Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Microprocessor and Interfacing J-Component

Collision Avoidance System for Vehicles

Final Report

TEAM MEMBERS:

SHASHWAT CHAUDHARY -19BCE0056

VARUN VEERBJADRA – 19BCB0121

DHARUN KARTHICK - 19BCE2316

ANKIT MAHTO - 19BCE2601

PRITI SRI S – 19BCE2433

Submitted to Prof. Saranya K.C.

Table of Contents

OBJECTIVE

The conventional sense-and-avoid collision avoidance mode of vehicles lack applicability and timeliness in a multi-threat environment. In this project, we are proposing a new efficient collision avoidance approach. The proposed collision avoidance pattern consists of a sensory layer, a logic layer and a development layer. The threat information is sensed using the sensory layer, and the path planning approach in the logical layer is applied to the output configuration of vehicle. In the implementation phase, the command is executed by matching the sensing information and action base.

INTRODUCTION

This system, very simply put, is designed to prevent or reduce the severity of a collision. In its very basic form, this system all the sensor data are collected all the once, and then according to the data collected our vehicle basically moves around. It is a small-scale application to show how we can deal with collisions in our own effective way. It can move right, left, forward, take a U-turn or wait for some time to decide what to do. We are testing it in different scenarios to prove whether our designed system is working up to the level we want. The working behind the scenes is all the sensors' data. We set a constant minimum distance to have our vehicle avoid collision. If sensor 1 data is greater than the minimum distance than the minimum distance then it will just go forward. If it is not the case then it will reverse a little and will take sensor 2 data. If sensor 2 data is greater than minimum distance then it will go right wait for some and forward again. If sensor 3 data is greater than minimum distance then it will go left wait for some time and forward again. If none of these are true then it will U turn and then take a forward path.

LITERNATURE SURVEY

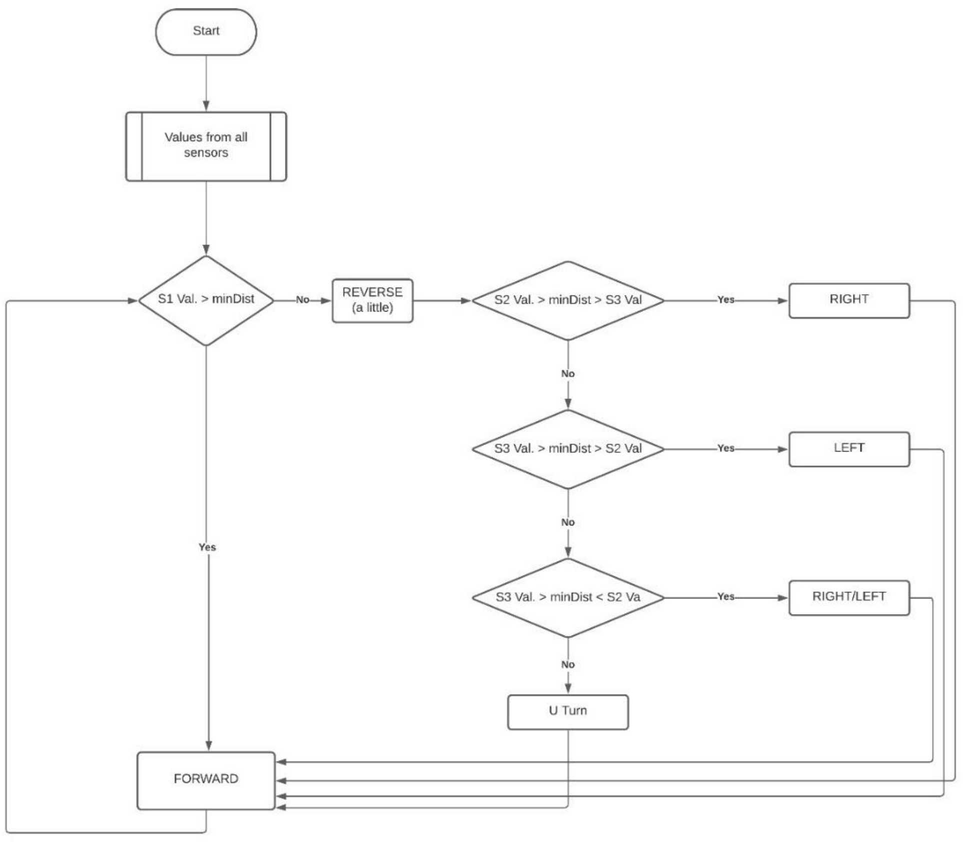
S. No	Title	Year of Publication	Journal Name	Technique	Disadvantages
1.	Binary Ostensibly-Implicit Trees for Fast Collision Detection	2020	Research Gate	Binary tree tranversal	If failure of capturing spatial structure occurs lots of storage issue occurs
2.	Fast collision detection between high resolution polygonal models	2019	IEEE	EBP-Octree	application to deformable models, and the need of rebuilding complete subtrees 472 bounding volumes in real time
3.	Design and Optimization of a Joint Torque Sensor for Robot Collision Detection	2019	IEEE	Sensor signal processing based on Wheatstone full bridge	Sensor signal processing based on Wheatstone full bridge
4.	A Novel Real-time Collision Avoidance System for On-road Vehicles	2018	IEEE	sensor based embedded system	there are still some more critical situations like how safe it is to overtake the front side vehicle or to take U-turn on the two-lane road, which have not beenconsidered by the collision avoidance system
5.	Research on Collision Avoidance Method Based	2019	IEEE	Infrared Image Recognition and Stereo Vision using OCSVM (Support Vector Machine)	Image recognition technology applied to unmanned surface vessel autonomous navigation and collision avoidance is important for improving the autonomy of unmanned surface vessel

