Embedded Systems Project
Vehicle Speed Management System

Names:	**Ediga Ajay Goud (B21EC010)**
	Bingi Rakesh(B21EC007)
	Mutyala Kalyani(B21EC015)
	Chethelli Ranjith(B21EC025)
Course Code:	**EC311**
performed on:	**20-09-2023 to 20-11-2023**
Date of Sub:	**2023.11.24**
Instructor:	**Dr. Shubhankar Majumdar**

1 Objective

The objective of implementing a vehicle speed management system using Arduino and ultrasonic sensors is to promote road safety and improve traffic management. Here are the primary objectives of such a system:

1.1 Speed Monitoring:

To monitor and measure the speed of vehicles passing through a specific location on the road.

1.2 Speed Enforcement:

To ensure that vehicles adhere to the prescribed speed limits in a designated area, contributing to safer road conditions.

1.3 Data Collection:

To collect data on vehicle speeds for analysis, reporting, and potential enforcement purposes. This data can be used for traffic studies and identifying speeding patterns.

1.4 Safety Awareness:

To raise awareness among drivers about their speed and the importance of adhering to speed limits, ultimately reducing the likelihood of accidents.

1.5 Traffic Control:

To help authorities manage traffic flow and congestion by identifying areas with high-speed violations and deploying resources as needed.

1.6 Customizable Alerts:

To provide real-time alerts or warnings to drivers who exceed the speed limit, encouraging them to slow down and drive safely.

1.7 Compliance Monitoring:

To support law enforcement efforts in monitoring and enforcing speed limits and regulations in specific zones.

1.8 Research and Analysis:

To gather data for research purposes, enabling the study of traffic behavior and the effectiveness of speed management measures.

1.9 Education and Public Safety:

To educate the public about the dangers of speeding and the benefits of adhering to speed limits, contributing to overall road safety.

1.10 Environmental Impact:

To potentially reduce fuel consumption and emissions by promoting smoother traffic flow and fewer abrupt accelerations and deaccelerations due to speeding.

By achieving these objectives, a vehicle speed management system can play a crucial role in reducing accidents, improving road safety, and creating a more efficient and orderly traffic environment. It can be particularly useful in areas where speeding is a common issue or where strict speed enforcement is necessary. In this project we have designed a bot car using ARDUINO UNO ,along with ultrasonic sensor,motors,battery and jumper wires.

2 Theory

Developing a robot car for vehicle speed management involves creating a self-contained, autonomous vehicle that can monitor and enforce speed limits or assist in speed management tasks. Here's a high-level overview of the components and steps you might consider when building such a bot car:

Hardware Components Required :

ARDUINO UNO :

Arduino Uno is an open-source microcontroller board based on the processor ATmega328P. It has 14 digital input/output pins(of which 6 can be used as PWM outputs),6 analog inputs,a USB connection,a power jack,an ICSP header and a reset button.It contains everything needed to support the microcontroller.Just plug it into a computer with a USB cable or power it with an adapter to get started.

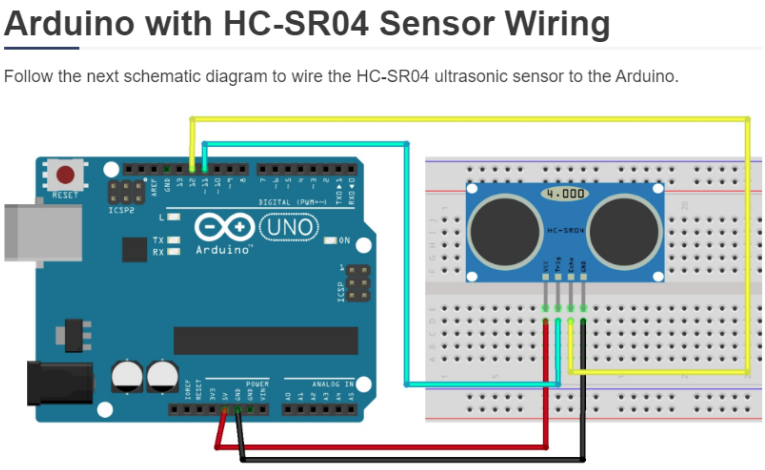


Figure 1: Arduino

ULTRASONIC SENSOR(HC-SR04):

