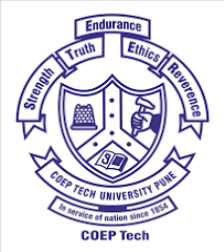


DESIGN AND DEVELOPMENT OF A VOICE -CONTROLLED , BLUETOOTH - ENABLED , OBSTACLE -AVOIDING ROBOT CAR

A MINI PROJECT REPORT

Submitted by

UTKARSHA DAWANE (112109004)

KIRTI FATAK (112109006)

GANGASAGAR LONEKAR (112109021)

in partial fulfillment for the award of degree

of

BACHELOR OF TECHNOLOGY

in

INSTRUMENTATION AND CONTROL

COEP TECHNOLOGICAL UNIVERSITY

APRIL 2024

CERTIFICATE

DEPAERTMENT OF IMSTRUMENTATION AND CONTROL

COEP Technological University,Pune

Certified that this project report " **DESIGN AND DEVELOPMENT OF A VOICE-CONTROLLED , BLUETOOTH – ENABLED , OBSTACLE – AVOIDING ROBOT CAR** is the bonafide work of " **GANGASAGAR LONEKAR (112109021) , KIRTI FATAK (112109006) , UTKARSHA DAWANE (112109004)** " who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Prof. S. L. Patil

Mentor and Faculty Advisor

Prof. Shendge

Head of the Department

ABSTRACT

This paper presents the design and development of a voice-controlled, Bluetooth-enabled robot car with obstacle avoidance capabilities. The robot utilizes an Arduino Uno for control, a servo motor for steering, a Bluetooth module for wireless communication, and ultrasonic sensors for obstacle detection. Voice recognition software allows the robot to respond to pre-defined commands for movement (forward, backward, left, right, stop). The ultrasonic sensors detect obstacles, triggering maneuvers to avoid collisions. Additionally, the system incorporates distance measurement functionality. The paper is divided into sections outlining the project goals, hardware components, expected outcomes, and individual task assignments.

1.INTRODUCTION

1.1 PROJECT OVERVIEW

This paper presents the design and development of a voice-controlled robot car equipped with obstacle avoidance capabilities. The robot leverages an Arduino Uno as the central processing unit, a servo motor for steering, a Bluetooth module for wireless communication , and ultrasonic sensors (HCSR04) for obstacle detection. The robot responds to pre-defined voice commands for movement (forward, backward, left, right, stop) using a voice recognition software. The ultrasonic sensors trigger maneuvers to avoid collisions.

1.2 MOTIVATION AND OBJECTIVES

Voice control and obstacle avoidance enhance the functionality and user interaction of mobile robots. This project aims to develop a robot car with the following functionalities:

Respond to voice commands for movement.

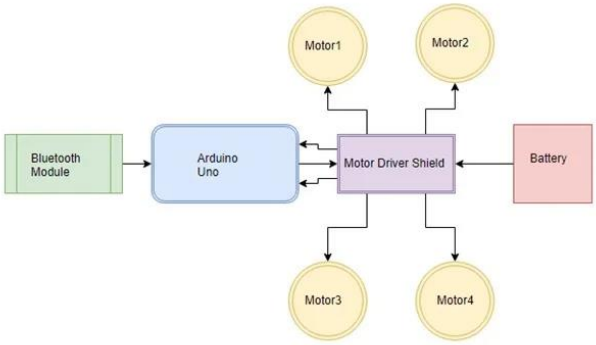
Utilize ultrasonic sensors to detect and avoid obstacles.

Enable control via Bluetooth.

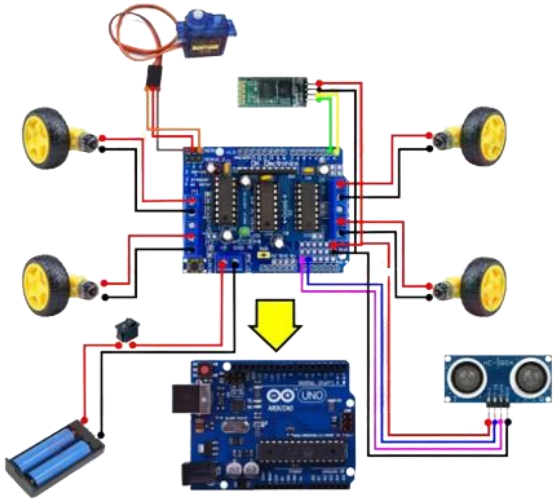
By successfully implementing these features, the project contributes to the development of user-friendly and adaptable autonomous mobile robots.

