**Heritage Institute of Technology**

***(An Autonomous Institute)***

**Department** **of**

**Electronics and Communication Engineering**



***This is to certify that the project report***

---------------SMART PARKING USING ARDUINO---------------

***Has been successfully completed by***

**Name: ANKESH KUMAR**

**ARINJIT SARKAR**

**GAURAV KUMAR**

**MD AMIR ARIF**

**Roll No: Respective university roll no.**

**Registration No: Respective registration no.**

***In partial fulfilment for the award of the degree in***

***Bachelor of Technology***

**In**

***Electronics and Communication Engineering***

### [*Maulana Abul Kalam Azad University of Technology*](https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=13&cad=rja&uact=8&ved=0ahUKEwiSuPnr8IrLAhUVkY4KHa5nDgUQFgg5MAw&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FMaulana_Abul_Kalam_Azad_University_of_Technology&usg=AFQjCNEkOWkuufv7TcM66ZvXxWdAunaASw&sig2=o5erv91mScoD8FHMGk2RRw)

### *(Formerly WBUT), 2018*

***Under the Supervision of***

(Name of the Project Guide) -SHIB SANKAR BHOWMICK

(Designation of Guide)-Assistant Professor

Dept. of Electronic and Communication Engineering

Heritage Institute of Technology, Kolkata-700107.



*Certificate of Recommendation*

This is to certify that the Thesis entitled ***“SMART PARKING USING ARDUINO”*** submitted by **ANKESH KUMAR, GAURAV KUMAR, ARINJIT SARKAR, MD AMIR ARIF** under the supervision of ***Prof. SHIB SANKAR BHOWMICK*** (Assistant Professor, Dept. of ECE, HITK), has been prepared according to the regulations of **B.Tech. Degree** in **Electronics and Communication Engineering Department**, awarded by [**Maulana Abul Kalam Azad University of Technology**](https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=13&cad=rja&uact=8&ved=0ahUKEwiSuPnr8IrLAhUVkY4KHa5nDgUQFgg5MAw&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FMaulana_Abul_Kalam_Azad_University_of_Technology&usg=AFQjCNEkOWkuufv7TcM66ZvXxWdAunaASw&sig2=o5erv91mScoD8FHMGk2RRw) **(Formerly WBUT)** and he/she has fulfilled the requirements for submission of thesis report and that neither his/her thesis report has been submitted for any degree/diploma or any other academic award anywhere before.

## *…………………………………………………….………*

**-----Prof.----Name of Project Guide----**

(Designation, Dept of ECE, HITK)

***Project Supervisor***

## *……………………………… ……………………….*

## Prof. Prabir Banerjee External Examiner

(HOD, Dept of ECE, HITK)

**Heritage Institute of Technology**

***(An Autonomous Institute)***

***Affiliated to***

### [Maulana Abul Kalam Azad University of Technology](https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=13&cad=rja&uact=8&ved=0ahUKEwiSuPnr8IrLAhUVkY4KHa5nDgUQFgg5MAw&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FMaulana_Abul_Kalam_Azad_University_of_Technology&usg=AFQjCNEkOWkuufv7TcM66ZvXxWdAunaASw&sig2=o5erv91mScoD8FHMGk2RRw)

(Formerly WBUT)



***Certificate of Approval\****

The foregoing thesis report is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned don’t necessarily endorse or approve any statement made opinion expressed or conclusion drawn therein but approve the project report only for the purpose for which it is submitted.

*Signature of the Examiners:*

*1…………………………………………….*

*2…………………………………………….*

*3…………………………………………….*

## *\*Only in the case the thesis report is approved*

**ACKNOWLEDGEMENTS**

*We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend my sincere thanks to all of them.*

*We are highly indebted to HOD of ECE department Dr. Prabir Banerjee and Prof. Shib Sankar Bhowmick for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in ongoing the project.*

*We would like to express our gratitude towards our parents for their kind co-operation and encouragement which is helping us in the ongoing project.*

*Our thanks and appreciations also go to our friends in developing the project and people who have willingly helped us out with their abilities.*

**Abstract**

Not finding a parking space for you sometimes is indeed a critical issue. The number of vehicles is also increasing daily adding to the parking vows at public places. Cities noticed that their drivers had real problems to find a parking space easily especially during peak hours, the difficulty roots from not knowing where the parking spaces are available at the given time. Even if this is known, many vehicles may pursue a small number of parking spaces which in turn leads to traffic congestion. The traffic on roads and parking space has been an area of concern in majority of cities. So, parking monitoring is an important solution. To avoid these problems, recently many new technologies have been developed that help in solving the parking problems to a great extent.

In recent times the concept of smart cities has gained great popularity. Thanks to the evolution of Internet of things the idea of smart city now seems to be achievable. Consistent efforts are being made in the field of IoT in order to maximize the productivity and reliability of urban infrastructure. Problems such as, traffic congestion, limited car parking facilities and road safety are being addressed by IoT. In this paper, we present an Android based smart parking system. The proposed Smart Parking system consists of an on-site deployment of an IoT module that is used to monitor and signalize the state of availability of each single parking space. A screenshot is also provided that allows an end user to check the availability of parking space and book a parking slot accordingly. The paper also describes a high-level view of the system architecture. Towards the end, the paper discusses the working of the system in form of a use case that proves the correctness of the proposed model.

**CONTENT**

1. **Introduction……………………………………………………..** 7
2. **Objective and Motivation………………………………………** 8-9
3. Proposed Planning……………………………………. 9
4. **Block Diagram………………………………………………….** 10-27
5. Car Parking Block Stage……………………………... 11-12
6. Ultrasonic Sensor Stage……………………………… 13-14
7. Arduino Stage………………………………………... 15-22
8. Database Stage………………………………………. 23-24
9. Webpage Stage………………………………………. 25-26
10. **Components Required…………………………………………** 27-32
11. Hardware……………………………………………... 27-32
12. Software………………………………………………. 32
13. **Outcome ………………………………………………………...** 33-36
14. **Application……………………………………………………...** 37
15. **Literature review……………………………………………….** 38

References……………………………………………………………...... 39

Appendix……………………………………………………………….... 40

1.0 **INTRODUCTION**

Modern automated parking complexes commonly include integrated electromechanical transmission control, security control, detection systems and automated placement. The systems can fully support vehicle measurement, image analysis, electronic payment scanning, and automatic retrieval.

In an automated parking complex, drivers need only park at an entry pallet or port. Images of the car are captured and transmitted to a controller where dimensions of the car are read and the license plate recorded. Meanwhile, the driver pays by card and the system provides authorization, possibly via voice recognition. Once sensors have verified the entry pallet or cabin is empty of occupants, the process of automatic moving and parking will be initiated. The car will be moved by an elevator or a carousel system up to the storage floors and parked into an appropriate space with security guarantee.

To command and coordinate all the controllers and subsystems, automated parking systems require precise control, robust and reliable networking, and secured system maintenance.

All these we will try to implement in our project and make it real world project.



Fig. 1. Smart Parking System



2.0 **OBJECTIVE​ ​& MOTIVATION​** ​

1. **To design a model of a smart car parking system**-​Our main aim is to design a smart parking system for 6 cars.
2. **To optimize parking** – Users find the best spot available, saving time, resources and effort. The parking lot fills up efficiently and space can be utilized properly by commercial and corporate entities.
3. **To reduce traffic** – Traffic flow increases as fewer cars are required to drive around in search of an open parking space.
4. **To reduce pollution** –​Searching for parking burns around one million barrels of oil a day. An optimal parking solution will significantly decrease driving time, thus lowering the amount of daily vehicle emissions and ultimately reducing the global environmental footprint.
5. **To Enhance User Experience** –​A smart parking solution will integrate the entire user experience into a unified action. Driver’s payment, spot identification, location search and time notifications all seamlessly become part of the destination arrival process.
6. **New​ ​Revenue​ ​Streams​** ​–​ ​Many new revenue streams are possible with smart parking technology. For example, lot owners can enable tiered payment options dependent on parking space location. Also, reward programs can be integrated into existing models to encourage repeat users.
7. **Integrated​ ​Payments​ ​and​ ​POS**​ ​–​ ​Returning users can replace daily, manual cash payments with account invoicing and application payments from their phone. This could also enable customer loyalty programs and valuable user feedback.
8. **To Increase Safety** – Parking lot employees and security guards contain real-time lot data that can help prevent parking violations and suspicious activity.

License

Plate recognition cameras can gather pertinent footage. Also, decreased spot searching traffic on the streets can reduce accidents caused by the distraction of searching for parking.

1. ​**Real-Time​ ​Data​ ​and​ ​Trend​ ​Insight**​ ​–​ ​Over time, a smart parking solution can produce data that uncovers correlations and trends of users and lots. These trends can prove to be invaluable to lot owners as to how to make adjustments and improvements to drivers.
2. **To Decrease Management Costs** – More automation and less manual activity saves on labour cost and resource exhaustion.
   1. **PROPOSED​ ​PLANNING​ ​**

The proposed system is the combination of hardware and software to form a complete module.

Exchanging of all the information or data between mobile and sensor circuitry is done by CLOUD. The algorithm defining the parking slot allotment is as follows:

Initially selection and checking for car parking is made from mobile or computer using cloud.

Checks for availability for parking slots P1, P2, P3, P4

If parking slot is free, the particular slot on HTML page will be green.

If all parking slots are full, all the slots on HTML page will be red and a pop-up will be generated “Parking​ ​full”.

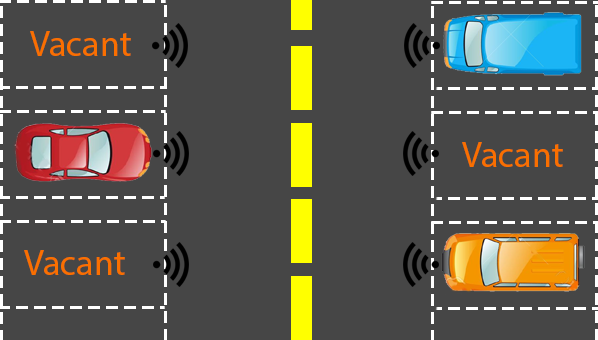


Fig 2. Basic pictorial view of Car Parking Block

3.0 **BLOCK​ ​DIAGRAM​ ​**



CAR PARKING BLOCK

SENSOR

N

ET

W

O

R

K

K

DATABASE

APPS

WEBPAGE

Fig. 3. Pictorial view of block diagram

3.1 **CAR PARKING BLOCK STAGE**

This is the first stage of the project in which we will build basic building block of the project.

