RAMAIAH Institute of Technology

Department of Electronics & Communication Engineering.

ROBOTICS (ECE632)

A Report on

Obstacle Avoiding Robot

By

Ravi N	**1MS20EC079**
Adithya R S	**1MS21EC400**
Syed Muzammil Sha	**1MS21EC410**

Under the guidance of

MAMTHA MOHAN

Assistant Professor
Department of ECE, RIT

M S RAMAIAH INSTITUTE OF TECHNOLOGY
(Autonomous Institute, Affiliated to VTU)

BANGALORE-560054

www.msrit.edu

2023

INTRODUCTION:

This project describes about an obstacle avoidance robot vehicle which is controlled by ultrasonic sensors. The robot is made using ultrasonic sensor and it is controlled by Arduino microcontroller. Ultrasonic sensor fixed in front portion of the robot vehicle. The sensor senses the obstacles and deviate its path to choose an obstacle free path. The sensor will be sending the data to the controller is compared with controller to decide the movement of the robot wheel. The robot wheel movement and direction will be based on the sensing of the ultrasonic sensor and also using a wheel encoder. This vehicle is used for detecting obstacle and avoiding the collision.

HARDWARE AND SOFTWARE REQUIREMENTS:

Hardware Requirements:

- Arduino Uno ATMEGA 328P
- L293D motor driver shield
- 100 RPM dual shaft BO motor
- SG90 mini servo motor
- HC-SR04 Ultrasonic distance sensor
- 18650 li-ion battery

Software Requirements:

- Arduino IDE
- Embedded C
- C program

COMPONENT DESCRIPTION:

ARDUINO UNO (ATMEGA 328P):

The Arduino uno is type of CPU it is used to control the entire operation of the system. It is connected to the all devices such as motor driver, BO motors, mini servo motor, ultrasonic distance sensor. This board can be programmed with the help of Arduino IDE (Integrated Development Environment) that supports embedded C. Arduino contains inbuilt ADC and DAC which is not the case with 8051.

MOTOR DRIVER SHIELD (L293D):

L293D motor driver shield is used to drive servo motor and 4 dual shaft BO motor also comes with extra or additional power supply terminal. It consists of 3 IC's L293D driver IC#1, IC#2, 74HC595 IC. The L293D is dual-channel H-Bridge motor driver capable of driving pair of dc motor or single stepper motor. The shield also comes with a 74HC595 shift register that extends 4 digital pins of the Arduino to the 8 direction control pins of two L293D chips. Pin 9, 10, 2 are used for servo motor, can be achieved by interface L293D motor driver shield with Arduino.

DUAL SHAFT BO MOTOR (100 RPM):

The 100 RPM dual shaft BO motor – straight motor gives good torque and RPM at lower operating voltages (3 to 12v). Here it is used as car function to move forward, backward, right, left. The small plastic wheels are connected across the shaft end.

SG90 MINI/MICRO SERVO MOTOR:

SG90 mini servo is 180 (degree) rotation servo. It is a digital servo motor that receives and processes PWM signal faster and better. It equips sophisticated internal circuitry that provides good torque, holding power, and faster updates in response to external forces. Here the servo motor is used to carry the ultrasonic distance sensor for rotation purpose to detect the obstacle.

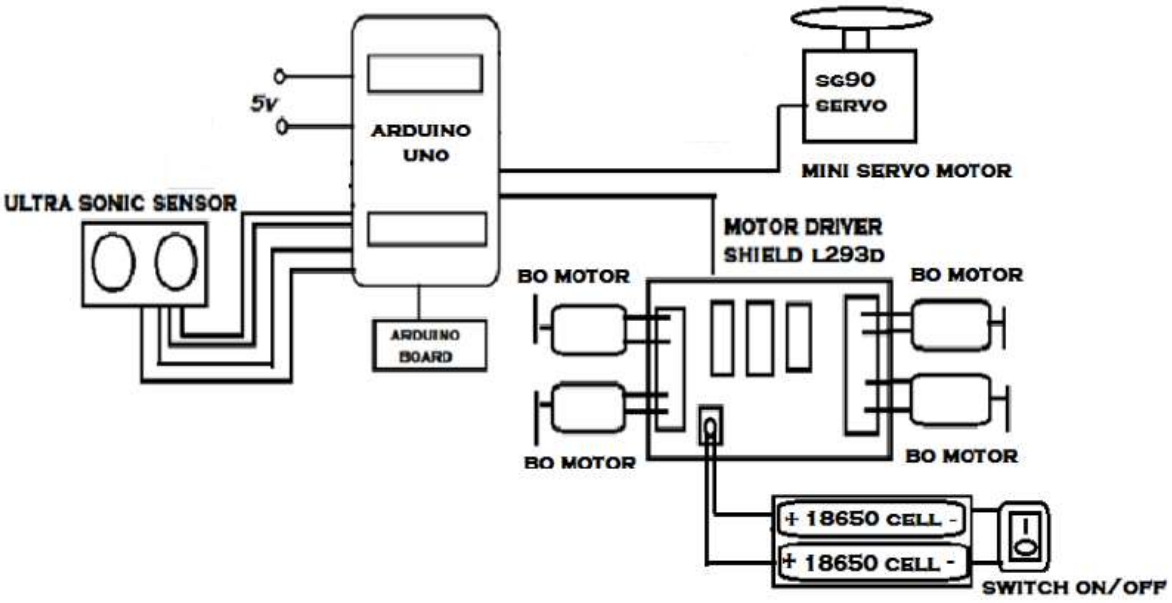
HC-SR04 ULTRASONIC DISTANCE SENSOR:

