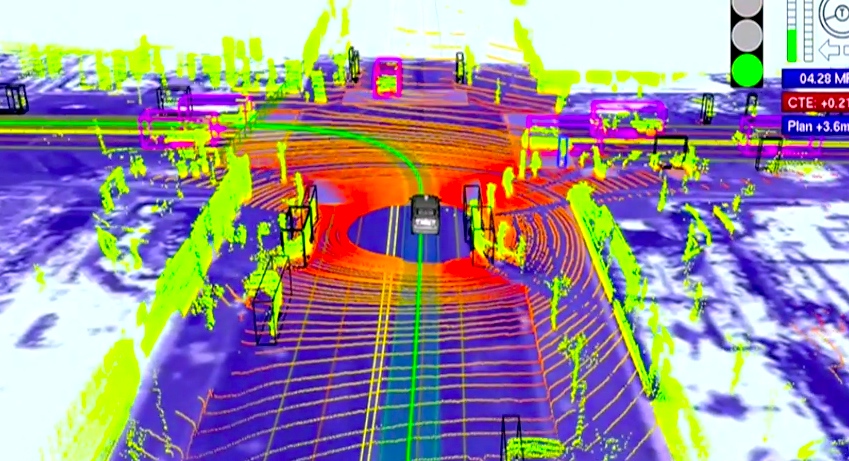


Figure 5- Lidar map

# 2 - FPGA AND ULTRASONIC

In our project we use only these two material, to complete ultrasonic vehicle project.

## 2.1 FPGA

The Basys2 board is a circuit design and implementation platform that anyone can use to gain experience building real digital circuits. Built around a Xilinx Spartan-3E Field Programmable Gate Array and a Atmel AT90USB2 USB controller, the Basys2 board provides complete, ready-to-use hardware suitable for hosting circuits ranging from basic logic devices to complex controllers. A large collection of on-board I/O devices and all required FPGA support circuits are included, so countless designs can be created without the need for any other components.

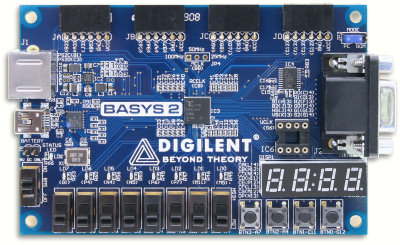


Figure 6- FPGA

## 2.2ULTRASONIC

Sound is a natural phenomenon which helps us to recognize our environment without physical contact over widely varying distances. SICK’s ultrasonic sensors use sound to accurately detect objects and measure distances. These sensors provide outstanding background suppression to reliably detect objects, regardless of the object’s appearance. The output used – switching, analog or both – is determined based on your application requirements.



Figure 7- Ultrasonic Sensors

Position and Speed Control of BLDC Motors Using Sensors PM motor drives require a rotor position sensor to properly perform phase commutation and/or current control. For PMAC motors, a constant supply of position information is necessary; thus a position sensor with high resolution, such as a shaft encoder or a resolver, is typically used. For BLDC motors, only the knowledge of six phase-commutation instants per electrical cycle is needed; therefore, low-cost Hall-effect sensors are usually used. Also, electromagnetic variable reluctance (VR) sensors or accelerometers have been extensively applied to measure motor position and speed. The reality is that angular motion sensors based on magnetic field sensing principles stand out because of their many inherent advantages and sensing benefits. 2.1. Position and Speed Sensors As explained before, some of the most frequently used devices in position and speed applications are Hall-effect sensors, variable reluctance sensors and accelerometers. Each of these types of devices is discussed further below.

# 3 – DESIGN AND BUILDING

## 3.1 PROGRAMING CODE

We programingcode in Verilog. Code include a commend to measure distance. It takes 7 value and compare it. If last value bigger than first value it means front car is speeding up and fpga command to speed up. If last value smaller than first value it means front car is slowing down and controller command to slow down.

