Robotics Society NITH

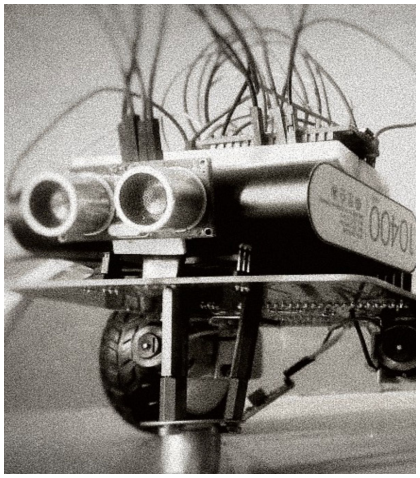
Instruction Manual for the preparation of bot to be used
on the day of Workshop **Deus Ex Machina** on 10th, 11th Sep, 2019.

DEUS EX MACHINA

September 10, 2019

Obstacle Avoidance Robot with AI Integration

Obstacle Avoiding Robot is an intelligent device which can automatically sense the obstacle in front of it and avoid them by turning itself in another direction. This design allows the robot to navigate in unknown environment by avoiding collisions, which is a primary requirement for any autonomous mobile robot.



How the bot works?

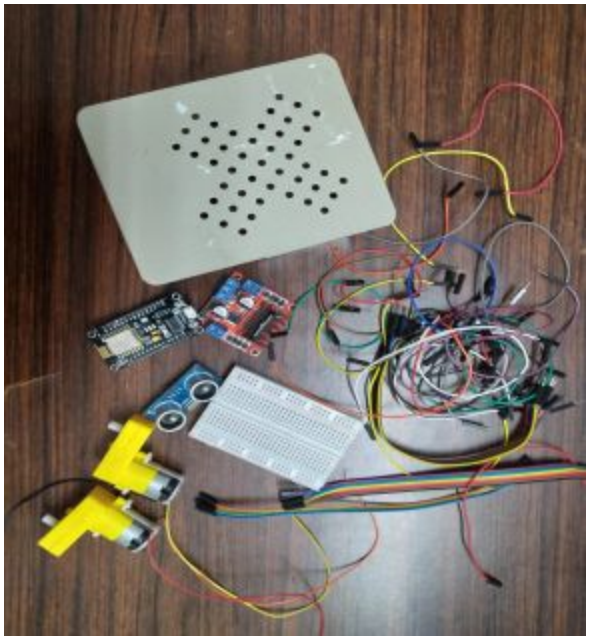
OVR uses Ultrasonic sensor for avoiding any obstacle in its path. The basic principle behind the working of ultrasonic sensor is to note down the time taken by sensor to transmit ultrasonic beams and receiving the ultrasonic beams after hitting the surface. According to the distance given by the Ultrasonic Sensor the robot decides whether it has to continue moving straight or change its current path to avoid obstacles. Motors are controlled using Node MCU with the help of a Motor Driver.

Use of Artificial Intelligence

Using Artificial Intelligence, Obstacle Avoidance Robot can avoid any obstacle without any human interference. But in order to do that we need to train this bot to avoid obstacles. While we train our bot by running it on the ground, the data will be stored on the server and we can download it on our mobile phones or laptops. That data can be trained and the model can be used for making our bot autonomous.

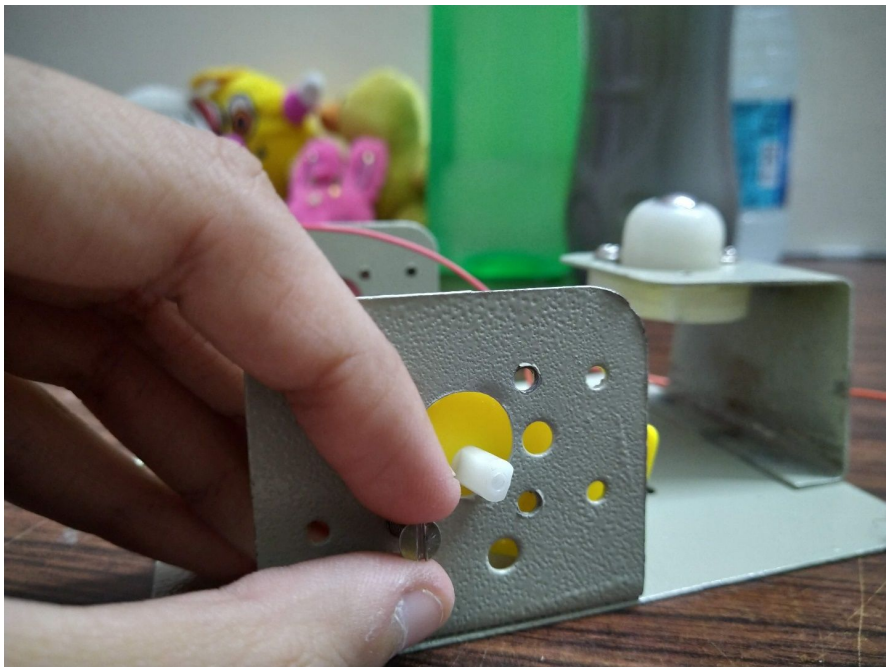
CONSTRUCTION

Step 1: Gather all components required.

