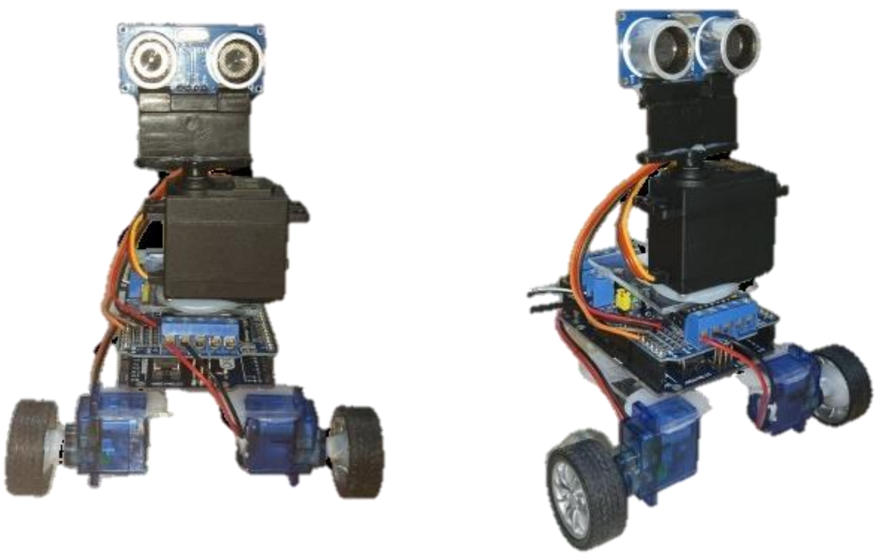
Simple Obstacle Avoidance Robot

Introduction

The Simple Obstacle Avoidance Robot is a basic robotics project designed to navigate its surroundings intelligently and autonomously. This project utilizes an Arduino Uno microcontroller, L293D Motor Driver Shield, servo motor, ultrasonic sensor, and two DC motors. The robot is capable of detecting obstacles using the ultrasonic sensor and making decisions to avoid collisions.



Components Used:

Arduino Uno: The brain of the project, responsible for processing sensor data and controlling the motors.

L293D Motor Driver Shield: Mounted directly on top of the Arduino Uno to drive the two DC motors independently.