S.No	Title	Year of Publication	Journal Name	Technique/algorithm used	Disadvantages
6.	Autonomous Navigation via Deep Reinforcement Learning for Resource Constraint Edge Nodes Using Transfer Learning	2019	IEEE	Deep Reinforcement Learning	Requires immense computation power to train the model and takes a lot of time too. Resources requirement is also pretty high.
7.	Aerial Robotics Competition Team Technical Paper	2019	KENNESAW STATE UNIVERSITY	Speech Recognition, Computer Vision and Machine Learning	Intense demand for GPU as it involves tensors.
8.	Autonomous Vision-based UAV Landing with Collision Avoidance using Deep Learning	2021	Cornell University	Computer Vision and Deep Learning	Energy consuming and time consuming
9.	A simple vision-based navigation and control strategy for autonomous drone racing	2021	Cornell University	Image and Video Processing, IoT	Components used are costly and architecture itself is very complex. Can be difficult to fix if an issue arises.
10.	Low Cost Autonomous UAV Swarm Application in Wildfire Surveillance and Suppression	2021	Association for Computing Machinery	Greedy Algorithm	It builds more weight on the drone and aerodynamics is affected as it needs an Jetson Nano on board. And Difficult to set up.

METHODOLOGY

The movement of the vehicle is dependent on the object in front of it. There are 3 ultrasonic sensors placed in the system. All the 3 sensors work together and the details to the microcontroller. The microcontroller decides which direction the vehicle should move and passes the command to the motor driver which rotates the motor accordingly. When there are no obstacles in all the 3 sides, the vehicle moves straight. When there is an obstacle in front of right sensor or left sensor or both the sensors but not in front of the middle sensor, the vehicle still continues to move straight. When there is obstacle in front of all the 3 sensors, the vehicle comes to stop, takes reverse and makes a u turn. When there is no obstacle in front of the right sensor but there are obstacles in front of left and middle, the vehicle turns to the right. Similarly, when there is no obstacle in front of left sensor but there are obstacles in front of middle and right sensor, the vehicle turns to the left. When there is an obstacle only in front of the middle sensor, the vehicle moves right.

METHOLOGY FLOWCHART



TECHNICAL REQUIREMENTS

❖ Arduino UNO R3:

Arduino is an open-source company that manufactures Arduino board and IDE. An Arduino is a microcontroller that can be used to program hard ware components in order to control them. In our project we have used Arduino UNO which has the ATmega328p. he board has 14 digital I/O pins, 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. The Arduino has 6 digital pins which are used to modulate the pulse width and they are known as PWM pins.