HC-SR04 is an ultrasonic distance sensor used for measuring the distance at which an object is located.The principle used by this sensor is called SONAR. [h]

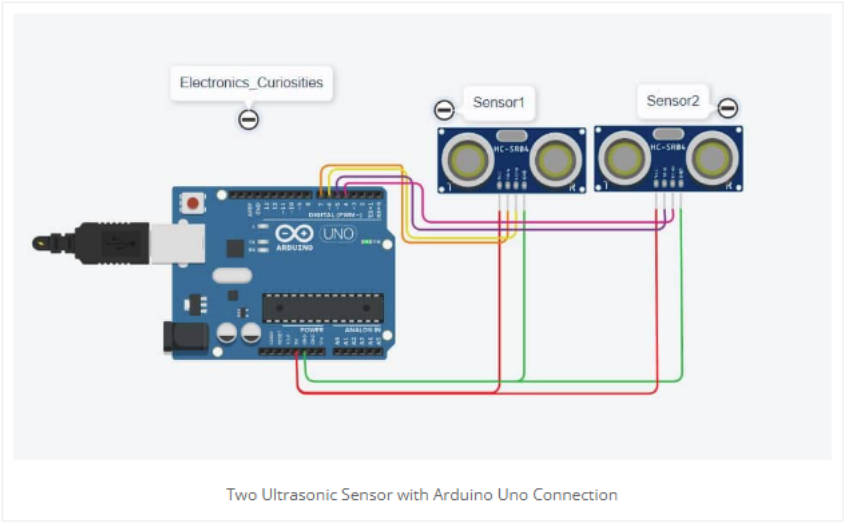
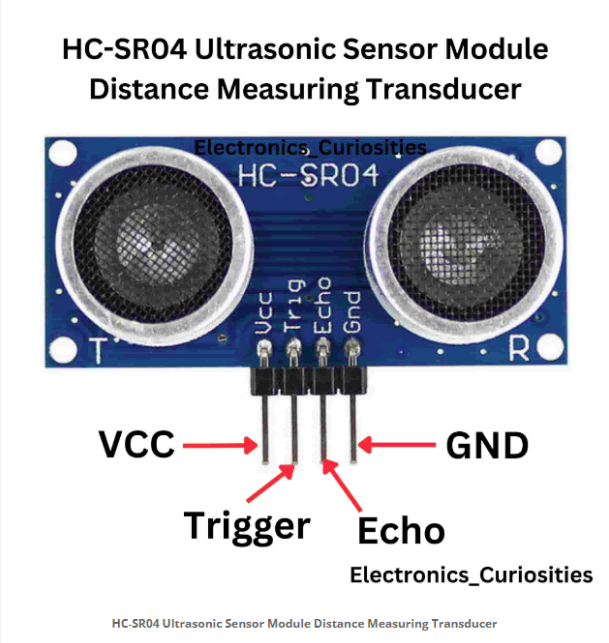