MAİN

module main(input clk,reset,echo,start,

output triger,

output [7:0] sseg,

output [3:0] an,

output ma,mb

);

wire [15:0] distance;

reg ready, done\_tick;

wire [15:0] bcdo;

wire hizlan, yavasla, dur;

led U1(.echo(echo), .clock(clk), .reset(reset), .enable(start), .triger(triger), .distance(distance));

bin2bcd U2(.clk(clk), .reset(reset), .start(start), .bin(distance), .bcdo(bcdo));

ssdScan U3(.clk(clk), .reset(reset), .d(bcdo), .sseg(sseg), .an(an), .en(start));

mkontrol U4(.clock(clk), .reset(reset), .distance(bcdo), .hizlan(hizlan), .yavasla(yavasla), .dur(dur));

motor U5(.hizlan(hizlan), .yavasla(yavasla), .dur(dur), .clock(clk), .reset(reset), .ma(ma), .mb(mb));

endmodule

SENSOR KODLARI

`timescale 1ns / 1ps

module led( input wire echo, clock, reset, enable,

output reg triger,

output wire [12:0] distance

// output wire [7:0] led

);

localparam [2:0]

sidle = 3'b000,

strig = 3'b001,

swait0 = 3'b010,

swait1 = 3'b110,

sdelay = 3'b111,

sdone = 3'b011;

reg [15:0] err;

reg [2:0] state\_reg, state\_next;

reg [15:0] or\_reg, or\_next;

reg [15:0] wc\_reg, wc\_next;

reg [31:0] tc\_reg, tc\_next;

reg [31:0] dc\_reg, dc\_next;

always @(posedge clock, posedge reset)

if(reset)

begin

state\_reg<= sidle;

tc\_reg <= 0;

wc\_reg <= 0;

or\_reg <= 0;

dc\_reg <= 0;

end

else

begin

state\_reg<= state\_next;

tc\_reg <= tc\_next;

wc\_reg <= wc\_next;

or\_reg <= or\_next;

dc\_reg <= dc\_next;

end

// next state

always @\*

begin

state\_next = state\_reg;

tc\_next = tc\_reg;

wc\_next = wc\_reg;

or\_next = or\_reg;

dc\_next = dc\_reg;

// triger = 0;

err =0;

case (state\_reg)

sidle:

if(enable == 1)

begin

tc\_next=0;

wc\_next=0;

dc\_next=0;

state\_next = strig;

end

strig:

begin

if(tc\_reg < 600)

begin

triger = 1;

tc\_next = tc\_reg + 1;

end

else

begin

triger = 0;

state\_next = swait1;

end

end

swait1:

begin

if(echo==1)

state\_next=swait0;

end

swait0:

begin

if(echo==0)

state\_next=sdelay;

else

wc\_next=wc\_reg + 1;

end

sdelay:

begin

if(dc\_reg==3000000)

state\_next=sdone;

else

dc\_next=dc\_reg+1;

end

sdone:

begin

//err = wc\_reg - or\_reg;

//or\_next=or\_reg+{err[15],err[15],err[15],err[15],err[15:4]};

or\_next= wc\_reg;

state\_next=sidle;

end

endcase

// always @\*

// begin

// if (echo == 1)

// echo\_next = echo\_reg + 1;

// else

// begin

// sayac2\_next = echo\_reg;

// echo\_next = 0;

// end

//

end

//assign triger = trig\_reg;

assign distance = or\_reg[15:3];

//assign distance = (sayac2\_reg \*2\*2\*2\*2\*2\*2\*2\*2\*2\*2\*2\*2\*2) >> 2'b11;

//assign led[0] = (sayac2\_reg < 9'b111000011) ? 1 : 0;

//assign led[1] = (sayac2\_reg > 10'b1001001010 && sayac2\_reg < 11'b10011100111) ? 1 : 0;

//assign led[2] = (sayac2\_reg > 11'b10110101000 && sayac2\_reg < 11'b11001100100) ? 1 : 0;

//assign led[3] = (distance > 30 && distance < 40) ? 1 : 0;

//assign led[4] = (distance > 40 && distance < 50) ? 1 : 0;

//assign led[5] = (distance > 50 && distance < 60) ? 1 : 0;

//assign led[6] = (distance > 60 && distance < 70) ? 1 : 0;

//assign led[7] = (distance > 70 && distance < 80) ? 1 : 0;

Endmodule

BIN to BCD ÇEVİRME KODU

`timescale 1ns / 1ps

module bin2bcd

(

input wire clk, reset,

input wire start,

input wire [12:0] bin,

output reg ready, done\_tick,

output wire [15:0] bcdo

);

// symbolic state declaration

localparam [1:0]

idle = 2'b00,

op = 2'b01,

done = 2'b10;