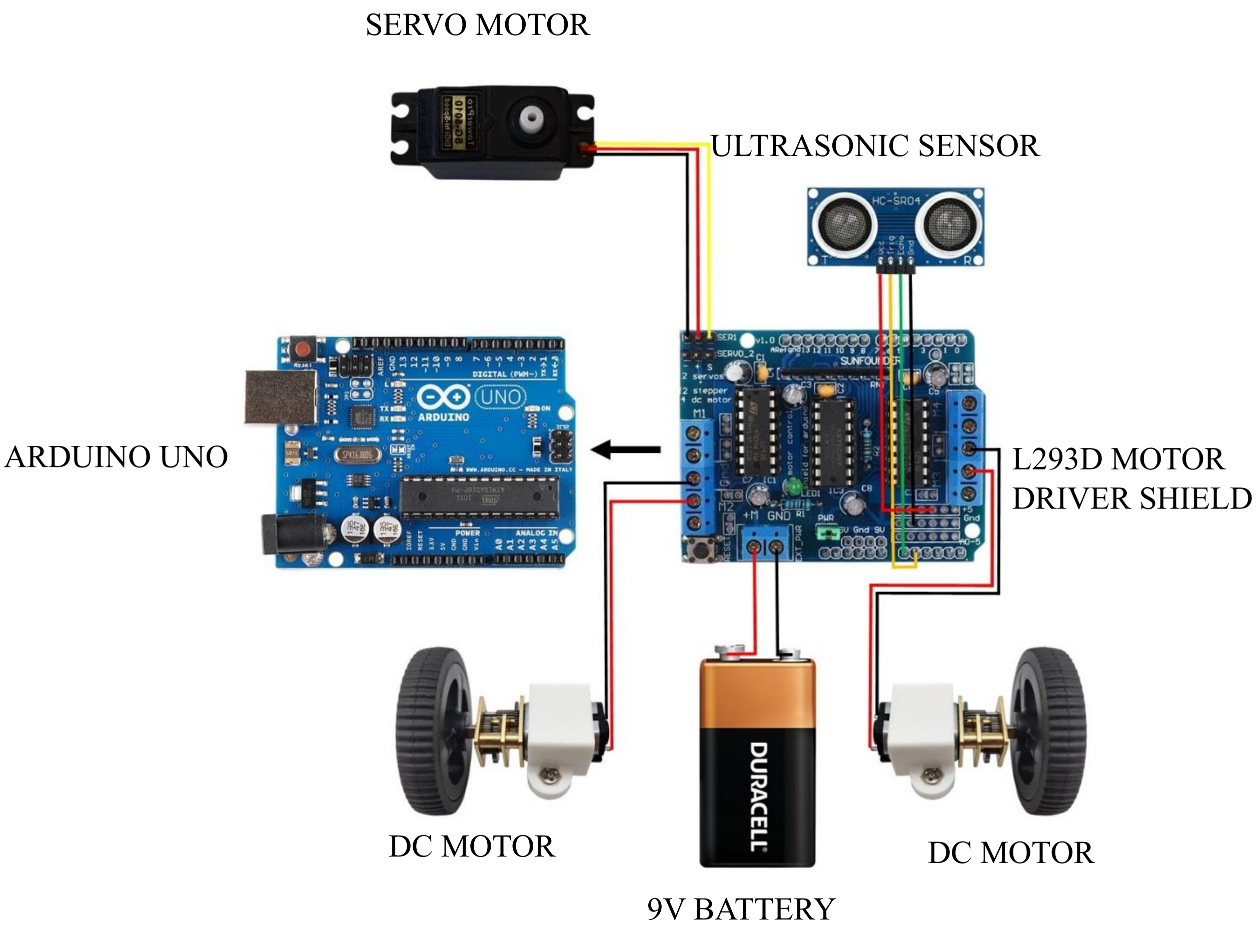
Servo Motor: Controls the movement of the ultrasonic sensor, allowing it to sweep and detect obstacles in different directions.

Ultrasonic Sensor (HC-SR04): Consists of a transmitter (trigger) and a receiver (echo) to measure the distance to obstacles by sending and receiving ultrasonic waves.

DC Motors: Two motors, one for each wheel, to drive the robot in different directions.

9V Battery: Powers the entire robot system.

Circuit Diagram



Project Operation:

The servo motor rotates the ultrasonic sensor, scanning the environment. The Arduino reads distance data from the ultrasonic sensor. If an obstacle is detected, the Arduino sends appropriate signals to the L293D Motor Driver Shield, controlling the DC motors to change the robot's direction and avoid the obstacle.

Intelligence System:

The intelligence in this system lies in the robot's ability to make decisions based on sensor data. The Arduino processes information from the ultrasonic sensor and adjusts the robot's movement, demonstrating a basic form of intelligent decision-making. This project represents a simple example of robotics and automation, showcasing the integration of sensors and actuators to create an autonomous system capable of navigating its environment intelligently.

How to use?

1. The robot must initially be powered by putting either an external power supply or a 9–12-volt battery to the motor driver power input.