This includes the following thing:

1. The real design model of parking block containing six parking lots.
2. In each parking lot there will be an ultrasonic sensor.
3. Each of these sensors will be connected to Arduino board to fetch real time data using wired connection on bread board.
4. Arduino will be connected to computer using data cable to update database.

Steps to connect ultrasonic sensor to Arduino: -

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.

1. Connect the Vcc pin to the positive rail on your breadboard.
2. Connect the Gnd pin to the negative rail on your breadboard.
3. Connect the Trig pin to any digital pin on the Arduino.
4. Connect the Echo pin to any digital pin on the Arduino.
5. Finally, connect the positive rail of the breadboard to 5V pin on the Arduino and the negative rail of the breadboard to the Gnd pin on the Arduino.

Arduino will be connected to computer using an Ethernet cable and Data cable.

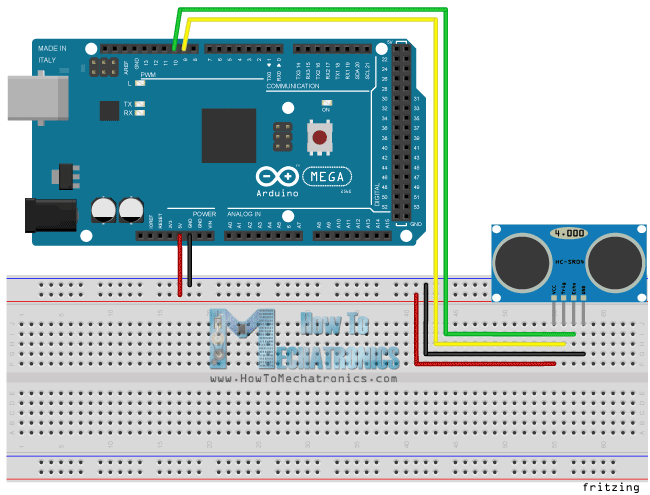


Fig. 4. Circuit Diagram of Arduino and sensor connection

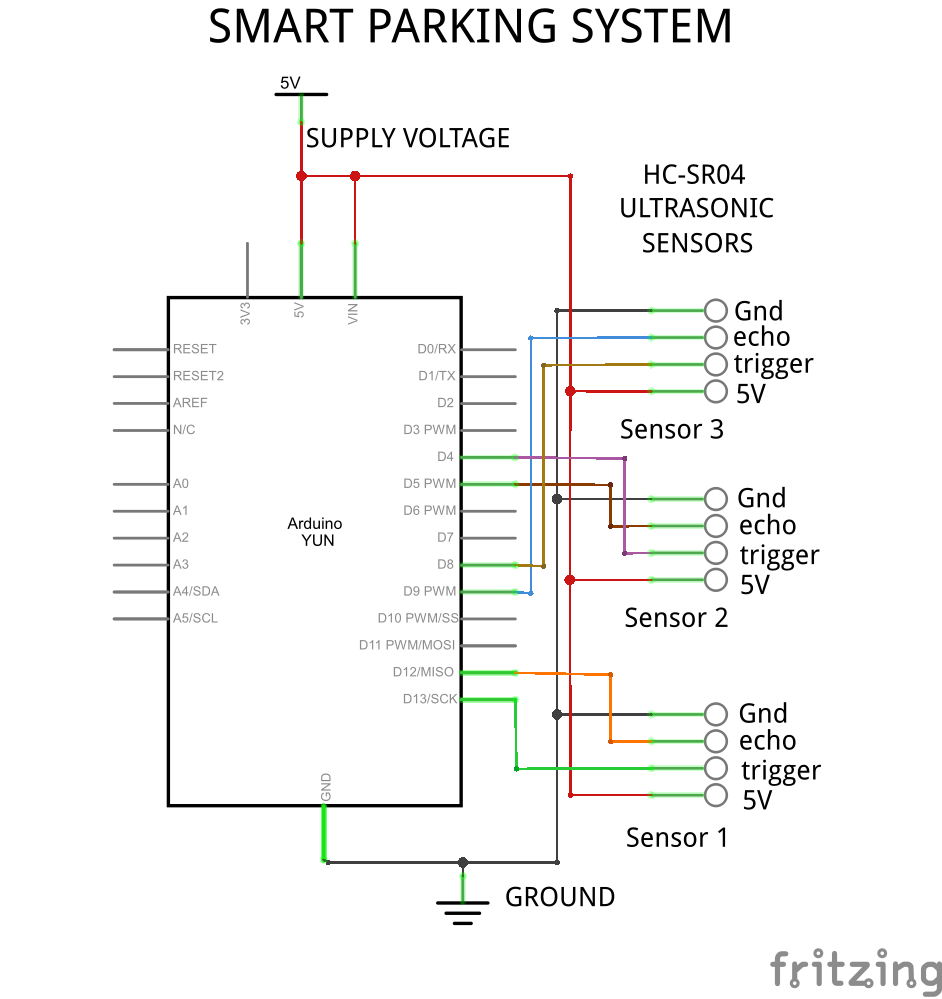


Fig. 5. Circuit Diagram of Arduino and sensor connection

3.2 **ULTRASONIC​ ​SENSOR STAGE**

​Ultrasonic sensors will be used in our project to detect if the parking slot is empty or free. In our project we plan to make 6 parking slots so we will be using 6 ultrasonic sensors which will be connected to the Arduino. Sensor connection with Arduino is explained in Sec 2.1.1.

An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object.

How It Works – Ultrasonic Sensor

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

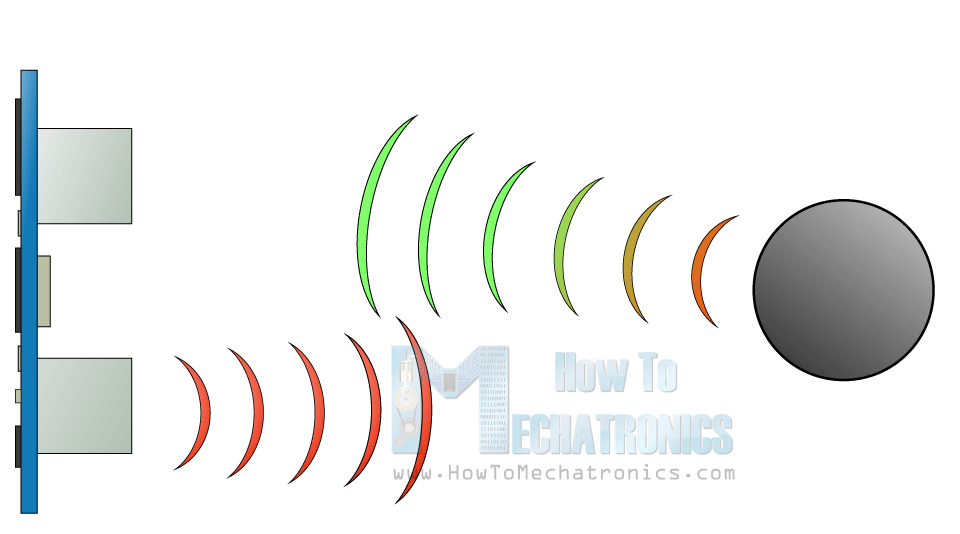


Fig 6. Working of Ultrasonic sensor

In order to generate the ultrasound, we need to set the Trig on a High State for 10 µs. That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received in the Echo pin. The Echo pin will output the time in microseconds the sound wave travelled.

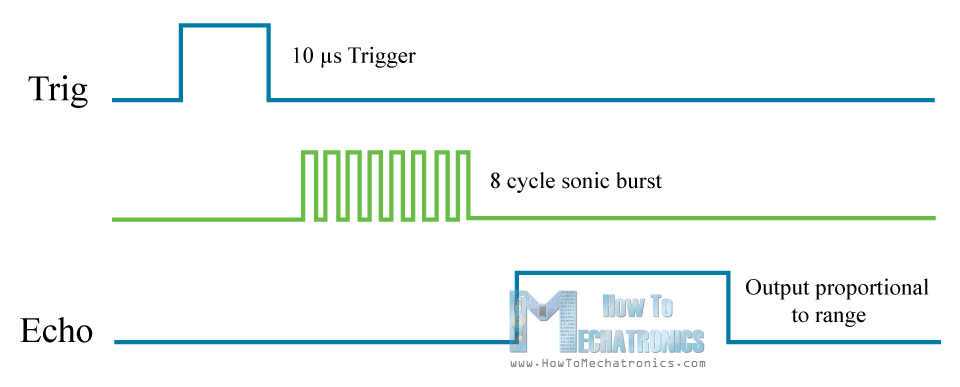


Fig 7. Timing diagram of Ultrasonic sensor

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 we 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.

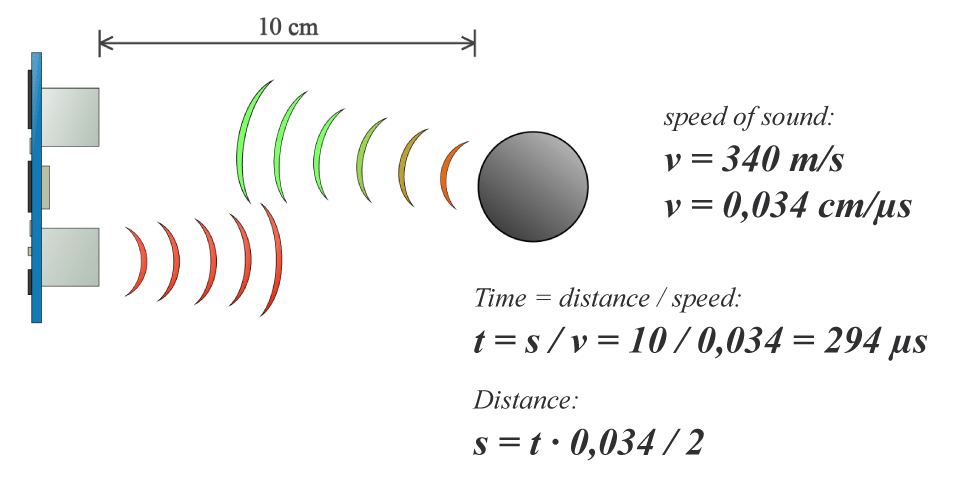


Fig. 8. Working of Ultrasonic sensor

3.3 **ARDUINO STAGE**

The main work of the Arduino in our project is to collect data from sensors connected to parking stations and to send the data to database. In order to fetch data from sensor and update it on database. We need to program the Arduino according to above sensor mechanism in order to regular update of sensor data.

SOURCE CODE

First we have to define the Trig and Echo pins and they are named trigPin and echoPin. Then we need a ‘Long’ variable, named “duration” for the travel time that we will get from the sensor and an integer variable for the distance.