It is also called Ultrasonic range sensor. This economical sensor provides 2cm to 400cm of non-contact measurement functionally with a ranging accuracy that can reach up to 3mm. It includes an ultrasonic transmitter, a receiver, and a control circuit. There are only four pins VCC, Trig, Echo and GND. Here for the use of project it is interface with Arduino UNO board, with the help of Arduino IDE software.

LI-ION BATTERY (18650):

18650 lithium-ion batteries are used due to superior capacity and discharging rates. Here it is used to supply the power for motor driver shield, because small amount of power will not be able to run 4 gear motors.

BLOCK DIAGRAM AND WORKING PRINCIPLE:



Working:

The working principle of this diagram is entirely carried out as shown in this figure the process is being mainly done by the micro controller called Arduino uno. Then the other parts are linked and controlled by this Arduino uno but, the 18650 battery cells are used here as the power supply for the motor driver shield where this runs 4 BO gear motors which are of 100 RPM. Then the servo motor is driven through same driver shield. Hence, it is justified that the 18650 battery cells provides sufficient amps. Now, we know that the same Arduino board provides power for both input and output ports, the ultrasonic distance sensor here acts as an input device.

Arduino is the main processing unit of the robot. Out of the 14 available digital I/O pins, 7 pins are used in this project design.

The ultrasonic sensor has 4 pins: VCC, Trig, Echo and Gnd. VCC and GND are connected to the +5v and GND pins of the Arduino. Trig (Trigger) is connected to the 9th pin and Echo is connected to 8th pin of the Arduino UNO respectively.

A Servo Motor is used to rotate the Ultrasonic Sensor to scan for obstacles. It has three pins namely Control, VCC and GND. The Servo Control Pin is connected to pin 11 of Arduino while the VCC and GND are connected to +5V and GND.

This information is processed by the Arduino. If the distance between the robot and the obstacle is less than 15cm, the Robot stops and scans in left and right directions for new distance using Servo Motor and Ultrasonic Sensor. If the distance towards the left side is more than that of the right side, the robot will prepare for a left turn. But first, it backs up a little bit and then activates the Left Wheel Motor in reversed in direction.

HARDWARE REQUIREMENTS:

ARDUINO UNO (ATMEGA 328P):

The Arduino Uno is a popularly used open-source microcontroller board that runs on the ATMEGA328P microcontroller. It was developed by Arduino.cc. Which is an Italy based Hardware Company. The board contains a set of digital and analog input/output (I/O) data pins that are used to interface this board with other electronic components. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and this programmable with the help of Arduino IDE (Integrated Development Environment) that supports embedded C, its back-end is constructed using JAVA. Uno consists of an USB port through which the code can be uploaded on to the board. This port can also be used to power the board by connecting it to a laptop, PC, etc. Along with a USB port, it also has a DC input power jack. An external battery of 9v can also be used to power Arduino board.