Figure 2: Arduino Interfacing with ultra sonic sensor

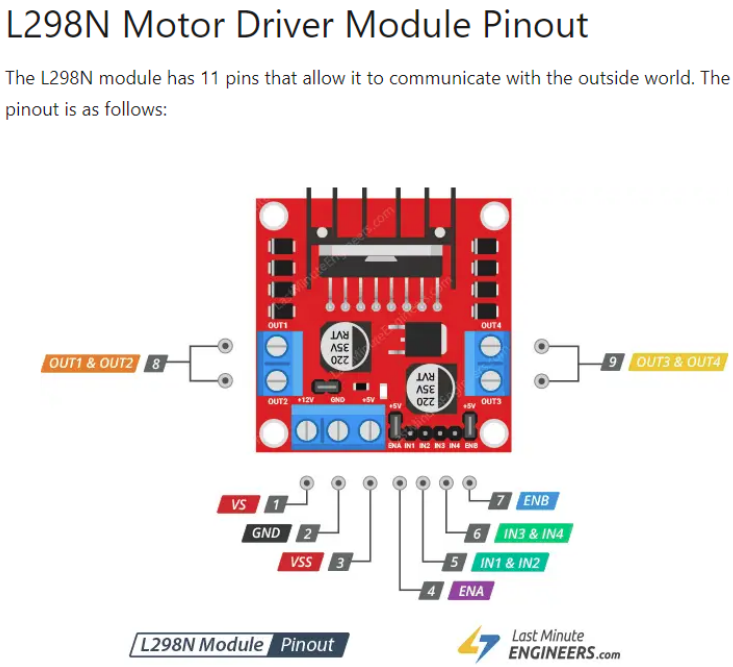


It is perfect for small robotics projects such as obstacle avoiding robot,distance measuring device etc.It has two parts,one emits the ultrasound sonar to measure the distance to an object.The other part is the receiver which listens for the echo.As soon as the ultrasound hits the object it bounces back and is detected by the receiver.The time taken for the wave to come back decides the distance of the object being measured.

Motors:
The motor is such an electric device that transforms electric power into mechanical power. The working of these motors depends on the interaction of the field at the stator with the flux generated

by the current armature windings at the rotor.The input to the motor can be provided according to their types if they are dc motor then input will be provided with the battery, rectifiers and if the motor is ac then its input will come from the ac power source, inverter, and ac generator induction generator or synchronous generator. Classifications of motors can according to input supply like AC or DC source, with that it can be classified with their internal structure, practical implementations, etc. With AC or DC motors, there are other types of electrical motors like brush motor, brushless motor, single-phase motor, 2 phase motor, 3 phase motor, etc.

L298 Motor Driver:
This L298 Based Motor Driver Module is a high power motor driver perfect for driving DC Motors and Stepper Motors. It uses the popular L298 motor driver IC and has the onboard 5V regulator which it can supply to an external circuit. It can control up to 4 DC motors, or 2 DC motors with directional and speed control This motor driver is perfect for robotics and mechatronics projects and perfect for controlling motors from microcontrollers, switches, relays, etc. Perfect for driving DC and Stepper motors for micro mouse, line following robots, robot arms, etc. An H-Bridge is a circuit that can drive a current in either polarity and be controlled by Pulse Width Modulation (PWM).
[h]



RECHARGEABLE BATTERY:

A rechargeable battery,storage battery,or secondary cell(formally a type of energy accumulator) is a type of electrical battery which can be charged,discharged into a load, and recharged many times,as opposed to a disposable or primary battery,which is supplied fully charged .

JUMPER WIRE:

A jumper is a tiny metal connector that is used to close or open part of an electrical circuit.It may be used as an alternative to a dual in-line package(DIP) switch.A jumper has two or more connecting points,which regulate an electrical circuit board.

SOFTWARE REQUIRED :

ARDUINO IDE:

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

3 CIRCUIT CONNECTION

:

- The circuit diagram for vehicle speed management (Figure-1) contains main components: Arduino Uno, Power supply ,motors,motor driver and an ultrasonic sensor .
- The Ultrasonic sensor HC-SR04 pins echo and trig are connected to Arduino Uno pins 5 and 6 respectively. The VCC pin is connected to 5V on Arduino Uno and both the grounds are connected together.
- then connect the motor driver's IN1 and IN2 pins to two digital output pins on the Arduino (e.g., pins 4 and 5) and the motor driver's OUT1 and OUT2 pins to the terminals of the DC motor.later on connect the motor driver's ENABLEA (ENA) pin to a PWM-enabled digital output pin on the Arduino (e.g., pin 3) and the motor driver's VCC and GND to the Arduino's 5V and GND respectively.An external power supply (e.g., a battery) is connected to the motor driver's motor power supply pins (usually labeled VMOT or similar) to power the motor. Make sure to connect the ground of the external power supply to the Arduino's ground as well.
- A 9v battery has been connected to Vin pin on the Arduino Uno and grounds are connected together.