❖ Driving Modules:

○ Steering Style:

Small wheel Paired with Servo Arm(motor) which, has a precise angular rotation movement, gives model car bot the direction of the model car bot.

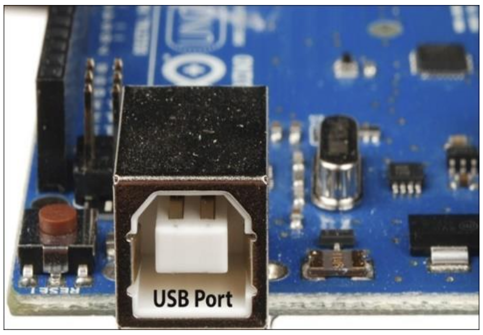
○ Driving Force:

Two big wheels paired with 2 motor allows the bot to move in forward and backward directions. Also travels in the speeds that have been assigned by Processing Unit.

❖ Power Modules:

○ Powering the Arduino UNO R3:

R3 is powered by USB B cable which serves as the default power input for the microcontroller, it also supplies current to the servo motor.



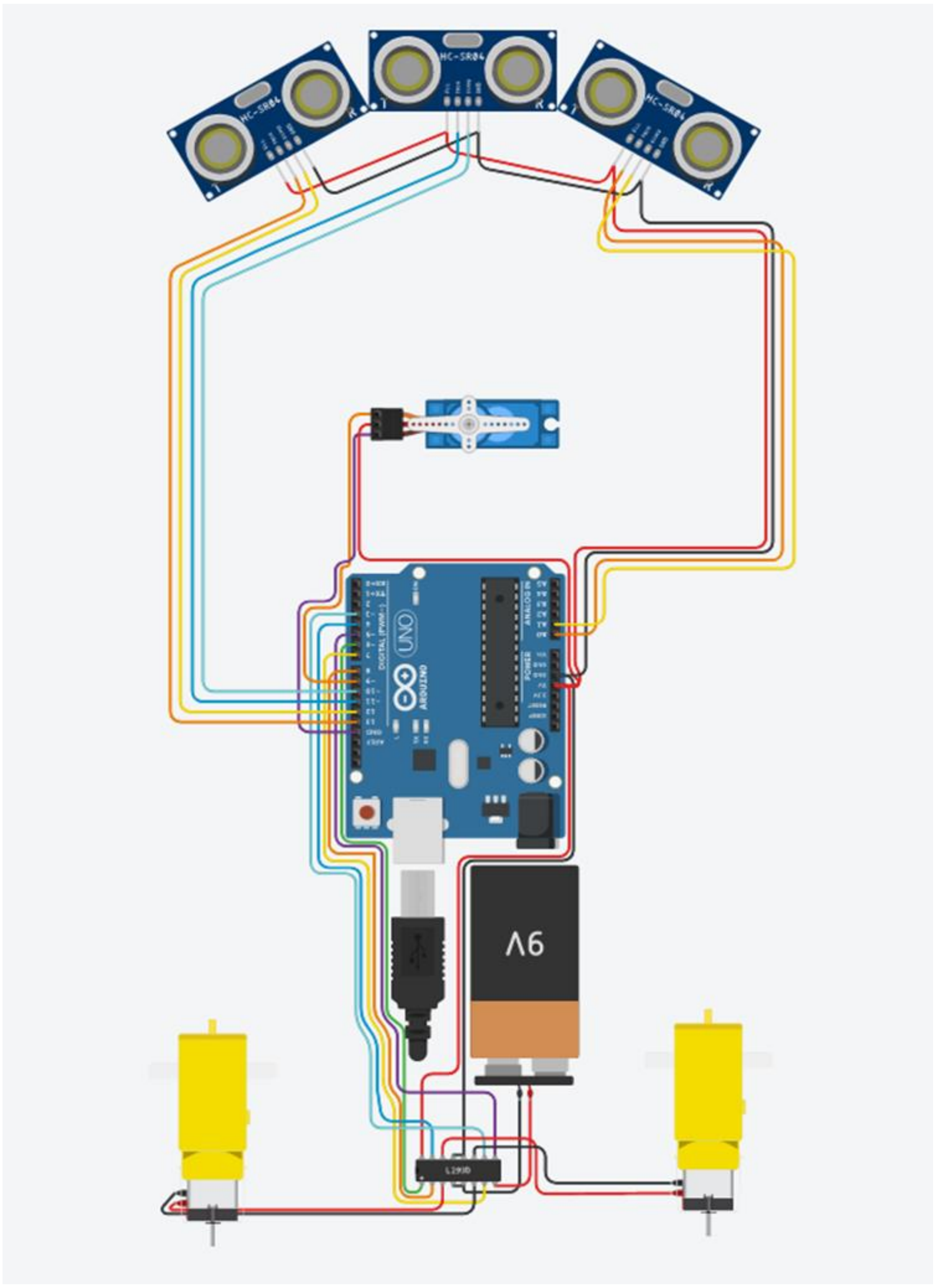
○ Powering the motors:

The motor gearbox is powered by 9v battery as the default input current does not meet its minimum power intake.

❖ Simulation tool used:

TIN KER CAD AUTODESK® TINKERCAD®

CIRCUIT DIAGRAM:



CODE WITH EXPLANATION

```
#include <Servo.h>

//Declaring all the variables
int frontdist = 0;
int leftdist = 0;
int rightdist = 0;
Servo servo_9;

long readUltrasonicDistance(int triggerPin, int echoPin) {
  pinMode(triggerPin, OUTPUT); //Clear the trigger
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  // Sets the trigger pin to HIGH state for 10 microseconds
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  // Reads the echo pin, and returns the sound wave travel time in microseconds
  return pulseIn(echoPin, HIGH);
}

void setup() {
  servo_9.attach(9, 500, 2500);

  Serial.begin(9600);

  pinMode(6, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(3, OUTPUT);
}

void loop() {
  servo_9.write(90);
  delay(1000); // Wait for 1000 millisecond(s)
  frontdist = 0.01723 * readUltrasonicDistance(11, 10);
  if (frontdist > 20) {
    // forward
    Serial.println("forward");
    servo_9.write(90);
    delay(1000); // Wait for 1000 millisecond(s)
    analogWrite(6, 255);
    analogWrite(5, 255);
    digitalWrite(8, HIGH);
    digitalWrite(4, HIGH);
    digitalWrite(7, LOW);
    digitalWrite(3, LOW);
  } else {
```

```
// 1stop
Serial.println("1stop");
analogWrite(6, 0);
analogWrite(5, 0);
digitalWrite(8, LOW);
digitalWrite(4, LOW);
digitalWrite(7, LOW);
digitalWrite(3, LOW);
delay(2000); // Wait for 2000 millisecond(s)
// 1reverse
Serial.println("1reverse");
analogWrite(6, 127);
analogWrite(5, 127);
digitalWrite(8, LOW);
digitalWrite(4, LOW);
digitalWrite(7, HIGH);
digitalWrite(3, HIGH);
delay(2000); // Wait for 2000 millisecond(s)
leftdist = 0.01723 * readUltrasonicDistance(13, 12);
rightdist = 0.01723 * readUltrasonicDistance(A0, A1);
if (leftdist < 20 && rightdist > 20) {
  // right
  Serial.println("right");
  servo_9.write(0);
  delay(1000); // Wait for 1000 millisecond(s)
  analogWrite(6, 127);
  analogWrite(5, 127);
  digitalWrite(8, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(7, LOW);
  digitalWrite(3, LOW);
  delay(2000); // Wait for 2000 millisecond(s)
} else {
  if (leftdist > 20 && rightdist < 20) {
    // left
    Serial.println("left");
    servo_9.write(180);
    delay(1000); // Wait for 1000 millisecond(s)
    analogWrite(6, 127);
    analogWrite(5, 127);
    digitalWrite(8, HIGH);
    digitalWrite(4, HIGH);
    digitalWrite(7, LOW);
    digitalWrite(3, LOW);
    delay(2000); // Wait for 2000 millisecond(s)
  } else {
    if (leftdist > 20 && rightdist > 20) {
      Serial.println("right");
      servo_9.write(0);
      delay(1000); // Wait for 1000 millisecond(s)
      analogWrite(6, 127);
      analogWrite(5, 127);
      digitalWrite(8, HIGH);
      digitalWrite(4, HIGH);
      digitalWrite(7, LOW);
```

