|  |
| --- |
| MIT Logo  ELECTRICAL AND COMPUTER  ENGINEERING DEPARTMENT  **114.609 Electronic Design**  **Microcontroller-based**  **Gate Automation System**  *By : Ben Joseph Hombrebueno*  *180005719*  *Supervisor : Zulfikar Hossain* |

Abstract

The development of a microcontroller-based automatic parking gate is discussed in this report. A cost effective and efficient system to improve security in parking lots was built during the course. The PIC18F25K22 microcontroller was used to control the display message sent to the LCD and the opening or closing of the gate upon successful detection of a car by the ultrasonic sensor. The final product was able to deliver the required specifications at the end.

Contents

[Abstract ii](#_Toc531378246)

[Contents iii](#_Toc531378247)

[List of Figures iv](#_Toc531378248)

[1 Planning 1](#_Toc531378249)

[2 Introduction 2](#_Toc531378250)

[3 Hardware Description 3](#_Toc531378251)

[3.1 Block Diagram 3](#_Toc531378252)

[3.2 Parts Description 3](#_Toc531378253)

[3.2.1 Ultrasonic Sensor 3](#_Toc531378254)

[3.2.2 Microcontroller 3](#_Toc531378255)

[3.2.3 Display 4](#_Toc531378256)

[3.2.4 Audio Playback 4](#_Toc531378257)

[3.2.5 Gate Control 4](#_Toc531378258)

[3.3 Circuit Diagram 4](#_Toc531378259)

[3.4 Final Product 6](#_Toc531378260)

[4 Software Description 7](#_Toc531378261)

[4.1 System Flowchart 7](#_Toc531378262)

[4.2 Microcontroller Code Listing 8](#_Toc531378263)

[Conclusion 9](#_Toc531378264)

[Bibliography 10](#_Toc531378265)

List of Figures

[Figure 1 Gantt Chart 1](#_Toc531378549)

[Figure 2 Sensing Method [7] 2](#_Toc531378550)

[Figure 3 System Block Diagram 3](#_Toc531378551)

[Figure 4 Final Circuit Diagram 5](#_Toc531378552)

[Figure 5 Board Layout 5](#_Toc531378553)

[Figure 6 Final Product 6](#_Toc531378554)

[Figure 7 System Flowchart 8](#_Toc531378555)

# Planning

The Gantt chart for the project is shown in Figure 1. There have been some changes in the schedule due to the PCB not arriving on time that is why the some of the tasks were moved around to compensate.



Figure Gantt Chart

# Introduction

The need for more parking spaces is one of the main problems in cities. The need for people to share these limited public facilities necessitates the need for improved security for their cars. Manually opening or closing the gate is time consuming and labour intensive, making automated parking spaces more cost efficient. This project will hopefully be an option to solve that problem.

Some implementations of automatic gate control systems from before used RFID tags [1] PLC technology [2], ,using infrared sensors[3] and even using different microprocessors like ATMEL [3] and Arduino [4]. For object detection, the project will use ultrasonic sensors. Ultrasonic High frequency waves generated from the transmitter will hit the car and the echo bouncing from it will be processed by the receiver. The distance by the sensor and the car can then be computed. [5]. Ultrasonic sensors are not affected by color, smoke, dust, and light and car material [6], making it ideal for use in car parks. The timing diagram is shown in Figure 2.