4 WORKING CONCEPT

Ultrasonic sensors work by sending out high-frequency sound waves and measuring the time it takes for the waves to bounce back after hitting an obstacle. The robot can calculate the distance to the obstacle using the speed of sound (approximately 343cm per second). In order to detect obstacle in front of robot you have to place once sensor on the front side but now you can't detect any object present on left or right side of your robot, so you have to place two sensors one on the left side of robot and one on the right side so in this project you need to use total three ultrasonic sensors, one on the front, one on left and one on right side of robot. The robot's

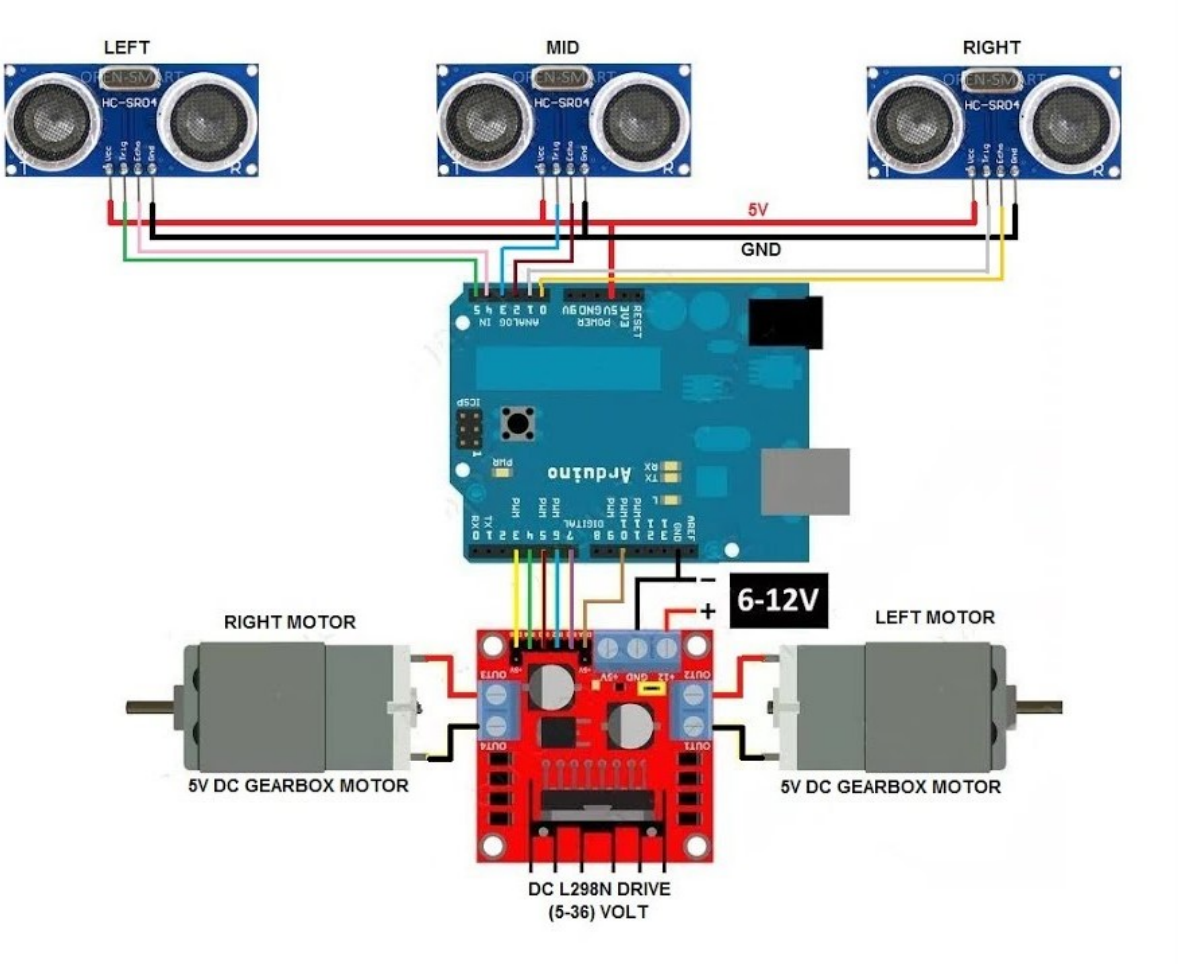


Figure 3: Circuit Diagram

microcontroller continuously triggers each ultrasonic sensor to send out a pulse and records the time it takes for the pulse to return (i.e., echo time) for each sensor. By analyzing the echo times from all three sensors, the robot can determine the distances to obstacles on its left, center, and right. It can identify the direction of the clearest path based on these measurements. The microcontroller processes the sensor data and decides on the robot's next move. If there is an obstacle detected in the center, the robot will turn in the direction with the most unobstructed path (left or right). The robot's motor drivers are used to control the speed and direction of the wheels. If the robot needs to turn, it adjusts the motor speeds on one side to make a controlled turn. For example, to turn left, it would decrease the speed of the right motor and increase the speed of the left motor. The robot repeats this process continuously, allowing it to navigate around obstacles and travel in the direction with the least obstruction. By setting enable A and Enable B to 80, speed of the bot is 30cm/sec.Similarly by setting enable A and enable B to 60, speed of the bot is 20cm/sec.