In the setup we 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.

In the loop first we have to make sure that the trigPin is clear so we have to set that pin on a LOW State for just 2 µs. Now for generating the Ultra sound wave we have to set the trigPin on HIGH State for 10 µs. Using the ***pulseIn()*** function we have to read the travel time and put that value into the variable “duration”. This function has 2 parameters, the first one is the name of the echo pin and for the second one we can write either HIGH or LOW. In this case, HIGH means that the ***pulsIn()*** function will wait for the pin to go HIGH caused by the bounced sound wave and it will start timing, then it will wait for the pin to go LOW when the sound wave will end which will stop the timing. At the end the function will return the length of the pulse in microseconds. For getting the distance we will multiply the duration by 0.034 and divide it by 2 as we explained this equation previously.  At the end we will store the value of the distance on the database.

We will need three software:

•**Arduino**: https://www.arduino.cc/en/Main/Software

•**Xampp**: https://www.apachefriends.org/download.html (this is the web server application)

•**Sublime Text Editor**: https://www.sublimetext.com/download

**Then in Arduino software we need to create a file and write the below given code. It will check the current status of the sensor and through Ethernet shield will try to communicate to the local machine connected.**

#include <Ethernet.h>

#include <SPI.h>

#include <Servo.h>

Servo servo;

int available1;

int available2;

int available3;

int available4;

int available5;

**//distance for the ultrasonic sensor**

long distance1;

long distance2;

long distance3;

long distance4;

long distance5;

**//duration taken for the sensor to transmit ultrasonic waves**

long duration1;

long duration2;

long duration3;

long duration4;

long duration5;

//**opening the servo for the cars to enter**

long servoduration;

long servodistance;

//**the output pins for the sensor**

#define trig1  26

#define trig2  28

#define trig3  30

#define trig4  32

#define trig5  34

//**the input pins for the sensor**

#define echo1  27

#define echo2  29

#define echo3  31

#define echo4  33

#define echo5  35

//**the out and in pins for the sensor**

#define servotrig 38

#define servoecho 39

//ethernet datapins

char server[] = "169.254.55.149";

EthernetClient client;

//**default computer address**

int led=5;

byte mac[]={

 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED

};

int response;

void setup() {

 Ethernet.begin(mac);

 Serial.begin (9600);

 delay(100);

 pinMode(trig1,OUTPUT);

 pinMode(trig2,OUTPUT);

 pinMode(trig3,OUTPUT);

 pinMode(trig4,OUTPUT);

 pinMode(trig5,OUTPUT);

 pinMode(echo1,INPUT);

 pinMode(echo2,INPUT);

 pinMode(echo3,INPUT);

 pinMode(echo4,INPUT);

 pinMode(echo5,INPUT);

 pinMode(servotrig,OUTPUT);

 pinMode(servoecho,INPUT);

servo.attach(37);

}

void loop() {

   digitalWrite(trig1, LOW);

 delayMicroseconds(2);

 digitalWrite(trig1, HIGH);

 delayMicroseconds(10);

 digitalWrite(trig1, LOW);

 duration1 = pulseIn(echo1, HIGH);

 distance1 = (duration1/2) / 29.1;

 digitalWrite(trig2, LOW);

 delayMicroseconds(2);

 digitalWrite(trig2, HIGH);

 delayMicroseconds(10);

 digitalWrite(trig2, LOW);

 duration2 = pulseIn(echo2, HIGH);

 distance2 = (duration2/2) / 29.1;

digitalWrite(trig3, LOW);

 delayMicroseconds(2);

 digitalWrite(trig3, HIGH);

 delayMicroseconds(10);

 digitalWrite(trig3, LOW);

 duration3 = pulseIn(echo3, HIGH);

 distance3 = (duration3/2) / 29.1;

 digitalWrite(trig4, LOW);

 delayMicroseconds(2);

 digitalWrite(trig4, HIGH);

 delayMicroseconds(10);

 digitalWrite(trig4, LOW);

 duration4 = pulseIn(echo4, HIGH);

 distance4 = (duration4/2) / 29.1;

 digitalWrite(trig5, LOW);

 delayMicroseconds(2);

 digitalWrite(trig5, HIGH);

 delayMicroseconds(10);

 digitalWrite(trig5, LOW);

 duration5 = pulseIn(echo5, HIGH);

 distance5 = (duration5/2) / 29.1;

 Serial.print("Distance 1:");

 Serial.print(distance1);

 Serial.println("cm");

Serial.print("Distance 2:");

 Serial.print(distance2);

 Serial.println("cm");

Serial.print("Distance 3:");

 Serial.print(distance3);

 Serial.println("cm");

Serial.print("Distance 4:");

 Serial.print(distance4);

 Serial.println("cm");

 Serial.print("Distance 5:");

 Serial.print(distance5);

 Serial.println("cm");

delay(100);

//**getting the actual value required to do the project successfully**

 if(distance1<6)

 {

   available1=0;

 }

 else{

   available1 =1;

 }

if(distance2<6)

 {

   available2=0;

 }

 else{

   available2 =1;

 }

 if(distance3<6)

 {

   available3=0;

 }

 else{

   available3 =1;

 }

if(distance4<6)

 {

   available4=0;

 }

 else{

   available4 =1;

 }

 if(distance5<6)

 {

   available5=0;

 }

 else{

   available5 =1;

 }

 digitalWrite(servotrig, LOW);

 delayMicroseconds(2);

 digitalWrite(servotrig, HIGH);

 delayMicroseconds(10);

 digitalWrite(servotrig, LOW);

 servoduration = pulseIn(servoecho, HIGH);

   if(servodistance<10)

 {

   servo.write(90);

 }

 else

 {

   servo.write(-90);

 }

 if (!client.connected())

   {

     Serial.print("Available1=");

     Serial.println(available1);

     Serial.print("Available2=");

     Serial.println(available2);

     Serial.print("Available3=");

     Serial.println(available3);

     Serial.print("Available4=");

     Serial.println(available4);

     Serial.print("Available5=");

     Serial.println(available5);

     client.stop();

    delay(100);

     if (client.connect(server, 80))

     {

       Serial.println("connected");

       // **Make a HTTP request:**

       client.print("GET /UpdateAvailability.php?Position=");

       client.print("1");

       client.print("&&");

       client.print("Available=");

       client.print(available1);

       client.println(" HTTP/1.1");

       client.println("Host: 169.254.55.149");

       client.println("Connection: close");

       client.println();

       client.stop();

      }

     else

     {

       // **if you didn't get a connection to the server:**

       Serial.println("connection failed");

     }

     if (client.connect(server, 80))

     {

       Serial.println("connected");

       // Make a HTTP request:

       client.print("GET /UpdateAvailability.php?Position=");

       client.print("2");

       client.print("&&");

       client.print("Available=");

       client.print(available2);

       client.println(" HTTP/1.1");

       client.println("Host: 169.254.55.149");

       client.println("Connection: close");

       client.println();

       client.stop();

      }

     else

     {

       // **if you didn't get a connection to the server:**

       Serial.println("connection failed");

     }

    if (client.connect(server, 80))

     {

       Serial.println("connected");

       // Make a HTTP request:

       client.print("GET /UpdateAvailability.php?Position=");

       client.print("3");

       client.print("&&");

       client.print("Available=");

       client.print(available3);

       client.println(" HTTP/1.1");

       client.println("Host: 169.254.55.149");

       client.println("Connection: close");

       client.println();

       client.stop();

      }

     else

     {

       // **if you didn't get a connection to the server:**

       Serial.println("connection failed");

     }

    if (client.connect(server, 80))

     {

       Serial.println("connected");

       // Make a HTTP request:

       client.print("GET /UpdateAvailability.php?Position=");

       client.print("4");

       client.print("&&");

       client.print("Available=");

       client.print(available4);

       client.println(" HTTP/1.1");

       client.println("Host: 169.254.55.149");

       client.println("Connection: close");

       client.println();

       client.stop();

      }

     else

     {

       // **if you didn't get a connection to the server:**

       Serial.println("connection failed");

     }

    if (client.connect(server, 80))

     {

       Serial.println("connected");

       // Make a HTTP request:

       client.print("GET /UpdateAvailability.php?Position=");

       client.print("5");

       client.print("&&");

       client.print("Available=");

       client.print(available5);

       client.println(" HTTP/1.1");

       client.println("Host: 169.254.55.149");

       client.println("Connection: close");

       client.println();

       client.stop();

      }

     else

     {

       // **if you didn't get a connection to the server:**

       Serial.println("connection failed");

     }

   }

}

3.4 **DATABASE STAGE**

While broadcasting data over a dedicated web page on our home network is useful for a home project, fundamentals of an advanced project are to be able to **store** data we read from our connected sensors. This way we can monitor live data, but are also able to get historic information. It also allows us to capture data from multiple data input devices and display them when and how we want. Even though this could also be done with a dedicated web page by adding a little more code to your Arduino, it is easier to store it to a database and create a web page (or user interface) that reads data from the database.

**After running the Xampp our laptop or machine will act like a server. Then we need to create a database in PHPMyadmin which you will get with the xampp. Here we created a database called carparking. The below piece of code will help us to create a database connectivity.**

File name : dbcon.php

<!DOCTYPE html>

<html>

<body>

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "carparking";

$conn = new mysqli($servername,$username,$password,$dbname);

if($conn->connect\_error){

die("Connection failed: ".$conn->connect\_error);

}

?>

</body>

</html>

**This code helps you to update the database with certain query running**

File name: UpdateAvailability.php

<!DOCTYPE html>

<html>

<body>

<?php

include 'dbcon.php';

$Pos = $\_GET["Position"];

$Available = $\_GET["Available"];

$sql = "UPDATE parkinglot SET Available = " . $Available . " WHERE Position = " . $Pos;

print $sql;

$result = $conn->query($sql);

     $conn->close();

?>

</body>

</html>

3.5 **WEBPAGE STAGE**

At last when our database is updated with the current status of the parking lots, we need to display the status on the screen. So we are going to create a webpage for displaying it.

We are going to use few coding languages for creating it like PHP, HTML, CSS and Bootstrap.

First we will create a database connection using SQL and PHP. Then we would fetch the data from the database and will show it by using PHP.