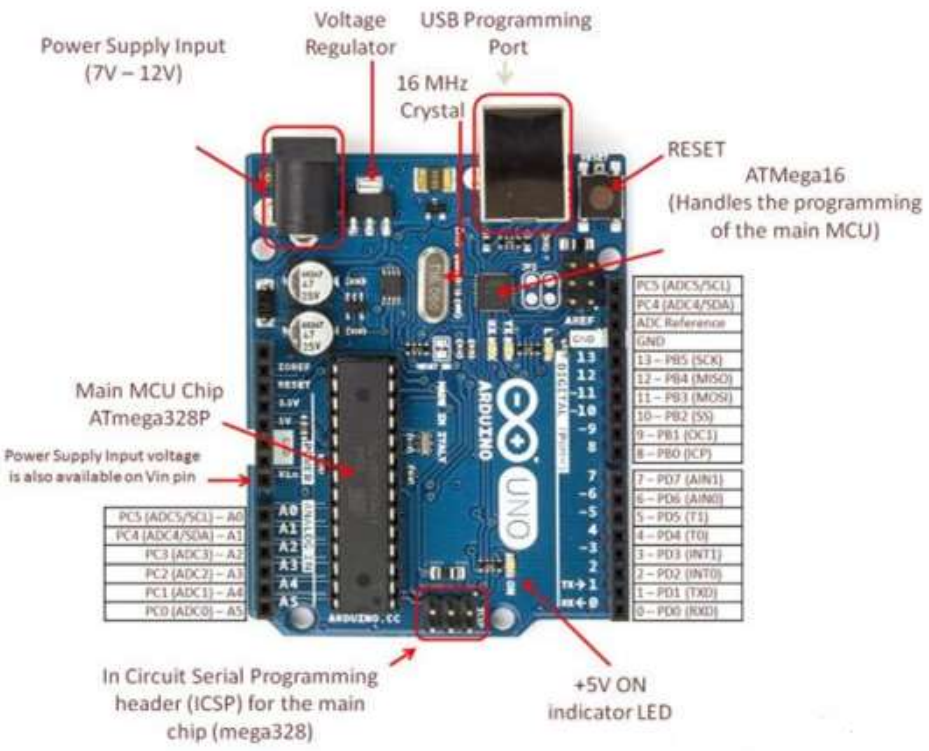


Fig : Arduino UNO controller

L293D MOTOR DRIVER SHIELD:

The L293D Motor driver shield is one of the best ways for controlling DC, Servo and Stepper motors especially if you are using Arduino UNO or MEGA in projects like robotics and CNC.

This shield is based on the L293D IC and can drive 4 bi-directional DC motors, 2 stepper motors and 2 servo motors. It is mainly compatible with the Arduino UNO and MEGA boards. Take note that each channel of this module has the maximum current of 1.2A and doesn't work if the voltage is more than 25v or less than 4.5v.

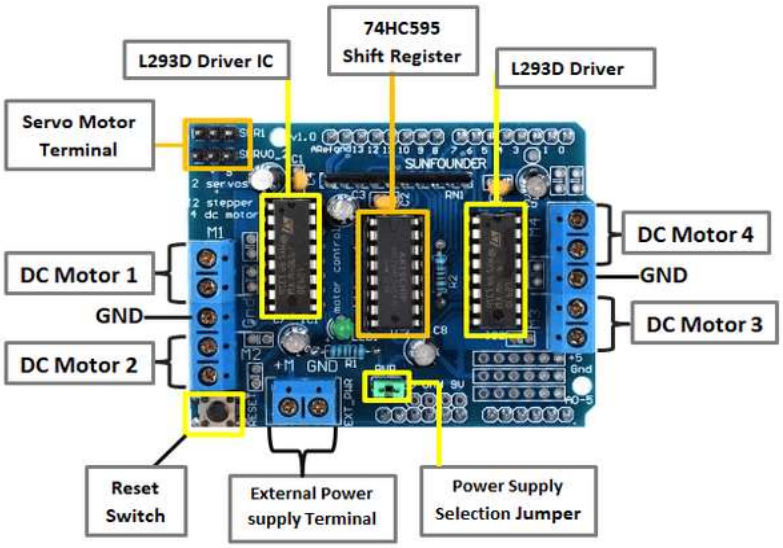


Fig L293D motor driver shield controller

Structure of the L293D motor driver shield:

The L293D is a dual-channel H-Bridge motor driver capable of driving a pair of DC motors or single stepper motor. This shield offers total four H-Bridges and each H-bridge can deliver up to 0.6A to the motor. The shield also comes with a 74HC595 shift register that extends 4 digital pins of the Arduino to the 8 direction control pins of two L293D chips.

Controlling motors using the L293D motor driver shield and Arduino: Before using the L293D motor driver shield with Arduino IDE, we need to install the AFMotor.h library. This contains the commands to control DC, stepper and servo motors.

Driving Servo Motors:

Driving servos with L293D motor shield is very easy. Just connect the three pins of the motor to the servo terminals of the shield. In this case Arduino pins 9, 10, 2 are in use and the power for the Servos comes from the Arduino's on-board 5V regulator, so no need of external power supply on the EXT_PWR terminal.