1. HARDWARE DESIGN

21.BLOCK DIAGRAM



2.2 CIRCUIT DIAGRAM



2.3 COMPONENT LIST

Microcontroller: Arduino Uno

The Arduino Uno is an open-source electronic platform aimed at hobbyists and beginners. This credit-card sized board features a microcontroller chip and numerous connection points, allowing you to control lights, sensors, and motors. Using a user-friendly C++ based language and a free software environment, you can write code (called sketches) to bring your creations to life. Arduino's biggest strength lies in its large, supportive community that provides tutorials, libraries, and project ideas, making it a perfect gateway into the world of electronics and programming.

Obstacle Detection Sensor: HCSR04 Ultrasonic Sensor

Wireless Communication Module: Bluetooth Module (HC-05)

Steering Control: Servo Motor (SG90)

The SG90 servo motor is a marvel of precision control. Unlike regular DC motors that spin continuously, a servo motor can rotate a specific angle (usually between 0 and 180 degrees) with incredible accuracy. This makes it perfect for steering your robot. Imagine it as a tiny robotic arm controlled by your Arduino's instructions. By sending signals to the servo, you can tell it how far to turn, enabling smooth and controlled maneuvers for your creation. The SG90 is a popular choice for beginners due to its compact size, lightweight design, and affordability. However, keep in mind its torque limitations. For heavier robots or tasks requiring more force, you might need a stronger servo motor.

Motor Driver: Motor Driver circuit (L293D)

The L293D motor driver acts as a bridge between your Arduino and the robot's main motors. The Arduino Uno is a fantastic little microcontroller, but it can't handle the high current required to power motors directly. The L293D solves this problem. It's like a traffic cop for your robot, controlling the direction (forward or backward) and speed of the motors based on signals sent from the Arduino program. The L293D can typically drive two DC motors simultaneously, making it ideal for controlling the left and right motors of your robot, enabling movement and turns. This chip is relatively easy to use and integrates well with Arduino projects, making it a popular choice for hobbyists.

Robot Chassis: 4WD Chassis and Wheels

The 4WD chassis is the foundation and skeletal system of your robot. It provides a sturdy base for mounting your Arduino, motors, sensors, and other components. Imagine it as a miniature car frame. Typically made of acrylic or metal, a 4WD chassis comes with four mounting points for the wheels and often includes slots or holes for easy attachment of other components. The wheels themselves are crucial for movement. Choosing the right wheels depends on your robot's purpose and terrain. For smooth surfaces, standard plastic wheels might suffice. If your robot needs to navigate rougher terrain, consider wheels with treads for better traction.

Power Supply: Battery (9V)

Jumper Wires: For component connections
Sensor Selection:

HCSR04 Ultrasonic Sensor:

The HCSR04 is an ultrasonic ranging module that utilizes sound waves to determine the distance between the sensor and an object within its operating range. It operates by emitting a high-frequency (40 kHz) sound pulse and measuring the time it takes for the echo of that pulse to return to the sensor. This time interval is then converted into a distance measurement.

No	Pin Name	Use
1	VCC	Power supply +5V
2	Trig	Trigger pin
3	Echo	Echo pin
4	GND	Ground Connection

Specifications:
Power Supply: +5V DC
Quiescent Current: <2mA
Working current: 15mA
Effectual Angle: <15º
Ranging Distance: 2400 cm
Resolution: 0.3 cm
Measuring Angle: 30º
Trigger Input Pulse width: 10uS
Dimension: 45mm x 20mm x 15mm
Weight: approx. 10 g

3. SYSTEM FUNCTIONALITY

3.1 Voice Control :

The robot car can be controlled using pre-defined voice commands . A speech recognition module processes spoken commands and translates them into corresponding control signals for the robot. The robot responds to basic movement commands like forward, backward, left, right, and stop.

