Smart and Ultra-Sound Car Parking System

Mohd. Rasadin

Department of Computer Science and Engineering
East West University, Dhaka, Bangladesh
Email: rasadin.ewu@gmail.com

***Abstract*— In this project, we will build a smart parking system using Arduino and Ultrasonic Distance Sensor. Our project will help the car drivers in parking time. The basic premise is that the closer a reversing car gets to an object, the greater the frequency of beeps from a buzzer. Eventually if the car gets too close to an object, the buzzer will remain on continuously. Not only buzzer but also LED will give warning and LCD will give the distance value.**

Keywords— Smart, Ultra, Sound, Parking, Warning.

I. INTRODUCTION

Internet of Things (IOT) plays a vital role in connecting the surrounding environmental things to the network and made easy to access those un-internet things from any remote location. IoT is inevitable for the people to update with the growing technology. And generally people are facing problems on parking vehicles in parking slots in a city. In this project we design a Smart Parking System (SPS) which enables the user to park car in a smart way. Our project will reduce the accident during parking time. The car will be free from hamper during parking. Our project will make a smart car parking place

Our system is derived from the idea of IoT. The system uses the Wireless Sensor Network (WSN) consisting of Radio Frequency Identification technology to monitor car parks.

II. OBJECTIVES

- To build a smart parking system using Arduino and Ultrasonic Distance Sensor.
- To help the car drivers in parking time.
- To know the use of HC-SR04.
- To find out a smart way of parking.
- To give extra benefit to the car drivers.

III. EQUIPMENT

I. Ultrasonic Sensor HC-SR04.

This is the HC-SR04 ultrasonic ranging sensor. This economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. Each HC-SR04 module includes an ultrasonic transmitter, a receiver and a control circuit. It emits an ultrasound at 40 000 Hz which travels through the air and if there is

an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance. The HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo.

II. Arduino Uno.

III. Arduino Software.

IV. 16X2 Character LCD.

V. Buzzer.

A buzzer or beeper is an audio signalling device, which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.

VI. 220 ohm resistor.

VII. Colored LED's.

VIII. Wires.

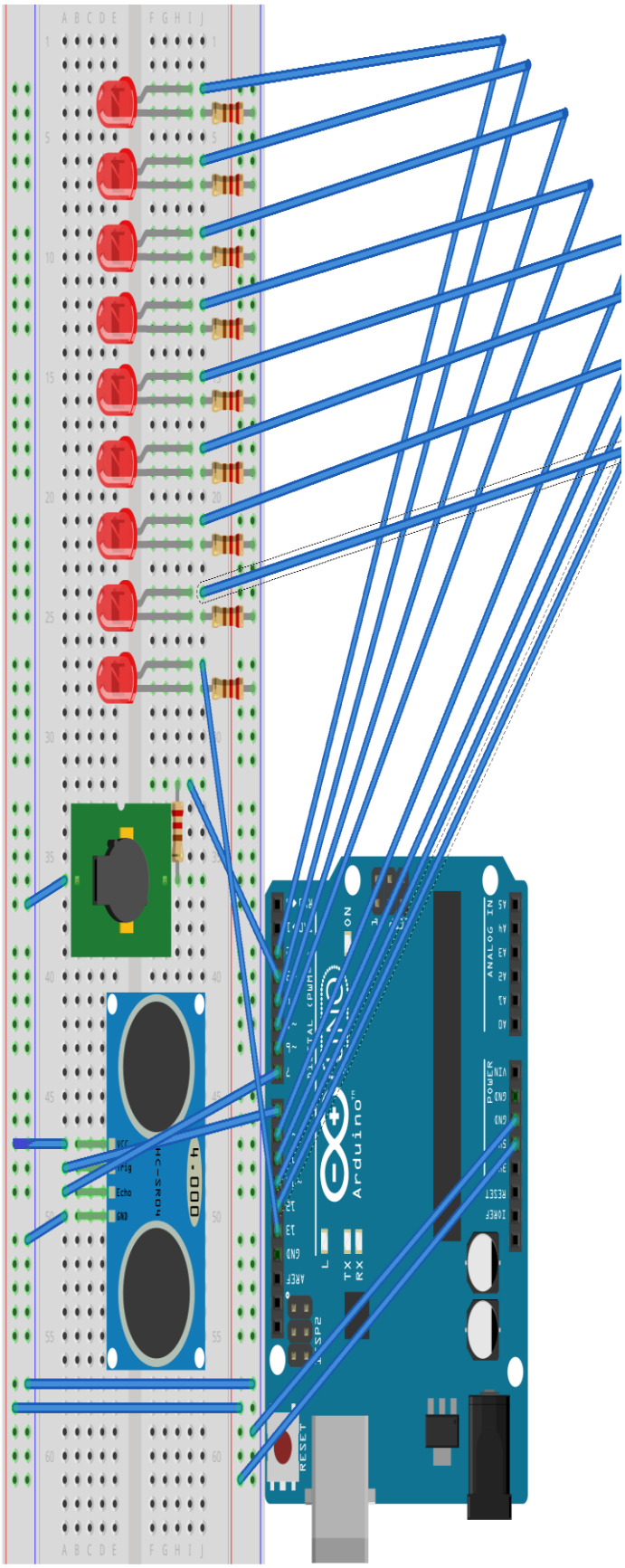
IX. Breadboard.

