Gesture-Controlled Robot using NRF24L01 Transceiver and Arduino

Abstract

This paper presents the design and development of a gesture-controlled robot capable of responding to hand movements. The system employs an accelerometer (ADXL335) as a transmitter to detect gestures, which are transmitted wirelessly using the NRF24L01 transceiver modules. The receiver interprets the data to control the motion of four DC motors using an L298N motor driver. The system demonstrates efficient real-time gesture recognition and motion control, showcasing applications in robotics and assistive technology.

---

1. Introduction

Robotics is revolutionizing industries ranging from manufacturing to healthcare. Gesture-based control systems offer an intuitive and user-friendly interface for operating robots, eliminating the need for traditional input devices. Gesture-controlled robots are particularly relevant in scenarios requiring contactless operation, such as healthcare environments or hazardous conditions.

This project, "GestureBot", focuses on the development of a robot controlled by hand gestures. By leveraging an ADXL335 accelerometer, NRF24L01 transceivers, and Arduino Nano microcontrollers, the system facilitates real-time wireless control of a robotic vehicle. The simplicity and cost-effectiveness of the design make it a viable solution for academic, industrial, and assistive applications.

---

2. System Design

The project comprises two main subsystems:

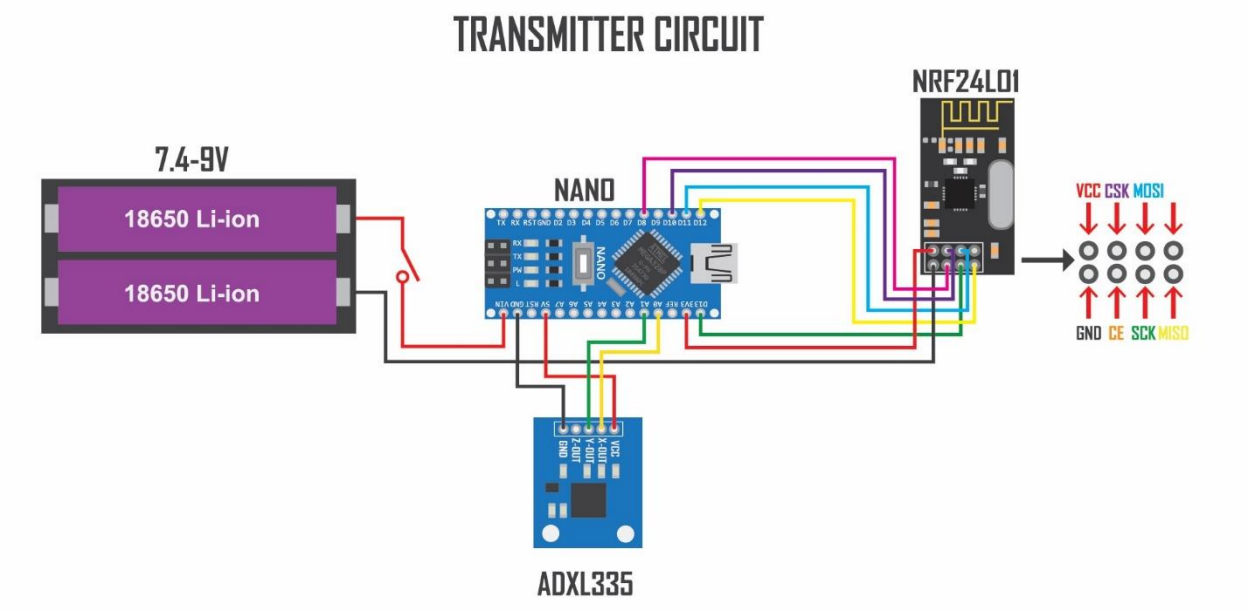
1. **Transmitter Unit**: Captures hand gestures and transmits the corresponding data wirelessly.
2. **Receiver Unit**: Interprets the transmitted data and controls the robot's motors accordingly.

2.1 Transmitter Circuit

The transmitter circuit consists of:

- **ADXL335 Accelerometer**: Detects the tilt of the user's hand and outputs analog signals proportional to the tilt along the X and Y axes.
- **Arduino Nano**: Reads the accelerometer data and sends it wirelessly via the NRF24L01 module.
- **NRF24L01 Transceiver**: Handles wireless communication with the receiver.

Transmitter Schematic



The schematic is designed to process hand movements and transmit data. It is powered by two 18650 Li-ion batteries providing a voltage of 7.4V.