100 RPM Dual Shaft BO motor:

The 100 RPM Dual Shaft BO Motor - Straight motor gives good torque and rpm at lower operating voltages, which is the biggest advantage of these motors.

Small shaft with matching wheels gives an optimized design for your application or robot. Mounting holes on the body & light weight makes it suitable for in-circuit placement. This motor can be used with 69mm Diameter Wheel for Plastic Gear Motors.

Fig 100 RPM Dual Shaft BO Motor - Straight

It is an alternative to our metal gear DC motors. It comes with an operating voltage of 3-12V and is perfect for building small and medium robots.

SG90 Mini/Micro Servo motor:

The SG90 9g mini servo motor is the most commonly used servo motor in RC applications. The servo motors are used for control applications which require precision control like robot arm positioning, tool position in machining equipment. The servo motors usually provide control over 180° range. This angular position control is performed by PWM technique so by varying its duty cycle you can control the angular position of the motor. This servo motor can lift a maximum of 1.6 kg when suspended at 1cm distance from the shaft. It can also be used in robotic arm, CNC machine, Steering systems on RC cars and other robotic or automation applications as well.

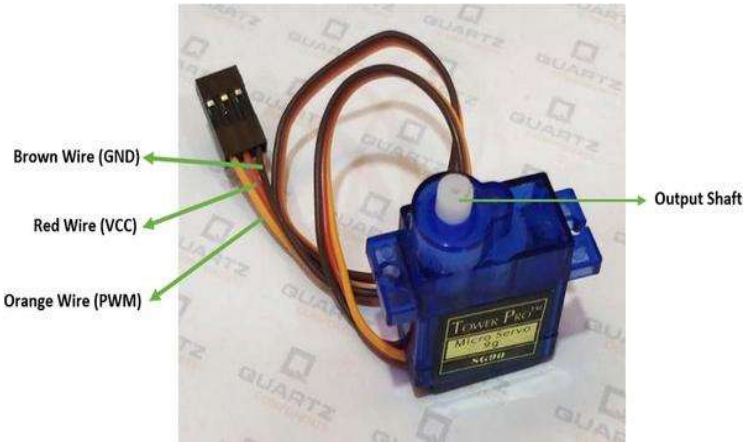


Fig SG90 Micro Servo motor

To use this motor we have used something which can generate PWM signals which can be anything from 555 Timer IC to other microcontrollers like Arduino, Raspberry Pi, PIC etc. Power the motor by VCC and GND pins and control the movement of the shaft by PWM input to it.

HC-SR04 Ultrasonic distance sensor:

The HC-SR04 Ultrasonic Distance Sensor is a sensor used for detecting the distance to an object using sonar. The HC-SR04 uses non-contact ultrasound sonar to measure the distance to an object, and consists of two ultrasonic transmitters (basically speakers), a receiver, and a control circuit.

The HC-SR04 ultrasonic sensor uses SONAR to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1” to 13 feet.

Ultrasonic sensors work by emitting sound waves at a frequency too high for humans to hear. They then wait for the sound to be reflected back, calculating distance based on the time required. This is similar to how radar measures the time it takes a radio wave to return after hitting an object.

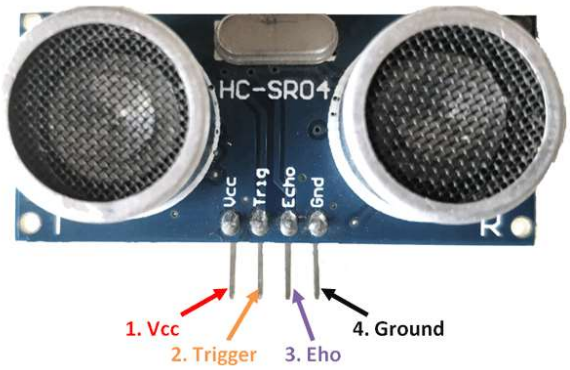


Fig HC-SR04 Ultrasonic distance sensor

18650 lithium-ion battery:

An 18650 battery is a lithium-ion battery. The name derives from the battery's specific measurements: 18mm x 65mm. For scale, that's ever so slightly larger than an AA battery. The 18650 battery has a voltage of 3.6v and has between 2600mAh and 3500mAh (mili-amp-hours). Lithium-Ion Battery for robotic applications. Very light weight and small size compared to Ni-Cd, Ni-MH and Lead acid batteries. Very long life without loosing the charging capacity. Weights just 41 gm.

The biggest users of 18650 cells are laptops and cordless tools. Laptops use low-amp cells that have a medium capacity, and they are sold based on the lowest price. The 18650 cells that are the most useful to e-bikes are from cordless tools, which have the dual needs of high amps and long range.