// signal declaration

reg [1:0] state\_reg, state\_next;

reg [12:0] p2s\_reg, p2s\_next;

reg [3:0] n\_reg, n\_next;

reg [3:0] bcd3\_reg, bcd2\_reg, bcd1\_reg, bcd0\_reg;

reg [3:0] bcd3\_next, bcd2\_next, bcd1\_next, bcd0\_next;

wire [3:0] bcd3\_tmp, bcd2\_tmp, bcd1\_tmp, bcd0\_tmp;

reg [15:0] bcdo\_reg, bcdo\_next;

// body

// FSMD state & data registers

always @(posedge clk, posedge reset)

if (reset)

begin

state\_reg <= idle;

p2s\_reg <= 0;

n\_reg <= 0;

bcd3\_reg <= 0;

bcd2\_reg <= 0;

bcd1\_reg <= 0;

bcd0\_reg <= 0;

bcdo\_reg <= 0;

end

else

begin

state\_reg <= state\_next;

p2s\_reg <= p2s\_next;

n\_reg <= n\_next;

bcd3\_reg <= bcd3\_next;

bcd2\_reg <= bcd2\_next;

bcd1\_reg <= bcd1\_next;

bcd0\_reg <= bcd0\_next;

bcdo\_reg <= bcdo\_next;

end

// FSMD next-state logic

always @\*

begin

state\_next = state\_reg;

ready = 1'b0;

done\_tick = 1'b0;

p2s\_next = p2s\_reg;

bcd0\_next = bcd0\_reg;

bcd1\_next = bcd1\_reg;

bcd2\_next = bcd2\_reg;

bcd3\_next = bcd3\_reg;

n\_next = n\_reg;

bcdo\_next = bcdo\_reg;

case (state\_reg)

idle:

begin

ready = 1'b1;

if (start)

begin

state\_next = op;

bcd3\_next = 0;

bcd2\_next = 0;

bcd1\_next = 0;

bcd0\_next = 0;

n\_next = 4'b1101; // index

p2s\_next = bin; // shift register

state\_next = op;

end

end

op:

begin

// shift in binary bit

p2s\_next = p2s\_reg << 1;

// shift 4 BCD digits

//{bcd3\_next, bcd2\_next, bcd1\_next, bcd0\_next}=

//{bcd3\_tmp[2:0], bcd2\_tmp, bcd1\_tmp, bcd0\_tmp,

// p2s\_reg[12]}

bcd0\_next = {bcd0\_tmp[2:0], p2s\_reg[12]};

bcd1\_next = {bcd1\_tmp[2:0], bcd0\_tmp[3]};

bcd2\_next = {bcd2\_tmp[2:0], bcd1\_tmp[3]};

bcd3\_next = {bcd3\_tmp[2:0], bcd2\_tmp[3]};

n\_next = n\_reg - 1;

if (n\_next==0)

state\_next = done;

end

done:

begin

done\_tick = 1'b1;

state\_next = idle;

bcdo\_next = {bcd3\_reg, bcd2\_reg, bcd1\_reg, bcd0\_reg};

end

default: state\_next = idle;

endcase

end

// data path function units

assign bcd0\_tmp = (bcd0\_reg > 4) ? bcd0\_reg+3 : bcd0\_reg;

assign bcd1\_tmp = (bcd1\_reg > 4) ? bcd1\_reg+3 : bcd1\_reg;

assign bcd2\_tmp = (bcd2\_reg > 4) ? bcd2\_reg+3 : bcd2\_reg;

assign bcd3\_tmp = (bcd3\_reg > 4) ? bcd3\_reg+3 : bcd3\_reg;

// BCD OUTPUT

assign bcdo = bcdo\_reg ;

endmodule

SevenSegmente Yazma KODU

`timescale 1ns / 1ps

module ssdScan(

input clk,

input reset,

input [15:0] d,

output [7:0] sseg,

output [3:0] an,

input en

);

// 4 bit memory, name=ssds

reg [3:0] ssds\_r; // Q output of the D flip flop

wire [3:0] ssds\_n; // D input of the D flip flop

// 32 bit memory, name=count

reg [31:0] count\_r; // Q output of the D flip flop

wire [31:0] count\_n; // D input of the D flip flop

reg [3:0] iout;

wire [7:0] ssp;

