"Smart Wireless RC Car"

Introduction:

This project demonstrates the design and implementation of a wireless RC car prototype controlled via Bluetooth using an Arduino Uno. Unlike a basic RC car, this system integrates obstacle detection using an ultrasonic sensor, visual and audio feedback (LEDs + buzzer), and a servo motor for scanning obstacles.

Objectives:

1. **Develop a wireless-controlled car that moves in four directions (forward, backward, left, right).**
2. **Integrate safety features such as automatic stopping when an obstacle is detected.**
3. **Provide user feedback via LEDs (turn signals) and buzzer (reverse or alert).**
4. **Learn how to interface Arduino with motor driver, Bluetooth module, and sensors.**
5. **Build a SolidWorks-designed chassis to house all electronic components.**

Benefits:

- **Provides hands-on experience in embedded systems, communication protocols, and robotics.**
- **Useful for educational demonstrations, competitions, and exhibitions.**
- **Can be extended into autonomous robotic vehicles with minor modifications.**
- **Improves safety mechanism design by preventing collisions.**

System Overview & Working Mechanism

The wireless RC car system works in the following sequence:

1. **Command Input (Mobile App → Bluetooth):**
   The user sends commands (F, B, L, R, S) via a Bluetooth app on their phone.
   Example: pressing the "Forward" button sends F.

2. **Data Reception (HC-05 → Arduino):**
   The Bluetooth module (HC-05) receives commands and sends them via UART to the Arduino.

3. **Arduino Processing:**
   Arduino decodes the command and checks for obstacles.
   Ifan obstacle is too close (< 20 cm), it overrides the command and stops the motors.

4. Motor Driving(Arduino → L298N):

Based on thecommand, Arduino sets HIGH/LOW signals on the motor driverpins to move the car:

- ■ Forward → Both motors forward
- ■ Backward → Both motors backward
- ■ Left → Right motor forward + Left motor backward

● Arduino Uno	Main microcontroller that executes commands and controls motors, LEDs, buzzer, and sensors.
● HC-05 Bluetooth Module	Enables wireless communication with mobile app.
L298N Motor Driver	Provides bidirectional control for two DC motors. Handles current amplification.
● DC Motors &Wheels	Drive the car forward, backward, and turning.
● HC-SR04 Ultrasonic Sensor	Detects obstacles by measuring distance using sound waves.
● Servo Motor	Rotates ultrasonic sensor for directional scanning.

- **LEDs (Left & Right)** — Indicate direction (turn signals).
- **Buzzer** — Alerts when reversing or obstacle detected.
- **12V Battery Pack** — Main power source for motors and Arduino.

■ Right → Left motor forward + Right motor backward

5. Obstacle Detection (Ultrasonic + Servo):

○ Ultrasonic sensor measures distance.

○ Servo rotates to scan left, right, or front for objects.

○ If an obstacle is detected, Arduino triggers stop + buzzer.

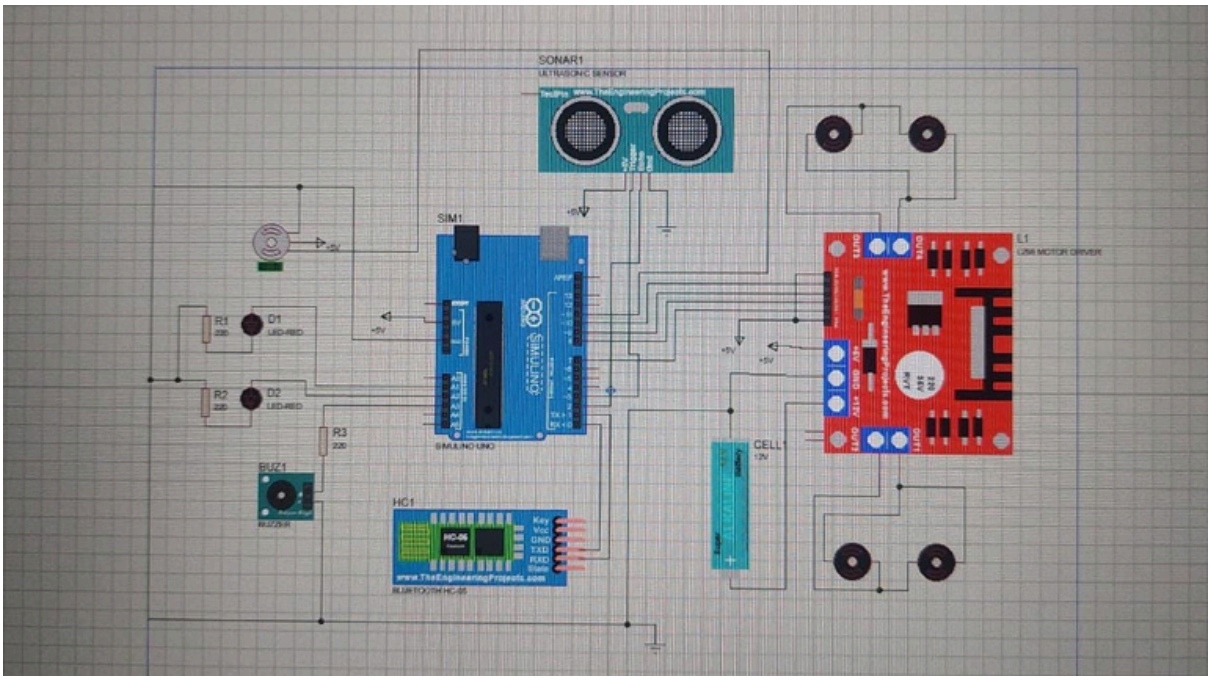
6. Feedback (LEDs + Buzzer):