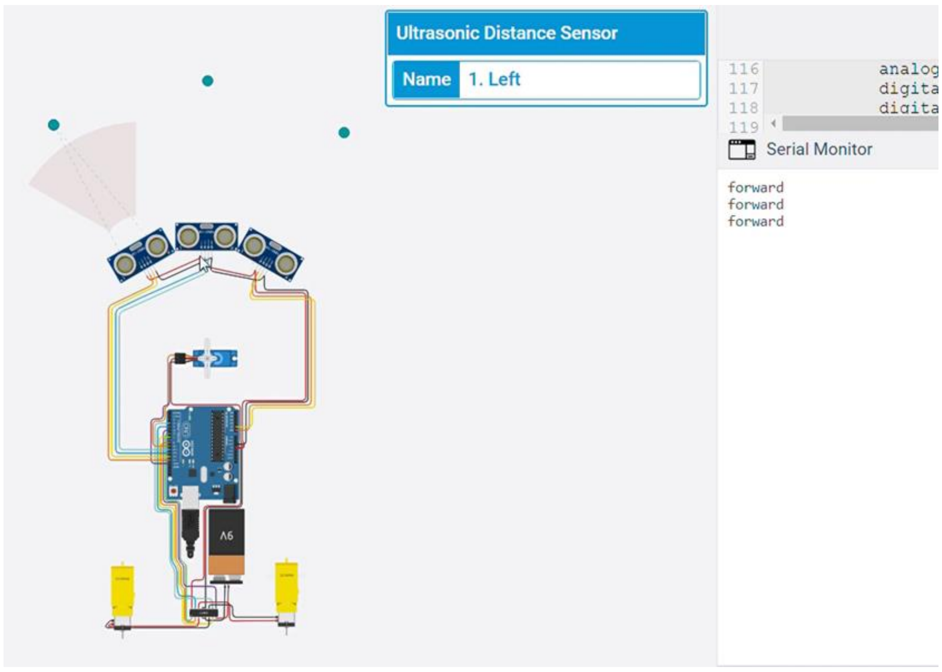
```
          digitalWrite(3, LOW);
          delay(2000); // Wait for 2000 millisecond(s)
        } else {

          // 2stop
          Serial.println("2stop");
          analogWrite(6, 0);
          analogWrite(5, 0);
          digitalWrite(8, LOW);
          digitalWrite(4, LOW);
          digitalWrite(7, LOW);
          digitalWrite(3, LOW);
          delay(1000); // Wait for 2000 millisecond(s)
          // 2reverse
          Serial.println("2reverse");
          analogWrite(6, 127);
          analogWrite(5, 127);
          digitalWrite(8, LOW);
          digitalWrite(4, LOW);
          digitalWrite(7, HIGH);
          digitalWrite(3, HIGH);
          delay(1000); // Wait for 2000 millisecond(s)


          // uturn
          Serial.println("uturn");
          analogWrite(6, 200);
          analogWrite(5, 150);
          digitalWrite(8, HIGH);
          digitalWrite(4, LOW);
          digitalWrite(7, LOW);
          digitalWrite(3, HIGH);
          delay(3000); // Wait for 3000 millisecond(s)
        }
      }
    }
  }
}
```