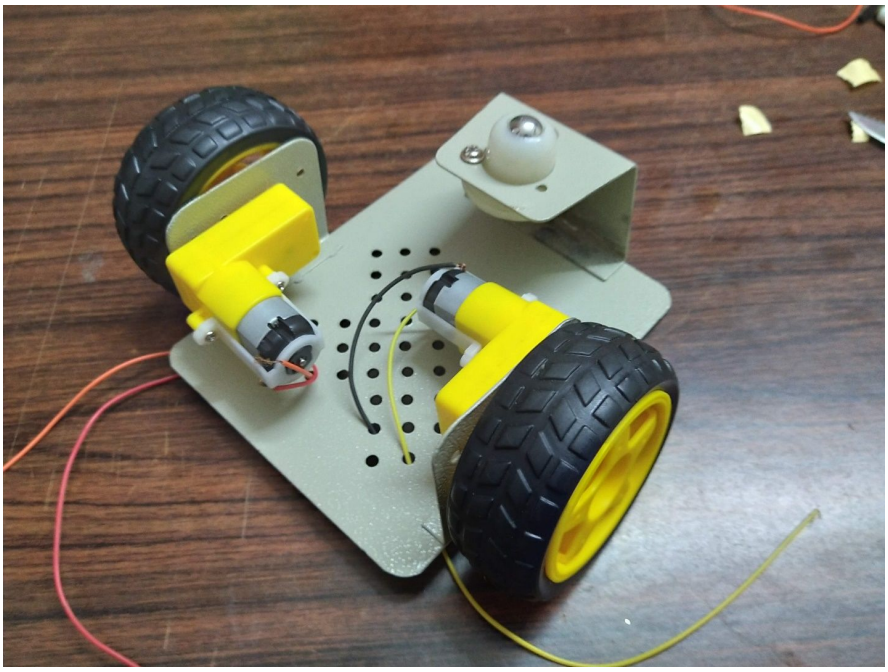


Step 2: Fix the castor wheel on its spot on the chassis.

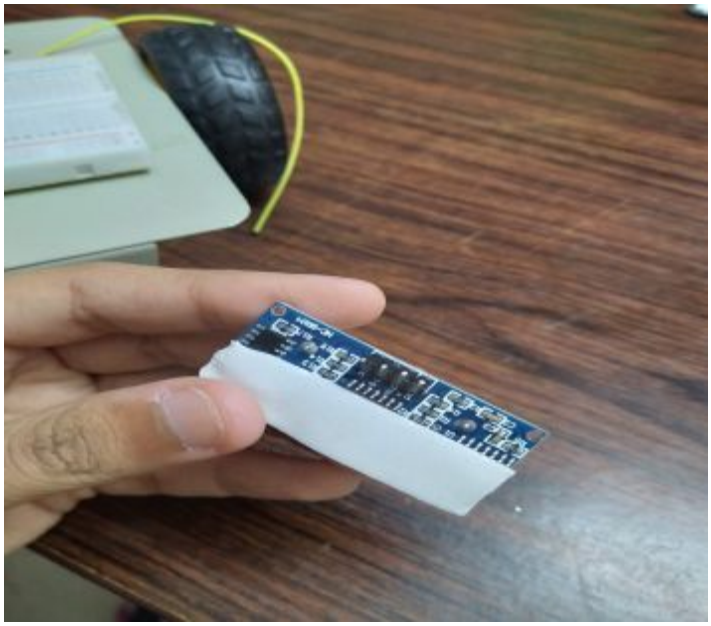
Step 3: Fix the motors on the chassis.



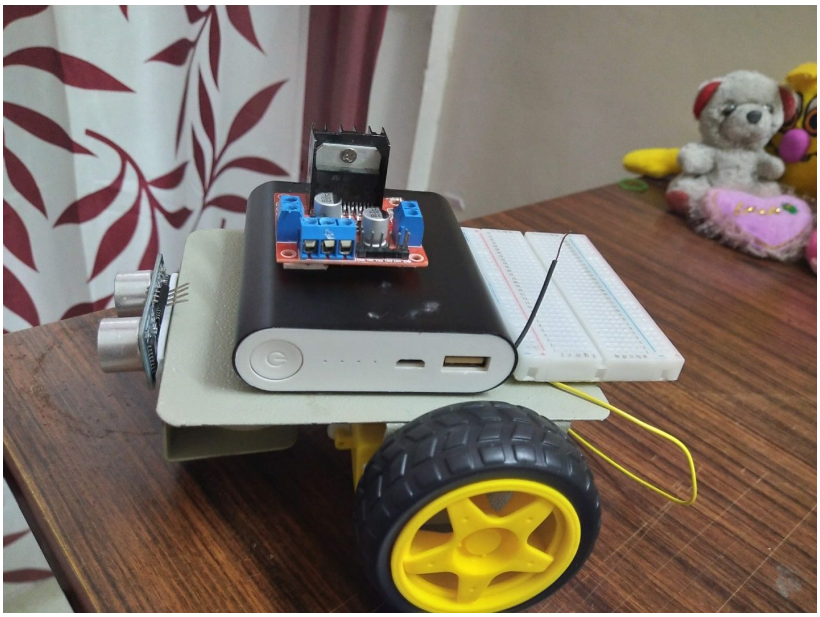
Step 4: Place the wheels on the Motors.



