DEVELOPMENT OF OBSTACLE DETECTION ROBOT USING ARDUINO MICROCONTROLLER

Preamble:

- *Introduction*
- *Components Used*
- *Code*
- *Explanation*
- *Real Time Usage*
- *Result/Video*
- *Bibliography*

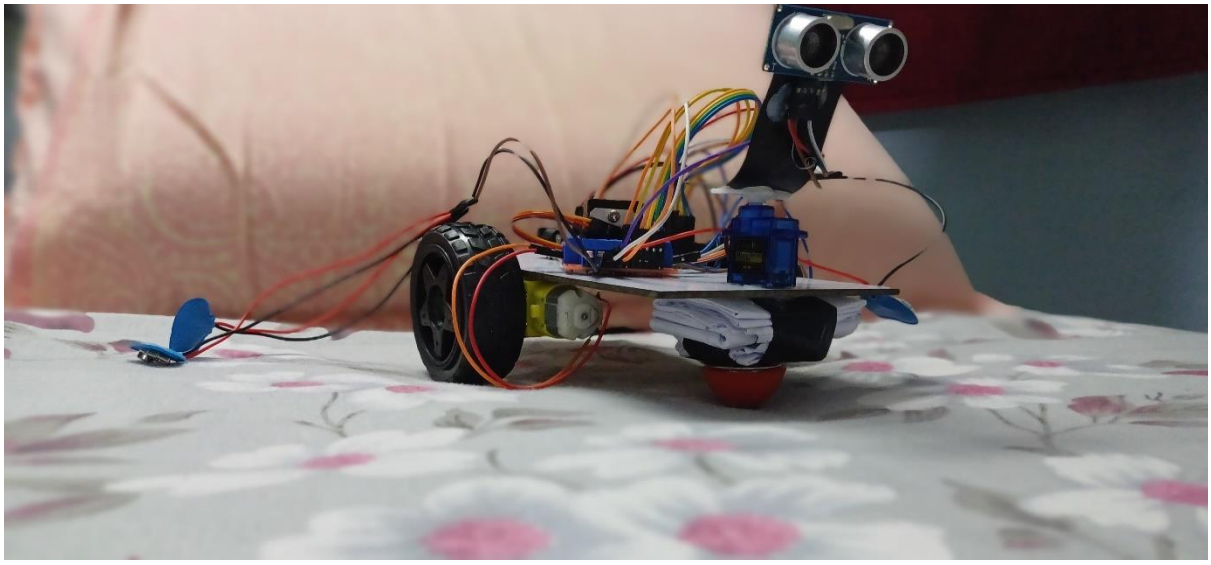
Introduction:

The Robots are playing an role in automation across many sectors such as construction, military, medical and manufacturing. After exploring the design and development of robots such as a line follower robot, a computer-controlled robot, we developed a automatic obstacle detection robot using an Arduino Uno. The key outcome of this project is to make the robot move on its own by sensing obstacles and move accordingly.

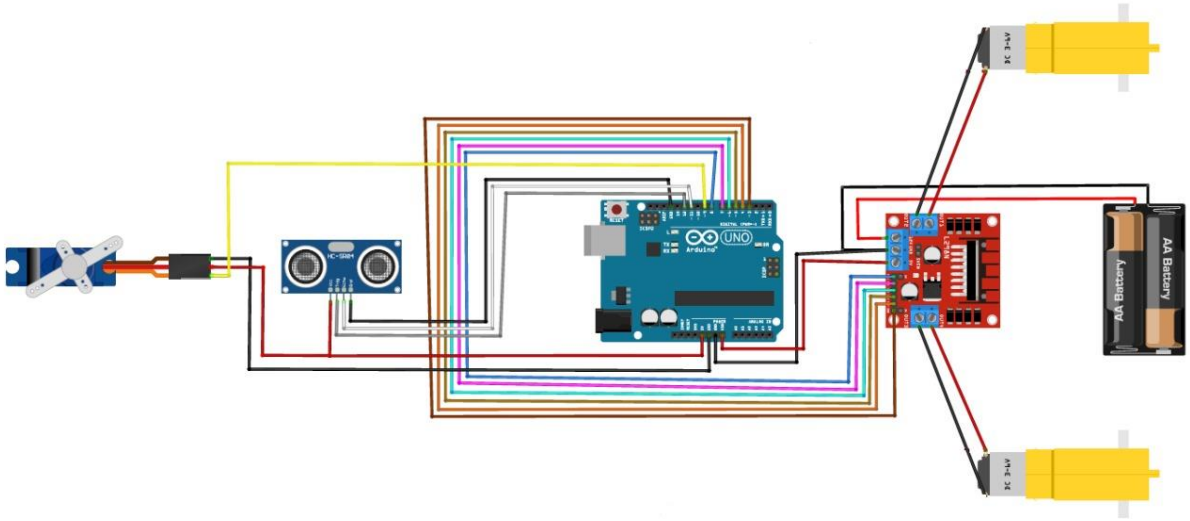
The ability to control things by a hand were first observed in science fiction movies and was like magic. We were excited and curious to know how it works

and initiated this project to implement such a functionality for robots.It is an inspiration from present tech advancements from Tesla innovating Automatic Car.