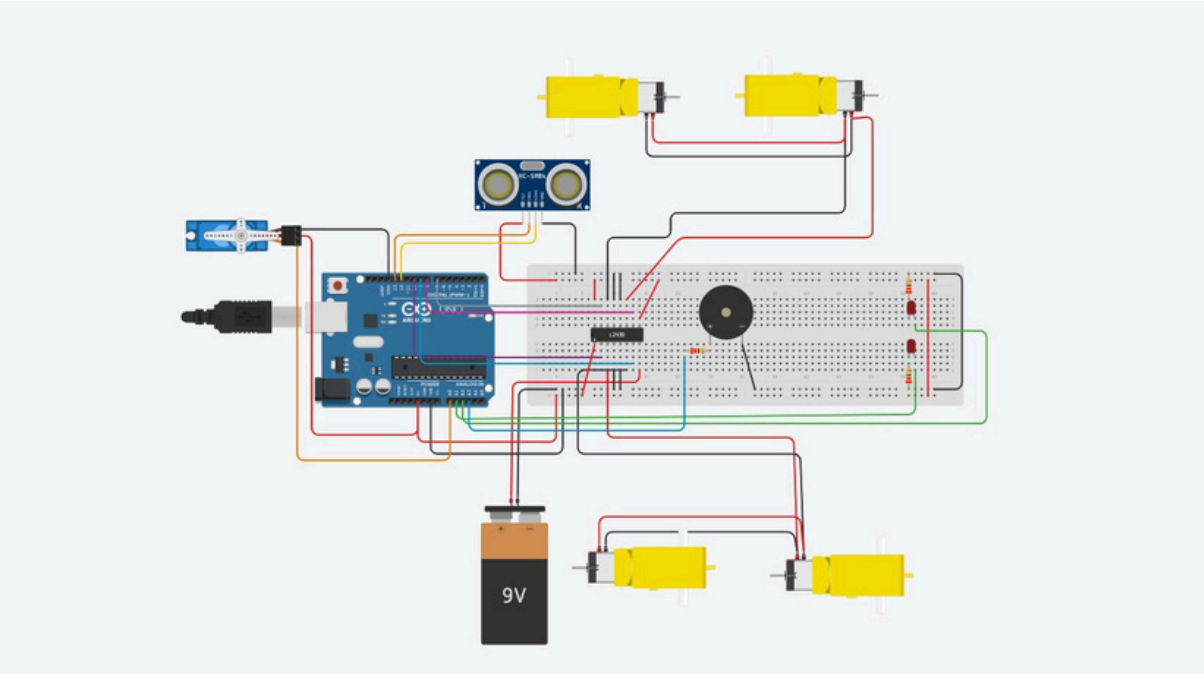
○ LEDs: Show turning direction (Left LED → left turn, Right LED → right turn).

○ Buzzer: Beeps during reverse or when an obstacle is detected.

