Arduino Bluetooth Controlled Car 🚘

Project Overview

The Arduino Bluetooth Controlled Car is a wireless robotic vehicle controlled using a smartphone via Bluetooth communication. It utilizes an Arduino Uno, an HC-05 Bluetooth module, and an L293D motor driver to receive movement commands and drive DC motors accordingly. The car can move forward, backward, left, right, and diagonally, making it a versatile DIY robotics project.

Components Required

Arduino Uno R3 – Microcontroller for processing commands.

L293D Motor Driver Module – Controls the speed and direction of the motors.

HC-05 Bluetooth Module – Enables wireless communication with a smartphone.

DC BO Motors (4) with Wheels – Provides movement capability.

Li-ion Battery with Connector – Powers the car.

Working Principle

The HC-05 Bluetooth module pairs with a smartphone.

A mobile app (like Arduino Bluetooth Controller) sends movement commands ('F' for forward, 'B' for backward, etc.).

The Arduino receives the Bluetooth signals and processes them.

Based on the command, the L293D motor driver controls the four DC motors to move in the desired direction.

Code Breakdown

1. Motor Initialization

The AFMotor library is used to control the motors connected to the L293D motor driver.

```
#include <AFMotor.h>

// motor pins

AF_DCMotor motor1(1, MOTOR12_1KHZ);

AF_DCMotor motor2(2, MOTOR12_1KHZ);

AF_DCMotor motor3(3, MOTOR34_1KHZ);

AF_DCMotor motor4(4, MOTOR34_1KHZ);
```

Each AF_DCMotor object represents one of the four DC motors. The MOTOR12_1KHZ and MOTOR34_1KHZ define the motor PWM frequency.

2. Defining Speed and Serial Communication

```
const int MAX_SPEED = 500;
const int REDUCED_SPEED = 500 / 3.1;
const int BAUD_RATE = 9600;
int val;
```

MAX_SPEED: The highest speed for motors (500 PWM).

REDUCED_SPEED: Lower speed used for diagonal movement.

BAUD_RATE: 9600 is the standard speed for Bluetooth communication.

3. Setup Function

```
void setup() {
  Serial.begin(BAUD_RATE);  // Sets baud rate to the Bluetooth module
}
```

Initializes serial communication for receiving Bluetooth commands.

4. Loop Function (Reading Bluetooth Commands)

```
void loop() {
  if (Serial.available() > 0) {
    val = Serial.read();
    Stop();  // Initialize with motors stopped
    switch (val) {
      case 'F': forward(); break;
      case 'B': back(); break;
      case 'L': left(); break;
      case 'R': right(); break;
      case 'I': topright(); break;
      case 'J': topleft(); break;
      case 'K': bottomright(); break;
      case 'M': bottomleft(); break;
      case 'T': Stop(); break;
    }
  }
}
```

Serial.available() checks for incoming data from the Bluetooth module.

The received character (val) determines the movement direction using a switch case.

The Stop() function is called before any movement to ensure the car stops before changing direction.

5. Movement Functions

Each function sets the motor speeds and directions accordingly.

Forward Movement

```
void forward() {
  setMotorsSpeed(MAX_SPEED, FORWARD);
}
```

All four motors move forward at full speed.

Backward Movement

```
void back() {
  setMotorsSpeed(MAX_SPEED, BACKWARD);
}
```

All four motors move backward at full speed.

Left Turn

```
void left() {
  motor1.setSpeed(MAX_SPEED);
  motor1.run(BACKWARD);
  motor2.setSpeed(MAX_SPEED);
  motor2.run(BACKWARD);
  motor3.setSpeed(MAX_SPEED);
  motor3.run(FORWARD);
  motor4.setSpeed(MAX_SPEED);
  motor4.run(FORWARD);
}
```

Left side motors move backward, right side motors move forward, causing the car to turn left.

Right Turn

```
void right() {
  motor1.setSpeed(MAX_SPEED);
  motor1.run(FORWARD);
  motor2.setSpeed(MAX_SPEED);
  motor2.run(FORWARD);
  motor3.setSpeed(MAX_SPEED);
  motor3.run(BACKWARD);
  motor4.setSpeed(MAX_SPEED);
  motor4.run(BACKWARD);
}
```

Right side motors move backward, left side motors move forward, turning the car right.

Diagonal Movements

These functions use reduced speed for a smoother turn:

Top-Left (Forward-Left)

```
void topleft() {
  motor1.setSpeed(MAX_SPEED);
  motor1.run(FORWARD);
  motor2.setSpeed(MAX_SPEED);
  motor2.run(FORWARD);
  motor3.setSpeed(REDUCED_SPEED);
  motor3.run(FORWARD);
  motor4.setSpeed(REDUCED_SPEED);
  motor4.run(FORWARD);
}
```

Right-side motors move at full speed, left-side at reduced speed for a diagonal motion.

Top-Right (Forward-Right)

```
void topright() {
  motor1.setSpeed(REDUCED_SPEED);
  motor1.run(FORWARD);
  motor2.setSpeed(REDUCED_SPEED);
  motor2.run(FORWARD);
  motor3.setSpeed(MAX_SPEED);
  motor3.run(FORWARD);
  motor4.setSpeed(MAX_SPEED);
  motor4.run(FORWARD);
}
```

Left-side motors move at full speed, right-side at reduced speed for a diagonal motion.

Bottom-Left (Backward-Left)

```
void bottomleft() {
  motor1.setSpeed(MAX_SPEED);
  motor1.run(BACKWARD);
  motor2.setSpeed(MAX_SPEED);
  motor2.run(BACKWARD);
  motor3.setSpeed(REDUCED_SPEED);
  motor3.run(BACKWARD);
  motor4.setSpeed(REDUCED_SPEED);
  motor4.run(BACKWARD);
}
```

Right-side motors move at full speed, left-side at reduced speed for a smooth backward-left motion.

Bottom-Right (Backward-Right)

```
void bottomright() {
  motor1.setSpeed(REDUCED_SPEED);
  motor1.run(BACKWARD);
  motor2.setSpeed(REDUCED_SPEED);
  motor2.run(BACKWARD);
  motor3.setSpeed(MAX_SPEED);
  motor3.run(BACKWARD);
  motor4.setSpeed(MAX_SPEED);
  motor4.run(BACKWARD);
}
```

Left-side motors move at full speed, right-side at reduced speed for a smooth backward-right motion.

Stopping the Car

```
void Stop() {
  setMotorsSpeed(0, RELEASE);
}
```

Sets all motor speeds to 0, stopping the car.

Helper Function to Set Motor Speeds

```
void setMotorsSpeed(int speed, int direction) {
  motor1.setSpeed(speed);
  motor1.run(direction);
  motor2.setSpeed(speed);
  motor2.run(direction);
  motor3.setSpeed(speed);
  motor3.run(direction);
  motor4.setSpeed(speed);
  motor4.run(direction);
}
```

This function allows setting speed and direction for all four motors simultaneously.

Applications

- ✅ DIY Robotics Projects
- ✅ Smart Car Development
- ✅ IoT & Wireless Control Systems
- ✅ Educational Learning in Embedded Systems

Future Enhancements

- 🔹 Obstacle Avoidance Sensors
- 🔹 Voice Control Integration
- 🔹 Camera Module for Live Streaming