2. Finally, place the robot on the ground and let it run on its own.

Source Code

```
#include <AFMotor.h>   // Include the AFMotor library for motor control
#include <NewPing.h>   // Include the NewPing library for ultrasonic sensor
#include <Servo.h>     // Include the Servo library for servo motor control

#define TRIG_PIN A0    // Define the trigger pin for the ultrasonic sensor
#define ECHO_PIN A1    // Define the echo pin for the ultrasonic sensor
```

```
#define MAX_DISTANCE 200   // Define the maximum distance for the ultrasonic sensor
#define MAX_SPEED 250      // Define the maximum speed for the motors


NewPing sonar(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);   // Create a NewPing object named 'sonar' for ultrasonic sensor


AF_DCMotor motor2(2, MOTOR12_1KHZ);   // Create a motor2 object to control the second DC motor
AF_DCMotor motor3(3, MOTOR34_1KHZ);   // Create a motor3 object to control the third DC motor
Servo myservo;   // Create a Servo object named 'myservo' for controlling the servo motor


boolean goesForward=false;   // Initialize a variable to track the robot's forward/backward direction
int distance = 120;          // Initialize a variable to store the current distance from obstacles
int speedSet = 0;            // Initialize a variable to store the motor speed


void setup() {
  myservo.attach(10);   // Attach the servo to pin 10
  myservo.write(115);   // Set the initial position of the servo to 115 degrees
  delay(1000);
  distance = readPing();  // Read the initial distance from the ultrasonic sensor
  delay(100);
  distance = readPing();  // Read the distance again with delays in between
  delay(100);
  distance = readPing();  // Read the distance again with delays in between
  delay(100);
  distance = readPing();  // Read the distance again with delays in between
  delay(100);
}


void loop() {
  int distanceR = 0;  // Initialize a variable to store the distance to the right
  int distanceL =  0;  // Initialize a variable to store the distance to the left
  delay(40);

  if(distance<=15) {   // Check if the robot is too close to an obstacle
    moveStop();        // Stop the robot
    delay(100);
    moveBackward();    // Move the robot backward
    delay(300);
    moveStop();        // Stop the robot
    delay(200);
    distanceR = lookRight();  // Look to the right and get the distance
    delay(200);
    distanceL = lookLeft();   // Look to the left and get the distance
    delay(200);

    if(distanceR >= distanceL) {
      turnRight();    // If the right distance is greater, turn right
      moveStop();     // Stop the robot
    } else {
      turnLeft();     // If the left distance is greater, turn left
      moveStop();     // Stop the robot
    }
  } else {
    moveForward();    // If no obstacle is detected, move the robot forward
  }
  distance = readPing();  // Read the current distance from the ultrasonic sensor
}

int lookRight() {
  myservo.write(50);       // Move the servo to the right position
  delay(500);
  int distance = readPing();   // Read the distance to the right
  delay(100);
  myservo.write(115);      // Reset the servo to the center position
  return distance;
}

int lookLeft() {
  myservo.write(170);      // Move the servo to the left position
  delay(500);
  int distance = readPing();   // Read the distance to the left
  delay(100);
  myservo.write(115);      // Reset the servo to the center position
  return distance;
  delay(100);   // This line is unreachable due to the 'return' statement above
}

int readPing() {
  delay(70);
  int cm = sonar.ping_cm();   // Get the distance in centimeters from the ultrasonic sensor
  if (cm == 0) {
    cm = 250;   // If the distance is 0 (out of range), set it to a large value (250)
  }
  return cm;    // Return the distance value
}

void moveStop() {
  motor2.run(RELEASE);   // Release the second motor to stop it
  motor3.run(RELEASE);   // Release the third motor to stop it
}

void moveForward() {
  if (!goesForward) {
    goesForward = true;   // Set the direction to forward
    motor2.run(FORWARD);  // Run the second motor forward
    motor3.run(FORWARD);  // Run the third motor forward
    for (speedSet = 0; speedSet < MAX_SPEED; speedSet += 2) {
      motor2.setSpeed(speedSet);   // Gradually increase the speed of the second motor
      motor3.setSpeed(speedSet);   // Gradually increase the speed of the third motor
      delay(5);
    }
  }
}

void moveBackward() {
  goesForward = false;     // Set the direction to backward
  motor2.run(BACKWARD);    // Run the second motor backward
  motor3.run(BACKWARD);    // Run the third motor backward
  for (speedSet = 0; speedSet < MAX_SPEED; speedSet += 2) {
    motor2.setSpeed(speedSet);   // Gradually increase the speed of the second motor
    motor3.setSpeed(speedSet);   // Gradually increase the speed of the third motor
    delay(10);
  }
}

void turnRight() {
  motor2.run(FORWARD);    // Run the second motor forward
  motor3.run(BACKWARD);   // Run the third motor backward
  delay(1000);
  motor2.run(FORWARD);    // Run the second motor forward
  motor3.run(FORWARD);    // Run the third motor forward
}

void turnLeft() {
  motor2.run(BACKWARD);   // Run the second motor backward
  motor3.run(FORWARD);    // Run the third motor forward
  delay(1000);
  motor2.run(FORWARD);    // Run the second motor forward
  motor3.run(FORWARD);    // Run the third motor forward
}
```