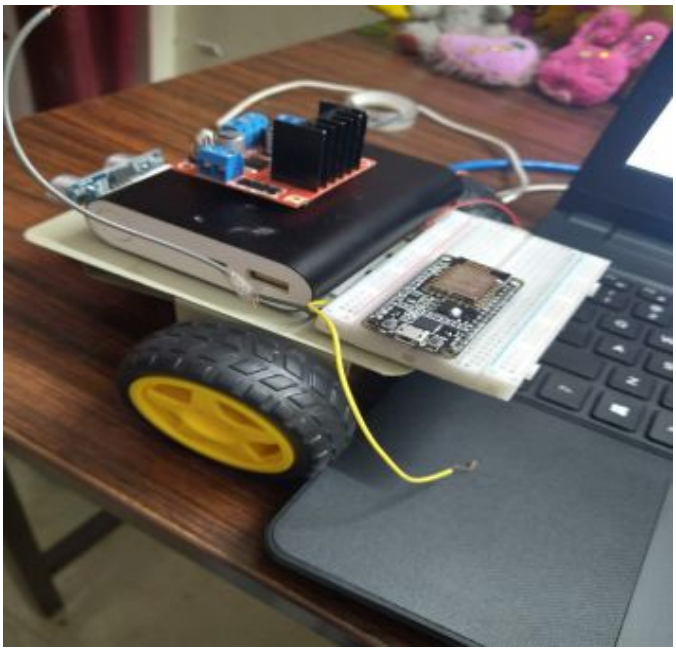
Step 5: Put double sided tape on UltraSonic Sensor and place it on the top of the bot.



Step 6: Place the Power Bank on the top of the bot on the chassis and put Motor Driver, Breadboard on top of it as shown below.



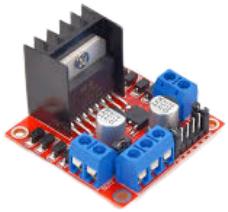
Step 7: Now fix the Node MCU on the breadboard as shown.



Step 8: JUMPER WIRE CONNECTIONS

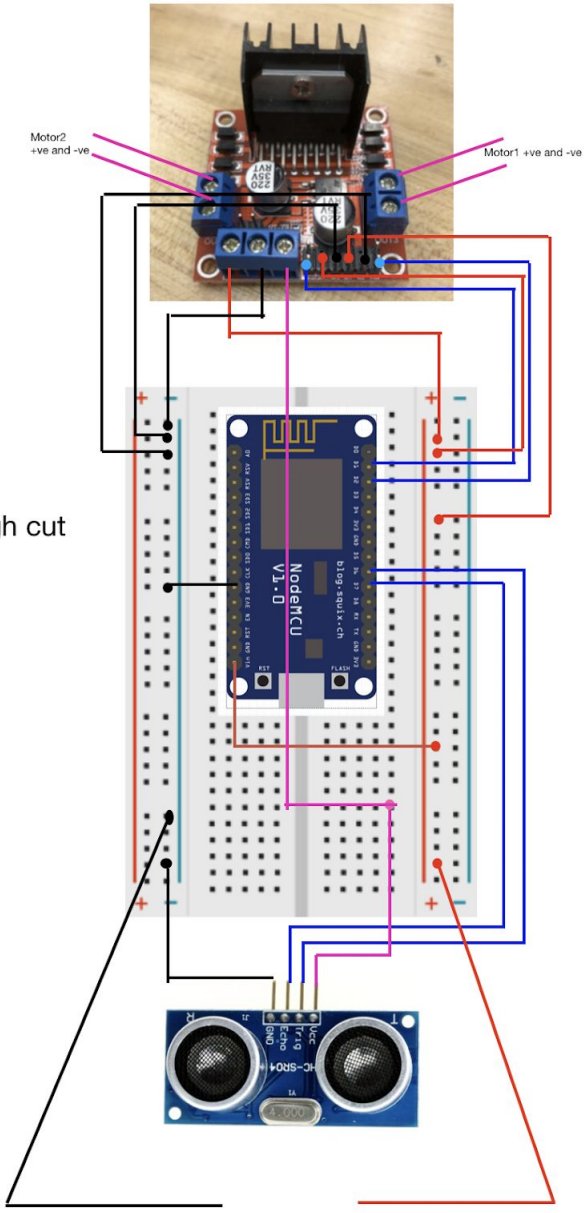
Firstly make separate rows for power, ground from the powerbank.
The positive wire and negative wires from the powerbank goes to the side rows of the breadboard which are already marked with red and blue color respectively.
Now whenever we need 5V we will take it from those two rows.

For **Motor Driver** L298N:
12 V Power Supply - 5V
Ground - GND on the breadboard
IN1, IN3 - 5V
IN2,IN4 - GND on the breadboard
EN1, EN2 - Output pins which goes to Node MCU(to the pins declared in the code)



For **Ultrasonic Sensor**:
VCC - 5V
GND - GND on the breadboard
TRIG - D6
ECHO - D7

Step 9: Verification of Circuit. Make sure you verify your connections with the help of the circuit diagram shown below.



Step 10: Make sure you have the **index.h** file before you start the next step. Place the 'index.h' file in the same folder with '.ino' file.