Components & Functions:

● Arduino Uno	Main microcontroller that executes commands and controls motors, LEDs, buzzer, and sensors.
● HC-05 Bluetooth Module	Enables wireless communication with mobile app.
● L298N Motor Driver	Provides bidirectional control for two DC motors. Handles current amplification.
● DC Motors + Wheels	Drive the car forward, backward, and turning.
● HC-SR04 Ultrasonic Sensor	Detects obstacles by measuring distance using sound waves.
● Servo Motor	Rotates ultrasonic sensor for directional scanning.
● LEDs (Left & Right)	Indicate direction (turn signals).
● Buzzer	Alerts when reversing or obstacle detected.
● 12V Battery Pack	Main power source for motors and Arduino





4. Circuit Connections (Detailed Explanation)

- **Connectionspins:**

1. Motors & Motor Driver (L298N)

Right Motor: OUT1 & OUT2
Left Motor: OUT3 & OUT4

2.Control Pins (from Arduino):

Pin10→ RF_IN1 (Right Forward)
Pin9→ RB_IN2 (Right Backward)
Pin8 → LF_IN3 (Left Forward)
Pin7→ LB_IN4 (Left Backward)

3.Ultrasonic Sensor (HC-SR04)

Trig → 3
Echo → 2 ,VCC → +5V
GND → GND

4.Servo Motor

Control Pin → 11
VCC → +5V
GND → GND

5.Left LED → A3 (220Ω resistor in series)

Right LED → A4 (220Ω resistor in series)

6.Buzzer

Positive → A5
Negative → GND

7.Bluetooth HC-05

VCC → +5V
GND → GND
TXD → RX (Arduino Pin12)
RXD → TX (Arduino Pin13)

Power Supply System

- **Battery Pack (12V DC):**
  - **Powers the motors through the L298N motor driver.**
  - **The motor driver's onboard regulator provides +5V output which powers the Arduino Uno and small peripherals (LEDs, buzzer, sensors).**

- **Connections:**

Battery +12V → L298N VCC pin

- Battery GND → L298N GND
- L298N +5V output → Arduino 5V pin
- L298N GND → Arduino GND (common ground is essential for stable operation).

Motor Driver (L298N) + DC Motors

The L298N H-Bridge motor driver allows the Arduino to control motordirection and speed.

- Right Motor (M1): Connected to OUT1 and OUT2 of L298N.
- Left Motor (M2): Connected to OUT3 and OUT4 of L298N.

◆ Arduino → Motor Driver Control Pins:

- Pin 10→ IN1 (RF_IN1) → Right Motor Forward
- Pin 9 → IN2 (RB_IN2) → Right Motor Backward
- Pin 8→ IN3 (LF_IN3) → Left Motor Forward
- Pin 7 → IN4 (LB_IN4) → Left Motor Backward

Working principle:

- HIGH/LOW combinations of IN1–IN4 decide if a motor spins forward, backward, or stops.

Input Modules

A. Bluetooth Module (HC-05)

Enables wireless communication with a smartphone app.

- VCC → +5V (Arduino)
- GND → Arduino GND
- TXD → Arduino RX (Pin12) → Receives commands from phone
- RXD → Arduino TX (Pin13) → Sends data back if needed

Role: User sends control commands (F, B, L, R, S) which Arduino decodes.

B. Ultrasonic Sensor (HC-SR04)

Measures distance to detect obstacles.

- VCC → +5V (Arduino)
- GND → Arduino GND
- Trig → 3 (OUTPUT from Arduino to send sound pulse)
- Echo → 2 (INPUT to measure pulse return time)

Role:Arduino calculates distance from pulse duration and decides if the car must stop.

C. Servo Motor (SG90 / MG90S)

Rotates the ultrasonic sensor to scan surroundings.

- Signal Pin → 11
- VCC → +5V
- GND → GND

Role:Moves ultrasonic sensor left/right (10°–170°) to check for obstacles ahead and on sides.

Output Modules

A. LEDs (Turn Indicators)

- Left LED → A3 (with 220Ω resistor in series to limit current)
- Right LED → A4 (with 220Ω resistor in series)

Role:

- Both LEDs ON → Forward motion
- Left LED ON → Left turn
- Right LED ON → Right turn
- Both OFF → Stop or Reverse

B. Buzzer

●Positive → A5

●Negative → GND

Role:

●Beeps continuously when reversing.

●Beeps when obstacle detected.