X. 10 K POT.

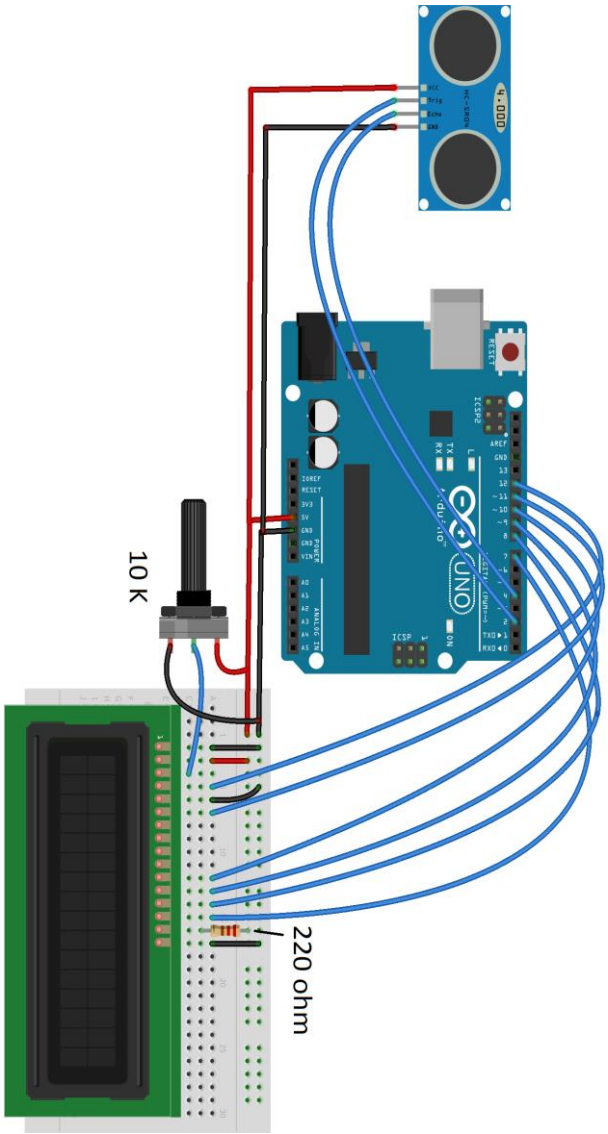
A potentiometer is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat.

IV. CIRCUIT DIAGRAM

We have two circuit diagrams. Circuit diagram(1) repesent that how we have create notification and warnig using different type of leds and buzzer. Circuit diagram(2) represent that how we have measure the distance by using 16X2 lcd display.



Circuit diagram (1)



Circuit diagram (2)

V. CONNECTION

efore wiring the LCD screen to our Arduino or uino board we add a pin header strip to the 14 16) pin count connector of the LCD screen. To our LCD screen to our board, connect the wing pins: LCD VSS pin to Arduino GND. VDD pin to Arduino 5V. LCD VO pin to 10k ntiometer center pin. LCD RS pin to digital pin CD RW pin to Arduino GND. LCD Enable pin igital pin 2. LCD D4 pin to digital pin 4. LCD pin to digital pin 5. LCD D6 pin to digital pin 6. LCD D7 pin to digital pin 7.

The 10k Potentiometer's other legs connect to +5V and GND. For the backlight of the display, pin 15 (A+) and 16 (K-) of the LCD connect to +5V and GND. We have used a 220 ohm resistor to power the backlight of the display. - The HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo. The Ground and the VCC pins of the module needs to be connected to the Ground and the 5 volts pins on the Arduino Board respectively and the trig and echo pins to any Digital I/O pin on the Arduino Board.