(You can get the index.h file from here : index.h)

```
const char MAIN_page[] PROGMEM = R"=====(
<!doctype html>
<html>

<head>
   <title>RealTime Graph|RoboSoc</title>

   <script src =
"https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.7.3/Chart.min.j
s"></script>
   <style>
   canvas{
        -moz-user-select: none;
        -webkit-user-select: none;
        -ms-user-select: none;
   }

   /* Data Table Styling */
   #dataTable {
     font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
     border-collapse: collapse;
     width: 100%;
   }

   #dataTable td, #dataTable th {
     border: 1px solid #ddd;
     padding: 8px;
   }

   #dataTable tr:nth-child(even){background-color: #f2f2f2;}
```

```
    #dataTable tr:hover {background-color: #ddd;}

    #dataTable th {
      padding-top: 12px;
      padding-bottom: 12px;
      text-align: left;
      background-color: #4CAF50;
      color: white;
    }
    </style>
</head>

<body>
        <div style="text-align:center;"><br>Real Time Data Logging
with Graphs on NodeMCU</div>
        <div class="chart-container" position: relative; height:350px;
width:100%">
        <canvas id="Chart" width="400" height="400"></canvas>
        </div> <div>
    <input id="inputFileNameToSaveAs"></input>
        <button onclick="saveTextAsFile()">Save Text to
File</button>
    <table id="dataTable">
      <tr><th>Date-Time</th><th>Left RPM</th><th>Right
RPM</th><th>UltraSonic Distance</th></tr>
    </table> </div> <br> <br>

<script type="text/javascript">
var valuesUS = [];
var timeStamp = [];
var entry="";
function showGraph() {
        for (i = 0; i < arguments.length; i++) {
         valuesUS.push(arguments[i]);
        }
```

```
	var ctx =
document.getElementById("Chart").getContext('2d');
	var Chart2 = new Chart(ctx, {
	type: 'line',
	data: {
	labels: timeStamp,    //Bottom Labeling
	datasets: [{
		label: "Distance",
		fill: false,    //Try with true
		backgroundColor: 'rgba( 243, 156, 18 , 1)', //Dot marker
color
		borderColor: 'rgba( 243, 156, 18 , 1)',    //Graph Line
Color
		data: valuesUS,
	}],
	},
	options: {
	title: {
		display: true,
		text: "UltraSonic"
		},
	maintainAspectRatio: false,
	elements: {
	line: {
		tension: 0.5 //Smoothening (Curved) of data lines
		}
	},
	scales: {
		yAxes: [{
			ticks: {
			beginAtZero:true
			}
		}]
	}
	}
	});
```

```
}

//On Page load show graphs
window.onload = function() {
    console.log(new Date().toLocaleTimeString());
    showGraph(0,0,0,0);
};
setInterval(function() {
  // Call a function repetatively with 5Second interval
  getData();
}, 500); //500mSeconds update rate

function saveTextAsFile() {
        var textToSave = entry;
        var textToSaveAsBlob = new Blob([textToSave],
{type:"text/plain"});
        var textToSaveAsURL =
window.URL.createObjectURL(textToSaveAsBlob);
        var fileNameToSaveAs =
document.getElementById("inputFileNameToSaveAs").value;

        var downloadLink = document.createElement("a");
        downloadLink.download = fileNameToSaveAs;
        downloadLink.innerHTML = "Download File";
        downloadLink.href = textToSaveAsURL;
        downloadLink.onclick = destroyClickedElement;
        downloadLink.style.display = "none";
        document.body.appendChild(downloadLink);

        downloadLink.click();
}

function destroyClickedElement(event) {
        document.body.removeChild(event.target);
}

function getData() {
```

```
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
        //Push the data in array
    var date = new Date().toLocaleDateString();
    var time = new Date().toLocaleTimeString();
    var filedata = String(this.responseText);
    var s1 = filedata.indexOf('#');
    var s2 = filedata.indexOf('%');
    var l = filedata.length;
    var IR1 = filedata.substring(0,s1);
    var IR2 = filedata.substring(s1+1,s2);
    var US = filedata.substring(s2+1,l);

        valuesUS.push(US);
        timeStamp.push(date+" "+time);
        showGraph();    //Update Graphs
    //Update Data Table
      var table = document.getElementById("dataTable");
      var row = table.insertRow(1);    //Add after headings
      entry = entry+"TIME:"+time+" "+"IR1:"+IR1+" "+"IR2:"+IR2+"
"+"US:"+US+"\r\n";
      var cell1 = row.insertCell(0);
      var cell2 = row.insertCell(1);
      var cell3 = row.insertCell(2);
      var cell4 = row.insertCell(3);
      cell1.innerHTML = date+"   "+time;
      cell2.innerHTML = IR1;
      cell3.innerHTML = IR2;
      cell4.innerHTML = US;
        }
  };
  xhttp.open("GET", "readADC", true);    //Handle readADC server
on ESP8266
  xhttp.send();
}
</script>
```

```
</body>
</html>
)=====";
```

Step 11: Copy and paste this code in the Arduino sketch area.

(You can also get the code from here : CODE)