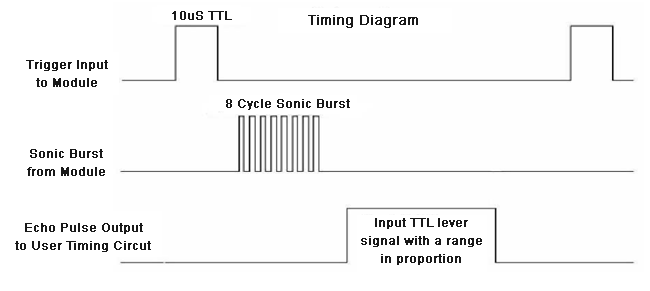


Figure Sensing Method [7]

Eight cycles of 40kHz waves will be generated by the transmitter when the module receives a trigger signal from the microcontroller of at least 10us in length. The time is recorded from the start of the transmission of the waves to the start of the reception of the echo. The distance is then computed by the following formula:

Distance = (recorded time × velocity of sound (340m/S) / 2,[8]

The gate will be controlled by a motor driver. The current project will be able to sense cars from both sides of the gate unlike the previous designs and LEDs can control the flow of traffic through the gate.

# Hardware Description

## Block Diagram

The system will have the following blocks included as shown in Figure 3. The blocks will be the Power Supply, Ultrasonic Sensors, the Microcontroller, the Liquid Crystal Display, the Audio Playback, the LED for traffic control and the Gate Control Servomotor.

Ultrasonic

Sensors

Entrance and Exit

PIC 18 Microcontroller

Liquid Crystal Display

Gate Control

Servomotor

Power

Supply

Audio

Playback

LED for

traffic control

Figure System Block Diagram

## Parts Description

### Ultrasonic Sensor

For this project, the sensor used was the HC-SR04 Ultrasonic module, and it is composed of a transmitter and receiver pair. This sensor is able to make non-contact measurements within 2cm - 400cm with an accuracy of 3mm.[8]

### Microcontroller

The PIC18F25K22 was chosen for this project for its flexibility in embedded systems applications like in cars and home appliances [5]. Some of its features include flexible 8- and 16-bit timers, 10-bitADC, MSSP, ESCI, CCP, PSP, WDT, POR and CAN 2.0B Active. The device also has RAM from 256 to 4 Kbytes and FLASH program memory from 8 to 128 Kbytes and data and. It has an operating range of 2.0 to 5.5V at speeds from DC to 40MHz. High-level languages like ANSI can be used to program the microcontroller. [9]

### Display

The LMC-SSC2A16-0,a 16character LCD with 2 lines display was used in this project. It is able to display each character in a 5X8 dots grid. It is an STN display with 1/16 Duty.[10].

### Audio Playback

For this project, the Nuvoton’s ISD1932 was used for audio playback. This device was chosen as it is a single chip capable of multiple- message record and playback. The chip has a wide voltage operation range from 2.4V to 5.5V and its sampling frequency can be set by external resistor within 4 to 12 kHz. The device can be used in standalone applications, but it can be used in conjunction with a microcontroller [11].

### Gate Control

The SG90 servo motor was used to control the opening and the closing of the gate. This model was chosen as it can be directly interfaced to the microcontroller and is ideal since it does not need a feedback and gear box. It can rotate approximately 180 degrees (90 in each direction), has a speed of 0.1s, a torque of 2.5 kg-cm and can be used with a supply around 4.8V-6V. [12]

## Circuit Diagram

The original design for this circuit included 2 sets of green and red LEDs for the entrance and exit sides of the gate to serve as traffic control signals for when the car entering or exiting can start to move or stop if they need to. Sensing the exiting cars from the other side of the gate was also part of the original design and this needed another ultrasonic sensor. The circuit diagram and PCB layout for this is shown in the Appendix B and C.

The final circuit of the project is shown in Figure 4. Here, the interconnection of the pins of the microcontroller to the ultrasonic sensor, the LCD, the servomotor, the audio playback and the power supply are shown. Also included is the connection to the PICKIT3 programmer. The board layout is shown in Figure 5.



Figure Final Circuit Diagram



Figure Board Layout

## Final Product

The final product is shown in Figure 6. The model is able to sense an object at the front of the gate and if the system successfully undergoes the checks, the gate is controlled by the servomotor and appropriate messages are displayed. The model was able to deliver the specifications, although in this case, the original plan of including a voice chip to deliver audible messages and was removed due to PCB layout issues.