// BODY

always @(posedge clk, posedge reset)

if (reset)

begin

count\_r <= 100\_000;

ssds\_r <= 4'b1110;

end

else

begin

count\_r <= count\_n;

ssds\_r <= ssds\_n;

end

// NEXT STATE LOGIC

assign count\_n = (count\_r == 0) ? 100\_000 : count\_r - 1;

assign ssds\_n = (count\_r == 0) ? {ssds\_r[2:0], ssds\_r[3]} : ssds\_r;

// OUTPUT LOGIC

assign an = ssds\_r;

always @\*

begin

case (an)

4'b1110: iout = d[3:0];

4'b1101: iout = d[7:4];

4'b1011: iout = d[11:8];

4'b0111: iout = d[15:12];

default: iout = 4'b0;

endcase

end

assign ssp = (iout == 4'b0000) ? 8'b11000000: // 0

(iout == 4'b0001) ? 8'b11111001: // 1

(iout == 4'b0010) ? 8'b10100100: // 2

(iout == 4'b0011) ? 8'b10110000: // 3

(iout == 4'b0100) ? 8'b10011001: // 4

(iout == 4'b0101) ? 8'b10010010: // 5

(iout == 4'b0110) ? 8'b10000010: // 6

(iout == 4'b0111) ? 8'b11111000: // 7

(iout == 4'b1000) ? 8'b10000000: // 8

(iout == 4'b1001) ? 8'b10010000: // 9

(iout == 4'b1010) ? 8'b10001000: // A

(iout == 4'b1011) ? 8'b10000011: // b

(iout == 4'b1100) ? 8'b11000110: // C

(iout == 4'b1101) ? 8'b10100001: // d

(iout == 4'b1110) ? 8'b10000110: // E

(iout == 4'b1111) ? 8'b10001110: // F

8'b11111111; // default i.e. all segments off!

assign sseg = (en) ? ssp : 8'b11111111;

endmodule

MOTOR KONTROL KODU

`timescale 1ns / 1ps

module mkontrol( input [15:0] distance,

input clock,

input reset,

output hizlan,yavasla,dur

);

//localparam[1:0]

//swait = 2'b00,

//sload = 2'b01,

//sop = 2'b10,

//sdone = 2'b11;

reg h\_reg, h\_next;

reg y\_reg, y\_next;

reg d\_reg, d\_next;

//reg [1:0] state\_reg, state\_next;

reg [15:0] dist1\_reg, dist1\_next;

reg [15:0] dist2\_reg, dist2\_next;

reg [15:0] dist3\_reg, dist3\_next;

//reg [15:0] dist4\_reg, dist4\_next;

//reg [15:0] dist5\_reg, dist5\_next;

//reg [15:0] dist6\_reg, dist6\_next;

//reg [15:0] dist7\_reg, dist7\_next;

//reg [15:0] dist8\_reg, dist8\_next;

//reg [15:0] dist9\_reg, dist9\_next;

//reg [15:0] dist10\_reg, dist10\_next;

//reg [15:0] dist11\_reg, dist11\_next;

//reg [15:0] dist12\_reg, dist12\_next;

//reg [15:0] dist13\_reg, dist13\_next;

//reg [15:0] dist14\_reg, dist14\_next;

//reg [15:0] dist15\_reg, dist15\_next;

//reg [15:0] dist16\_reg, dist16\_next;

//reg [15:0] dist17\_reg, dist17\_next;

always @(posedge clock, posedge reset)

if(reset)

begin

// state\_reg <= sload;

dist1\_reg <= 0;

dist2\_reg <= 0;

dist3\_reg <= 0;

// dist4\_reg <= 0;

// dist5\_reg <= 0;

// dist6\_reg <= 0;

// dist7\_reg <= 0;

// dist8\_reg <= 0;

// dist9\_reg <= 0;

// dist10\_reg <= 0;

// dist11\_reg <= 0;

// dist12\_reg <= 0;

// dist13\_reg <= 0;

// dist14\_reg <= 0;

// dist15\_reg <= 0;

// dist16\_reg <= 0;

// dist17\_reg <= 0;

h\_reg <= 0;

y\_reg <= 0;

d\_reg <= 0;

end

else

begin

// state\_reg<= state\_next;

dist1\_reg <= dist1\_next;

dist2\_reg <= dist2\_next;

dist3\_reg <= dist3\_next;