Obstacle detection in robotics involves using sensors to identify and avoid obstacles in a robot's path, ensuring safe and efficient navigation. Common sensors include ultrasonic Distance sensor which collect data about the surroundings. The robot processes this data to determine object distance, size, and position, allowing it to adjust its movement in real-time to avoid collisions. Obstacle detection is essential for autonomous and mobile robots, enhancing their ability to operate in dynamic environments, such as warehouses, streets, or even households.



Circuit Diagram:

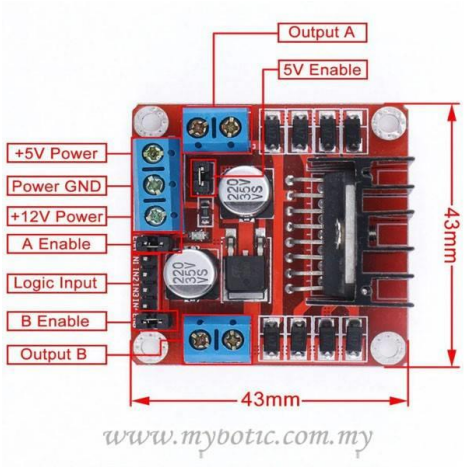


Components used :

1) Arduino UNO Microcontroller:-

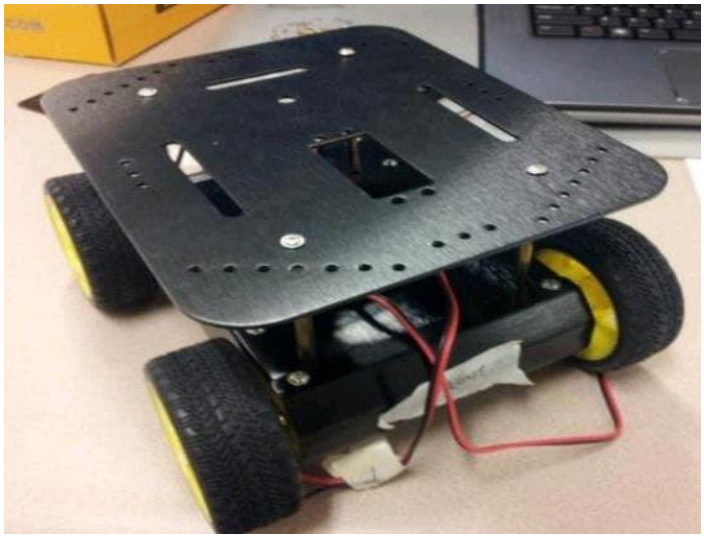
The Arduino Uno features 14 digital I/O pins and 6 analog input pins, making it versatile for a range of projects. Six digital pins support PWM output to simulate analog signals, useful for controlling devices like motors and LEDs. It can be powered via USB or an external power source, typically between 7-12V. Known for its simplicity and compatibility, the Uno is widely used in electronics, prototyping, and educational applications.

2) Motor driver L298N:-



The L298N is a dual H-bridge motor driver IC that controls the speed and direction of two DC or stepper motors simultaneously. It operates at 5V to 35V and includes built-in diodes for back EMF protection. Easy to interface with microcontrollers like Arduino, it uses PWM signals for speed control and direction management, making it ideal for robotics projects.

3) **Chassis:-**



The chassis was made up of wood or plastic which gives the support to the car and helps to place the Arduino and motor driver on it.

4) **Motors and wheels:-**

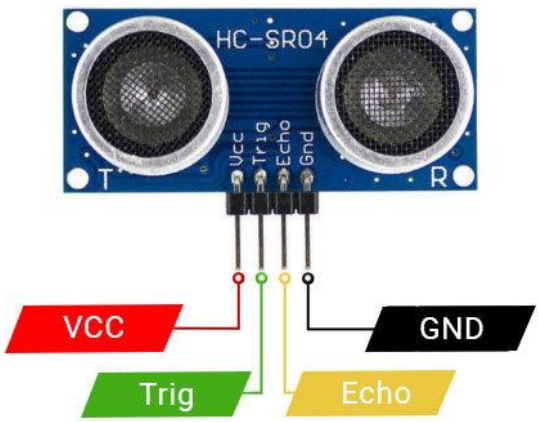
Motors, such as DC motors and stepper motors, convert electrical energy into rotational motion, enabling movement in various applications. DC motors provide variable speed control, while stepper motors allow for precise positioning. Wheels attached to these motors facilitate movement,

with different designs influencing speed and maneuverability. Together, motors and wheels are crucial for mobile robotics and automation, enabling navigation in diverse environments.