RESULT SCREENSHOT

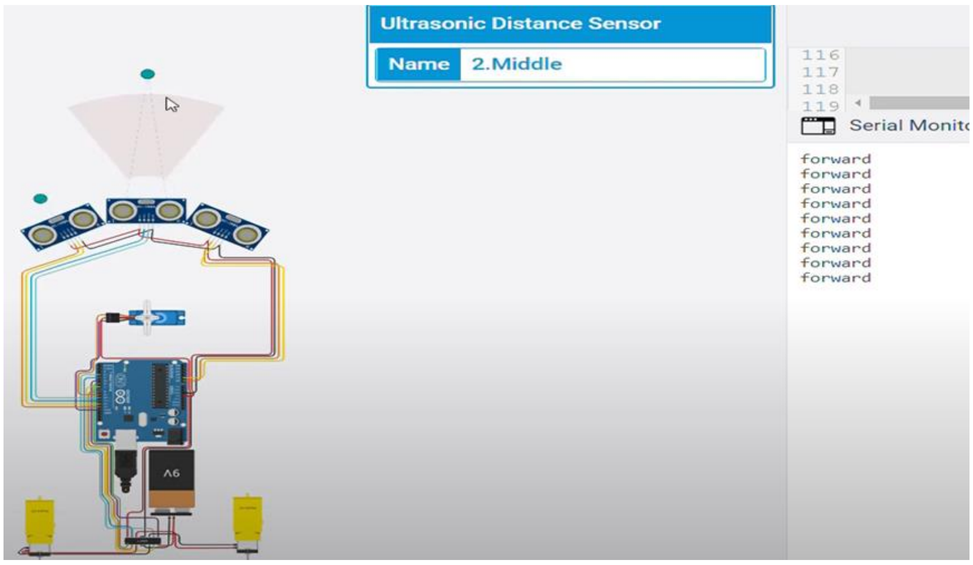
- When there are no obstacles in all the 3 sides, the vehicle moves straight forward.



```
analog
digita
digita
```

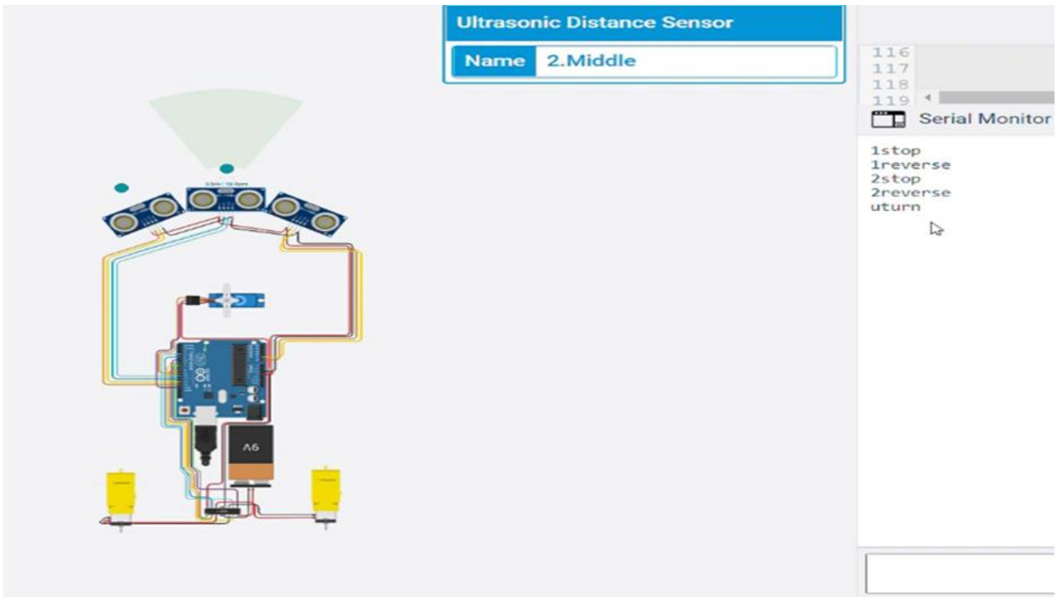
```
forward
forward
forward
```

- When there is an obstacle in front of right sensor or left sensor or both the sensors but not in front of the middle sensor, the vehicle still continues to move straight.