- The HC-SR04 sensor attach to the Breadboard.
- The Sensor VCC connect to the Arduino Board +5V.
- The Sensor GND connect to the Arduino Board GND.
- The Sensor Trig connect to the Arduino Board Digital I/O 9.
- The Sensor Echo connect to the Arduino Board Digital I/O 10.

In this way we have complete our two circuit.

VI. CODE

The LiquidCrystal library allows you to control LCD displays that are compatible.

- First you have to define the Trig and Echo pins. In this case they are the pins number 9 and 10 on the Arduino Board and they are named trigPin and echoPin. Then you need a Long variable, named "duration" for the travel time that you will get from the sensor and an integer variable for the distance.
- In the setup you have to define the trigPin as an output and the echoPin as an Input and also start the serial communication for showing the results on the serial monitor.
- If the object is 10 cm away from the sensor, and the speed of the sound is 340 m/s or 0.034 cm/µs the sound wave will need to travel about 294 u seconds. But what you will get from the Echo pin will be double that number because the sound wave needs to travel forward and bounce backward. So in order to get the distance in cm we need to multiply the received travel time value from the echo pin by 0.034 and divide it by 2.

CODE PART 1:

```
#define echoPin 7 // Echo Pin
#define trigPin 8 // Trigger Pin
//Mohd. Rasadin



const int freq = 500;
const int dur = 20;
const int buzzer = 3;
const int red3 = 2;
const int red2 = 4;
const int red1 = 5;
const int yellow3 = 6;
const int yellow2 = 9;
const int yellow1 = 10;
const int green3 = 11;
const int green2 = 12;
const int green1 = 13;



int maximumRange = 200; // Maximum range needed
int minimumRange = 0; // Minimum range needed
long duration, distance; // Duration used to calculate distance



void setup() {
 pinMode(buzzer, OUTPUT);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 pinMode(green1, OUTPUT);
 pinMode(green2, OUTPUT);
 pinMode(green3, OUTPUT);
 pinMode(yellow1, OUTPUT);
 pinMode(yellow2, OUTPUT);
 pinMode(yellow3, OUTPUT);
 pinMode(red1, OUTPUT);
 pinMode(red2, OUTPUT);
 pinMode(red3, OUTPUT);



 digitalWrite(green1, HIGH);
 digitalWrite(green2, HIGH);
 digitalWrite(green3, HIGH);
 digitalWrite(yellow1, HIGH);
 digitalWrite(yellow2, HIGH);
 digitalWrite(yellow3, HIGH);
 digitalWrite(red1, HIGH);
 digitalWrite(red2, HIGH);
```

```
 digitalWrite(red3, HIGH);

 delay(300);
 digitalWrite(green1, LOW);
 digitalWrite(green2, LOW);
 digitalWrite(green3, LOW);
 digitalWrite(yellow1, LOW);
 digitalWrite(yellow2, LOW);
 digitalWrite(yellow3, LOW);
 digitalWrite(red1, LOW);
 digitalWrite(red2, LOW);
 digitalWrite(red3, LOW);

 tone(buzzer, freq, duration);

}

void loop() {
/* The following trigPin/echoPin cycle is used to determine the
 distance of the nearest object by bouncing soundwaves off of it. */
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);


 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);

 digitalWrite(trigPin, LOW);
 duration = pulseIn(echoPin, HIGH);

 //Calculate the distance (in cm) based on the speed of sound.
 distance = duration/58.2;


 if (distance < 7){
   digitalWrite(green1, HIGH);
   digitalWrite(green2, HIGH);
   digitalWrite(green3, HIGH);
   digitalWrite(yellow1, HIGH);
   digitalWrite(yellow2, HIGH);
   digitalWrite(yellow3, HIGH);
   digitalWrite(red1, HIGH);
   digitalWrite(red2, HIGH);
   digitalWrite(red3, HIGH);



 } else if (distance < 15){
   digitalWrite(green1, HIGH);
   digitalWrite(green2, HIGH);
   digitalWrite(green3, HIGH);