5)Servo motor:-

A servo motor provides precise control of angular position and speed, consisting of a DC motor, feedback mechanism, and control circuit. Commonly used in robotics and remote-controlled vehicles, it operates by receiving PWM signals to adjust its position accurately. Its ability to maintain position under varying loads makes it ideal for steering and positioning tasks.

6) Ultrasonic sensor:-



An ultrasonic sensor measures distance by emitting ultrasonic pulses and detecting their echoes. It consists of a transmitter and a receiver, making it useful for applications like obstacle detection and robotic navigation. These sensors are accurate, non-contact devices suitable for various environments

7) Battery:-

Battery is a portable energy storage device that converts stored chemical energy into electrical energy through electrochemical reactions. It consists of one or more cells that generate voltage and provide power for various applications.

8) Connecting wires:-

Connecting wires are used to link between Arduino and motor driver, Arduino and ultra-sonic sensor, motor driver and motors.

Libraries Used:

- #include <Servo.h>
- #include <NewPing.h>

Code:

```
#include <Servo.h>          //standard library for the servo
#include <NewPing.h>        //for the Ultrasonic sensor function library.

//L298N motor control pins
```

```
const int LeftMotorForward = 6;
const int LeftMotorBackward = 7;
const int RightMotorForward = 5;
const int RightMotorBackward = 4;
const int enA = 8;
const int enB = 3;

//sensor pins
#define trig_pin 11 //
#define echo_pin 12 //

#define maximum_distance 200
boolean goesForward = false;
int distance = 100;

NewPing sonar(trig_pin, echo_pin, maximum_distance); //sensor function
Servo servo_motor;

void setup(){

  pinMode(RightMotorForward, OUTPUT);
  pinMode(LeftMotorForward, OUTPUT);
```

```
  pinMode(LeftMotorBackward, OUTPUT);
  pinMode(RightMotorBackward, OUTPUT);
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);

  servo_motor.attach(9); //our servo pin

  servo_motor.write(115);
  delay(2000);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
}

void loop(){

  int distanceRight = 0;
```

```
int distanceLeft = 0;
delay(50);

if (distance <= 15){
  moveStop();
  delay(300);
  moveBackward();
  delay(300);
  moveStop();
  delay(300);
  distanceRight = lookRight();
  delay(300);
  distanceLeft = lookLeft();
  delay(300);

  if (distance >= distanceLeft){
    turnRight();
    moveStop();
  }
  else{
    turnLeft();
    moveStop();
```

```
    }
  }
  else{
    moveForward();

  }
    distance = readPing();
}

int lookRight(){
  servo_motor.write(50);
  delay(300);
  int distance = readPing();
  delay(100);
  servo_motor.write(115);
  return distance;
}

int lookLeft(){
  servo_motor.write(170);
  delay(300);
  int distance = readPing();
```

```
  delay(100);
  servo_motor.write(115);
  return distance;
  delay(100);
}

int readPing(){
  delay(70);
  int cm = sonar.ping_cm();
  if (cm==0){
    cm=250;
  }
  return cm;
}

void moveStop(){

  digitalWrite(RightMotorForward, LOW);
  digitalWrite(LeftMotorForward, LOW);
  digitalWrite(RightMotorBackward, LOW);
  digitalWrite(LeftMotorBackward, LOW);
  digitalWrite(enA,HIGH);
```

```
  digitalWrite(enB,HIGH);
}

void moveForward(){

  if(!goesForward){

    goesForward=true;

    digitalWrite(LeftMotorForward, HIGH);
    digitalWrite(RightMotorForward, HIGH);

    digitalWrite(LeftMotorBackward, LOW);
    digitalWrite(RightMotorBackward, LOW);
    digitalWrite(enA,HIGH);
    digitalWrite(enB,HIGH);
  }
}

void moveBackward(){

  goesForward=false;
```

```
  digitalWrite(LeftMotorBackward, HIGH);
  digitalWrite(RightMotorBackward, HIGH);

  digitalWrite(LeftMotorForward, LOW);
  digitalWrite(RightMotorForward, LOW);
  digitalWrite(enA,HIGH);
  digitalWrite(enB,HIGH);

}

void turnRight(){

  digitalWrite(LeftMotorForward, HIGH);
  digitalWrite(RightMotorBackward, HIGH);

  digitalWrite(LeftMotorBackward, LOW);
  digitalWrite(RightMotorForward, LOW);
  digitalWrite(enA,HIGH);
  digitalWrite(enB,HIGH);

  delay(300);
```

```
  digitalWrite(LeftMotorForward, HIGH);
  digitalWrite(RightMotorForward, HIGH);

  digitalWrite(LeftMotorBackward, LOW);
  digitalWrite(RightMotorBackward, LOW);
  digitalWrite(enA,HIGH);
  digitalWrite(enB,HIGH);




}

void turnLeft(){

  digitalWrite(LeftMotorBackward, HIGH);
  digitalWrite(RightMotorForward, HIGH);

  digitalWrite(LeftMotorForward, LOW);
  digitalWrite(RightMotorBackward, LOW);
  digitalWrite(enA,HIGH);
  digitalWrite(enB,HIGH);
```

```
  delay(300);


  digitalWrite(LeftMotorForward, HIGH);
  digitalWrite(RightMotorForward, HIGH);


  digitalWrite(LeftMotorBackward, LOW);
  digitalWrite(RightMotorBackward, LOW);
  digitalWrite(enA,HIGH);
  digitalWrite(enB,HIGH);
}
```