SOFTWARE REQUIREMENTS:

Arduino IDE:

Arduino IDE (Integrated Development Environment) is a software platform that enables a user to program Arduino or any controller of ATMEGA family. The back-end of this software is developed using JAVA. This IDE provides a user the liberty to program the Arduino using C language. It connects to the Arduino and hardware to upload programs and communicate with them.

- **void setup ()**: This is the location where a user can initialize all the variables that will be required during the course of programming a system. As the name suggests, this function is used to set up the Arduino before interfacing it with other circuits. This area can also be used to include libraries of various sensors. The popularly used functions in void setup are:
- **pin mode**: This function is used to declare pins of Arduino as input or output
- **serial began**: this function is used when Arduino is communicating with other sensors or devices. This enables a user to set a specific band rate for communication purpose.
- **void loop()**: The code written in this space will run over and over again unless Arduino is interrupted using an interrupt or the USB cable is disconnected from USB port. The different functions that are often used in void loop are:

- **digital wire**: This function is used to make a specific pin on Arduino logically HIGH or LOW.
- **digital Read**: This function is used when there is a need to read digital data from a sensor or when we have to control something using a switch/push button.
- **Analog Read:** This function comes in handy when we have to read analog data from a sensor example Analog read is used when there is a need to read data from a potentiometer.
- **Analog Write:** This function is used when a user wants to supply analog voltages to a component. The best example of analog write is when the intensity of LED is controlled using a potentiometer and analog write function.

Flow chart:

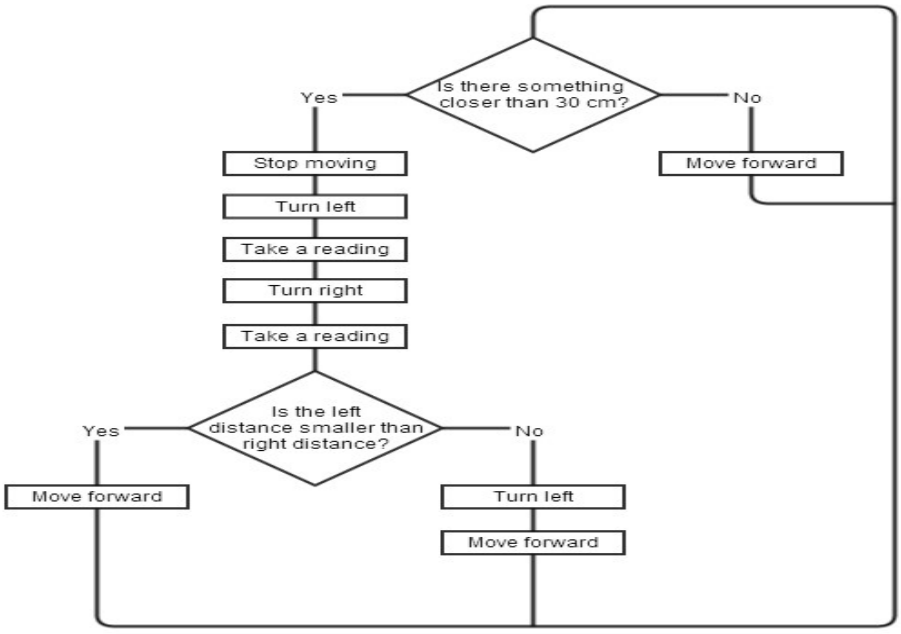
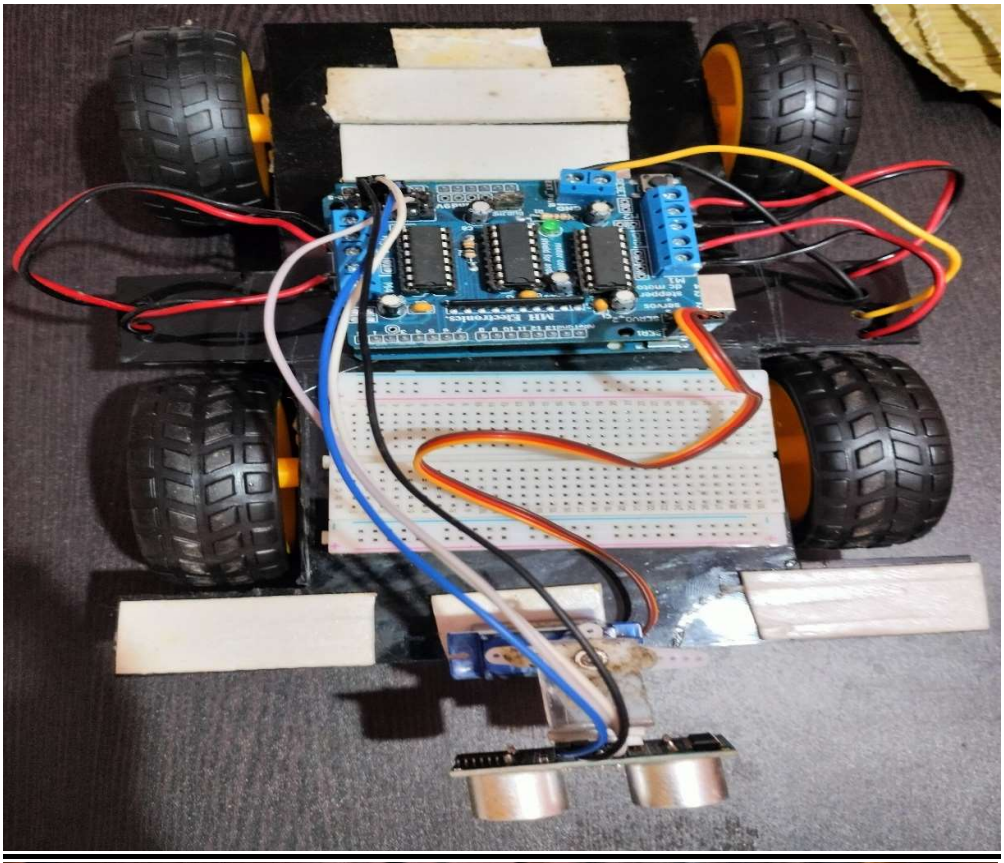
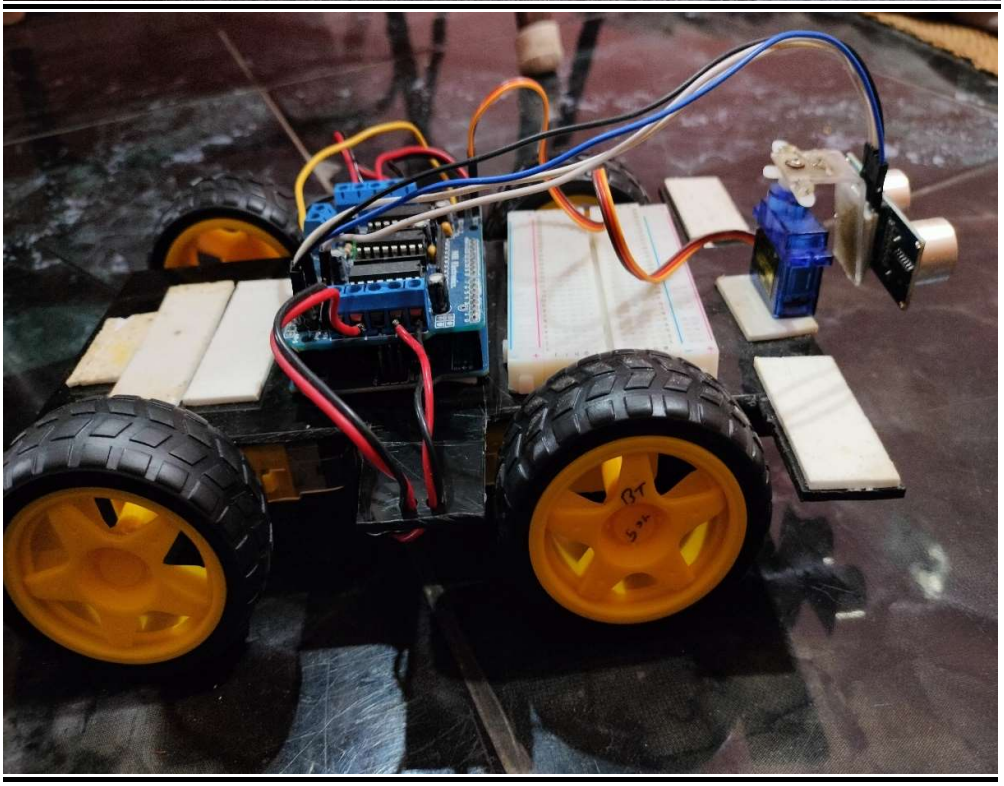


Figure shows the Flow Chart of the working of the obstacle avoidance robot. Initially it checks obstacle within 30cm.If there is an obstacle it stops moving and turns towards left and checks if there is an object closer than 30 cm. The check has two possible outcomes, yes or no. Yes, meaning that there is indeed some object closer than 30 cm. No, meaning that there is no objects detected within 30cm. If there is nothing within 30 cm the robot can simply move forward as the path is clear. If there is something closer than 30 cm the robot must perform obstacle avoidance. The first stage of obstacle avoidance is to stop the robot! If you don't stop the robot immediately it will crash! After the robot has stopped it needs to see what way it should go. It does this by looking both directions, much like you should when you cross the road. First the robot turns left, takes a reading, turns right, and takes a reading. Another check occurs to see what direction is the best way to go. If left is the way to go it has to turn back to the left and then go forward. If right is the way to go the robot simply moves forward as it is already facing in the right direction.