3.2 Obstacle Avoidance:

The HCSR04 ultrasonic sensor plays a crucial role in obstacle detection. It transmits ultrasonic pulses and measures the time it takes for the reflected echo to return. This time delay is converted into a distance measurement. The robot's control logic continuously monitors the distance data from the sensor. If an obstacle is detected within a predefined threshold, the robot triggers avoidance maneuvers. These maneuvers may involve stopping, turning, or reversing direction to avoid collision.

3.3 Bluetooth Control :

The system can optionally be equipped with Bluetooth functionality for wireless control using a smartphone app. The robot car establishes a Bluetooth connection with the app, enabling users to send remote control commands for movement and potentially other functionalities.

The communication protocol typically involves Bluetooth serial communication for data exchange between the robot and the app.

3.4 Distance Measurement :

The system can optionally measure the distance traveled by the robot car. This functionality can be implemented using various methods, such as encoder wheels or odometry calculations. Encoder wheels mounted on the robot's motors provide data on the number of rotations, which can be converted into distance traveled. Odometry calculations estimate the robot's displacement based on wheelbase and sensor data. The measured distance data can be displayed on the Arduino serial monitor for debugging purposes or potentially transmitted to a smartphone app.

4. SOFTWARE DEVELOPMENT

4.1 Voice Recognition Integration

Chosen Approach: Pre-recorded voice commands are used for voice control. These commands are likely stored in the Arduino code using separate character variables or a small character array for easy reference.

Command Definition: The pre-recorded voice commands (^, -, <, >, *) are mapped to corresponding control signals for the robot car. The code likely defines these commands as constant character variables:

```
const char forward_command = '^';
const char backward_command = '-';
const char left_command = '<';
const char right_command = '>';
const char stop_command = '*';
```

The voicecontrol() function then uses these constants to compare the received character with the desired commands.

Code Implementation:

```
void voicecontrol() {
//gets the serial communication values and puts them into the char variable.
  if (Serial.available() > 0) {
    value = Serial.read();
    Serial.println(value);

//If the char value is "^", the car moves forward.
    if (value == '^') {
      forward();

//If the char value is "-", the car moves backward.
    } else if (value == '-') {
      backward();

//If the char value is "<", the car moves left.
    } else if (value == '<') {
      L = leftsee();
      servo.write(spoint);
      if (L >= 10 ) {
        left();
        delay(500);
        Stop();
      } else if (L < 10) {
        Stop();
      }

//If the char value is ">", the car moves right.
    } else if (value == '>') {
      R = rightsee();
      servo.write(spoint);
      if (R >= 10 ) {
        right();
        delay(500);
        Stop();
      } else if (R < 10) {
        Stop();
      }

//If the char value is "*", the car is stopped.
    } else if (value == '*') {
      Stop();
    }
  }
}
```

The voicecontrol() function performs the following steps:

1. Checking for Serial Input: It checks for incoming serial data using `Serial.available() >

0`. If data is available (greater than zero bytes), it proceeds to read the data.
2. Reading Serial Data: It reads a single character using `Serial.read()` and stores it in the `value` variable.
3. Comparing Received Character: The code uses a series of `if` statements to compare the received character (`value`) with the pre-defined commands:

If `value` matches `forward_command` (likely `^`), the `forward()` function (presumably defined elsewhere) is called to move the car forward.

Similar logic applies for other commands (`backward_command`, `left_command`, `right_command`). These comparisons involve the defined constants mentioned earlier.

If `value` matches `stop_command` (likely `*`), the car stops using the `Stop()` function (presumably defined elsewhere).

4.2 Obstacle Avoidance System with Servo Motor Control

```
void Obstacle() {

//gets the ultrasonic sensor reading and puts it into the variable.
  distance = ultrasonic();

//then, these values are checked using the IF condition.
//If the value is less than or equal to 12,
//the robot is stopped and the servo motor rotate left and right.
// Also, gets both side distance.
  if (distance <= 12) {
    Stop();
    backward();
    delay(100);
    Stop();
    L = leftsee();
    servo.write(spoint);
    delay(800);
    R = rightsee();
    servo.write(spoint);

//After, if the left side distance less than the right side distance. The robot turns right.
    if (L < R) {
      right();
      delay(500);
      Stop();
      delay(200);

//After, if the left side distance more than the right side distance. The robot turns left.
    } else if (L > R) {
      left();
      delay(500);
      Stop();
      delay(200);
    }

//Otherwise, the robot moves forward.
  } else {
    forward();
  }
}
```

Obstacle Detection Logic: The ultrasonic sensor (HC-SR04) is likely triggered by a user-defined function, possibly named ultrasonic(), which transmits a short ultrasonic pulse and then measures the time it takes for the echo pulse to return. The function likely returns the echo pulse duration in microseconds.