// dist4\_reg <= dist4\_next;

// dist5\_reg <= dist5\_next;

// dist6\_reg <= dist6\_next;

// dist7\_reg <= dist7\_next;

// dist8\_reg <= dist8\_next;

// dist9\_reg <= dist9\_next;

// dist10\_reg <= dist10\_next;

// dist11\_reg <= dist11\_next;

// dist12\_reg <= dist12\_next;

// dist13\_reg <= dist13\_next;

// dist14\_reg <= dist14\_next;

// dist15\_reg <= dist15\_next;

// dist16\_reg <= dist16\_next;

// dist17\_reg <= dist17\_next;

h\_reg <= h\_next;

y\_reg <= y\_next;

d\_reg <= d\_next;

end

always @\*

begin

// state\_next = state\_reg;

dist1\_next = distance;

dist2\_next = dist1\_reg;

dist3\_next = dist2\_reg;

// dist4\_next = dist3\_reg;

// dist5\_next = dist4\_reg;

// dist6\_next = dist5\_reg;

// dist7\_next = dist6\_reg;

// dist8\_next = dist7\_reg;

// dist9\_next = dist8\_reg;

// dist10\_next = dist9\_reg;

// dist11\_next = dist10\_reg;

// dist12\_next = dist11\_reg;

// dist13\_next = dist12\_reg;

// dist14\_next = dist13\_reg;

// dist15\_next = dist14\_reg;

// dist16\_next = dist15\_reg;

// dist17\_next = dist16\_reg;

h\_next = h\_reg;

y\_next = y\_reg;

d\_next = d\_reg;

// case (state\_reg)

//

//

// sload:

// begin

if(dist1\_reg > dist3\_reg)

begin

h\_next = 1'b1;

y\_next = 1'b0;

d\_next = 1'b0;

end

else if(dist3\_reg > dist1\_reg)

begin

h\_next = 1'b0;

y\_next = 1'b1;

d\_next = 1'b0;

end

else

begin

h\_next = 1'b0;

y\_next = 1'b0;

d\_next = 1'b1;

end

// state\_next = sload;

// end

// endcase

end

assign hizlan = h\_reg;

assign yavasla = y\_reg;

assign dur = d\_reg;

endmodule

PWM KODU

`timescale 1ns / 1ps

module motor(input hizlan,yavasla,dur,clock,reset,

output ma, mb

);

localparam [1:0]

swait = 2'b00,

sload = 2'b01,

sop = 2'b10,

sdone = 2'b11;

parameter sd=100;

reg [1:0] state\_reg, state\_next;

reg [7:0] switches\_reg, switches\_next ;

reg pwm\_reg, pwm\_next;

reg [15:0] counter = 0;

always @(posedge clock, posedge reset)

if(reset)

begin

state\_reg <= swait;

switches\_reg <= 0;

pwm\_reg <= 0;

end

else

begin

state\_reg<= state\_next;

switches\_reg <= switches\_next;

pwm\_reg <= pwm\_next;

end

always @ (posedge clock)

begin

case (state\_reg)

swait:

begin

if(hizlan == 1)

switches\_next = switches\_reg + 3;

else if(yavasla == 1)

switches\_next = switches\_reg - 3;

else

switches\_next = switches\_reg;

counter = counter + 1;

if (counter <= switches\_reg\*sd) pwm\_next = 1;

else pwm\_next = 0;

if (counter >= 50\_000\_000) counter = 0;

end

endcase

end

assign ma = pwm\_reg;

assign mb = 0;

endmodule

UCF DOSYASI

# This file is a general .ucf for Basys2 rev C board

# To use it in a project:

# - remove or comment the lines corresponding to unused pins

# - rename the used signals according to the project

# clock pin for Basys2 Board

NET "clk" LOC = "B8"; # Bank = 0, Signal name = CLK

#NET "uclk" LOC = "M6"; # Bank = 2, Signal name = UCLK

#NET "mclk" CLOCK\_DEDICATED\_ROUTE = FALSE;

#NET "uclk" CLOCK\_DEDICATED\_ROUTE = FALSE;