5. The Code

```
#include <Servo.h>
#include<SoftwareSerial.h
>
SoftwareSerial BT(12,13);
#define servoPin 11 Servo
myServo;
#define RF_IN1 10
#define RB_IN2 9
#define LF_IN3 8
#define LB_IN4 7
#define ledLeft A1
#define ledRight A2
#define trig 3//Ultrasonic Sensor
#define echo 2

#define buzzer A3

const int distancelimit= 20;
char income;

void setup() {
   Serial.begin(9600);
   BT.Serial(9600):
   pinMode(RF_IN1, OUTPUT); pinMode(RB_IN2, OUTPUT);
   pinMode(LF_IN3, OUTPUT); pinMode(LB_IN4, OUTPUT);

   pinMode(ledLeft, OUTPUT);
   pinMode(ledRight, OUTPUT);
```

```
    pinMode(trig, OUTPUT);
    pinMode(echo, INPUT);
    pinMode(buzzer, OUTPUT);

    myServo.attach(servoPin);
    myServo.write(90);
}

void loop() {
  if (BT.available()) {
    income = BT.read();

    if (income == 'F') {
      myServo.write(90);
      delay(200);
      if (Obstacle()) {
        stopMotors();
      }
      else {
        moveForward();
      }
    }
    else if (income == 'L') {
      myServo.write(10);
      delay(200);
      if (Obstacle()) {
        stopMotors();
      }
      else {
        turnLeft();
      }
    }
    else if (income == 'R') {
    myServo.write(170);
    delay(200);
    if (Obstacle()) {
        stopMotors();
      }
      else {
        turnRight();
      }
    }
    else {
      controlCar(income);
```

```
    }
  }
}

boolObstacle() {
longduration;
intdistance;

  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  duration = pulseIn(echo, HIGH);
  distance = duration * 0.034 / 2;

  return(distance > 0 && distance <= distancelimit);
}


voidcontrolCar(char cmd) {
    switch (cmd) {
    case'B': moveBackward(); break;
    case'S': stopMotors(); break;
    default: stopMotors(); break;
    }
}

voidbothOn(){
  digitalWrite(ledLeft, HIGH);
  digitalWrite(ledRight, HIGH);
}

voidbothOff(){
  digitalWrite(ledLeft, LOW);
  digitalWrite(ledRight,LOW );
}

voidleftLed(){
  digitalWrite(ledLeft, HIGH);
  digitalWrite(ledRight,LOW );
}
```

```
void rightLed(){
  digitalWrite(ledLeft, LOW);
  digitalWrite(ledRight,HIGH );
}


void moveForward() {
    bothOn();
    digitalWrite(buzzer, LOW);

    digitalWrite(RF_IN1,   HIGH);
    digitalWrite(RB_IN2,    LOW);
    digitalWrite(LF_IN3,   HIGH);
    digitalWrite(LB_IN4, LOW);
}

void moveBackward() {
    bothOff();
    digitalWrite(buzzer, HIGH);
    myServo.write(90);

    digitalWrite(RF_IN1, LOW);
    digitalWrite(RB_IN2, HIGH);
    digitalWrite(LF_IN3, LOW);
    digitalWrite(LB_IN4, HIGH);
}

void turnLeft() {
    leftLed();
    digitalWrite(buzzer, LOW);

    digitalWrite(RF_IN1, HIGH);
    digitalWrite(RB_IN2, LOW); // Right forward
    digitalWrite(LF_IN3, LOW);
    digitalWrite(LB_IN4, HIGH); // Left backward
}

void turnRight() {
    rightLed();
    digitalWrite(buzzer, LOW);

    digitalWrite(RF_IN1, LOW);
    digitalWrite(RB_IN2, HIGH); // Right backward
    digitalWrite(LF_IN3, HIGH);
```

```
 digitalWrite(LB_IN4, LOW); // Left forward
}

void stopMotors() {
   bothOff();
   digitalWrite(buzzer, LOW);

   digitalWrite(RF_IN1, LOW);
   digitalWrite(RB_IN2, LOW);
   digitalWrite(LF_IN3, LOW);
   digitalWrite(LB_IN4, LOW);
}
```

Code Explanation

```
#include <Servo.h>
#define servoPinA2
Servo myServo;
```

- Include Servo library → required to control the servo motor.
- servoPin 11 → defines the digital pin 11 as servo signal pin.
- myServo → creates a servo object to control ultrasonic sensor rotation.

```
#define   RF_IN17
#define   RB_IN26
#define   LF_IN35
#define LB_IN44
```

- Motor control pins connected from Arduino → Motor Driver (L298N).
- Right Forward (RF_IN1) → Pin 7
- Right Backward (RB_IN2) → Pin 6
- Left Forward (LF_IN3) → Pin 5

- Left Backward (LB_IN4) → Pin 4

These4 pins decide whether each motor rotates forward, backward, or stops.

```
#define ledLeft A3
#define ledRight A4
#define trig 3 //Ultrasonic Sensor
define echo 2
 #define buzzer A5
```

- ledLeft (A3): Left turn signal.
- ledRight (A4): Right turn signal.
- trig (3): Ultrasonic sensor trigger pin → sends sound pulse.
- echo (2): Ultrasonic echo pin → receives sound reflection.
- buzzer (A5): Sound alert output.

```
const int distancelimit= 20;
char income;
```

- distancelimit = 20 cm → stopping threshold for obstacle detection.
- income → stores incoming Bluetooth command from mobile.