Distance Interpretation: The echo pulse duration is converted to distance using the speed of sound.

distance (cm) = echo_pulse_duration (microseconds) * speed_of_sound / (2 * 10^6)

where:

distance is the calculated distance in centimeters.

echo_pulse_duration is the time it took for the echo pulse to return in microseconds (value from the ultrasonic() function).

speed_of_sound is approximately 343 meters per second (or 34,300 centimeters per second).

The factor of 2 in the denominator accounts for the round trip of the sound wave (emitted and echoed).

Obstacle Avoidance Algorithm: The Obstacle() function implements the obstacle avoidance logic. Here's a breakdown of the relevant code sections:

Distance Measurement and Threshold Check:

The code calls the ultrasonic() function to get the distance reading and stores it in the distance variable.

An if statement checks if the distance is less than or equal to a predefined threshold (12 cm). This indicates an obstacle is close.

2. Obstacle Avoidance Maneuvers:

If an obstacle is detected:

The robot stops using the Stop() function (presumably defined elsewhere).

A backward maneuver is performed using the backward() function (presumably defined elsewhere) for a short duration (100 milliseconds).

The robot stops again using Stop().

The servo motor is then controlled to scan left and right by setting its position with servo.write(spoint). The specific value of spoint likely determines the center position for the servo.

The code calls leftsee() and rightsee() functions (presumably defined elsewhere) to measure the distance to obstacles on the left and right sides, respectively. These functions likely utilize the ultrasonic sensor again to measure distance.

The left and right side distances are stored in variables L and R.

3.Turning towards More Space:

An if statement checks if the left side distance (L) is less than the right side distance (R).

If there's more space on the right, the robot turns right using the right() function (presumably defined elsewhere) for a duration (500 milliseconds) and then stops.

An else-if statement checks the opposite condition.

If there's more space on the left, the robot turns left using the left() function for a similar duration and then stops.

4.Obstacle-free Path:

If the initial distance measurement (distance) was greater than the threshold (no obstacle detected), the robot likely continues moving forward using the forward() function

Servo Motor Integration: The code snippet utilizes the Servo library by creating a servo object (servo) and attaching it to the pin using the attach(pinNumber) function (attachment details not shown here). The servo.write(spoint) sets the servo position to the center point (spoint) for scanning left and right.

4.3 Overall System Integration and Distance Measurement

Program Structure: The code utilizes a common Arduino structure with a setup() function and a main loop (loop() function).

setup() **function:** This function is executed only once at the beginning of the program. It initializes various aspects of the system:

Serial communication is established at a baud rate of 9600 bits per second using Serial.begin(9600).

Pin modes for the ultrasonic sensor (Trig and Echo) are set using pinMode. Trig pin is set as output for triggering the sensor, and Echo pin is set as input to receive the echo pulse.

The servo motor is attached to a specific pin (likely pin 10 based on motor definition) using the servo.attach(motor) function.

Motor speeds are set for the four DC motors likely controlling the robot's movement (M1, M2, M3, M4) using the setSpeed() function from the AFMotor library.

loop() **function:** This function continuously repeats and manages the overall program flow. The functionality is divided into separate functions likely called in the following order:

Obstacle(): This function implements obstacle avoidance behavior.