**This code just help to show the data from the database to the users in a representational way. And this page refreshes every second giving the current status of the database.**

Filename : index.php

<!DOCTYPE html>

<html lang="en">

<head>

 <title>Bootstrap Example</title>

 <meta charset="utf-8">

 <meta name="viewport" content="width=device-width, initial-scale=1">

 <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">

 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>

 <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>

<style>

.red-icon {

color:#ff0000;

}

.green-icon {

color:#009933;

}

.table-responsive

{

width:50%;

}

</style>

</head>

<body>

<?php include 'dbcon.php';

$sql = "SELECT Position, Available FROM ParkingLot Order by Position";

$result = $conn->query($sql);

header("Refresh:1");

?>

<div class="container-fluid">

<div class="table-responsive">

<table class="table">

<thead>

<tr>

<th> Position </th>

<th> Availability </th>

</tr>

</thead>

<?php while($row = $result->fetch\_assoc()) { ?>

<tbody>

<tr>

<td> P <?php echo $row['Position']; ?>  </td>

<td>

<?php if ( $row['Available'] == 1 ) { ?>

<p><span class="glyphicon glyphicon-ok green-icon"></span></p>

<?php } else { ?>

<p><span class="glyphicon glyphicon-remove red-icon"></span></p>

<?php } ?>

</td>

</tr>

</tbody>

<?php } ?>

</table>

</div>

</div>

<?php $conn->close(); ?>

</body>

</html>

**EXTRA SETUP FOR CONNECTIVITY**

LAN Connectivity step:

* Ethernet shield need to be connected to the system using LAN wire to update the data from the sensor.
* If the system is connected to some wireless network, the sharing of wireless network should be on so that both can work simultaneously.
* To connect the shield, first of all we to provide dynamic Ip to the shield using DHCP address printer provided in Arduino.
* Now we need to upload the code of Web Server given in Arduino which will start the web server at some Ip. We can view that Ip in serial monitor.
* Now we are ready to upload the main project code. After uploading the project’s code, the system will work according to its functionality.

Data Updating:

* We have a local Xampp web server to update the data of the sensor and display it on a web page.
* The Arduino code which runs in a loop continuously read the data from the sensor and send it to Xampp web server to update the table.
* There is a table made in Xampp server which stores the data of availability of parking space corresponding to its position. It contains two columns as position and availability. Availability = 1 means that space is vacant for parking and Availability = 0 means that space is occupied.
* Now we need a web page which displays the data fetched from the web server. The web page need to refresh on regular interval of time so that there is very less time lag in the correctness of the result displayed.

4.0 **COMPONENTS​ ​REQUIRED**

4.1 **HARDWARE**

1. **HC-SR04 Ultrasonic sensors (x5)**

### **Ultrasonic Sensor Pin Configuration**

|  |  |  |
| --- | --- | --- |
| **Pin Number** | **Pin Name** | **Description** |
| 1 | Vcc | The Vcc pin powers the sensor, typically with +5V |
| 2 | Trigger | Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave. |
| 3 | Echo | Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor. |
| 4 | Ground | This pin is connected to the Ground of the system. |

### **HC-SR04 Sensor Features**

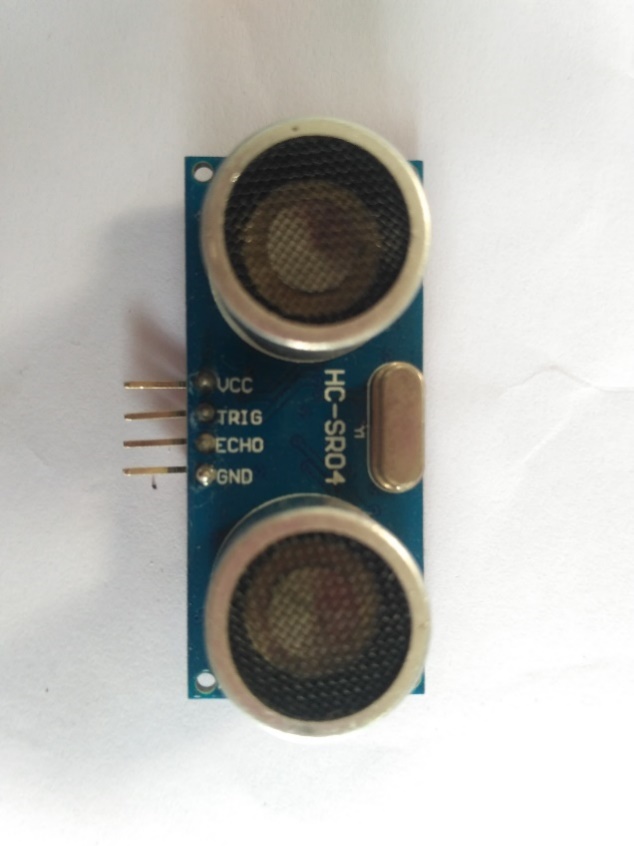
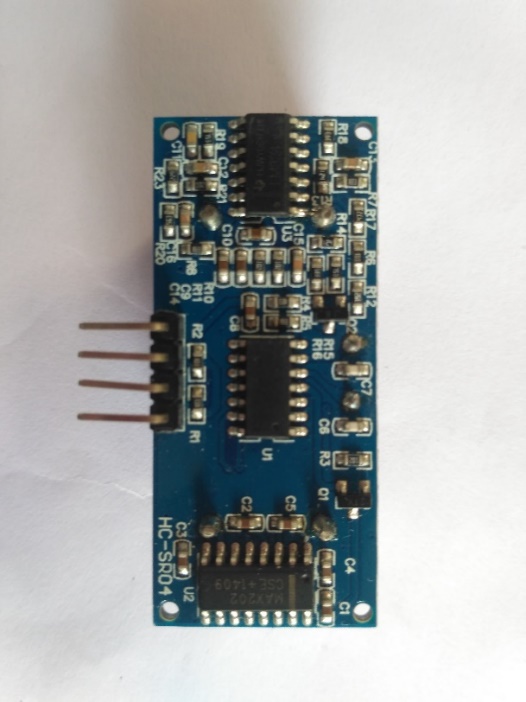
* Operating voltage: +5V
* Theoretical Measuring Distance: 2cm to 450cm
* Practical Measuring Distance: 2cm to 80cm
* Accuracy: 3mm
* Measuring angle covered: <15°
* Operating Current: <15mA
* Operating Frequency: 40Hz

Fig. 9. HC-SR04 ultrasonic sensors

1. **Arduino Mega 2560 (x1)**

Arduino Mega 2560 Rev. 3 Microcontroller Board is based on the Atmel ATmega2560 8-bit microcontroller (MCU). Arduino Mega 2560 features 54 digital input/output pins (15 of which can be used as PWM outputs) and 16 analog inputs. This Arduino MCU board also includes 4 UARTs (hardware serial ports), a 16MHz crystal oscillator, a USB connection, a power jack, an In-Circuit Serial Programming (ICSP) header, and a reset button.   
  
Mega 2560 includes everything the user needs to support the MCU. The user can get started by connecting the Mega 2560 to a computer with a USB cable or by powering it with an AC-to-DC adapter or battery. Arduino Mega 2560 board is compatible with most shields designed for the Uno and former boards Duemilanove or Diecimila. Mega 2560 is an update to the earlier Arduino Mega board.

|  |
| --- |
| **Features**   * [ATmega2560 Microcontroller (MCU)](https://www.mouser.in/Microchip/_/N-1z0zl2w?Keyword=ATmega2560&FS=True) * 5V operating voltage * 7-12V input voltage (recommended) * 6-20V input voltage (limit) * 54 digital input/output (I/O) pins   + 15 of which provide PWM output * 16 analog input pins * 20mA DC current per I/O pin * 50mA DC current for 3.3V pin * 256KB Flash memory   + 8KB used by bootloader * 8KB SRAM * 4KB EEPROM * 16MHz clock speed * 101.52mm x 53.3mm (Length x Width) * 37g weight   **Power**   * Can be powered via USB connection or with external power supply * Power source selected automatically   **Communication**   * Features a number of facilities for communicating with a computer, another board, or other MCUs * [ATmega2560](https://www.mouser.in/Microchip/_/N-1z0zl2w?Keyword=ATmega2560&FS=True) provides four hardware UARTs for TTL (5V) serial communication * An [ATmega16U2](https://www.mouser.in/Microchip/_/N-1z0zl2w?Keyword=ATmega16U2&FS=True) ([ATmega 8U2](https://www.mouser.in/Microchip/_/N-1z0zl2w?Keyword=ATmega8U2&FS=True" \o "ATmega 8U2) on the revision 1 and revision 2 boards) on the board channels one of these over USB and provides a virtual com port to software on the computer |
|  |