NET "reset" LOC = "A7"; # Bank = 1, Signal name = BTN3

#NET "btn<2>" LOC = "M4"; # Bank = 0, Signal name = BTN2

#NET "btn<1>" LOC = "C11"; # Bank = 2, Signal name = BTN1

#NET "enable" LOC = "G12"; # Bank = 0, Signal name = BTN0

# Pin assignment for LEDs

#NET "distance[15]" LOC = "G1" ; # Bank = 3, Signal name = LD7

#NET "distance[14]" LOC = "P4" ; # Bank = 2, Signal name = LD6

#NET "distance[13]" LOC = "N4" ; # Bank = 2, Signal name = LD5

#NET "distance[12]" LOC = "N5" ; # Bank = 2, Signal name = LD4

#NET "distance[11]" LOC = "P6" ; # Bank = 2, Signal name = LD3

#NET "distance[10]" LOC = "P7" ; # Bank = 3, Signal name = LD2

#NET "distance[9]" LOC = "M11" ; # Bank = 2, Signal name = LD1

#NET "distance[8]" LOC = "M5" ; # Bank = 2, Signal name = LD0

# Pin assignment for SWs

NET "start" LOC = "N3"; # Bank = 2, Signal name = SW7

#NET "A<2>" LOC = "E2"; # Bank = 3, Signal name = SW6

#NET "A<1>" LOC = "F3"; # Bank = 3, Signal name = SW5

#NET "A<0>" LOC = "G3"; # Bank = 3, Signal name = SW4

#NET "B<3>" LOC = "B4"; # Bank = 3, Signal name = SW3

#NET "B<2>" LOC = "K3"; # Bank = 3, Signal name = SW2

#NET "B<1>" LOC = "L3"; # Bank = 3, Signal name = SW1

#NET "B<0>" LOC = "P11"; # Bank = 2, Signal name = SW0

## Pin assignment for DispCtl

## Connected to Basys2 onBoard 7seg display

NET "sseg<0>" LOC = "L14"; # Bank = 1, Signal name = CA

NET "sseg<1>" LOC = "H12"; # Bank = 1, Signal name = CB

NET "sseg<2>" LOC = "N14"; # Bank = 1, Signal name = CC

NET "sseg<3>" LOC = "N11"; # Bank = 2, Signal name = CD

NET "sseg<4>" LOC = "P12"; # Bank = 2, Signal name = CE

NET "sseg<5>" LOC = "L13"; # Bank = 1, Signal name = CF

NET "sseg<6>" LOC = "M12"; # Bank = 1, Signal name = CG

NET "sseg<7>" LOC = "N13"; # Bank = 1, Signal name = DP

NET "an<3>" LOC = "K14"; # Bank = 1, Signal name = AN3

NET "an<2>" LOC = "M13"; # Bank = 1, Signal name = AN2

NET "an<1>" LOC = "J12"; # Bank = 1, Signal name = AN1

NET "an<0>" LOC = "F12"; # Bank = 1, Signal name = AN0

## Loop back/demo signals

## Pin assignment for PS2

#NET "PS2C" LOC = "B1" | DRIVE = 2 | PULLUP ; # Bank = 3, Signal name = PS2C

#NET "PS2D" LOC = "C3" | DRIVE = 2 | PULLUP ; # Bank = 3, Signal name = PS2D

## Pin assignment for VGA

#NET "HSYNC" LOC = "J14" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = HSYNC

#NET "VSYNC" LOC = "K13" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = VSYNC

#NET "OutRed<2>" LOC = "F13" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = RED2

#NET "OutRed<1>" LOC = "D13" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = RED1

#NET "OutRed<0>" LOC = "C14" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = RED0

#NET "OutGreen<2>" LOC = "G14" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = GRN2

#NET "OutGreen<1>" LOC = "G13" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = GRN1

#NET "OutGreen<0>" LOC = "F14" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = GRN0

#NET "OutBlue<2>" LOC = "J13" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = BLU2

#NET "OutBlue<1>" LOC = "H13" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = BLU1

## Loop Back only tested signals

#NET "PIO<72>" LOC = "B2" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JA1

#NET "PIO<73>" LOC = "A3" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JA2

#NET "sig" LOC = "J3" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JA3

#NET "x" LOC = "B5" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JA4

#

NET "triger" LOC = "B5" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JB1

NET "echo" LOC = "J3" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JB2

NET "ma" LOC = "B2" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JB3

NET "mb" LOC = "A3" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JB4

#

#NET "PIO<80>" LOC = "A9" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JC1