   digitalWrite(yellow1, HIGH);
   digitalWrite(yellow2, HIGH);
   digitalWrite(yellow3, HIGH);
   digitalWrite(red1, HIGH);
   digitalWrite(red2, HIGH);
   digitalWrite(red3, LOW);


 } else if (distance < 20){
   digitalWrite(green1, HIGH);
   digitalWrite(green2, HIGH);
   digitalWrite(green3, HIGH);
   digitalWrite(yellow1, HIGH);
   digitalWrite(yellow2, HIGH);
   digitalWrite(yellow3, HIGH);
   digitalWrite(red1, HIGH);
   digitalWrite(red2, LOW);
   digitalWrite(red3, LOW);


 } else if (distance < 25){
   digitalWrite(green1, HIGH);
   digitalWrite(green2, HIGH);
   digitalWrite(green3, HIGH);
   digitalWrite(yellow1, HIGH);
   digitalWrite(yellow2, HIGH);
   digitalWrite(yellow3, HIGH);
   digitalWrite(red1, LOW);
   digitalWrite(red2, LOW);
   digitalWrite(red3, LOW);


 } else if (distance < 30){
   digitalWrite(green1, HIGH);
   digitalWrite(green2, HIGH);
   digitalWrite(green3, HIGH);
   digitalWrite(yellow1, HIGH);
   digitalWrite(yellow2, HIGH);
   digitalWrite(yellow3, LOW);
   digitalWrite(red1, LOW);
   digitalWrite(red2, LOW);
   digitalWrite(red3, LOW);


 } else if (distance < 35){
   digitalWrite(green1, HIGH);
   digitalWrite(green2, HIGH);
   digitalWrite(green3, HIGH);
   digitalWrite(yellow1, HIGH);
   digitalWrite(yellow2, LOW);
```

```
  digitalWrite(yellow3, LOW);
  digitalWrite(red1, LOW);
  digitalWrite(red2, LOW);
  digitalWrite(red3, LOW);


} else if (distance < 40){
  digitalWrite(green1, HIGH);
  digitalWrite(green2, HIGH);
  digitalWrite(green3, HIGH);
  digitalWrite(yellow1, LOW);
  digitalWrite(yellow2, LOW);
  digitalWrite(yellow3, LOW);
  digitalWrite(red1, LOW);
  digitalWrite(red2, LOW);
  digitalWrite(red3, LOW);


} else if (distance < 45){
  digitalWrite(green1, HIGH);
  digitalWrite(green2, HIGH);
  digitalWrite(green3, LOW);
  digitalWrite(yellow1, LOW);
  digitalWrite(yellow2, LOW);
  digitalWrite(yellow3, LOW);
  digitalWrite(red1, LOW);
  digitalWrite(red2, LOW);
  digitalWrite(red3, LOW);


} else if (distance < 50){
  digitalWrite(green1, HIGH);
  digitalWrite(green2, LOW);
  digitalWrite(green3, LOW);
  digitalWrite(yellow1, LOW);
  digitalWrite(yellow2, LOW);
  digitalWrite(yellow3, LOW);
  digitalWrite(red1, LOW);
  digitalWrite(red2, LOW);
  digitalWrite(red3, LOW);


} else if (distance < 55){
  digitalWrite(green1, LOW);
  digitalWrite(green2, LOW);
  digitalWrite(green3, LOW);
  digitalWrite(yellow1, LOW);
  digitalWrite(yellow2, LOW);
  digitalWrite(yellow3, LOW);
  digitalWrite(red1, LOW);
  digitalWrite(red2, LOW);
  digitalWrite(red3, LOW);
```

```
 } else if (distance > 55){
   digitalWrite(green1, LOW);
   digitalWrite(green2, LOW);
   digitalWrite(green3, LOW);
   digitalWrite(yellow1, LOW);
   digitalWrite(yellow2, LOW);
   digitalWrite(yellow3, LOW);
   digitalWrite(red1, LOW);
   digitalWrite(red2, LOW);
   digitalWrite(red3, LOW);
 }

 if(distance < 7){
   tone(buzzer, freq, dur);
   delay(5);
 } else if (distance < 15){
   tone(buzzer, freq, dur);
   delay(10);
 } else if (distance < 20){
   tone(buzzer, freq, dur);
   delay(15);
 } else if (distance < 25){
   tone(buzzer, freq, dur);
   delay(20);
 } else if (distance < 30){
   tone(buzzer, freq, dur);
   delay(25);
 } else if (distance < 35){
   tone(buzzer, freq, dur);
   delay(30);
 } else if (distance < 40){
   tone(buzzer, freq, dur);
   delay(35);
 } else if (distance < 45){
   tone(buzzer, freq, dur);
   delay(40);
 } else if (distance < 50){
   tone(buzzer, freq, dur);
   delay(45);
 } else if (distance < 55){
   tone(buzzer, freq, dur);
   delay(50);
 }



 //Delay 50ms before next reading.
 delay(50);
}
```