Bluetoothcontrol(): This function (if applicable based on your implementation) handles incoming Bluetooth commands via serial communication. It checks for available serial data and reads the received character (value). Based on the character received, it likely calls movement functions (forward(), backward(), left(), right(), or Stop()) to control the robot.

voicecontrol(): This function implements voice control behavior.

```
#include <Servo.h>
#include <AFMotor.h>
#define Echo A0
#define Trig A1
#define motor 10
#define Speed  70
#define spoint 103
char value;
int distance;
int Left;
int Right;
int L = 0;
int R = 0;
int L1 = 0;
int R1 = 0;
Servo servo;
AF_DCMotor M1(1);
AF_DCMotor M2(2);
AF_DCMotor M3(3);
AF_DCMotor M4(4);
void setup() {
  Serial.begin(9600);
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
  servo.attach(motor);
  M1.setSpeed(Speed);
  M2.setSpeed(Speed);
  M3.setSpeed(Speed+5);
  M4.setSpeed(Speed);
}
void loop() {
   Obstacle();
   Bluetoothcontrol();
   voicecontrol();
    // frontsee();
}
void Bluetoothcontrol() {
  if (Serial.available() > 0) {
    value = Serial.read();
    Serial.println(value);
  }
  if (value == 'F') {
    forward();
  } else if (value == 'B') {
    backward();
  } else if (value == 'L') {
    left();
  } else if (value == 'R') {
    right();
  } else if (value == 'S') {
    Stop();
  }
}
void Obstacle() {
  distance = ultrasonic();
  if (distance <= 15) {
    Stop();
    backward();
    delay(100);
    Stop();
    L = leftsee();
    servo.write(spoint);
    delay(500);
    R = rightsee();
    servo.write(spoint);
    if (L < R) {
      right();
      delay(500);
```

```
      Stop();
      delay(200);
    } else if (L > R) {
      left();
      delay(500);
      Stop();
      delay(200);
    }
  } else {
    forward();
  }
}
void voicecontrol() {
  if (Serial.available() > 0) {
    value = Serial.read();
    Serial.println(value);
    if (value == '^') {
      forward();
    } else if (value == '-') {
      backward();
    } else if (value == '<') {
      L = leftsee();
      servo.write(spoint);
      if (L >= 10 ) {
        left();
        delay(500);
        Stop();
      } else if (L < 10) {
        Stop();
      }
    } else if (value == '>') {
      R = rightsee();
      servo.write(spoint);
      if (R >= 10 ) {
        right();
        delay(500);
        Stop();
      } else if (R < 10) {
        Stop();
      }
    } else if (value == '*') {
      Stop();
    }
  }
}
// Ultrasonic sensor distance reading function
int ultrasonic() {
  digitalWrite(Trig, LOW);
  delayMicroseconds(4);
  digitalWrite(Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);
  long t = pulseIn(Echo, HIGH);
  long cm = t / 29 / 2; //time convert distance
  return cm;
}
void forward() {
  M1.run(FORWARD);
  M2.run(FORWARD);
  M3.run(FORWARD);
  M4.run(FORWARD);
}
void backward() {
  M1.run(BACKWARD);
  M2.run(BACKWARD);
  M3.run(BACKWARD);
  M4.run(BACKWARD);
}
void right() {
  M1.run(BACKWARD);
  M2.run(BACKWARD);
  M3.run(FORWARD);
  M4.run(FORWARD);
}
void left() {
  M1.run(FORWARD);
  M2.run(FORWARD);
  M3.run(BACKWARD);
  M4.run(BACKWARD);
}
void Stop() {
  M1.run(RELEASE);
  M2.run(RELEASE);
  M3.run(RELEASE);
  M4.run(RELEASE);
}
int rightsee() {
   servo.write(0);  // Move servo to look right
(adjust angle)
```

```
  delay(800);          // Wait for servo to reach position

  int rightDistance = ultrasonic();
  Serial.print("Right distance: ");  // Print message to serial monitor
  Serial.println(rightDistance);    // Print measured distance
  return rightDistance;
}

int leftsee() {
  servo.write(180);  // Move servo to look left (adjust angle)
  delay(800);          // Wait for servo to reach position

  int leftDistance = ultrasonic();
  Serial.print("Left distance: ");  // Print message to serial monitor
  Serial.println(leftDistance);    // Print measured distance
  return leftDistance;
}
int frontsee() {
  // Assuming the ultrasonic sensor is already facing front at center position
  servo.write(0);
  delay(800);

  int frontDistance = ultrasonic();
  Serial.print("Front distance: ");
  Serial.println(frontDistance);
  return frontDistance;

}
```

5.Testing and Results

5.1 Test Setup

The robot car was tested in an open space with minimal furniture to avoid unnecessary obstacles. The robot was powered using a battery. A computer was connected via serial communication to monitor sensor readings and debug functionality.