Figure Final Product

# Software Description

## System Flowchart

In the design of the original system, upon start up, the ultrasonic sensor for the entrance will try to sense if there is a car at the entrance then the same goes for the ultrasonic sensor for the exit. If there is a car exiting, priority will go to that first, else the system proceeds in displaying the welcome message, controlling the LEDs for the car at the entrance to proceed while signalling any exiting cars to stop, playback of the audio and opening of the gate. The gate closes after 5 seconds to allow the car to pass, then the system goes back into sensing. The reverse is true when there is an exiting car. This program was not implemented in the final product as the board only has room for the entrance sensor and the program was modified as such. The proposed microcontroller program is shown in the Appendix D.

The final system flowchart is shown in Figure 4. At the start of the program, the status at the gate is checked by the ultrasonic sensor. If there is no car sensed, then the system waits for 1 second then goes back to sensing. If there is a car sensed, then it will wait for 5 seconds and then check again if the car is still able to be sensed. This step is taken to prevent false opening of the gate. If the car cannot be sensed after this 5 seconds waiting time, then it waits for 1 second and goes back to sensing. Otherwise, it opens the gate and displays the welcome message for 3 seconds, and prompts the thank you message and closes the gate after 5 seconds. Then the system goes back to sensing.



Figure System Flowchart

## Microcontroller Code Listing

The final microcontroller code used in the final product is shown in the Appendix A.

# Conclusion

A functioning model of the automatic gate was developed in this course. The system was able to detect a car, control the servomotor for the gate and display a message on the LCD. Changes in the original specifications were made due to some problems with the PCB layout, specifically, only the entrance of the gate was monitored and the audio capability was removed. The board layout should be carefully checked in the future so that the required specifications would be met; specifically the system should have the audio playback and the ability to sense cars that are exiting the parking area.

# Bibliography

[1] Anusha, "RFID Based Car Parking System", *Electronics Hub*, 2018. [Online]. Available: https://www.electronicshub.org/rfid-based-car-parking-system/. [Accessed: 12- Aug- 2018].

[2] F. Islam, M. Adil and S. Alvi, "PLC Based Automatic Intelligent Car Parking System", *International Journal of Computer Theory and Engineering*, 2017. [Online]. Available: http://www.ijcte.org/vol9/1111-MS15108.pdf. [Accessed: 12- Aug- 2018].

[3] O. Abu kebiru, "Design and Development of an Automatic Vehicle Gate Opener", *IOSR Journal of Engineering (IOSRJEN)*, 2017. [Online]. Available: http://www.iosrjen.org/Papers/vol7\_issue11/Version-1/C0711011217.pdf. [Accessed: 12- Aug- 2018].

[4] J. ENOKELA and M. TYOWUAH, "An electronically controlled automatic security access gate", *Leonardo Journal of Sciences*, 2014. [Online]. Available: http://ljs.academicdirect.org/A25/085\_096.pdf. [Accessed: 12- Aug- 2018].

[5] H.Al\_Issa, S. Thuneibat, A. Ijjeh, M. Abdesalam,“Sensors Application Using PIC16F877A Microcontroller”*American Journal of Remote Sensing*,Vol. 4, No. 3, 2016, pp. 13-18. [Online].Available: http://article.sciencepublishinggroup.com/pdf/10.11648.j.ajrs.20160403.11.pdf [Accessed: August 15, 2018]

[6] R. Burnett, "Understanding How Ultrasonic Sensors Work - MaxBotix Inc.", *MaxBotix Inc*., 2018. [Online]. Available: https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm. [Accessed: 15- Aug- 2018].

[7] B. Raj, "Interfacing Ultrasonic Sensor HC-SR04 with PIC Microcontroller (PIC16F877A)", *Circuitdigest.com*, 2017. [Online]. Available: https://circuitdigest.com/microcontroller-projects/interfacing-ultrasonic-sensor-hc-sr04-with-pic16f877a. [Accessed: 15- Aug- 2018].