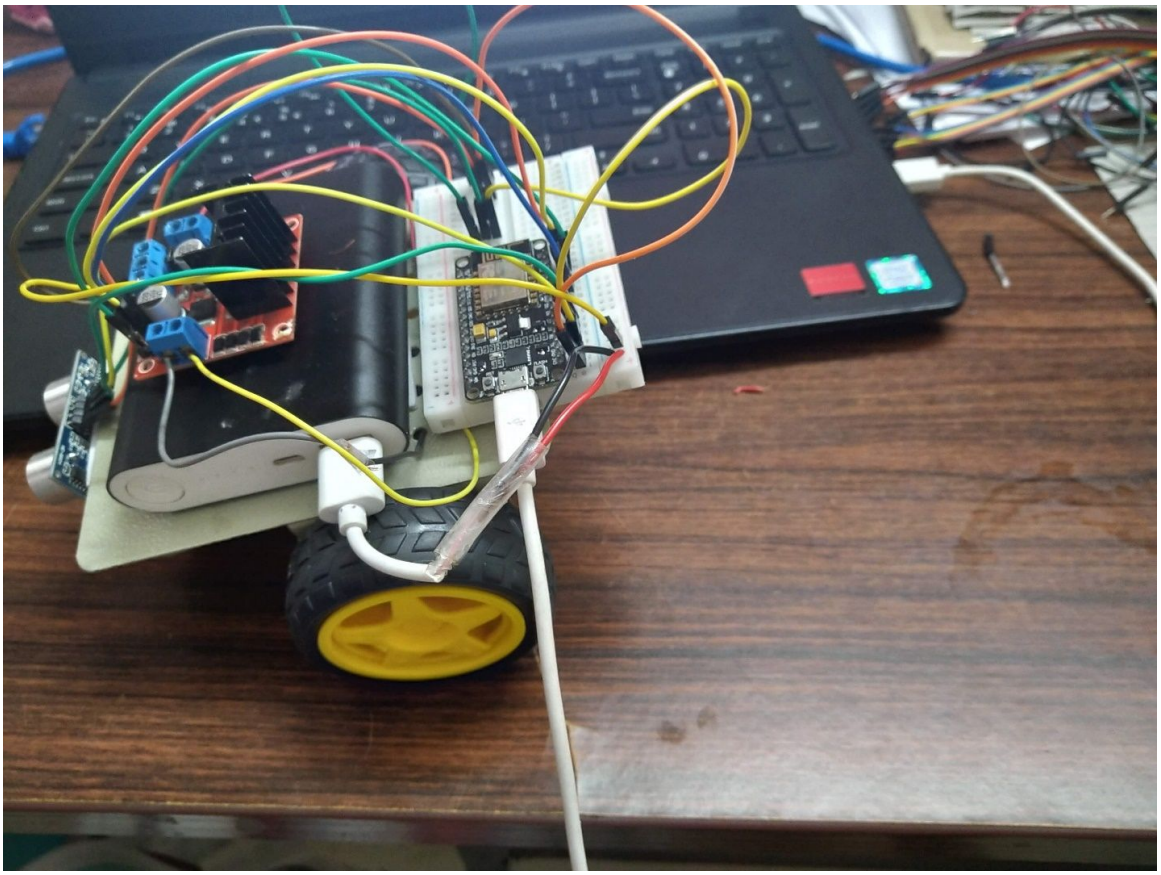
```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include "index.h" //HTML webpage to show Real Time Graphs

#define EN1 D1
#define EN2 D2
#define TRIG D6    //Defining pins
#define ECHO D7

const char* SSID = "YourHotspotNameWillComeHere";
const char* PASSWORD = "YourHotspotPasswordWillComeHere";
int lms = 1023, rms = 1023;
String data = "";

WiFiClient client;
WiFiServer server1(85);
ESP8266WebServer server(80); //Server for Real Time WebPage

void handleRoot() {
  String s = MAIN_page; //Read HTML contents
  server.send(200, "text/html", s); //Loads Real Time WebPage
}

int ultrasonic() { //Finding distance using Ultrasonic Sensor
  digitalWrite(TRIG, HIGH);
```

```
  delayMicroseconds(2);
  digitalWrite(TRIG, LOW);
  delayMicroseconds(10);
  int duration = pulseIn(ECHO, HIGH);
  int distance = duration / 34 / 2;
  Serial.println(distance);
  return distance;
}
String speed_() { //Returns a string containing both speeds
  int LS = map(lms, 400, 1023, 15, 150);
  int RS = map(rms, 400, 1023, 15, 150);
  return String(LS) + "#" + String(RS) + "%";
}

void handleADC() { //Handling web requests from WebPage
  int a = ultrasonic();
  String iR = speed_();
  String adcValue = iR + String(a);
  server.send(200, "text/plane", adcValue); //data for web page
}

String checkClient (void) //Checks for any new requests
{
  while (!client.available()) delay(1);
  String request = client.readStringUntil('\r');
  Serial.println(request);
  client.flush();
  return request;
}

void setup(void) {
  Serial.begin(115200);
  pinMode(EN1,OUTPUT);
  pinMode(EN2,OUTPUT);
```