5 PROGRAM CODE

```
int enA = 3;
int in1 = 5;
int in2 = 6;
// Motor B connections
int enB = 11;
int in3 = 9;
int in4 = 10;
const int trigPin1 = 2; // Trigger pin for sensor 1
const int echoPin1 = 4; // Echo pin for sensor 1
const int trigPin2 = 8; // Trigger pin for sensor 2
const int echoPin2 = 12; // Echo pin for sensor 2
const int trigPin3 = 7; // Trigger pin for sensor 3
const int echoPin3 = 13; // Echo pin for sensor 3
long duration1, distance1, duration2, distance2,duration3, distance3,height,width,obj_size;
voidsetup()
   Serial.begin(9600);
pinMode(enA, OUTPUT);
pinMode(enB, OUTPUT);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
pinMode(in3, OUTPUT);
pinMode(in4, OUTPUT);
pinMode(trigPin1, OUTPUT);
pinMode(echoPin1, INPUT);
pinMode(trigPin2, OUTPUT);
pinMode(echoPin2, INPUT);
pinMode(trigPin3, OUTPUT);
pinMode(echoPin3, INPUT);

   void loop()

int centerSensor = sensorThree();
int leftSensor = sensorOne();
int rightSensor = sensorTwo();

if ( 30 ≥ centerSensor)

   analogWrite(enA, 60);
analogWrite(enB, 60);
```

```
digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);
digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);
   // Check the distance using the IF condition
else if ((30≥ rightSensor)||(30 ≥ leftSensor))
   Stop();
backward();
delay(1);
Serial.println("Stop");
delay(1000);

if (leftSensor ¿ rightSensor)

   right();
Serial.println("Right");
delay(1000);

else

   left();
Serial.println("Left");
delay(1000);

   Serial.println("Forward");
forward();

//Get the sensor values

   int sensorOne()

//pulse output

digitalWrite(trigPin1, LOW);
delayMicroseconds(2);
digitalWrite(trigPin1, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin1, LOW);
duration1 = pulseIn(echoPin1, HIGH);
distance1 = (duration1 / 2) / 29.1;
return distance1; // Return the values from the
```