[8] Elecfreaks, "HC-SR04 User Guide", *Elecfreaks.com*. [Online]. Available: https://elecfreaks.com/estore/download/EF03085-HC-SR04\_Ultrasonic\_Module\_User\_Guide.pdf. [Accessed: 19- Aug- 2018].

[9]Microchip Technology Inc., "PIC18 Microcontroller Family", *ww1.microchip.com*, 2002. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/30327b.pdf. [Accessed: 19- Aug- 2018].

[10] SDEC Technology Corp., "DOT MATRIX LIQUID CRYSTAL DISPLAY MODULE LMC-SSC2A16-01 Serial USER‘ MANUAL", *Images.100y.com.tw*. [Online]. Available: http://images.100y.com.tw/pdf\_file/SDEC\_LMC-SSC2A16DLYY-E01.pdf. [Accessed: 19- Aug- 2018].

[11] Nuvoton, "ISD1900 Single - chip, Multiple - Message ,Voice Record/Playback Device", *Nuvoton.com*, 2009. [Online]. Available: http://www.nuvoton.com/resource-files/EN\_ISD1900\_Datasheet\_Rev-0.51.pdf. [Accessed: 19- Aug- 2018].

[12] Tower Pro, "SERVO MOTOR SG90 Data sheet", *Ee.ic.ac.uk*. [Online]. Available: http://www.ee.ic.ac.uk/pcheung/teaching/DE1\_EE/stores/sg90\_datasheet.pdf. [Accessed: 19- Aug- 2018].

[13]O. Oshoewu and O. Olatinwo, "Design and Implementation of a Microcontroller Based Automatic Gate", *African Journal of Computing & ICT*, 2013. [Online]. Available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.301.6287&rep=rep1&type=pdf. [Accessed: 12- Aug- 2018].

[14] J. Kamid, "Microprocessor Based Automated Gate Barrier with Vehicle Counter", *Academia.edu*, 2018. [Online]. Available: http://www.academia.edu/4456781/Microprocessor\_Based\_Automated\_Gate\_Barrier\_with\_Vehicle\_Counter. [Accessed: 12- Aug- 2018].

[15] D. Ibrahim, *PIC BASIC projects*, 1st ed. Great Britain: MPG Books Ltd., 2006, pp. 248-259.

[16] T. Agarwal, "Automatic Door Opening System using PIR Sensor", *ElProCus - Electronic Projects for Engineering Students*, 2018. [Online]. Available: https://www.elprocus.com/automatic-door-opening-system-project-circuit/. [Accessed: 11- Aug- 2018].

[17]D. Ebegba and F. Anyasi, "Microprocessor Based Controller For A Motor Car Park", *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*, 2013. [Online]. Available: http://www.iosrjournals.org/iosr-jece/papers/Vol5-Issue1/G0514551.pdf. [Accessed: 12- Aug- 2018].

[18] J. Fabros, D. Tabanag, A. Espra and O. Gerasta, "Automated Car Park Management System", *IOP Conf. Series: Materials Science and Engineering*, 2015. [Online]. Available: http://iopscience.iop.org/article/10.1088/1757-899X/79/1/012009/pdf. [Accessed: 12- Aug- 2018].

#### Microcontroller Code listing

#include <xc.h>

#include "LCD.h"

#pragma config WDTEN=OFF, FOSC=INTIO67,MCLRE=EXTMCLR,IESO=OFF, LVP=OFF

#define \_XTAL\_FREQ 4000000

#define CENTRE 0

#define LEFT 1

#define RIGHT 2

unsigned int i=0;

unsigned int state=0, count=0,sec=0,prevstate=1;

bit flag=0;

unsigned int Sensor\_Time\_Entry=0,Sensor\_Time\_Exit=0,Sensor\_Distance2\_Entry=0,Sensor\_Distance2\_Exit=0, Sensor\_Distance4=0,state;

float Sensor\_Distance\_Entry=0,Sensor\_Distance\_Exit=0, Sensor\_Distance3=0;

void delay\_sec(unsigned char s)