#NET "PIO<81>" LOC = "B9" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JC2

#NET "PIO<82>" LOC = "A10" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JC3

#NET "PIO<83>" LOC = "C9" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JC4

#

#NET "PIO<84>" LOC = "C12" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JD1

#NET "PIO<85>" LOC = "A13" | DRIVE = 2 | PULLUP ; # Bank = 2, Signal name = JD2

#NET "PIO<86>" LOC = "C13" | DRIVE = 2 | PULLUP ; # Bank = 1, Signal name = JD3

#NET "PIO<87>" LOC = "D12" | DRIVE = 2 | PULLUP ; # Bank = 2, Signal name = JD4

## Pin assignment for EppCtl

## Connected to Basys2 onBoard USB controller

#NET "EppAstb" LOC = "F2"; # Bank = 3

#NET "EppDstb" LOC = "F1"; # Bank = 3

#NET "EppWR" LOC = "C2"; # Bank = 3

#NET "EppWait" LOC = "D2"; # Bank = 3

#NET "EppDB<0>" LOC = "N2"; # Bank = 2

#NET "EppDB<1>" LOC = "M2"; # Bank = 2

#NET "EppDB<2>" LOC = "M1"; # Bank = 3

#NET "EppDB<3>" LOC = "L1"; # Bank = 3

#NET "EppDB<4>" LOC = "L2"; # Bank = 3

#NET "EppDB<5>" LOC = "H2"; # Bank = 3

#NET "EppDB<6>" LOC = "H1"; # Bank = 3

#NET "EppDB<7>" LOC = "H3"; # Bank = 3

### 3.2 MOUNTING SYSTEMS

At this stage we mounted the ultrasonic sensor front of the vehicle then we find a suitable place for fpga. Then we completed other connection.

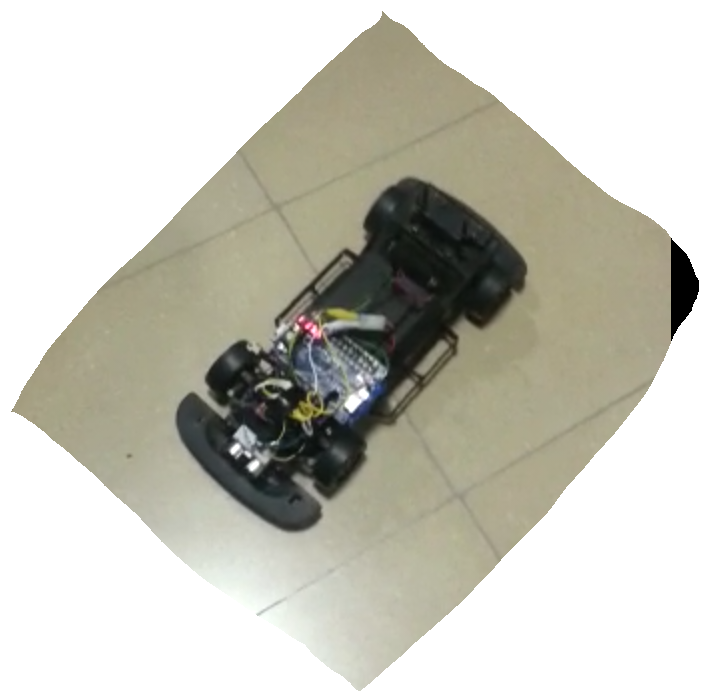


Figure 8-Our Prototype

# 

# 4- CONCLUSION

At the end of the project; We completed the system to follow front car. Test car can be driven by FPGAs according to information received from the sensor.This system open a new era in vehicle driving. Can we see cars how ranked as the train. Actually there is a continuation of this project. We complete very small part of this project. We have been able to gain a lot of expirience in the project. We will therefore like to express the fact that the project is very interesting and will be very important in our daily live as far as the use of vehicles are concerned. Trafic accidents which occur frequently in our transportatıon activities will be reduced to its minal when this project is iplemented in the automobile sector.

# 

# 5-REFERANCES

<http://www.acikbilim.com/2012/02/dosyalar/surucusuz-arabalar.html>

<http://autonomos-labs.com/vehicles/made-in-germany/>

<http://en.wikipedia.org/wiki/Main_Page>