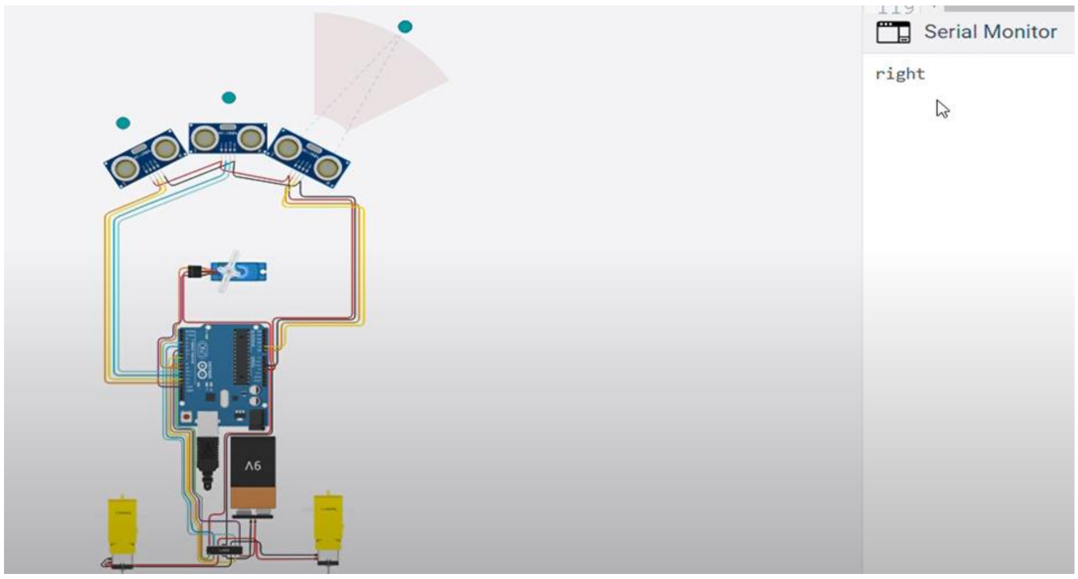
```
forward
forward
forward
forward
forward
forward
forward
forward
forward
```

- When there is obstacle in front of all the 3 sensors, the vehicle comes to stop, takes reverse and makes a u turn.



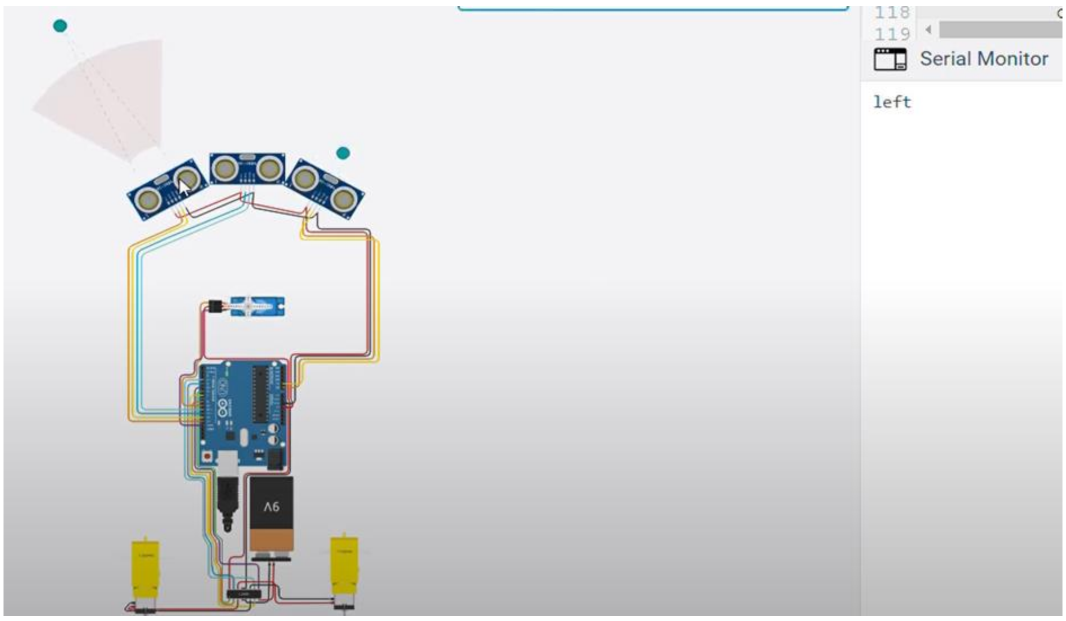
```
1stop
1reverse
2stop
2reverse
uturn
```

- When there is no obstacle in front of the right sensor but there are obstacles in front of left and middle, the vehicle turns to the right.