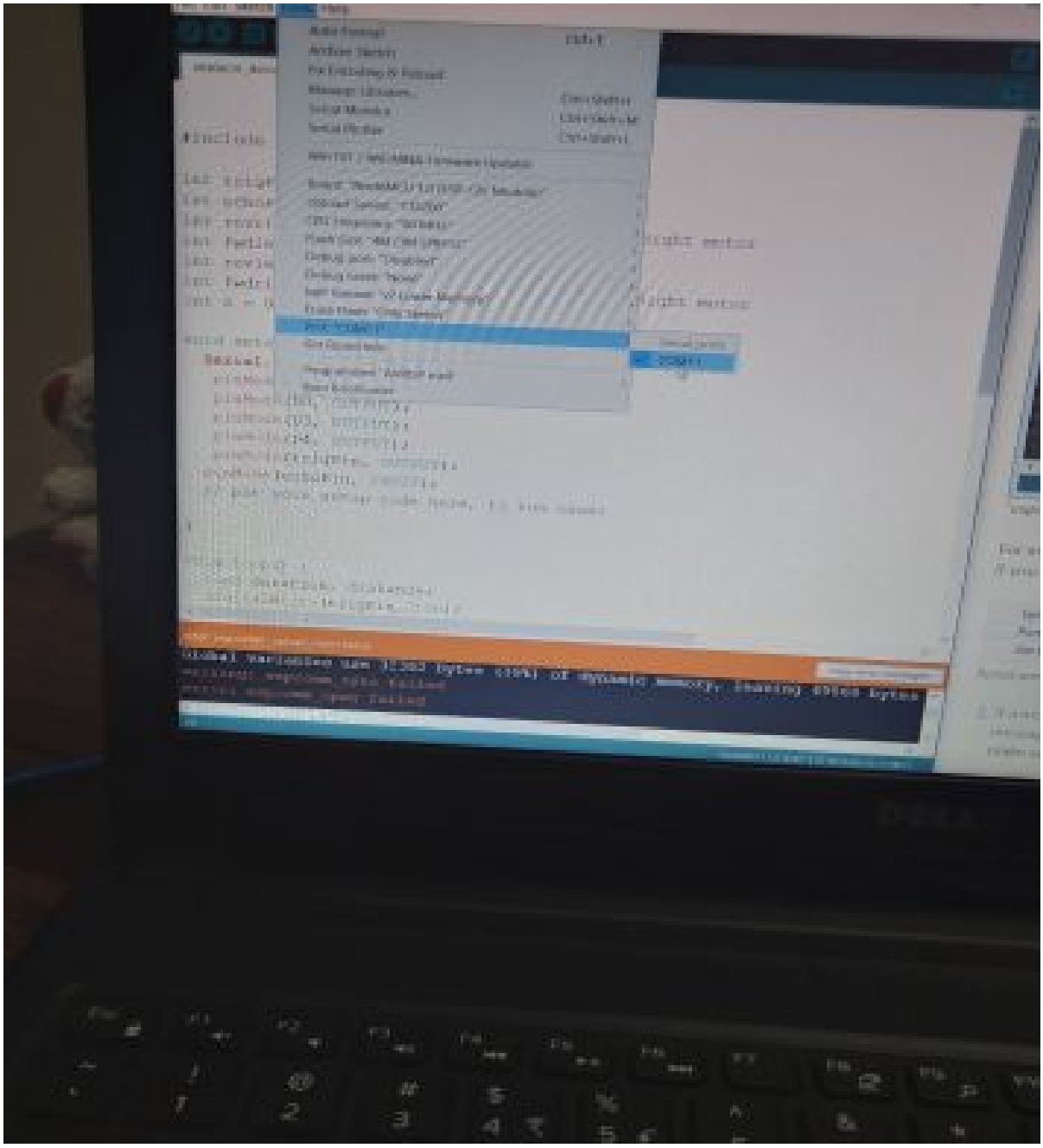
```
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
  WiFi.begin(SSID, PASSWORD);         //Connect to your Device
  Serial.println("");
  server1.begin();
   // Waiting for connection
  while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
  }
  //If connection successful show IP address in serial monitor
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());  //IP address assigned to your ESP
  server.on("/", handleRoot);  //Which routine to handle at root
location. This is display page
  server.on("/readADC", handleADC);
//This page is called by java Script AJAX
  server.begin(); //Start server
  Serial.println("HTTP server started");
}

void loop(void) {
  server.handleClient();
  client = server1.available();
  if (!client) {
        return;
  }
  data = checkClient (); //Handle client requests
  if (data.indexOf("forward") != -1)
  {
        lms = 1023;
```

```
        rms = 1023;
  }
  else if (data.indexOf("backward") != -1)
{
  if (lms >= 500 && rms >= 500 && lms == rms)
        {
        lms = lms - 100;
        rms = rms - 100;
        }
else if (rms > lms)
        lms -= 100;
else if (lms > rms)
        rms -= 100;
}
  else if (data.indexOf("left") != -1)
{
  lms = 800;
  rms = 1023;
}
else if (data.indexOf("right") != -1)
{
  rms = 800;
  lms = 1023;
}
else if (data.indexOf("grip") != -1)
{
  rms = 0;
  lms = 0;
}
analogWrite(EN1,rms);
analogWrite(EN2,lms);
}
```

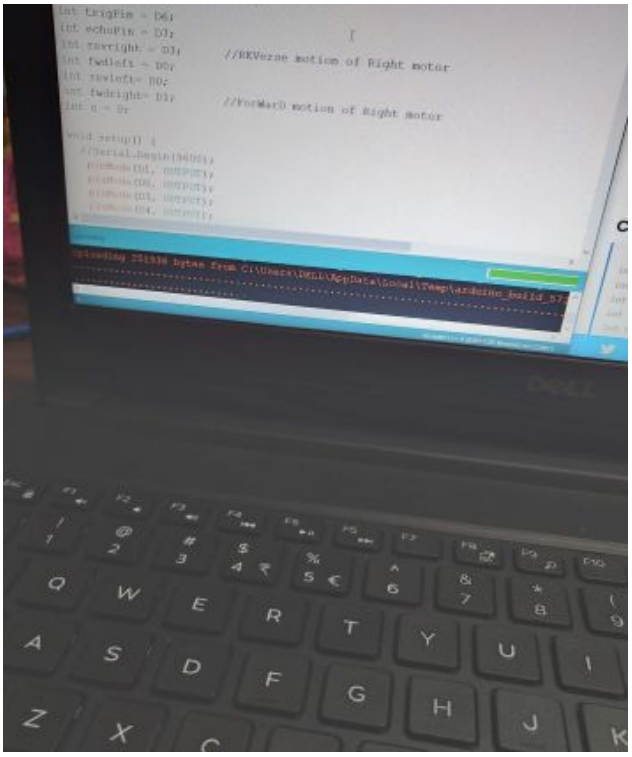
Step 12: Now connect the Node MCU to your computer using USB Cable .