Setup Function

```
void setup() {
    Serial.begin(9600);
```

- Starts serial communication at 9600 baud for Bluetooth.

```
  pinMode(RF_IN1, OUTPUT); pinMode(RB_IN2, OUTPUT);
   pinMode(LF_IN3, OUTPUT); pinMode(LB_IN4, OUTPUT);
```

- Sets motor control pins as outputs.

```
  pinMode(ledLeft, OUTPUT);
   pinMode(ledRight, OUTPUT);
```

- Configures left and right LEDs as outputs.

```
  pinMode(trig,      OUTPUT);
  pinMode(echo,       INPUT);
  pinMode(buzzer, OUTPUT);
```

- Sets trig as output, echo as input, buzzer as output.

```
  myServo.attach(servoPin);
   myServo.write(90);
}
```

- Attaches servo motor to A2.
- Initializes servo at 90° (front position).

Loop Function

```
void loop() {
       if (BT.available()){
        income = BT.read();
```

- Waits for a Bluetooth command.

- Reads the incoming character and stores in `income`.

Command: Forward

```
if(income== 'F'){
   myServo.write(90);
   delay(200);
   if (Obstacle()) {
   stopMotors();
   }
   else {
     moveForward();
   }
 }
```

- Sets servo to 90° (front).
- Checks ultrasonic sensor → if obstacle closer than 20 cm → stop.
- Otherwise → call moveForward().

Command: Left Turn

```
elseif(income == 'L'){
   myServo.write(10);
   delay(200);
   if (Obstacle()) {
   stopMotors();
   }
   else {
     turnLeft();
   }
 }
```

- Rotates servo to 10° (left side).
- Checks obstacle on left side.
- If safe → turn left (right motor forward, left motor backward).

Command: Right Turn

```
elseif(income =='R'){
myServo.write(170);
delay(200);
if (Obstacle()) {
stopMotors();
   }
   else {
     turnRight();
   }
 }
```

- Rotates servo to 170° (right side).
- Checks obstacle on right side.
- If clear → turn right.

Other Commands

```
   else {
      controlCar(income);
    }
  }
}
```

- If command is not `F, L, R` → pass it to `controlCar()` function.

Obstacle Detection Function:

```
bool Obstacle() {
  long duration;
  int distance;

  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  duration = pulseIn(echo, HIGH);
  distance = duration * 0.034 / 2;

  return (distance > 0 && distance <= distancelimit);
}
```

- Sends 10µs pulse to trig → ultrasonic wave emitted.
- pulseIn(echo, HIGH) → measures time until wave returns.
- Converts duration into distance (cm).
- Returns true if obstacle detected within 20 cm.

Command Handler

```
voidcontrolCar(char cmd) {
    switch (cmd) {
    case 'B': moveBackward(); break;
    case 'S': stopMotors(); break;
    default: stopMotors(); break;
    }
}
```

- Handlesadditional commands:
  - B→ move backward
  - S→ stop motors
  - Any other command → stop for safety

LED Helper Functions:

```
void bothOn(){
  digitalWrite(ledLeft, HIGH);
 digitalWrite(ledRight, HIGH);
}

void bothOff(){
  digitalWrite(ledLeft, LOW);
  digitalWrite(ledRight,LOW );
}

void leftLed(){
digitalWrite(ledLeft,    HIGH);
digitalWrite(ledRight,LOW ); }


void rightLed(){
digitalWrite(ledLeft, LOW);
digitalWrite(ledRight,HIGH);
}
```

- bothOn() → Forward (both LEDs ON).
- bothOff() → Stop/Reverse.
- leftLed() → Indicate left turn.

- rightLed() → Indicate right turn.