PROTOTYPE MODEL:





ARDUINO UNO CODE:

```
#include <AFMotor.h>                    //Import library to control motor shield
#include <Servo.h>                      //Import library to control the servo

AF_DCMotor rightBack(1);                //Create an object to control each motor
AF_DCMotor rightFront(2);
AF_DCMotor leftFront(3);
AF_DCMotor leftBack(4);
Servo servoLook;                        //Create an object to control the servo

byte trig = 2;                          //Assign the ultrasonic sensor pins
byte echo = 13;
byte maxDist = 150;       //Maximum sensing distance (Objects further than this distance are
ignored)
byte stopDist = 50;               //Minimum distance from an object to stop in cm
float timeOut = 2*(maxDist+10)/100/340*1000000;       //Maximum time to wait for a return
signal

byte motorSpeed = 55;             //The maximum motor speed
int motorOffset = 10;             //Factor to account for one side being more powerful
int turnSpeed = 50;               //Amount to add to motor speed when turning


void setup()
{
 rightBack.setSpeed(motorSpeed);          //Set the motors to the motor speed
 rightFront.setSpeed(motorSpeed);
 leftFront.setSpeed(motorSpeed+motorOffset);
 leftBack.setSpeed(motorSpeed+motorOffset);
 rightBack.run(RELEASE);                  //Ensure all motors are stopped
 rightFront.run(RELEASE);
 leftFront.run(RELEASE);
 leftBack.run(RELEASE);
 servoLook.attach(10);                    //Assign the servo pin
 pinMode(trig,OUTPUT);                    //Assign ultrasonic sensor pin modes
 pinMode(echo,INPUT);
}

void loop()
{
 servoLook.write(90);                     //Set the servo to look straight ahead
 delay(750);
```

```
  int distance = getDistance();                  //Check that there are no objects ahead
  if(distance >= stopDist)          //If there are no objects within the stopping distance, move
forward
  {
    moveForward();
  }
  while(distance >= stopDist)                    //Keep checking the object distance until it is
within the minimum stopping distance
  {
    distance = getDistance();
    delay(250);
  }
  stopMove();                                    //Stop the motors
  int turnDir = checkDirection();       //Check the left and right object distances and get the
turning instruction
  Serial.print(turnDir);
  switch (turnDir)                //Turn left, turn around or turn right depending on the instruction
  {
    case 0:                                      //Turn left
      turnLeft (400);
      break;
    case 1:                                      //Turn around
      turnLeft (700);
      break;
    case 2:                                      //Turn right
      turnRight (400);
      break;
  }
}

void accelerate()                              //Function to accelerate the motors from 0 to full speed
{
  for (int i=0; i<motorSpeed; i++)               //Loop from 0 to full speed
  {
    rightBack.setSpeed(i);                     //Set the motors to the current loop speed
    rightFront.setSpeed(i);
    leftFront.setSpeed(i+motorOffset);
    leftBack.setSpeed(i+motorOffset);
    delay(10);
  }
}

void decelerate()                              //Function to decelerate the motors from full speed to zero
{
  for (int i=motorSpeed; i!=0; i--)              //Loop from full speed to 0
  {
```

```
   rightBack.setSpeed(i);                    //Set the motors to the current loop speed
   rightFront.setSpeed(i);
   leftFront.setSpeed(i+motorOffset);
   leftBack.setSpeed(i+motorOffset);
   delay(10);
  }
}

void moveForward()                          //Set all motors to run forward
{
 rightBack.run(FORWARD);
 rightFront.run(FORWARD);
 leftFront.run(FORWARD);
 leftBack.run(FORWARD);
}

void stopMove()                             //Set all motors to stop
{
 rightBack.run(RELEASE);
 rightFront.run(RELEASE);
 leftFront.run(RELEASE);
 leftBack.run(RELEASE);
}

void turnLeft(int duration)          //Set motors to turn left for the specified duration then stop
{
 rightBack.setSpeed(motorSpeed+turnSpeed);              //Set the motors to the motor speed
 rightFront.setSpeed(motorSpeed+turnSpeed);
 leftFront.setSpeed(motorSpeed+motorOffset+turnSpeed);
 leftBack.setSpeed(motorSpeed+motorOffset+turnSpeed);
 rightBack.run(FORWARD);
 rightFront.run(FORWARD);
 leftFront.run(BACKWARD);
 leftBack.run(BACKWARD);
 delay(duration);