{

for(i = 0; i < 10 \* s; i++)

{

\_\_delay\_ms(100);

}

}

void interrupt multi(void)

{

if(TMR1IF==1)

{

count=count+1;

if (count==15)

{

sec=sec+1;

count=0;

}

if ((sec==1)&& (state==0))

{

state=3;

sec=0;

count=0;

TMR1ON=0;

}

if ((sec==3)&& (state==1))

{

state=2;

sec=0;

count=0;

TMR1ON=0;

}

if ((sec==5)&& (state==2))

{

state=0;

sec=0;

count=0;

TMR1ON=0;

}

TMR1IF=0;

}

}

void close\_door(void)

{

for(i=0;i<=5;i++)

{

LATC0=1;

\_\_delay\_ms(1);

LATC0=0;

\_\_delay\_ms(19);

}

}

void open\_door(void)

{

for(i=0;i<=5;i++)

{

LATC0=1;

\_\_delay\_ms(2);

LATC0=0;

\_\_delay\_ms(18);

}

}

void init\_timer1(void)

{

TMR1CS1 = 0;

TMR1CS0 = 0;

T1CKPS0 = 0;

T1CKPS1 = 0;

T1SOSCEN = 0;

T1SYNC = 0;

T1RD16 = 0;

TMR1IF = 0;

TMR1ON = 0;

TMR1IE = 1;

PEIE = 1;

GIE=1;

}

void main(void)

{

//setting frequencey and I/O Pins

OSCCON=0b01010010;//this to select internal osc at 4MHz

//ANSELX to set the type of signal - i.e. digital (0) or analog (1)

//TRISX - to set the pin direction - i.e. input (1) or output (0)

//PORTX - to set value at the pin - i.e. high (1) or low (0).

//X will be replaced by the specific pin name- i.e. A or B or C.

//For example - setting for port A would be as below

TRISA=0b00000010; //make pins directions - i.e. input or output

ANSELA=0x00; //make PORTA digital or analogue

PORTA=0x00; //clear all PORTA digital pins

TRISB=0b00000000; //make pins directions - i.e. input or output

ANSELB=0x00; //make PORTB digital or analogue

PORTB=0x00; //clear all PORTB digital pins

TRISC=0b00000000; //make pins directions - i.e. input or output

ANSELC=0x00; //make PORTB digital or analogue

PORTC=0x00; //clear all PORTB digital pins

//timer interrupt settings

T0PS0=1; // set PS to divide by 4

T0PS1=0; // set PS to divide by 4

T0PS2=0; // set PS to divide by 4

PSA=0; //enable PS

T0CS=0;//use as timer

T08BIT=0; //make TMR0 an 8 bit timer

TMR0ON=0;//turn on TMR0

// T0IE=1;//enable the interrupt

// T0IF=0;//clear the flag

//PORTXbits.RX0-7 - to check the value at the pic. e.g. PORTAbits.RA2 is to check

//the value at the PORTA2 pin. It's useful to check the button status.

init\_timer1();

init\_display();

while(1)

{//starting contineous Loop.

//resetting timer

TMR0H=0x00;

TMR0L=0x00;

TMR1H=0x00;

TMR1L=0x00;

switch(state)

{

case 0:

TMR1H=0x00;

TMR1L=0x00;

TMR1ON=1;

close\_door();

set\_address\_line1(1);

send\_msg(" ");

set\_address\_line2(1);

send\_msg(" ");

break;

case 1:

TMR1H=0x00;

TMR1L=0x00;

TMR1ON=1;

set\_address\_line1(1);

send\_msg("Welcome!");

set\_address\_line2(1);

send\_msg("Carpark");

open\_door();

break;

case 2:

TMR1H=0x00;

TMR1L=0x00;

TMR1ON=1;

set\_address\_line1(1);

send\_msg("ThankYou ");

set\_address\_line2(1);

send\_msg("Goodbye ");

break;

case 3:

TMR1H=0x00;

TMR1L=0x00;

TMR0H=0x00;

TMR0L=0x00;

//trigger pulse

TRISAbits.RA0=0; //RA0 pin as output

LATA0=1;

\_\_delay\_us(14);

LATA0=0;

//getting an echo and calculating distance.

