Aim:

Design and implement a smart car capable of detecting obstacles using an ultrasonic sensor and automatically avoiding them by altering its path. This project demonstrates the principles of distance measurement and decision-making in autonomous navigation.

Requirements:

1. Hardware Components:

- Arduino (Uno/Nano) or Raspberry Pi
- Ultrasonic sensor (HC-SR04)
- Motor driver (L298N or similar)
- DC motors (4-wheel or 2-wheel differential drive)
- Wheels
- Chassis
- Power supply (e.g., LiPo battery)
- Jumper wires

2. Software Requirements:

- Arduino IDE / Python (if using Raspberry Pi)
- Basic motor control and sensor libraries

3. Connections:

- **Ultrasonic sensor:** The HC-SR04 sensor has four pins (VCC, GND, TRIG, and ECHO):
  - VCC → 5V
  - GND → GND
  - TRIG → Digital pin 9 (or another available pin)
  - ECHO → Digital pin 10 (or another available pin)
- **Motor driver (L298N):**
  - IN1, IN2, IN3, IN4 for motor control
  - ENA and ENB for motor speed control using PWM
  - Connect DC motors to the motor driver

Procedure:

1. Assemble the Car:

- Build the car chassis with DC motors and wheels.
- Mount the ultrasonic sensor on the front of the car for obstacle detection.
- Connect the motors to the motor driver, and the motor driver to the Arduino.

2. Ultrasonic Sensor:

- The ultrasonic sensor will emit sound waves and measure the time it takes for

them to bounce back from an object. Using this time, the distance to the object is calculated.

3. Obstacle Avoidance Logic:

- The car should move forward unless an obstacle is detected within a specified range (e.g., 20 cm).
- If an obstacle is detected, the car should stop, then turn either left or right to avoid it.

4. Test the System:

- Upload the code to the Arduino.
- Place obstacles in the path of the car and verify that it stops and avoids them by turning.

CODE:

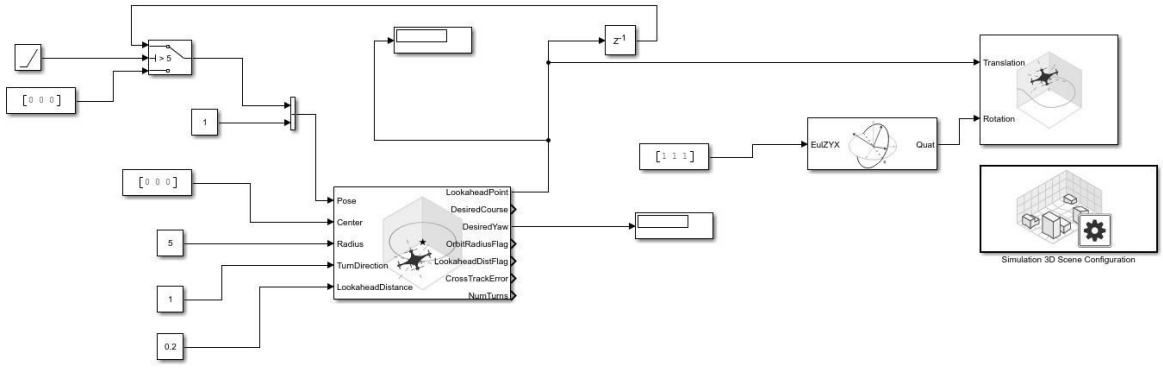
```
// Pin assignments
#define trigPin 9
#define echoPin 10
#define ENA 3  // Speed control for motor A
#define IN1 7  // Motor A forward
#define IN2 6  // Motor A backward
#define ENB 5  // Speed control for motor B
#define IN3 4  // Motor B forward
#define IN4 2  // Motor B backward

long duration;
int distance;
int motorSpeed = 200;  // PWM speed value

void setup() {
  // Set motor control pins as output
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  // Set ultrasonic sensor pins as output/input
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  Serial.begin(9600);  // Initialize serial communication for debugging
}

void loop() {
  // Measure distance using the ultrasonic sensor
```

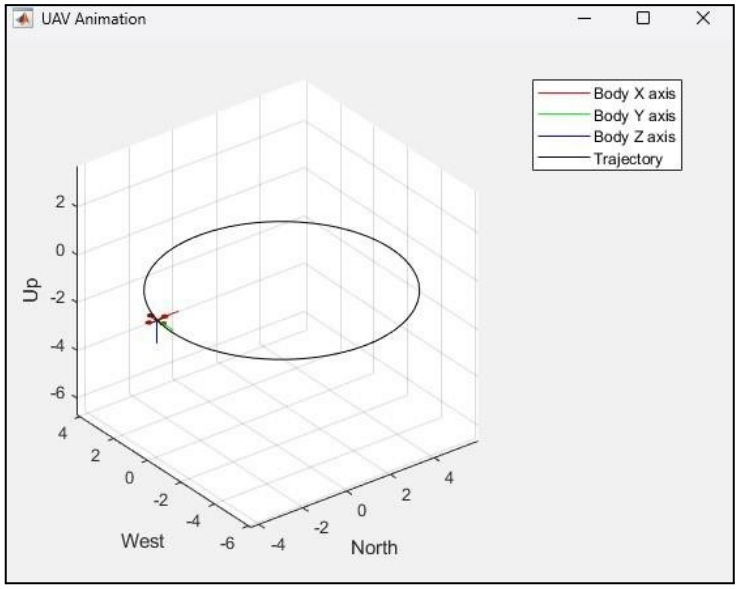
```
  distance = getDistance();
  Serial.print("Distance: ");
  Serial.println(distance);

  // If an obstacle is detected within 20 cm, stop and avoid it
  if (distance < 20) {
    stopCar();         // Stop the car
    delay(500);        // Pause for a moment
    turnRight();       // Turn right to avoid the obstacle
    delay(500);
  } else {
    moveForward();     // Continue moving forward if the path is clear
  }

  delay(100);  // Short delay before next loop
}

int getDistance() {
  // Send a pulse to trigger the ultrasonic sensor
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echo time
  duration = pulseIn(echoPin, HIGH);

  // Calculate distance in cm (duration * 0.034 / 2)
  return duration * 0.034 / 2;
}

void moveForward() {
  analogWrite(ENA, motorSpeed);
  analogWrite(ENB, motorSpeed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}

void stopCar() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}

void turnRight() {
```

```
  // Turn right by moving one motor backward and the other forward
  analogWrite(ENA, motorSpeed);
  analogWrite(ENB, motorSpeed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}
```

Orbit Follower -2

- Constant
- Coordinate Transformation Conversion
- Display
- Orbit Follower
- Ramp
- Simulation 3D Scene Configuration
- Switch
- UAV Animation
- Vector Concatenate





Waypoint Follower

- Constant
- Coordinate Transformation Conversion
- Display
- Waypt Follower
- Ramp
- Simulation 3D Scene Configuration

- Switch
- UAV Animation ● Vector Concatenate