 rightBack.setSpeed(motorSpeed);                        //Set the motors to the motor speed
 rightFront.setSpeed(motorSpeed);
 leftFront.setSpeed(motorSpeed+motorOffset);
 leftBack.setSpeed(motorSpeed+motorOffset);
 rightBack.run(RELEASE);
 rightFront.run(RELEASE);
 leftFront.run(RELEASE);
 leftBack.run(RELEASE);

}
```

```
void turnRight(int duration)        //Set motors to turn right for the specified duration then stop
{
 rightBack.setSpeed(motorSpeed+turnSpeed);              //Set the motors to the motor speed
 rightFront.setSpeed(motorSpeed+turnSpeed);
 leftFront.setSpeed(motorSpeed+motorOffset+turnSpeed);
 leftBack.setSpeed(motorSpeed+motorOffset+turnSpeed);
 rightBack.run(BACKWARD);
 rightFront.run(BACKWARD);
 leftFront.run(FORWARD);
 leftBack.run(FORWARD);
 delay(duration);
 rightBack.setSpeed(motorSpeed);                        //Set the motors to the motor speed
 rightFront.setSpeed(motorSpeed);
 leftFront.setSpeed(motorSpeed+motorOffset);
 leftBack.setSpeed(motorSpeed+motorOffset);
 rightBack.run(RELEASE);
 rightFront.run(RELEASE);
 leftFront.run(RELEASE);
 leftBack.run(RELEASE);
}

int getDistance()                           //Measure the distance to an object
{
 unsigned long pulseTime;                   //Create a variable to store the pulse travel time
 int distance;                              //Create a variable to store the calculated distance
 digitalWrite(trig, HIGH);                  //Generate a 10 microsecond pulse
 delayMicroseconds(10);
 digitalWrite(trig, LOW);
 pulseTime = pulseIn(echo, HIGH, timeOut);        //Measure the time for the pulse to return
 distance = (float)pulseTime * 340 / 2 / 10000;    //Calculate the object distance based on the
pulse time
 return distance;
}

int checkDirection()                                //Check the left and right directions and
decide which way to turn
{
 int distances [2] = {0,0};                         //Left and right distances
 int turnDir = 1;                                   //Direction to turn, 0 left, 1 reverse, 2 right
 servoLook.write(180);                              //Turn servo to look left
 delay(500);
 distances [0] = getDistance();                     //Get the left object distance
 servoLook.write(0);                                //Turn servo to look right
 delay(1000);
```

```
  distances [1] = getDistance();                          //Get the right object distance
  if (distances[0]>=200 && distances[1]>=200)             //If both directions are clear, turn left
    turnDir = 0;
  else if (distances[0]<=stopDist && distances[1]<=stopDist)    //If both directions are
blocked, turn around
    turnDir = 1;
  else if (distances[0]>=distances[1])                    //If left has more space, turn left
    turnDir = 0;
  else if (distances[0]<distances[1])                     //If right has more space, turn right
    turnDir = 2;
  return turnDir;
}
```

ADVANTAGES AND APPLICATION:

Advantages:

- Improves the safety factor and driving experience at night.
- It is a low cost circuit, easy to install, easily adaptable, cost effective and high reliability.
- The programming of the microcontroller is easy.
- Whenever robot senses any obstacles automatically diverts its position to left or right and follows the path without human guidance.
- Less human effort and low maintenance cost.

Applications:

- Can be used in all automobiles
- Can be used in almost all mobile robot navigation systems
- With proper programming we can use it as an auto parking assistance
- Just by making small changes in software this system can be used for avoiding concealed paths.
- By adding the IR sensors we can use this for distance measurement, surface feature detection.

CONCLUSION:

This project provides an obstacle avoiding robot that detects obstacles coming in its path and avoids it by moving in another direction. The robot is built with Arduino that processes the information to various parts. For object detection, ultrasonic sensors have been used that provides a wider field of view. Servo motor has been used for rotating the sensor. The robot is able to move by using four geared motors. It is perfectly avoiding the obstacles coming in its path.

Thus the implementation of this device in every vehicle in future will not only avoid accidents but also provide a safe and a comfortable driving.