sensor

```
   //Get the sensor values
int sensorTwo()

//pulse output
digitalWrite(trigPin2, LOW);
delayMicroseconds(2);
digitalWrite(trigPin2, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin2, LOW);
duration2 = pulseIn(echoPin2, HIGH);
distance2 = (duration2 / 2) / 29.1;
return distance2; // Return the values from the sensor


   //Get the sensor values
int sensorThree()
//pulse output
digitalWrite(trigPin3, LOW);
delayMicroseconds(2);
digitalWrite(trigPin3, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin3, LOW);
duration3 = pulseIn(echoPin3, HIGH);
distance3 = (duration3 / 2) / 29.1;

   return distance3; // Return the values from the sensor


void forward()
analogWrite(enA, 80);
analogWrite(enB, 80);
digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);
digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);

void backward()
analogWrite(enA, 80);
```

```
analogWrite(enB, 80);
digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);
digitalWrite(in3, HIGH);
digitalWrite(in4, LOW);

void left()
analogWrite(enA, 80);
analogWrite(enB, 80);
digitalWrite(in2,HIGH);

void right()
analogWrite(enA, 80);
analogWrite(enB, 80);
digitalWrite(in4,HIGH);

void Stop()
analogWrite(enA, 0);
analogWrite(enB, 0);
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
digitalWrite(in3, LOW);
digitalWrite(in4, LOW);
```

[h]

```
int enA = 3;
int in1 = 5;
int in2 = 6;
// Motor B connections
int enB = 11;
int in3 = 9;
int in4 = 10;
const int trigPin1 = 2; // Trigger pin for sensor 1
const int echoPin1 = 4; // Echo pin for  sensor 1
const int trigPin2 = 8; // Trigger pin for sensor 2
const int echoPin2 = 12; // Echo pin for sensor 2
const int trigPin3 = 7; // Trigger pin for sensor 3
const int echoPin3 = 13; // Echo pin for sensor 3
long duration1, distance1, duration2, distance2,duration3, distance3,height,width,obj_size;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
    pinMode(enA, OUTPUT);
    pinMode(enB, OUTPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(in3, OUTPUT);
    pinMode(in4, OUTPUT);
    pinMode(trigPin1, OUTPUT);
    pinMode(echoPin1, INPUT);
    pinMode(trigPin2, OUTPUT);
    pinMode(echoPin2, INPUT);
    pinMode(trigPin3, OUTPUT);
  pinMode(echoPin3, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  // Sensor 1

   int centerSensor = sensorThree();
  int leftSensor = sensorOne();
  int rightSensor =  sensorTwo();
  if ( 30 >= centerSensor )
  {
    analogWrite(enA, 60);
     analogWrite(enB, 60);
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);

  }
// Check the distance using the IF condition
 else if ((30 >= rightSensor) || (30 >= leftSensor) ) {
    Stop();
    backward();
    delay(1);
    Serial.println("Stop");
    delay(1000);

    if (leftSensor > rightSensor) {
      right();
      Serial.println("Right");
      delay(1000);
    } else {
      left();
      Serial.println("Left");
      delay(1000);
    }
```

[h]

```
  }
  Serial.println("Forward");
  forward();
}
//Get the sensor values
int sensorOne() {
  //pulse output
  digitalWrite(trigPin1, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin1, LOW);
  duration1 = pulseIn(echoPin1, HIGH);
  distance1 = (duration1 / 2) / 29.1;
  return distance1; // Return the values from the sensor
}

//Get the sensor values
int sensorTwo() {
  //pulse output
  digitalWrite(trigPin2, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin2, LOW);
  duration2 = pulseIn(echoPin2, HIGH);
  distance2 = (duration2 / 2) / 29.1;
  return distance2; // Return the values from the sensor
}

//Get the sensor values
int sensorThree() {
  //pulse output
  digitalWrite(trigPin3, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin3, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin3, LOW);
  duration3 = pulseIn(echoPin3, HIGH);
  distance3 = (duration3 / 2) / 29.1;

  return distance3; // Return the values from the sensor
}
```

[h]

```
110  void forward() {
111   analogWrite(enA, 80);
112      analogWrite(enB, 80);
113      digitalWrite(in1, LOW);
114      digitalWrite(in2, HIGH);
115      digitalWrite(in3, LOW);
116      digitalWrite(in4, HIGH);
117  }
118  void backward(){
119      analogWrite(enA, 80);
120      analogWrite(enB, 80);
121      digitalWrite(in1, HIGH);
122      digitalWrite(in2, LOW);
123      digitalWrite(in3, HIGH);
124      digitalWrite(in4, LOW);
125  }
126  void left() {
127    analogWrite(enA, 80);
128      analogWrite(enB, 80);
129   digitalWrite(in2,HIGH);
130  }
131  void right() {
132     analogWrite(enA, 80);
133      analogWrite(enB, 80);
134    digitalWrite(in4,HIGH);
135  }
136  void Stop() {
137     analogWrite(enA, 0);
138     analogWrite(enB, 0);
139      digitalWrite(in1, LOW);
140      digitalWrite(in2, LOW);
141      digitalWrite(in3, LOW);
142      digitalWrite(in4, LOW);
143  }
```