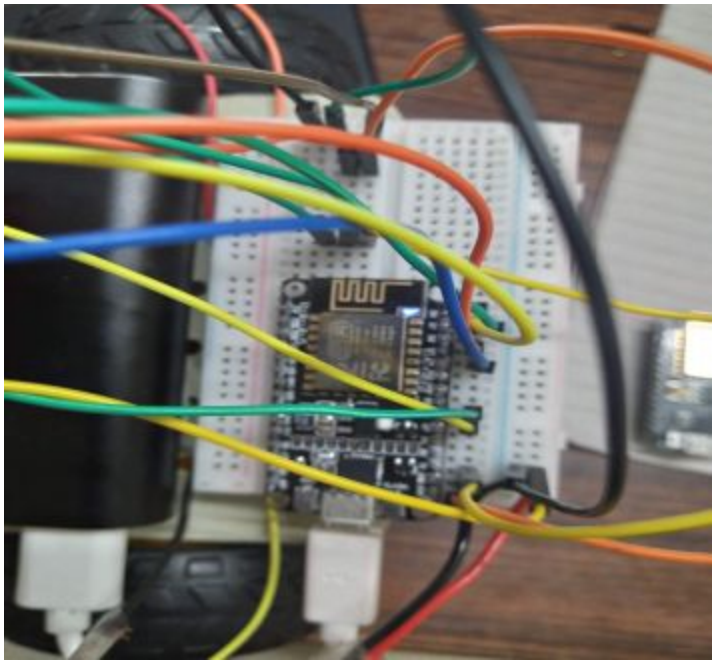


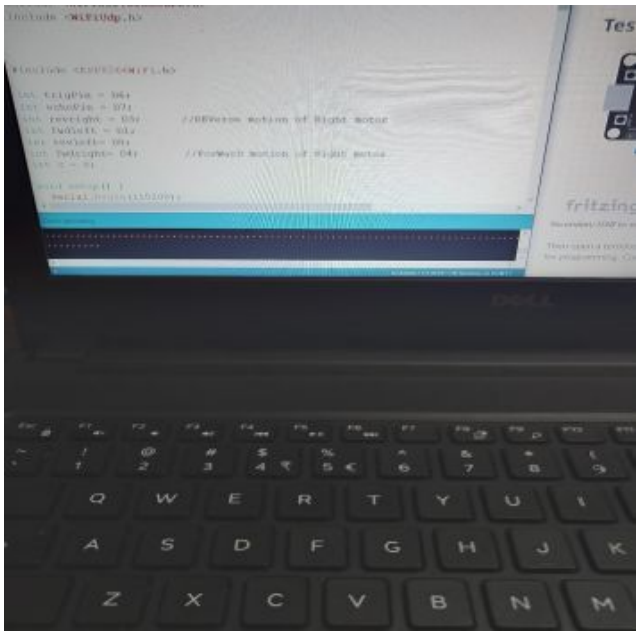
Step 13: Select NODEMCU 1.0(ESP-12E Module) as your board in the Board Manager (Arduino IDE >> Tools >> Boards Manager). Also don't forget to select the port at which Node MCU is connected.



Step 16: Click on the “Tick Sign” to verify/compile the code and the “Right Arrow” to upload the code on the board.
(The builtin LED which is just beside D0 will start blinking when code is uploading, and after the code has been uploaded a message will be displayed)