TRISAbits.RA1=1; //RB0 pin as input

while(PORTAbits.RA1==0);

TMR0ON=1;

while(PORTAbits.RA1==1);

TMR0ON=0;

Sensor\_Time\_Entry=TMR0;

Sensor\_Distance\_Entry=Sensor\_Time\_Entry/15;

if(Sensor\_Distance\_Entry<11)

{

delay\_sec(5);

state=4;

}

else

{

state=0;

}

break;

case 4:

TMR0H=0x00;

TMR0L=0x00;

//trigger pulse

TRISAbits.RA0=0; //RA0 pin as output

LATA0=1;

\_\_delay\_us(14);

LATA0=0;

//getting an echo and calculating distance.

TRISAbits.RA1=1; //RB0 pin as input

while(PORTAbits.RA1==0);

TMR0ON=1;

while(PORTAbits.RA1==1);

TMR0ON=0;

Sensor\_Time\_Entry=TMR0;

Sensor\_Distance\_Entry=Sensor\_Time\_Entry/15;

if(Sensor\_Distance\_Entry<11)

{

state=1;

}

else

{

state=0;

}

break;

}

}//ending contineous Loop.

} //end of main function

#### Proposed Circuit Diagram



#### Proposed Board Layout



#### Proposed Microcontroller Code Listing

#include <xc.h>

#include "LCD.h"

#pragma config WDTEN=OFF, FOSC=INTIO67,MCLRE=EXTMCLR,IESO=OFF, LVP=OFF

#define \_XTAL\_FREQ 4000000

#define CENTRE 0

#define LEFT 1

#define RIGHT 2

unsigned int i=0;

unsigned int state=4, count=0,sec=0,prevstate=1;

bit flag=0;

unsigned int Sensor\_Time\_Entry=0,Sensor\_Time\_Exit=0,Sensor\_Distance2\_Entry=0,Sensor\_Distance2\_Exit=0, Sensor\_Distance4=0,state;

float Sensor\_Distance\_Entry=0,Sensor\_Distance\_Exit=0, Sensor\_Distance3=0;

void delay\_sec(unsigned char s)

{

for(i = 0; i < 10 \* s; i++)

{

\_\_delay\_ms(100);

}

}

void centre(void)

{

for(i=0;i<=5;i++)

{

LATB3=1;

\_\_delay\_ms(1);

\_\_delay\_us(500);

LATB3=0;

\_\_delay\_ms(18);

\_\_delay\_us(500);

}

}

void close\_door(void)

{

for(i=0;i<=5;i++)

{

LATB3=1;

\_\_delay\_ms(1);

LATB3=0;

\_\_delay\_ms(19);

}

}

void open\_door(void)

{

for(i=0;i<=5;i++)

{

LATB3=1;

\_\_delay\_ms(2);

LATB3=0;

\_\_delay\_ms(18);

}

}

void main(void)

{

//setting frequencey and I/O Pins

OSCCON=0b01010010;//this to select internal osc at 4MHz

//ANSELX to set the type of signal - i.e. digital (0) or analog (1)

//TRISX - to set the pin direction - i.e. input (1) or output (0)

//PORTX - to set value at the pin - i.e. high (1) or low (0).

//X will be replaced by the specific pin name- i.e. A or B or C.