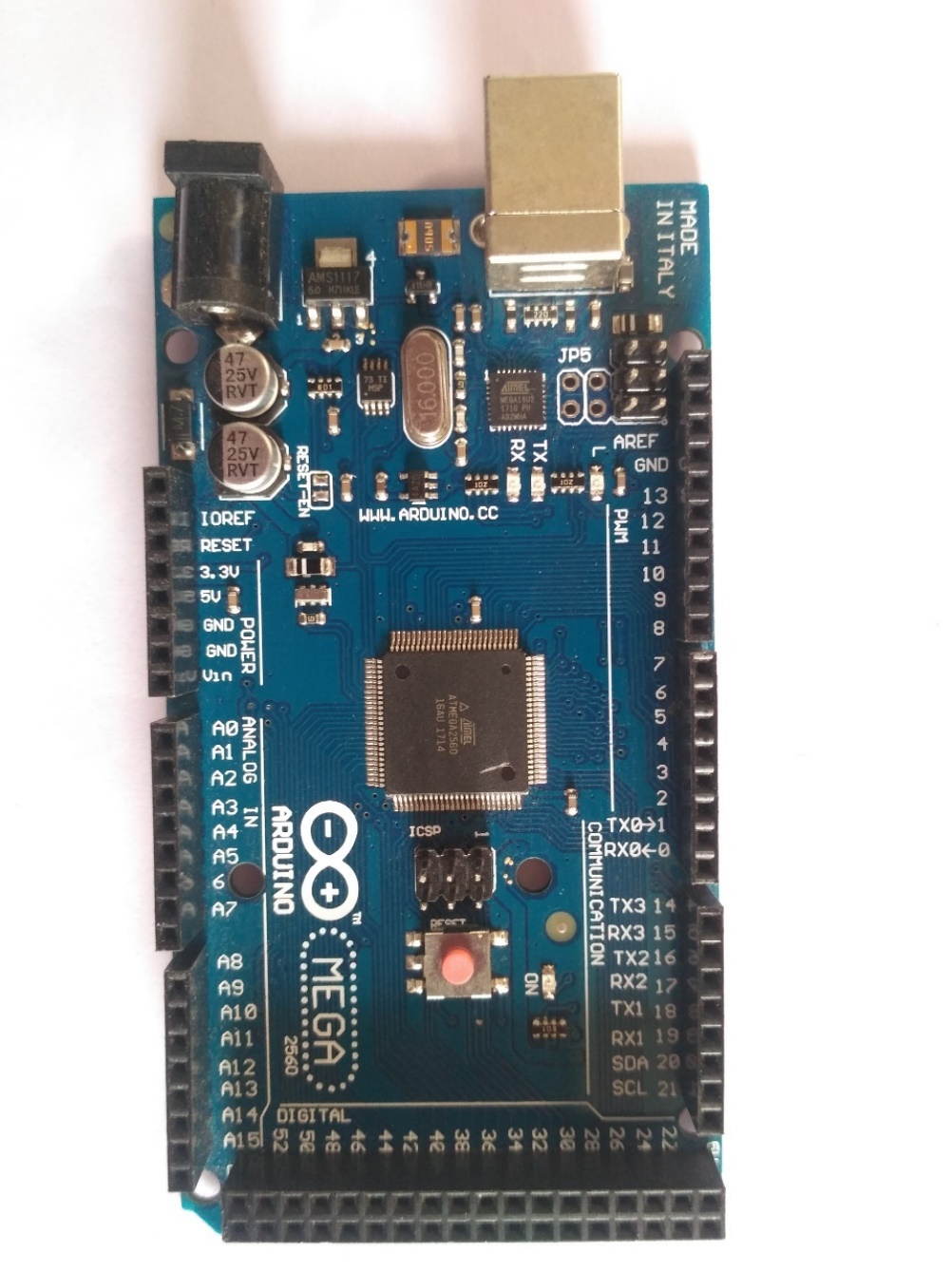


Fig. 10. Arduino Mega 2560

1. **Breadboard (x1)**

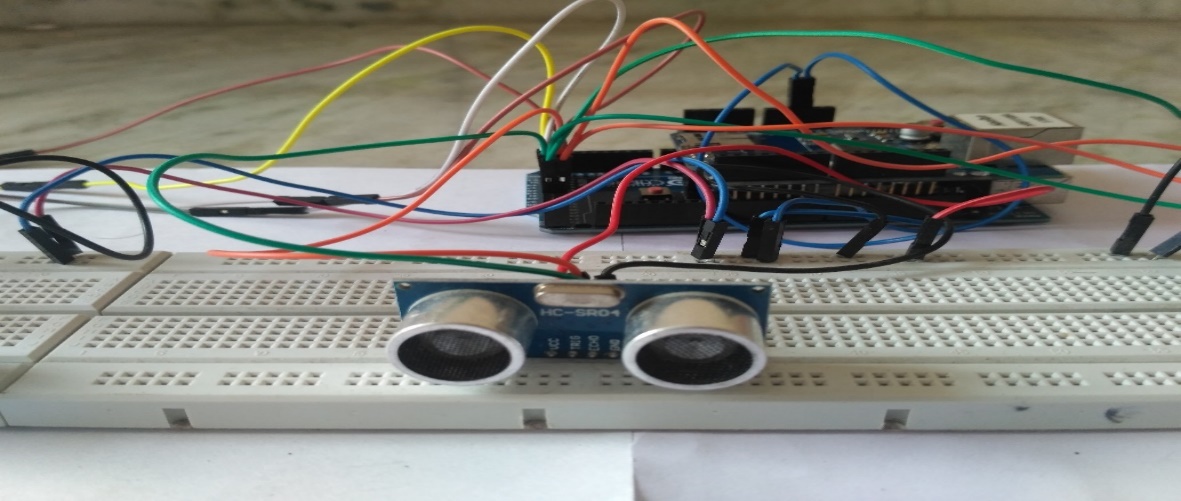


Fig. 11. Breadboard

1. **Arduino Ethernet Shield**

The **Arduino Ethernet** is a little nifty board, attractive because of its compact size - it has 32kb ROM and 2kb RAM. It's best for small projects

he **Arduino Mega** (with Ethernet Shield) has more memory and is great for larger project with more code line. It has 256kb ROM and 8kb RAM. You need this if you want to drive an LCD display with a large menu structure. Or embed a web server!



Fig. 12. Arduino Ethernet Shield

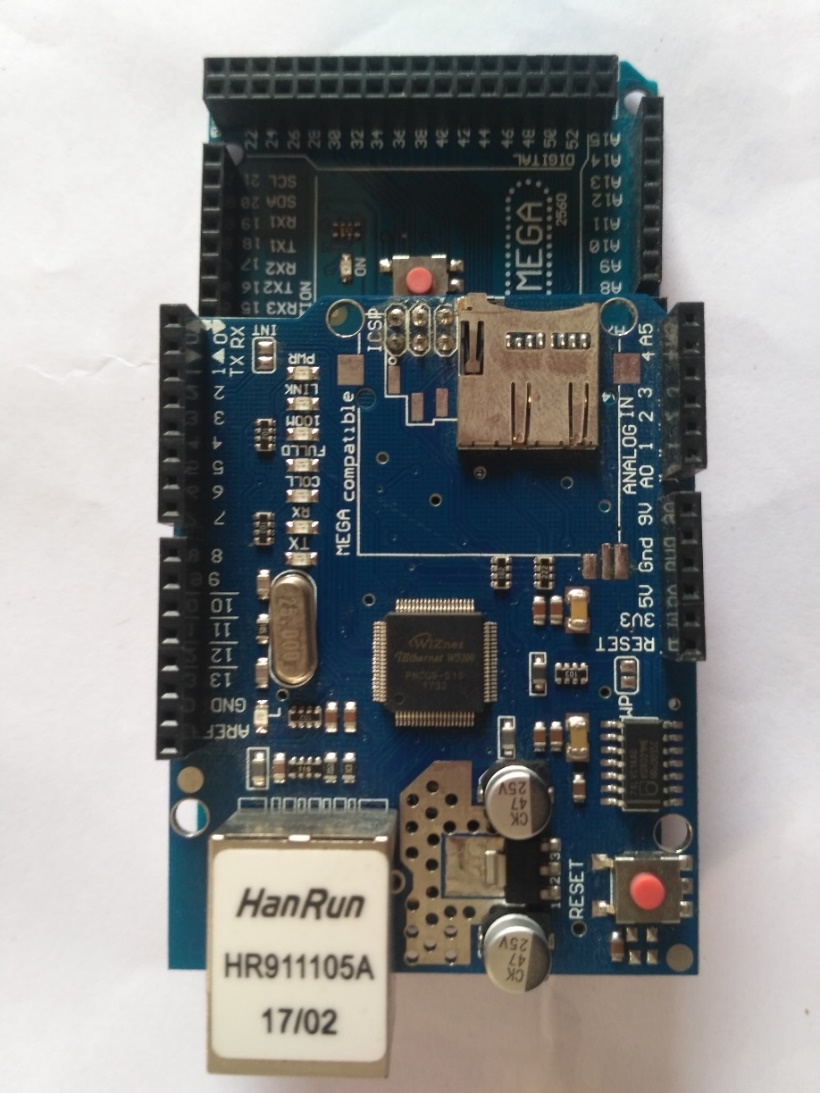


Fig. 13. Arduino Ethernet shield mounted on Arduino Mega 2560

1. **Ethernet cable (x1)**



Fig. 14. Ethernet Cable

1. **Wires (+30)**

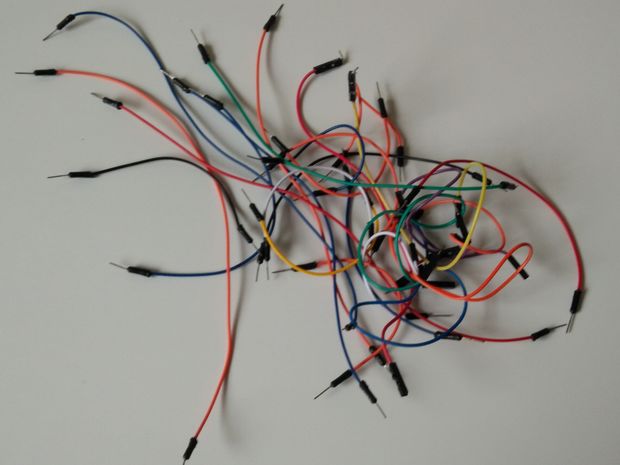


Fig. 15. Wires

4.2 **SOFTWARE**

1. Arduino-​Required to program the Arduino.
2. Xampp-​We will be using phpMyAdmin database and the PHP interpreter available in Xampp.
3. Sublime​ ​text​ ​editor.

5.0 **OUTCOME​ ​**

After successfully setting up hardware, making proper connections and running the codes, we expect the following output:

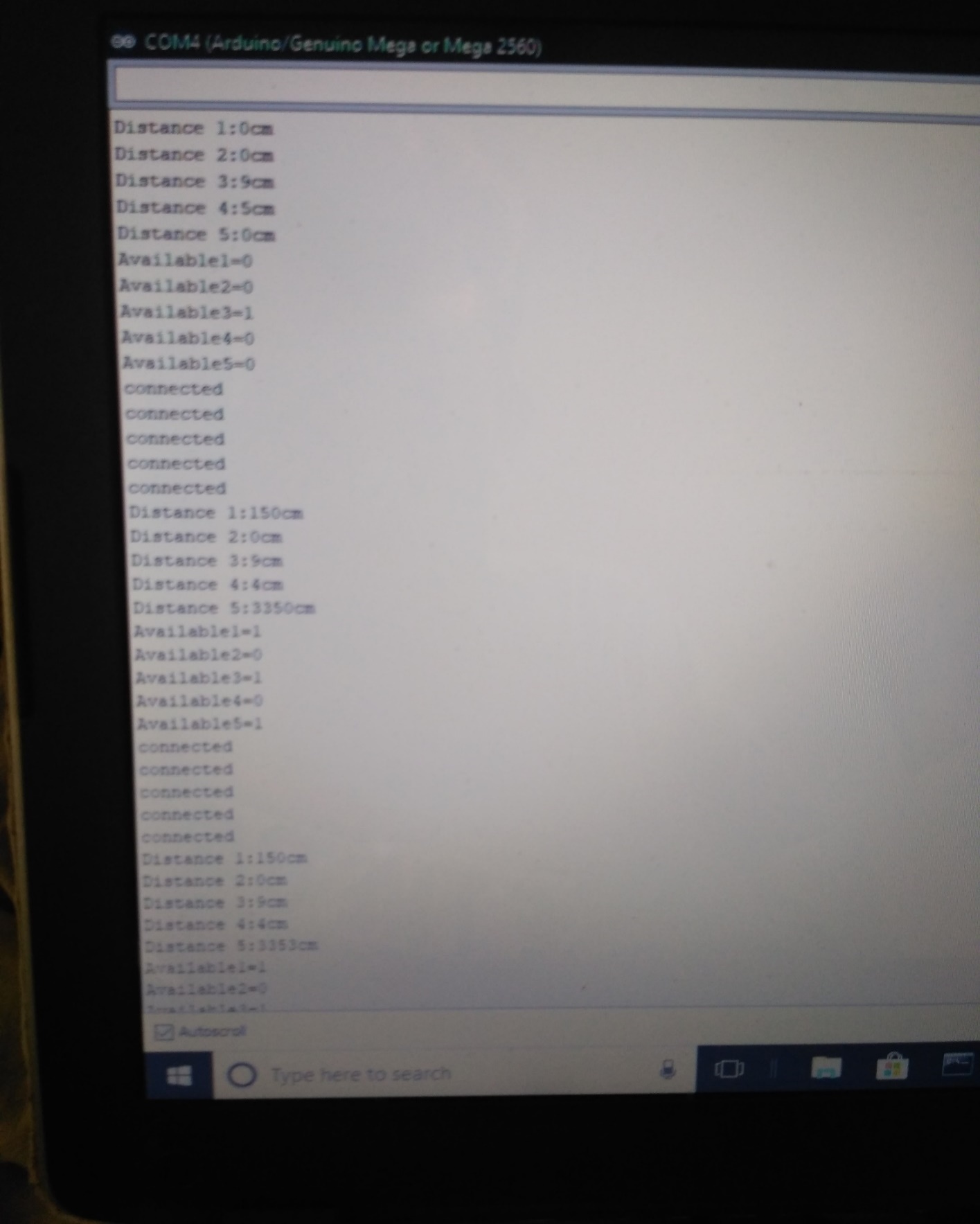


Fig. 16. Data extracted by Arduino from ultrasonic sensors

Since we are designing the parking system for only 5 cars so if there are 5 cars parked in our parking area and if we try to park one more car then the webpage will show the pop up “'parking full” meaning there are no available slots.

If there are 5 cars in 5 slots then the webpage will show that there are 5 filled slots and 1 available slot. It will also show which parking lot is empty.

Similarly, if there are 4 cars already parked, 2 slots will be available for a new car to be parked and respective slot detail will also be given.

LETS SEE SOME REAL SCENARIO EXAMPLE IN NEXT PAGES……….

**EXAMPLE 1**

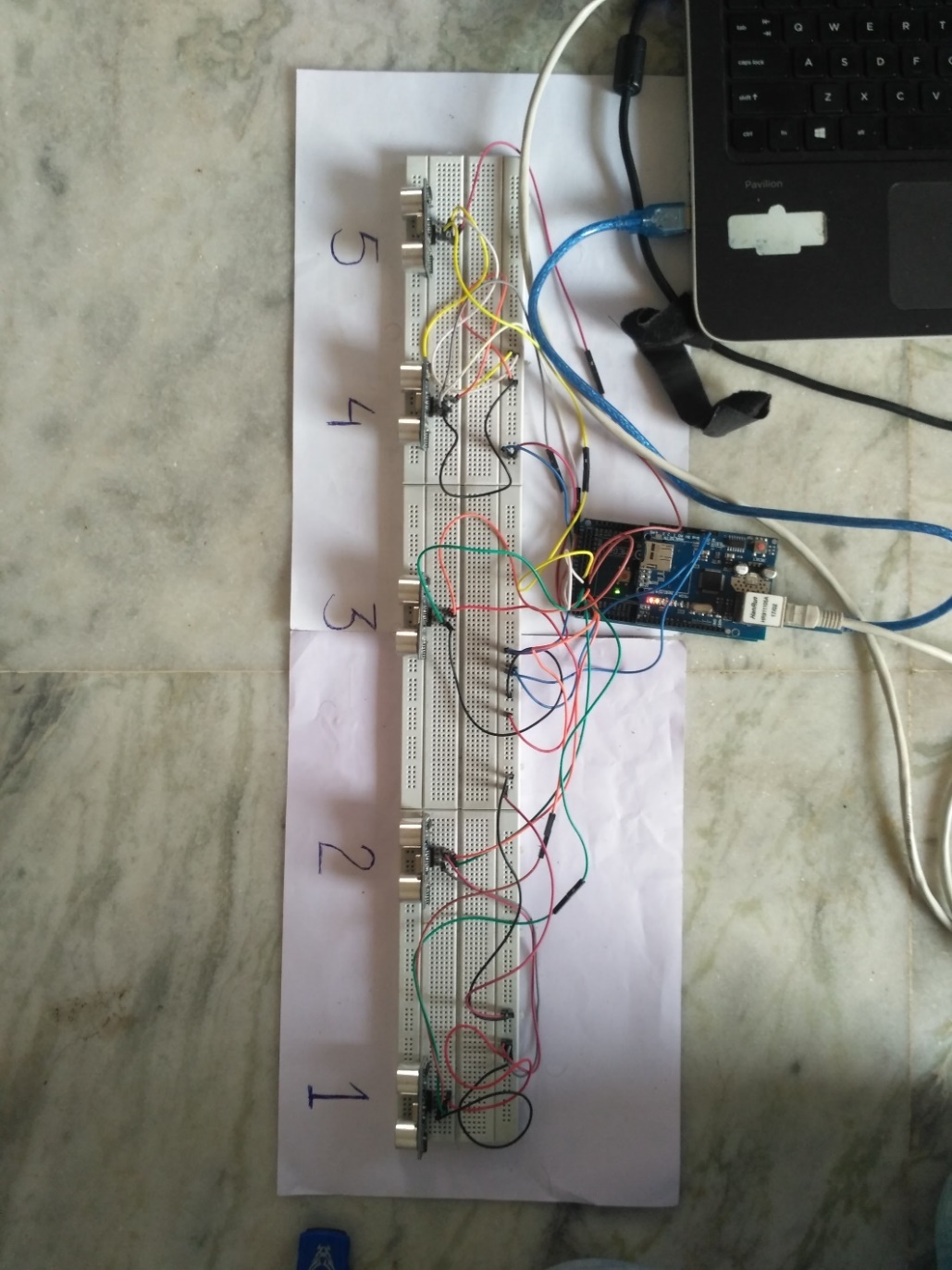


Fig. 17. SITUATION 1

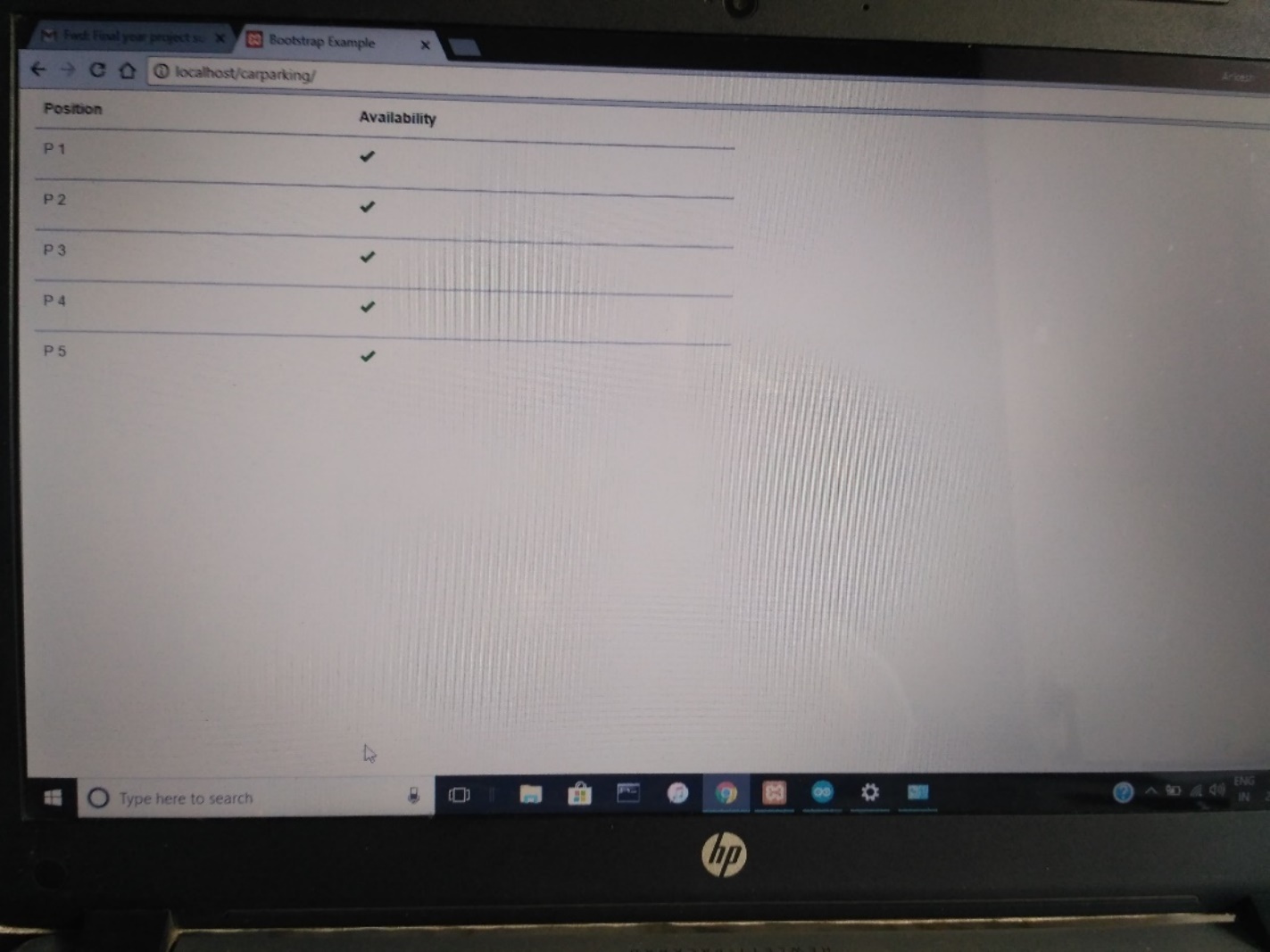
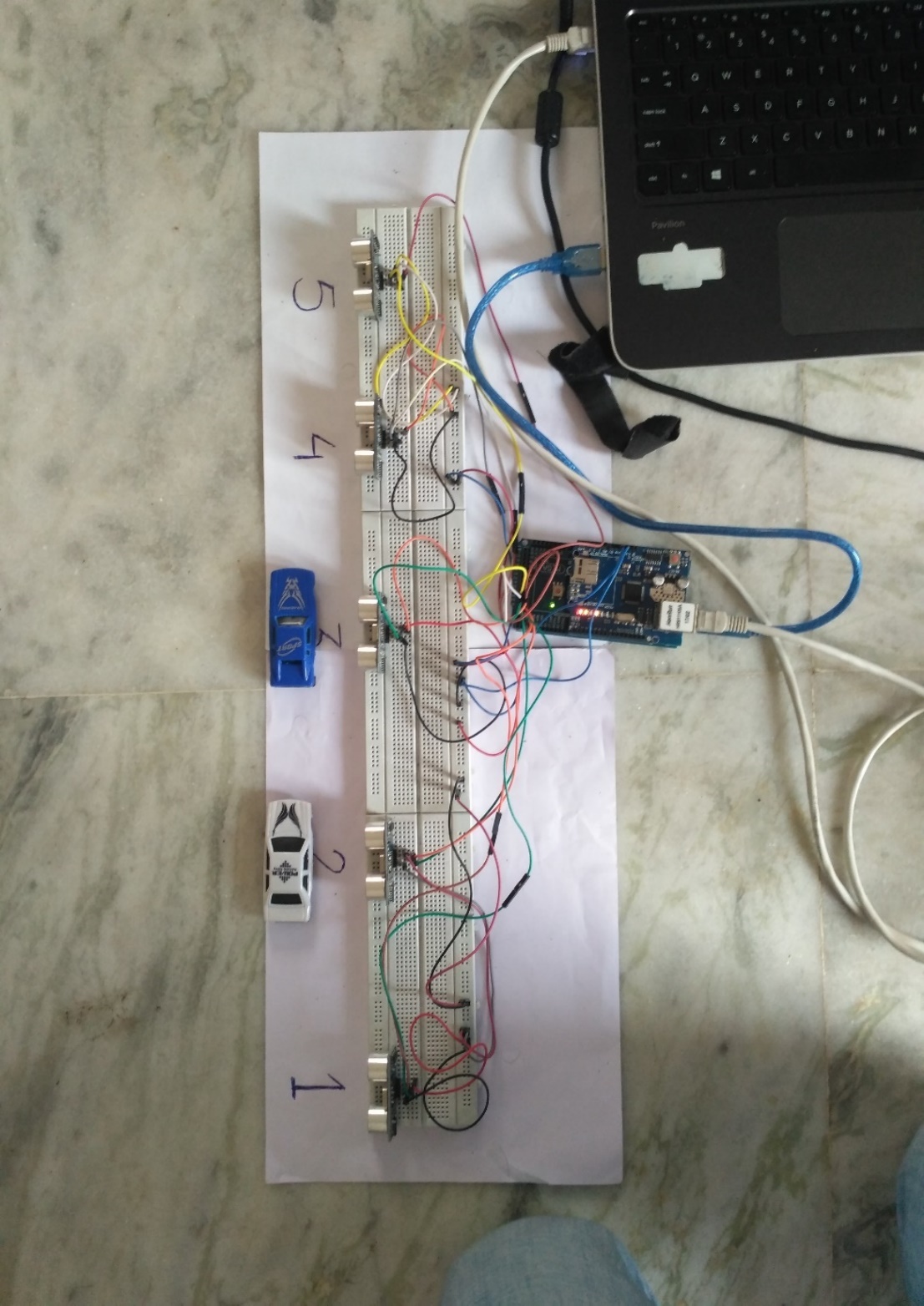


