```
#include <AFMotor.h> // Library for motor control
// Define motors
AF_DCMotor motorFrontLeft(1);
AF_DCMotor motorFrontRight(2);
AF_DCMotor motorRearLeft(3);
AF_DCMotor motorRearRight(4);
String receivedCommand = ""; // Store the received command
char direction = 'S';       // Default direction is Stop
int motorSpeed = 0;         // Default speed is 0
void setup() {
  Serial.begin(9600);  // USB communication
  Serial1.begin(9600); // Bluetooth module communication
  Serial.println("Bluetooth motor control initialized. Send commands!");
  stopMotors(); // Ensure motors are stopped initially
}
void loop() {
  // Check for incoming data from Bluetooth
  if (Serial1.available()) {
    receivedCommand = Serial1.readStringUntil('\n'); // Read full command until newline
    receivedCommand.trim(); // Remove extra spaces or newlines
    Serial.print("Received Command: ");
    Serial.println(receivedCommand);
    // Parse the command
    if (parseCommand(receivedCommand)) {
      executeCommand(); // Execute the parsed command
    } else {
      Serial.println("Invalid command received!");
    }
  }
}
```

```
// Parse the command into direction and speed
bool parseCommand(String command) {
  if (command.length() < 1) return false; // Ensure command is not empty
  direction = command.charAt(0); // First character is the direction
  if (command.length() > 2) { // Check if a speed value is included
    String speedStr = command.substring(2); // Extract the speed part
    motorSpeed = speedStr.toInt();          // Convert speed to integer
    motorSpeed = constrain(motorSpeed, 0, 255); // Ensure speed is within 0-255
  } else {
    motorSpeed = 0; // Default speed if not provided
  }
  return true; // Command parsed successfully
}
// Execute the parsed command
void executeCommand() {
  switch (direction) {
    case 'F': // Forward
      setMotors(motorSpeed, FORWARD, FORWARD, FORWARD, FORWARD);
      break;
    case 'B': // Backward
      setMotors(motorSpeed, BACKWARD, BACKWARD, BACKWARD, BACKWARD);
      break;
    case 'R': // Right
      setMotors(motorSpeed, FORWARD, BACKWARD, BACKWARD, FORWARD);
      break;
    case 'L': // Left
      setMotors(motorSpeed, BACKWARD, FORWARD, FORWARD, BACKWARD);
      break;
    case 'E': // North-East
      setMotors(motorSpeed, FORWARD, RELEASE, BACKWARD, RELEASE);
      break;
```

```
    case 'Q': // North-West
      setMotors(motorSpeed, RELEASE, FORWARD, RELEASE, BACKWARD);
      break;
    case 'C': // South-East
      setMotors(motorSpeed, BACKWARD, RELEASE, FORWARD, RELEASE);
      break;
    case 'Z': // South-West
      setMotors(motorSpeed, RELEASE, BACKWARD, RELEASE, FORWARD);
      break;
    case 'CW': // Clockwise rotation
      setMotors(motorSpeed, FORWARD, BACKWARD, FORWARD, BACKWARD);
      break;
    case 'CCW': // Counterclockwise rotation
      setMotors(motorSpeed, BACKWARD, FORWARD, BACKWARD, FORWARD);
      break;
    case 'S': // Stop
    default:
      stopMotors(); // Stop all motors
      break;
  }
}
// Control all motors with specified speeds and directions
void setMotors(int speed, int frontLeftDir, int frontRightDir, int rearLeftDir, int rearRightDir) {
  motorFrontLeft.setSpeed(speed);
  motorFrontRight.setSpeed(speed);
  motorRearLeft.setSpeed(speed);
  motorRearRight.setSpeed(speed);
  motorFrontLeft.run(frontLeftDir);
  motorFrontRight.run(frontRightDir);
  motorRearLeft.run(rearLeftDir);
  motorRearRight.run(rearRightDir);
```

```
  Serial.print("Command: ");
  Serial.print(direction);
  Serial.print(", Speed: ");
  Serial.println(speed);
}
// Stop all motors
void stopMotors() {
  motorFrontLeft.setSpeed(0);
  motorFrontRight.setSpeed(0);
  motorRearLeft.setSpeed(0);
  motorRearRight.setSpeed(0);
  motorFrontLeft.run(RELEASE);
  motorFrontRight.run(RELEASE);
  motorRearLeft.run(RELEASE);
  motorRearRight.run(RELEASE);
  Serial.println("Motors stopped.");
}
```