//For example - setting for port A would be as below

TRISA=0b00001010; //make pins directions - i.e. input or output

ANSELA=0x00; //make PORTA digital or analogue

PORTA=0x00; //clear all PORTA digital pins

TRISB=0b00000000; //make pins directions - i.e. input or output

ANSELB=0x00; //make PORTB digital or analogue

PORTB=0x00; //clear all PORTB digital pins

TRISC=0b00000000; //make pins directions - i.e. input or output

ANSELC=0x00; //make PORTB digital or analogue

PORTC=0b01000011; //clear all PORTB digital pins

//timer interrupt settings

T0PS0=1; // set PS to divide by 4

T0PS1=0; // set PS to divide by 4

T0PS2=0; // set PS to divide by 4

PSA=0; //enable PS

T0CS=0;//use as timer

T08BIT=0; //make TMR0 an 8 bit timer

TMR0ON=0;//turn on TMR0

// T0IE=1;//enable the interrupt

// T0IF=0;//clear the flag

//PORTXbits.RX0-7 - to check the value at the pic. e.g. PORTAbits.RA2 is to check

//the value at the PORTA2 pin. It's useful to check the button status.

init\_display();

while(1)

{//starting contineous Loop.

//resetting timer

TMR0H=0x00;

TMR0L=0x00;

switch(state)

{

case 0:

set\_address\_line1(1);

send\_msg(" ");

set\_address\_line2(1);

send\_msg(" ");

LATA4=0;

LATA5=0;

LATA6=0;

LATA7=0;

close\_door();

delay\_sec(2);

//state=3;

state=4;

break;

case 1:

set\_address\_line1(1);

send\_msg("Welcome ");

set\_address\_line2(1);

send\_msg("Carpark ");

LATA5=1;

LATA6=1;

LATC0=1;

LATC1=0;

delay\_sec(5);

LATC1=1;

open\_door();

delay\_sec(5);

state=0;

break;

case 2:

set\_address\_line1(1);

send\_msg("ThankYou ");

set\_address\_line2(1);

send\_msg("Goodbye ");

LATA4=1;

LATA7=1;

LATC0=1;

LATC2=0;

delay\_sec(5);

LATC2=1;

open\_door();

delay\_sec(5);

state=0;

break;

case 3:

TMR0H=0x00;

TMR0L=0x00;

//trigger pulse

TRISAbits.RA0=0; //RA0 pin as output

LATA0=1;

\_\_delay\_us(14);

LATA0=0;

//getting an echo and calculating distance.

TRISAbits.RA1=1; //RB0 pin as input

while(PORTAbits.RA1==0);

TMR0ON=1;

//TMR1ON=1;

while(PORTAbits.RA1==1);

TMR0ON=0;

//TMR1ON=0;

Sensor\_Time\_Entry=TMR0;

//Sensor\_Time=TMR1;

Sensor\_Distance\_Entry=Sensor\_Time\_Entry/15;

//Sensor\_Distance=Sensor\_Time/50;

Sensor\_Distance2\_Entry=Sensor\_Distance\_Entry;

if(Sensor\_Distance\_Entry<11)

{

state=1;

}

else

{

state=4;

}

break;

case 4:

TMR0H=0x00;

TMR0L=0x00;

TMR1H=0x00;

TMR1L=0x00;

//trigger pulse

TRISAbits.RA2=0; //RA0 pin as output

LATA2=1;

\_\_delay\_us(14);

LATA2=0;

//getting an echo and calculating distance.

TRISAbits.RA3=1; //RB0 pin as input

while(PORTAbits.RA3==0);

TMR0ON=1;

while(PORTAbits.RA3==1);

TMR0ON=0;

Sensor\_Time\_Exit=TMR0;

Sensor\_Distance\_Exit=Sensor\_Time\_Exit/15;

Sensor\_Distance2\_Exit=Sensor\_Distance\_Exit;

if(Sensor\_Distance\_Exit<11)

{

state=2;

}

else

{

state=3;

}

break;

}

}//ending contineous Loop.

} //end of main function