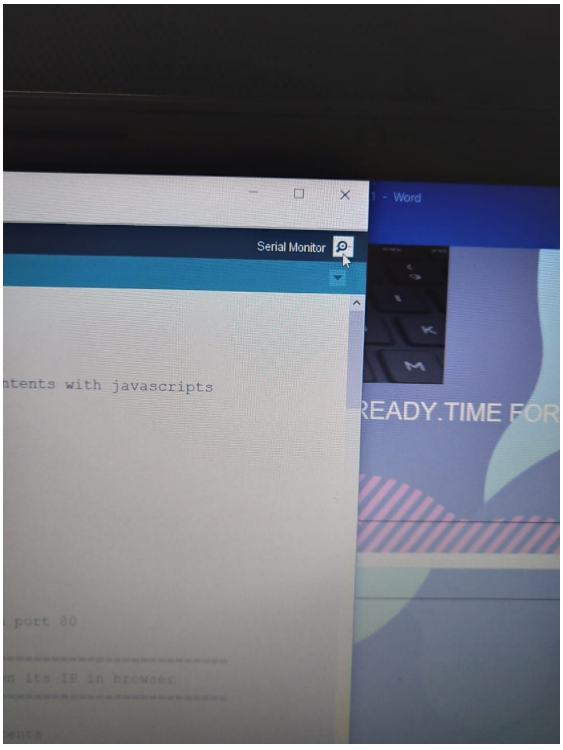




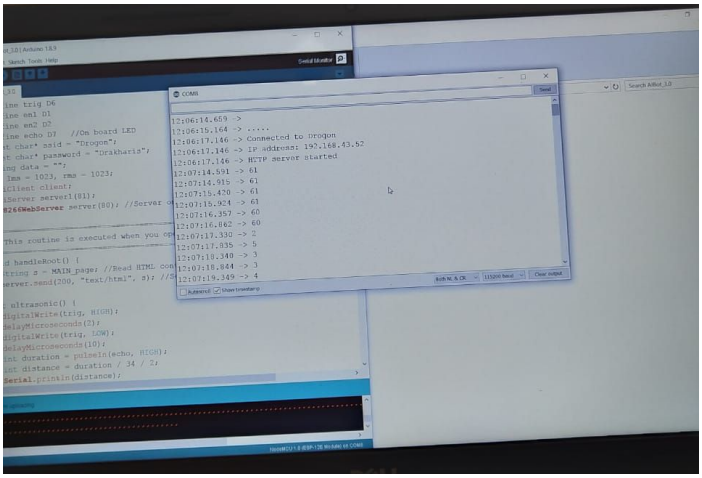


TRAINING BOT

Step 1: Open the serial monitor by pressing ‘Ctrl+Shift+M’ or by clicking on the ‘magnifying icon’ on the top right corner of the arduino ide.

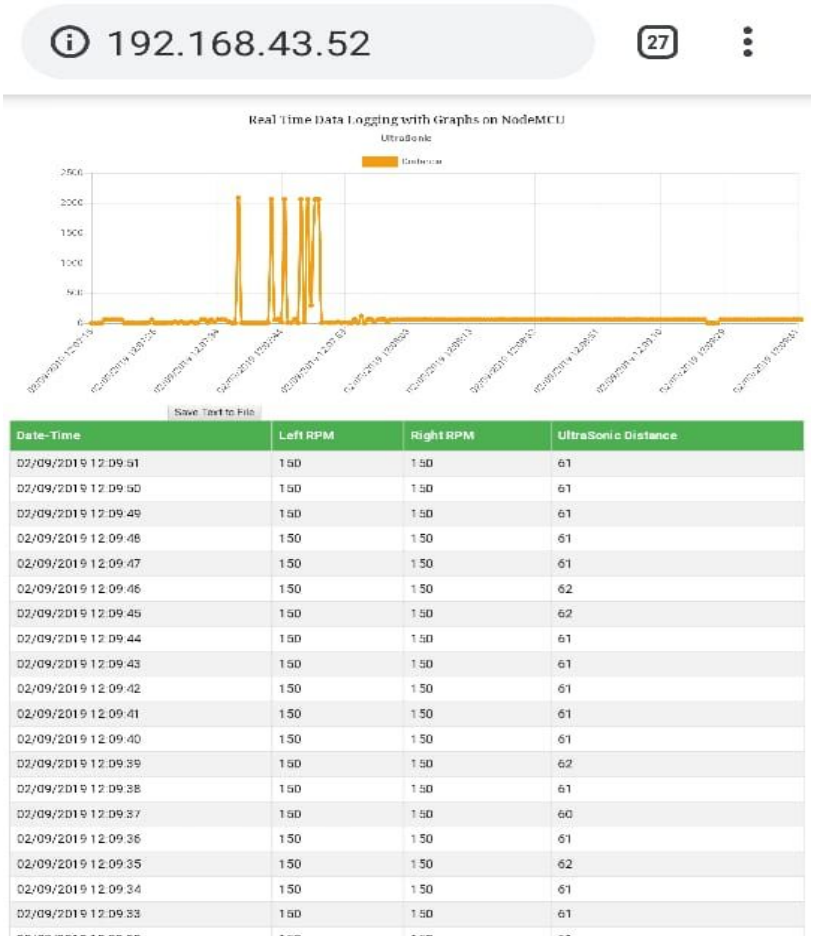


Step 2: Set the baud rate as 115200 and turn ON your mobile hotspot. Your serial monitor will show you the IP Address of your Node MCU once it is connected to your hotspot. After that distance as calculated by Ultrasonic Sensor will start flooding.



Step 3: Once the Node MCU is connected to your mobile hotspot, you can control your bot using the Mobile App provided to you.

Step 4: To view the real time graph, you must open Google Chrome and type in the IP Address of your Node MCU like "http://192.168.43.43:80".



Date-Time	Left RPM	Right RPM	UltraSonic Distance
02/09/2019 12:09:51	150	150	61
02/09/2019 12:09:50	150	150	61
02/09/2019 12:09:49	150	150	61
02/09/2019 12:09:48	150	150	61
02/09/2019 12:09:47	150	150	61
02/09/2019 12:09:46	150	150	62
02/09/2019 12:09:45	150	150	62
02/09/2019 12:09:44	150	150	61
02/09/2019 12:09:43	150	150	61
02/09/2019 12:09:42	150	150	61
02/09/2019 12:09:41	150	150	61
02/09/2019 12:09:40	150	150	61
02/09/2019 12:09:39	150	150	62
02/09/2019 12:09:38	150	150	61
02/09/2019 12:09:37	150	150	60
02/09/2019 12:09:36	150	150	61
02/09/2019 12:09:35	150	150	62
02/09/2019 12:09:34	150	150	61
02/09/2019 12:09:33	150	150	61

Step 5: Using the Mobile App, train your bot to avoid the obstacle by running it on different courses. Once you have enough data, click on "Save Text to File" to download your own dataset.

FINISH