6 Advantages

- Enhanced road safety by enforcing speed limits.
- Consistent and impartial enforcement.
- Continuous operation (24/7).

- Reduced workload for human law enforcement.
- Data collection for traffic analysis.
- Cost-effective over the long term.
- Raises public awareness of responsible driving.
- Precise and accurate speed measurements.
- Real-time alerts for drivers.
- Potential environmental benefits.
- Flexibility to target specific areas.
- Remote operation for safety.

7 Reference

- Chatgpt.
- Wikipedia.
- Youtube.
- Stackoverflow.
- Electronics curiosities.
- Last minute engineers.

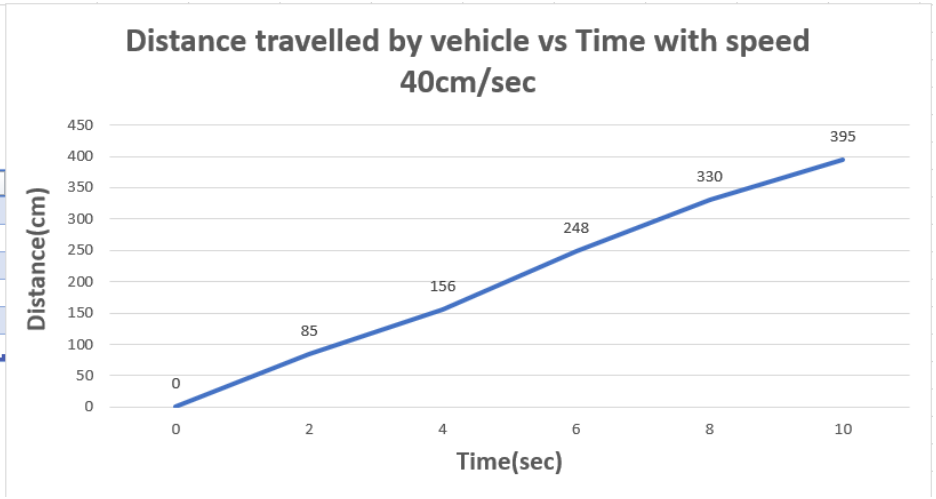
8 Result

- The minimum height of the obstacle required to be detected by the ultrasonic sensors of the bot is 9 cm.
- The minimum width of the obstacle required to be detected by the ultrasonic sensors of the bot is 1 cm.
- 180 cm is the maximum distance and 3cm is the minimum distance,the obstacle can be placed for the bot's ultrasonic sensors to detect it.
- By setting Enable A and Enable B to 100 rev/min, speed of the bot is 40cm/sec.

9 Related Graphs

Distance from the obstacle	width of the ob
3	1cm - 1.5cm
6	2cm-3cm
12	3cm-5cm
32	5cm-15cm
40	>15cm

Time	Distance
0	0
2	85
4	156
6	248
8	330
10	395



10 SUMMARY

After wiring and attaching all the devices and setting up the robot for vehicle speed managment,we observe all the important setup whether they are well connected or something missed. After connection set up ,now next step is to submit/upload code in Arduino and supply power to the circuit.When system is powered ON,Arduino keeps monitoring for any things that come near the sensor at given range and the bot will move with a speed of 30cm/sec.When ultrasonic sensor detect any object for example any box,then Arduino calculates its distance and if it is less than a certain predefines value than the bot will turn either left or right. The microcontroller processes the sensor data and decides on the robot's next move. If there is an obstacle detected in the center, the robot will turn in the direction with the most unobstructed path (left or right). The robot repeats this process continuously, allowing it to navigate around obstacles and travel in the direction with the least obstruction.