Fig. 10. Arduino Mega 2560

Fig. 18. Screenshot of output

**EXAMPLE 2**

Fig. 19. SITUATION 2

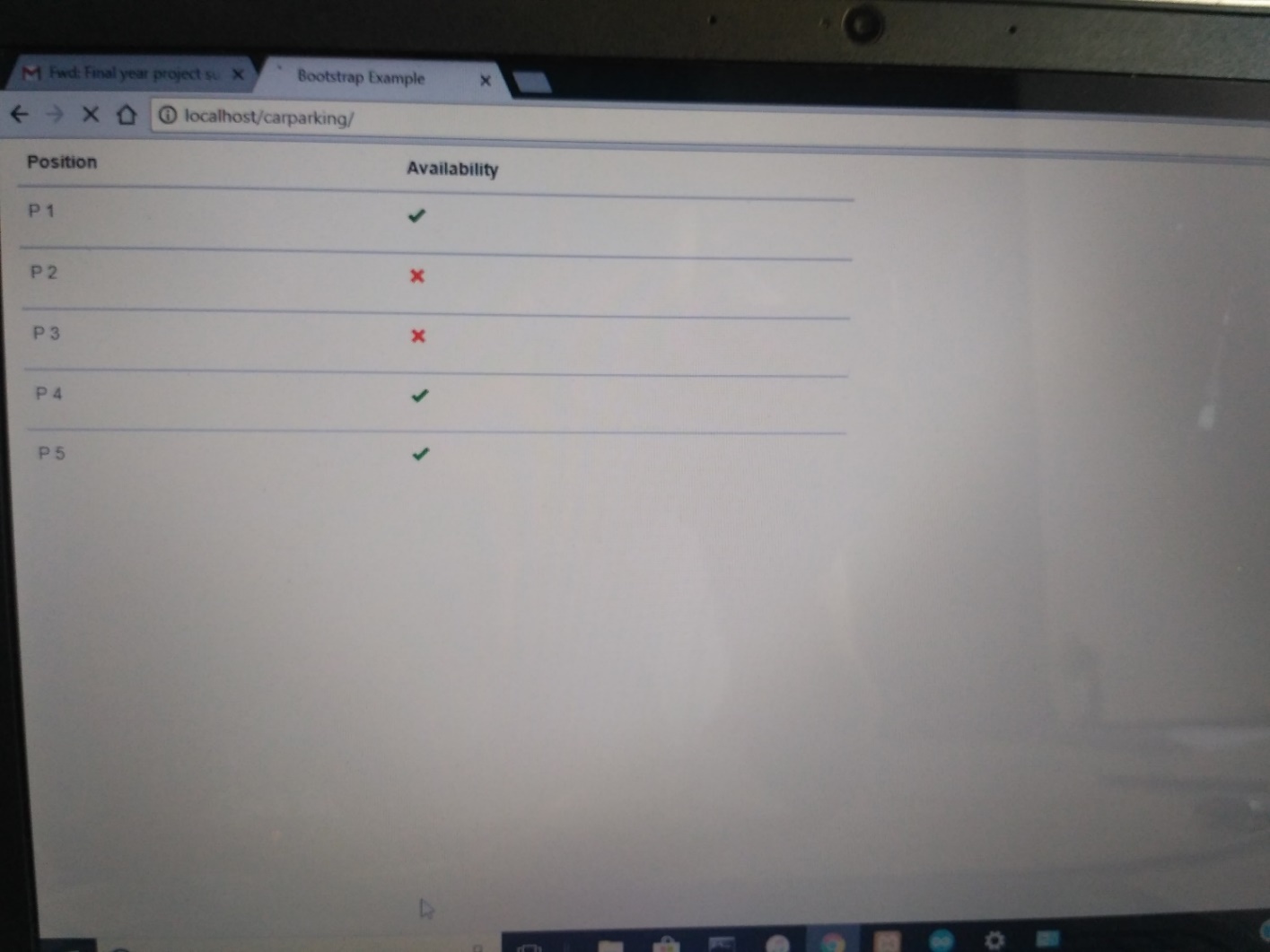


Fig. 20. Screenshot of output

**EXAMPLE 3**

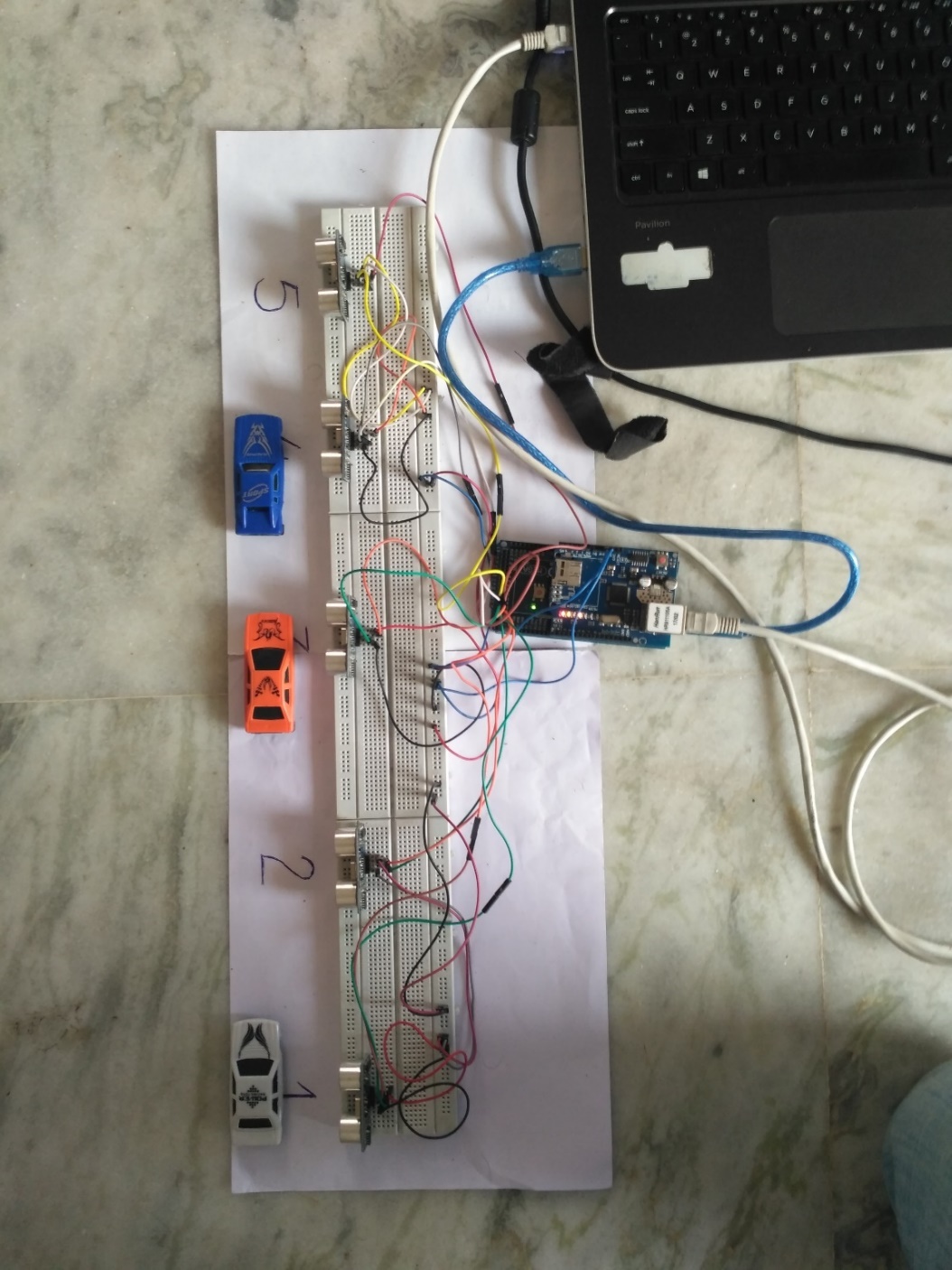


Fig. 21. SITUATION 3

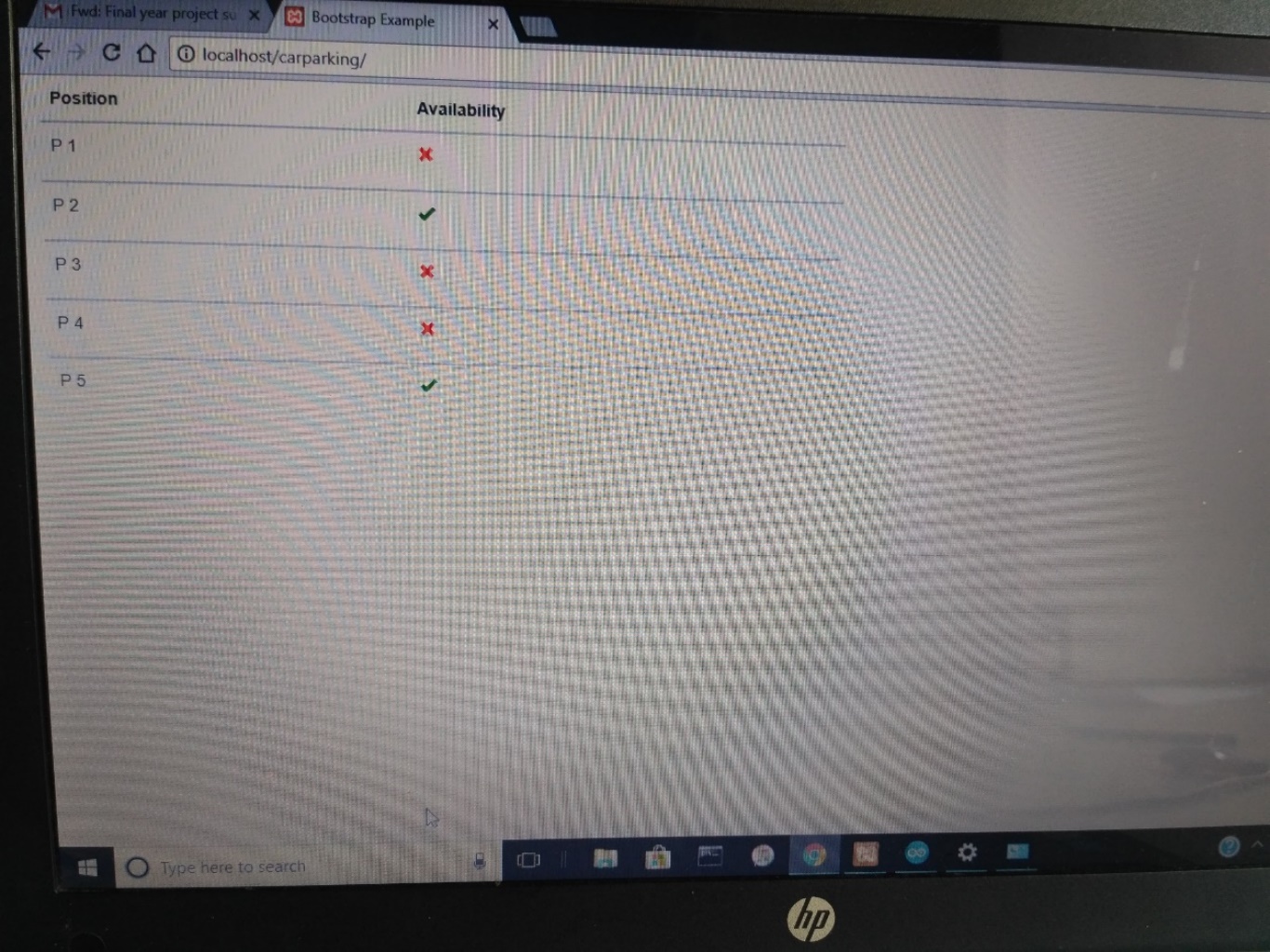


Fig. 22. Screenshot of output

6.0 **APPLICATIONS**

A smart parking system is a complex system composed of several hardware devices able to detect the city occupancy level of parking spaces, and software components integrated to manage the allocation of these parking spaces by redirecting cars accordingly. Usually, such systems are designed to assist motorists in the localization of available parking spaces, so that they can decide which space to select according to their own needs.

In the present work, we assume that Smart Cities will be equipped with such a complex system, and we propose to extend it with a software module implementing an application that can make decisions on where to park on behalf of a motorist, taking into account not only his/her needs, but also the social benefit for the city. In the proposed approach, a decision on where to park is the result of an automated negotiation process between two software agents: the User Agent (UA) acting on behalf of the motorist, and the Parking Manager (PM) who is responsible for managing parking spaces belonging to different car parks located in the city, which are offered to users as a global city facility. This means that different car parks owners agreed to subscribe to a City Parking System, managed by the Parking Manager, by delegating the selling of their parking spaces (partially or globally) to it. Hence, the Parking Manager is the authority responsible for allocating the parking spaces, virtually belonging to the City Parking System, but it is also responsible for collecting the information concerning specific city needs regarding transportation that will be gathered from the city council offices managing it.

**E-PARKING**

E-parking employs advanced technologies to combine and streamline parking reservation and payment systems. Using this system, a driver could inquire about the availability, reserve for a parking space at a given destination, and pay when leaving. The system is accessed via cell phone PDA and/or internet. Still conventional detectors are needed to detect approaching vehicles. However, the system must be able to identify customers and/or their vehicles making reservation and allows them the access to reserved space. The identification process at the parking lot may employ confirmation code access that the customer receives on cell phone. This will work on RFID technology. Vehicles will be installed with RFID tags and when they approach the parking area, required amount will be debited from their account.



Fig. 23. E-Parking

7.0 **LITERATURE REVIEW**

7.1 **FUTURE DIRECTIONS**

As said earlier, the smart parking solutions are still in the infancy state. As these solutions mature, additional features will be offered that will make parking easier for customers.

1. **Payment modes:** Both on-street and off-street parking spaces charge for parking. This entire transaction can be made from the mobile phone. Also specific parking lots have a time limit, usually on-street parking. In case a car over-stays the amount of time it is parked, an alert can be sent out to the towing company / police in order to take the car to the impound lot.
2. **Parking reservation**: Instead of driving to the parking lot and then getting directions, customers will be able to reserve parking from their home, even before leaving. For this, the server will have to be connected to the Internet, and must recognize the appropriate customer.