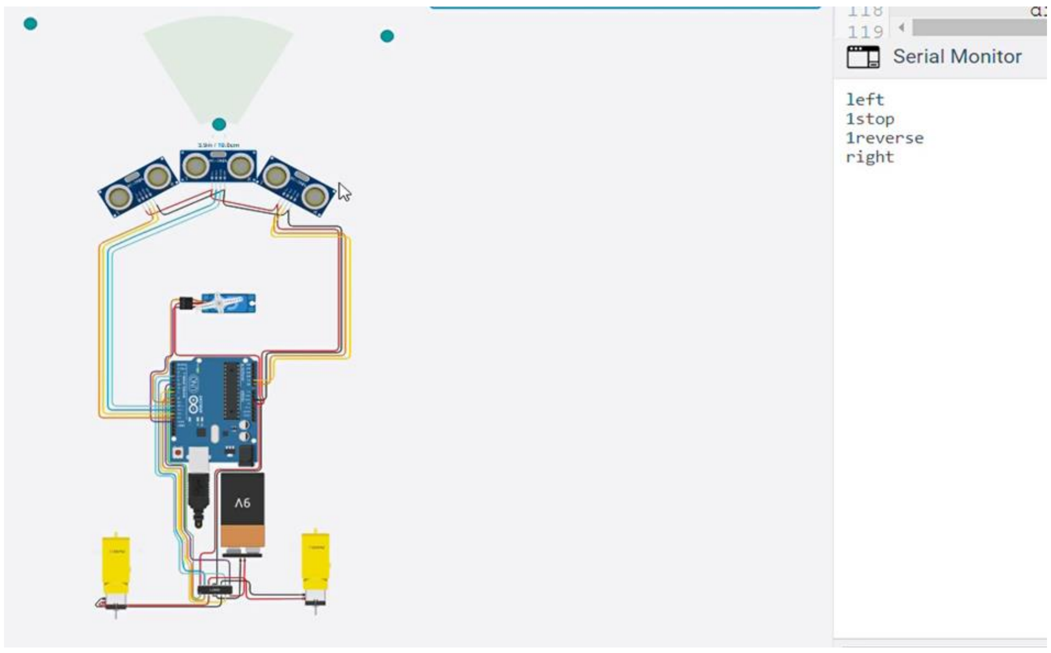
```
right
```

- Similarly, when there is no obstacle in front of left sensor but there are obstacles in front of middle and right sensor, the vehicle turns to the left.



```
left
```

- When there is an obstacle only in front of the middle sensor, the vehicle moves right.



```
left
1stop
1reverse
right
```

LINK OF THE DEMO

YOUTUBE

Demonstration - (https://www.youtube.com/watch?v=mm960fzvDU0)

GOOGLE DRIVE

https://drive.google.com/file/d/17zMttCqrMqSAIwWTW7Nm1D2kWgdgxk7h/view?usp=sharing

REFERENCE

1. https://www.semanticscholar.org/paper/Autonomous-Navigation-via-Deep-Reinforcement-for-Anwar-Raychowdhury/94ae5015c549b0bd0658c70fee612a21523f5f7f

2. .http://www.aerialroboticscompetition.org/assets/downloads/2017SymposiumPapers/HarbinInstituteofTechnology.pdf

3. https://arxiv.org/abs/2109.08628

4. https://www.semanticscholar.org/paper/A-simple-vision-based-navigation-and-control-for-Cyba-Szolc/3072c20c9148a8a717f5528a9528eefaaa8a94e2

5. https://cs.csub.edu/~dpeters/research/AutonomousUAVSwarm.html