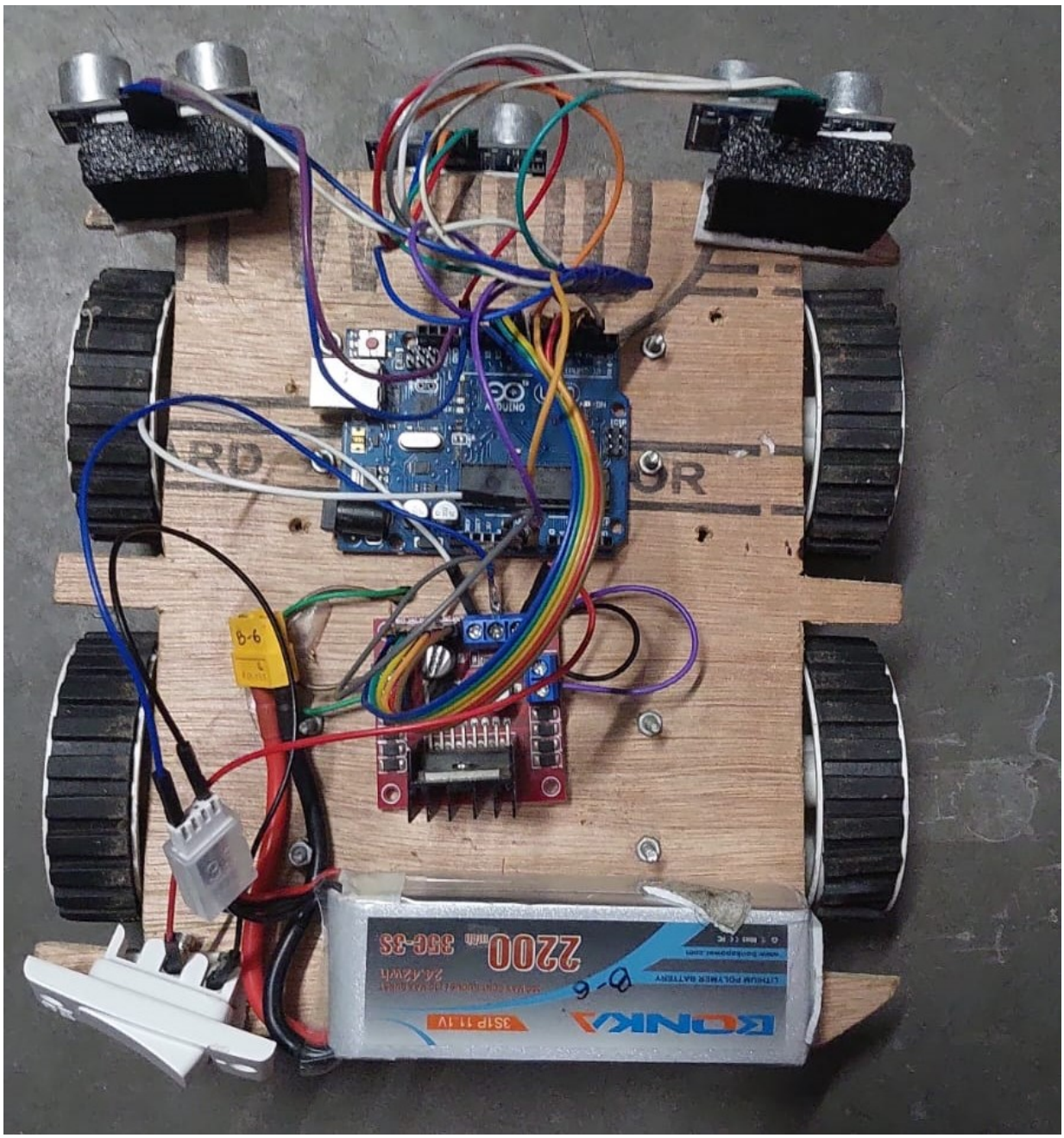


Figure 4: bot image