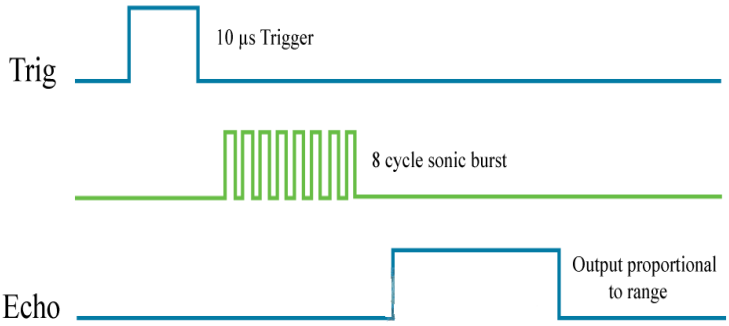
CODE PART 2:

```
#include 'LiquidCrystal.h' //Please replace the single quote characters (") with the parenthesis character (<>)

LiquidCrystal lcd(1, 2, 4, 5, 6, 7); // Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7)

const int trigPin = 9;
const int echoPin = 10;
long duration;
int distanceCm, distanceInch;

void setup() {

lcd.begin(16,2); // Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);

}

void loop() {

digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);
distanceCm= duration*0.034/2;
distanceInch = duration*0.0133/2;

lcd.setCursor(0,0); // Sets the location at which subsequent text written to the LCD will be displayed
lcd.print("Distance: "); // Prints string "Distance" on the LCD
lcd.print(distanceCm); // Prints the distance value from the sensor
lcd.print(" cm");
delay(10);
lcd.setCursor(0,1);
lcd.print("Distance: ");
lcd.print(distanceInch);
lcd.print("inch");
delay(10);

}
```

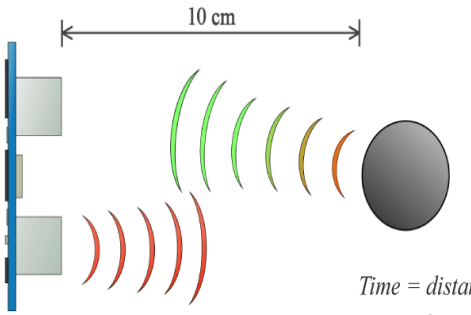
VII. HOW IT WORKS

It emits an ultrasound at 40 000 Hz which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.

The HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo. The Ground and the VCC pins of the module needs to be connected to the Ground and the 5 volts pins on the Arduino Board respectively and the trig and echo pins to any Digital I/O pin on the Arduino Board.



For example, if the object is 10 cm away from the sensor, and the speed of the sound is 340 m/s or 0.034 cm/µs the sound wave will need to travel about 294 u seconds. But what you will get from the Echo pin will be double that number because the sound wave needs to travel forward and bounce backward. So in order to get the distance in cm we need to multiply the received travel time value from the echo pin by 0.034 and divide it by 2.



speed of sound:

$v = 340\ m/s$

$v = 0{,}034\ cm/\mu s$

Time = distance / speed:

$t = s / v = 10 / 0{,}034 = 294\ \mu s$

Distance:

$s = t \cdot 0{,}034 / 2$

VIII. RESULT AND ANALYSIS

When any car enter parking area our project will start to work. At first HC-SR04 sensor take the value from the object or car. Firstly buzzer will give audio sound. This sound indicate that there are car which is entering in tne parking area. When the car start to move close the wall or another car our led wil be on. The led will maintain the serial. The serial is green, white, blue, yellow and red. Every color has its own warning. With led our buzzer also continuously giving audio sound. When our target car going very much close of wall or another car all the led and sound will be on. Not only sound and led warning driver can alos see the distance value from the wall or other car. Our 16X2 lcd give the distance values continuously.

IX. CONCLUSION

In this IOT project, we have built a smart parking system using Arduino and Ultrasonic Distance Sensor. Our project will help the car drivers in parking time. The basic premise is that the closer a reversing car gets to an object, the greater the frequency of beeps from a buzzer. Eventually if the car gets too close to an object, the buzzer will remain on continuously. Not only buzzer but also LED will give warning and LCD will give the distance value.Our project give benefit to the car drivers. Our project will be able to create a smart car parking place. This poject will be able to reduce any type of accident.

References

[1] https://howtomechatronics.com
[2] https://www.wikipedia.org/
[3] https://www.arduino.cc/
[4] http://mertarduinotutorial.blogspot.com/