3. **Connected lighting**: Parking lots are kept illuminated throughout the day. As smart solutions will keep track of which parking spaces are occupied and which aren’t this will change. Sectors with no occupancies will be dimmed out. Similarly, office parking lots can be dimmed during work hours. As every new car entering will be detected, appropriate sectors can be illuminated, and others kept at diffused setting.
4. **On-street parking**: On-street parking will be the next sector that will be targeted once off-street parking has reached a better state. Already pilot projects are being implemented in the cities of Los Angeles and San Francisco.
5. **GPS based directions**: For both on-street and off-street directions, GPS based systems will no doubt be the best possible solution. The user will be able to get real time directions that will guide to the parking space. It won’t be just a route drawn on a map, but rather arrows pointing towards the possible destinations. However at the present moment getting GPS connection inside parking lots is difficult. It will take lots of innovation and technological wizardry to achieve.
6. **Better governance and traffic jam avoidance**: The data of parking spaces, especially the day of the week and time of parking can be useful for both local governing bodies and for citizens. For local government bodies the patterns of high and low parking density will result in deciding tariff rates that will discourage customers from using cars on certain days. This will help reduce traffic congestions. Variable tariff rates will also encourage carpooling. For the citizens, the availability of data will help them plan better while going to a particular destination. They will be able to opt for public transport when parking spaces are less. The historic trends will work in the favour of citizen as well.

REFERENCES

[*http://www.icreateproject.info*](http://www.icreateproject.info)

[*http://www.instructables.com*](http://www.instructables.com)

[*http://www.youtube.com*](http://www.youtube.com)

[*http://www.ieeexplore.ieee.org*](http://www.ieeexplore.ieee.org)

[*http://www.wikipedia.com*](http://www.wikipedia.com)

[*https://www.arduino.cc/en*](https://www.arduino.cc/en)

**APPENDIX**

Fig. 1. Smart Parking System……………………………………………………………..7

Fig. 2. Basic pictorial view of car parking block…………………………………………9

Fig. 3. Pictorial view of block diagram…………………………………………………..10

Fig. 4. Circuit Diagram of Arduino and sensor connection……………………………...12

Fig. 5. Circuit Diagram of Arduino and sensor connection……………………………...12

Fig 6. Working of Ultrasonic sensor …………………………………………………….13

Fig 7. Timing diagram of Ultrasonic sensor……………………………………………. 14

Fig. 8. Working of Ultrasonic sensor…………………………………………………….14

Fig. 9. HC-SR04ultrasonic sensors……………………………………………………... 28

Fig. 10. Arduino Mega 2560…………………………………………………………….30

Fig. 11. Breadboard……………………………………………………………………...30

Fig. 12. Arduino Ethernet Shield………………………………………………………...31

Fig. 13. Arduino Ethernet Shield mounted on Arduino Mega 2560…………………….31

Fig. 14. Ethernet cable…………………………………………………………………...32

Fig. 15. Wires……………………………………………………………………………32

Fig. 16. Data extracted by Arduino from ultrasonic sensors……………………………33

Fig. 17. SITUATION 1………………………………………………………………….34

Fig. 18. Screenshot of output……………………………………………………………34

Fig. 19. SITUATION 2………………………………………………………………….35

Fig. 20. Screenshot of output……………………………………………………………35

Fig. 21. SITUATION 3………………………………………………………………….36

Fig. 22. Screenshot of output……………………………………………………………36

Fig. 23. E-Parking……………………………………………………………………….37