Explanation:

- ***Snippet (read ping):***

readPing() gives the robot an awareness of the surroundings, enabling it to respond to nearby obstacles and navigate effectively. It provides reliable feedback for distance measurements, making the robot capable of obstacle avoidance.

- ***Snippet (void loop function (conditions)):***

It manages the movement of a robot based on distance measurements. It checks the distance to an object in front of it and makes decisions on whether to move forward, backward, or turn based on those measurements.

- **Snippet (The look functions)**

They are used to check distances in different directions using a servo-mounted ultrasonic sensor. By turning the servo to the left or right, the robot can scan for obstacles, gather distance data, and decide which direction to move based on nearby objects.

- **Snippet(The movement functions)**

They control the robot's motion. By adjusting motor pins, these functions enable the robot to navigate forward, backward, and make turns, allowing it to avoid obstacles dynamically and continue along a clear path.

- **Snippet(Initializing)**

This code imports necessary libraries for controlling a servo motor (Servo.h) and an ultrasonic sensor (NewPing.h). It defines the pin connections for the L298N motor driver, which controls the motors' forward nd backward movements. The ultrasonic sensor is set up with trigger a and echo pins to measure distances up to 200 cm. Two instances, sonar and servo_motor, are created to handle sensor and servo functionalities, respectively. Finally, global variables are used to track the robot's movement direction (goesForward) and the distance detected (distance).

Real Life Scenarios:

Obstacle detection robots are used in various real-life applications to navigate safely and efficiently in complex environments. Here are some key areas where they are applied:

1. Warehousing and Logistics: Robots in warehouses use obstacle detection to move goods, avoid collisions, and navigate through aisles, improving efficiency in inventory management and order fulfillment.

2. Healthcare: Hospitals use obstacle-detecting robots to transport medication, equipment, and supplies, ensuring they can move safely around patients, staff, and other obstacles.

3. Self-Driving Cars: Autonomous vehicles rely heavily on obstacle detection to identify pedestrians, other vehicles, and road obstacles, ensuring safe driving on roads.

4. Military and Search & Rescue: In hazardous environments, robots are used to locate and assist people, avoid hazards, and navigate rubble or rough terrains, making them valuable in disaster response.

5. Home Cleaning Robots : Household robots, such as robotic vacuums, use obstacle detection to avoid furniture, stairs, and walls, efficiently cleaning homes without getting stuck.

In all these applications, obstacle detection improves the robot's ability to operate autonomously, ensuring safety and efficiency.

Results/Video:

The obstacle avoidance robot successfully detects obstacles within a set distance using an ultrasonic sensor. When an obstacle is detected, it stops, then turns left or right to find a clear path before resuming forward movement. This allows it to navigate autonomously through an environment with obstacles. During testing, it consistently avoided collisions with a response time of approximately 0.5 seconds.

Kindly find the below attachment for the video of obstacle detection car(Project):

https://shorturl.at/WdI5Y

Bibliography:

Arduino - Home

Obstacle Detection in Self-Controlled Cars

Obstacle Detection and Track Detection in Autonomous Cars | IntechOpen

BY:

K.Lohith(CSE23033)

M.Himanshu(CSE23040)

P.Shanmukha Gautam(CSE23044)

B.Kalyan(CSE23068)