Bluetooth Controlled Arduino RC Car

By

Varshith Peddineni

RC Car Arduino Code Documentation

Overview

This code controls an RC car using an Arduino Uno in combination with an HC-05 Bluetooth module. It receives commands via Bluetooth and drives a pair of DC motors through an L298N motor driver. The code supports different driving directions, varying speed levels, and includes a basic framework for an electronic braking system.

---

1. Pin Definitions and Global Variables

- **Motor Control Pins:**
  - **in1 (Pin 5)** and **in2 (Pin 6):** Control one motor.
  - **in3 (Pin 10)** and **in4 (Pin 11):** Control the other motor.
- **LED Pin:**
  - **LED (Pin 13):** Used for status indication (could be used to show connectivity or error states).
- **Global Variables:**
  - **command:** Stores the command character received via Bluetooth.
  - **Speed:** Controls the base speed of the motors (range 0–255). Initially set to 204.
  - **Speedsec:** Holds a secondary speed value (used for turning operations).
  - **buttonState** and **lastButtonState:** Used in the braking system to detect changes in command state.
  - **Turnradius:** Sets the radius of a turn; adjusts the speed of one motor to enable smooth turning.
  - **brakeTime:** Time delay (in milliseconds) for the brake action.
  - **brkonoff:** A flag to activate (1) or bypass (0) the braking system.

2. Setup Function

```
void setup() {
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(LED, OUTPUT); // Configure LED pin.
  Serial.begin(9600);   // Start serial communication at 9600 baud (matches HC-05 module).
}
```

- **Purpose:**
  - Initializes the motor control pins and the LED pin as outputs.
  - Starts the serial communication, enabling the Arduino to receive Bluetooth commands.

3. Main Loop

```
void loop() {
  if (Serial.available() > 0) {
    command = Serial.read();
    Stop(); // Stop motors to prepare for the new command.

    switch (command) {
      case 'F':
        forward();
        break;
      case 'B':
        back();
        break;
```

```
case 'L':
  left();
  break;
case 'R':
  right();
  break;
case 'G':
  forwardleft();
  break;
case 'I':
  forwardright();
  break;
case 'H':
  backleft();
  break;
case 'J':
  backright();
  break;
// Cases '0' to '9' and 'q' change the Speed value.
case '0':
  Speed = 100;
  break;
case '1':
  Speed = 140;
  break;
case '2':
  Speed = 153;
  break;
case '3':
  Speed = 165;
  break;
case '4':
  Speed = 178;
  break;
case '5':
  Speed = 191;
  break;
case '6':
  Speed = 204;
  break;
case '7':
  Speed = 216;
  break;
case '8':
  Speed = 229;
  break;
case '9':
  Speed = 242;
  break;
case 'q':
  Speed = 255;
  break;
```

```
    }


    // For turning: Adjust secondary speed based on Turnradius.

    Speedsec = Turnradius;


    // Execute braking if enabled.

    if (brkonoff == 1) {

      brakeOn();

    } else {

      brakeOff();

    }

  }

}
```

- **Purpose:**
  - Checks if a new Bluetooth command is available.
  - Reads the command and immediately stops the motors to ensure a safe state.
  - Uses a switch statement to determine which action to perform based on the command character.
  - Adjusts speed values if a number command is received.
  - Optionally triggers the braking system based on the brkonoff flag.

4. Direction Functions

These functions control the motion of the car by sending appropriate speed values to the motor driver pins:

- **Forward:**

```
void forward() {
  analogWrite(in1, Speed);
  analogWrite(in3, Speed);
}
```

Drives both motors forward.

- **Back:**

```
void back() {
  analogWrite(in2, Speed);
  analogWrite(in4, Speed);
}
```

Drives both motors in reverse.

- **Left:**

```
void left() {
  analogWrite(in3, Speed);
  analogWrite(in2, Speed);
}
```

Drives the car to turn left by running one motor in reverse and the other forward (depending on motor orientation).

- **Right:**

```
void right() {
  analogWrite(in4, Speed);
  analogWrite(in1, Speed);
}
```

Drives the car to turn right.

- **Forward Left / Right:**
  Adjust the speed of one motor using the Turnradius value (Speedsec) to create a smoother turn:
  - **Forward Left:**

    ```
    void forwardleft() {
      analogWrite(in1, Speedsec);
      analogWrite(in3, Speed);
    }
    ```

  - **Forward Right:**

    ```
    void forwardright() {
      analogWrite(in1, Speed);
      analogWrite(in3, Speedsec);
    }
    ```

- **Back Left / Right:**
  Similar to forward turning, but while moving backwards:
  - **Back Left:**

    ```
    void backleft() {
      analogWrite(in2, Speedsec);
      analogWrite(in4, Speed);
    }
    ```

  - **Back Right:**

    ```
    void backright() {
      analogWrite(in2, Speed);
      analogWrite(in4, Speedsec);
    }
    ```

- **Stop:**

```
void Stop() {
  analogWrite(in1, 0);
  analogWrite(in2, 0);
  analogWrite(in3, 0);
  analogWrite(in4, 0);
}
```

Immediately stops all motor activity by setting motor control pins to 0.

---

5. Braking Functions

- **brakeOn:**

```
void brakeOn() {
  buttonState = command;
  if (buttonState != lastButtonState) {
    if (buttonState == 'S') {
      if (lastButtonState != buttonState) {
        digitalWrite(in1, HIGH);
        digitalWrite(in2, HIGH);
       digitalWrite(in3, HIGH);
       digitalWrite(in4, HIGH);
       delay(brakeTime);
          Stop();
          }
          }
         lastButtonState = buttonState;
        }
       }
```

Implements an electronic braking mechanism triggered by receiving the 'S' command. It briefly sets all motor pins to HIGH, causing a rapid deceleration, then stops the motors after a short delay defined by brakeTime.

- **brakeOff:**

  ```
  void brakeOff() {
    // Empty function when electronic braking is disabled.
  }
  ```

A placeholder function that does nothing when the braking system is turned off.

6. Summary

- **Initialization:** The code sets up necessary pins and serial communication.
- **Main Loop:** Continuously checks for Bluetooth commands, stops the motors, and processes commands to drive the car in various directions.
- **Speed Control:** Specific commands change the motor speed, providing a way to adjust performance.
- **Directional Movement:** Different functions manage forward, backward, and turning maneuvers.
- **Braking System:** A simple electronic braking system is implemented to quickly stop the car when required.