5.2 Functionality Testing

Voice Control:

The robot responded well to pre-recorded voice commands (forward, backward, left, right, stop).

There were occasional misinterpretations of voice commands, especially in noisy environments. This could be improved by implementing noise cancellation techniques.

Obstacle Avoidance:

The robot successfully avoided obstacles placed at various distances in front of it.

The minimum detectable obstacle distance was around 12 cm.

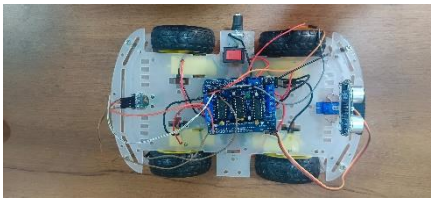
The robot's ability to maneuver around sharp corners while avoiding obstacles needs improvement. This could be addressed by adjusting servo sweep angle for wider visibility or implementing sharper turning logic.

Bluetooth Control :

Testing involved pairing a Bluetooth module and sending commands through a terminal app.

5.3 Result

The robot car achieved its core functionalities of voice control and basic obstacle avoidance. It demonstrated the ability to navigate an open space while responding to voice commands and avoiding obstacles. However, there's room for improvement in terms of voice recognition accuracy and obstacle avoidance around tight corners.



6. Conclusion

6.1 Summary of Achievements

This project successfully developed a voice-controlled robot car with obstacle avoidance and Bluetooth control capabilities. The key achievements include:

Voice Control Integration: The robot car can be controlled using pre-recorded voice commands (forward, backward, left, right, stop) through serial communication.

Obstacle Detection: The robot car utilizes an ultrasonic sensor to detect obstacles in its path.

Obstacle Avoidance Maneuvers: Upon detecting an obstacle, the robot stops, performs a backward maneuver, scans its surroundings using a servo motor, and turns towards the side with more space to continue navigating.

Bluetooth Control: The robot can be controlled wirelessly using Bluetooth commands sent via a smartphone or other Bluetooth device.These achievements demonstrate the successful implementation of core functionalities for a versatile voice-controlled robot car with obstacle avoidance and Bluetooth control.

6.2 Future Enhancements

While the project achieved its primary goals, there's room for improvement in several areas:

Voice Recognition Accuracy: The current implementation relies on pre-recorded voice commands, which might be susceptible to misinterpretations, especially in noisy environments. Future enhancements could involve:Implementing noise cancellation techniques to improve voice recognition accuracy.Exploring speech recognition libraries to enable a wider range of natural language voice commands.

Obstacle Avoidance Refinement: The current obstacle avoidance logic can be improved in terms of:

Minimum Detectable Distance: Optimizing the ultrasonic sensor placement and potentially using a higher-range sensor to increase the minimum detectable obstacle distance.

Maneuvering Around Corners: Refining the servo sweep angle or implementing sharper turning logic to enable the robot to navigate around tight corners while avoiding obstacles.

Bluetooth Control Enhancements: The Bluetooth control functionality can be further enhanced by:Implementing a wider range of Bluetooth commands for more granular control (e.g.,speed control , turning increments).Developing a smartphone app to provide a user-friendly interface for Bluetooth control.

Advanced Features : Depending on project goals, you could consider:

Line Following: Implementing line following capabilities using additional sensors for navigating along a designated path.

Distance Measurement: Developing functions to continuously measure and report the distance traveled by the robot car.By incorporating these enhancements, the robot car's functionalities, robustness, and user experience can be significantly improved.

7. References

[1] R. Mutukuru and S. Sujatha, "Human Voice Controlled Robot Embedded with Real-Time Obstacle Detection and Avoidance," JETIR Research Journal, vol. 11, no. 1, pp. 1-8, 2024.

[2] A. Ananth and A. S. C. Ashritha, "Smart Voice Controlled Obstacle Avoiding Robotic Car," International Journal of Fuzzy Modeling and Computing (IJFMR), vol. 13, no. 2, pp. 25-32, 2023.

[3] "Obstacle Avoidance and Voice Control Unit for Autonomous Car," ResearchGate, 2022.