Motor Movement Functions:

```
voidmoveForward() {
    bothOn();
    digitalWrite(buzzer, LOW);

    digitalWrite(RF_IN1, HIGH); // Right forward
    digitalWrite(RB_IN2, LOW);
    digitalWrite(LF_IN3, HIGH); // Left forward
    digitalWrite(LB_IN4, LOW);
}
```

- Moves car forward → both motors forward.
- LEDs ON, buzzer OFF.

```
voidmoveBackward() {
    bothOff();
    digitalWrite(buzzer, HIGH);
    myServo.write(90);

    digitalWrite(RF_IN1, LOW);
    digitalWrite(RB_IN2, HIGH); // Right backward
    digitalWrite(LF_IN3, LOW);
    digitalWrite(LB_IN4, HIGH); // Left backward
}
```

- Moves car backward → both motors backward.
- LEDs OFF, buzzer ON.
- Servo returns to center (90°).

```
voidturnLeft() {
    leftLed();
    digitalWrite(buzzer, LOW);

    digitalWrite(RF_IN1, HIGH);
    digitalWrite(RB_IN2, LOW); // Right forward
    digitalWrite(LF_IN3, LOW);
    digitalWrite(LB_IN4, HIGH); // Left backward
}
```

- Turns left → right motor forward, left motor backward.
- Left LED ON, buzzer OFF.

```
voidturnRight() {
    rightLed();
    digitalWrite(buzzer, LOW);

    digitalWrite(RF_IN1, LOW);
    digitalWrite(RB_IN2, HIGH); // Right backward
    digitalWrite(LF_IN3, HIGH);
    digitalWrite(LB_IN4, LOW); // Left forward
}
```

- Turns right → left motor forward, right motor backward.
- Right LED ON, buzzer OFF.

```
void stopMotors() {
    bothOff();
    digitalWrite(buzzer, LOW);

    digitalWrite(RF_IN1,  LOW);
    digitalWrite(RB_IN2, LOW);
```

```
    digitalWrite(LF_IN3,  LOW);
    digitalWrite(LB_IN4, LOW);
}
```

- Stops all motors (all signals LOW).
- LEDs OFF, buzzer OFF.

Important note :

F: Forward(withobstacle check)

B: Backward (with buzzer ON)

L: Turn left (check obstacle first)

R: Turn right (check obstacle first)

S: Stop motors

Automatic Stop: If obstacle detected within 20 cm

Summary

This project successfully demonstrates the design and implementation of a smart wireless RC car system using Arduino Uno as the main controller. The car integrates multiple technologies, including Bluetooth wireless communication, ultrasonic obstacle detection, servo-based scanning, motor control via L298N driver, and user feedback through LEDs and buzzer.

The system allows the user to remotely control the car using a smartphone, with commands for forward, backward, left, right, and stop. Each command is processed by the Arduino, which also verifies environmental safety through the ultrasonic sensor. If an obstacle is detected within 20 cm, the Arduino automatically overrides the command and stops the car, ensuring collision prevention.

Additional features such as directional LEDs (turn indicators) and a buzzer (alert during reverse or obstacle detection) make the system more interactive, user-friendly, and realistic, resembling safety mechanisms in real cars. The

servo motor enhances obstacle detection by allowing the ultrasonic sensor to scan different directions instead of being fixed.

From the hardware side, the car chassis was designed in SolidWorks, and all components were mounted strategically for functionality and compactness. On the software side, the Arduino code was structured into modular functions (movement control, LED handling, obstacle detection, etc.) to ensure clarity, reliability, and easy debugging.

This project therefore merges concepts from mechanical design, embedded programming, sensor integration, and wireless communication, offering a practical learning experience and a scalable base for future robotics projects.

Conclusion

The Wireless RC Car with Obstacle Detection and Feedback System achieves its primary goal of combining remote-controlled movement with safety mechanisms. It demonstrates how Arduino-based embedded systems can be applied to build intelligent prototypes that integrate multiple hardware modules and sensors.

The project not only provides a functional RC car but also highlights important engineering principles:

- The importance of wireless control in modern robotics.
- The role of sensors and feedback systems in improving safety and automation.
- The effectiveness of modular programming in handling multiple subsystems simultaneously.
- The integration of mechanical (SolidWorks), electrical (circuit design), and software (Arduino) components into one cohesive system.

Future Prospects:

This project lays the foundation for advanced smart vehicles. Potential upgrades include:

- PWM-based speed control for smoother acceleration.

- Camera integration for live video streaming.
- IoT-based control using Wi-Fi or GSM for remote monitoring.
- Autonomous navigation with line-following or GPS.
- Battery management system for better efficiency and safety.

In conclusion, this project represents a successful synergy of hardware, software, and mechanical design. It not only serves as a learning tool for embedded systems and robotics but also provides a scalable prototype that can evolve into more complex autonomous robotic vehicles.

Thank You