Transmitter Code

The transmitter's code is responsible for capturing the X and Y-axis data from the ADXL335 and transmitting it. The data structure ensures efficient communication.

```
#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>
```

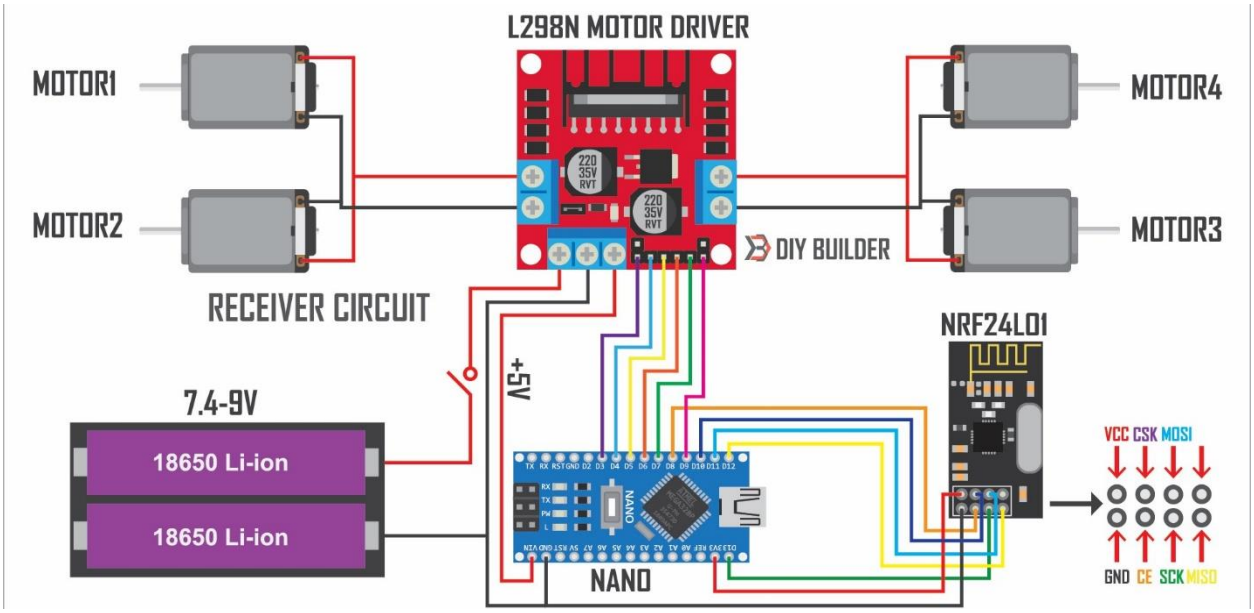
```
const int x_out = A0;
const int y_out = A1;
RF24 radio(8, 10);
const byte address[6] = "00001";
struct data {
  int xAxis;
  int yAxis;
};
data send_data;

void setup() {
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);
  radio.setDataRate(RF24_250KBPS);
  radio.stopListening();
}

void loop() {
  send_data.xAxis = analogRead(x_out);
  send_data.yAxis = analogRead(y_out);
  radio.write(&send_data, sizeof(data));
}
```

2.2 Receiver Circuit

The receiver circuit processes data and controls the robot's motion. It consists of:

- **NRF24L01 Transceiver**: Receives the gesture data.
- **Arduino Nano**: Decodes the data and sends commands to the motor driver.
- **L298N Motor Driver**: Controls four DC motors.
- **DC Motors**: Drive the robot in forward, backward, left, and right directions.

Receiver Schematic



The receiver schematic is powered by two 18650 Li-ion batteries. The L298N motor driver ensures adequate power delivery to the motors while maintaining control precision.

Receiver Code

The receiver's code interprets the data and executes the corresponding motor operations.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

int ENA = 3;
int ENB = 9;
int MotorA1 = 4;
int MotorA2 = 5;
int MotorB1 = 6;
int MotorB2 = 7;

RF24 radio(8, 10);
const byte address[6] = "00001";

struct data {
  int xAxis;
  int yAxis;
};
data receive_data;

void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MIN);
  radio.setDataRate(RF24_250KBPS);
```

```
  radio.startListening();
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(MotorA1, OUTPUT);
  pinMode(MotorA2, OUTPUT);
  pinMode(MotorB1, OUTPUT);
  pinMode(MotorB2, OUTPUT);
}

void loop() {
  while (radio.available()) {
    radio.read(&receive_data, sizeof(data));

    if (receive_data.yAxis > 400) {
      digitalWrite(MotorA1, LOW);
      digitalWrite(MotorA2, HIGH);
      digitalWrite(MotorB1, HIGH);
      digitalWrite(MotorB2, LOW);
      analogWrite(ENA, 150);
      analogWrite(ENB, 150);
    } else if (receive_data.yAxis < 320) {
      digitalWrite(MotorA1, HIGH);
      digitalWrite(MotorA2, LOW);
      digitalWrite(MotorB1, LOW);
      digitalWrite(MotorB2, HIGH);
      analogWrite(ENA, 150);
      analogWrite(ENB, 150);
    } else if (receive_data.xAxis < 320) {
```

```
      digitalWrite(MotorA1, HIGH);
      digitalWrite(MotorA2, LOW);
      digitalWrite(MotorB1, HIGH);
      digitalWrite(MotorB2, LOW);
      analogWrite(ENA, 150);
      analogWrite(ENB, 150);
    } else if (receive_data.xAxis > 400) {
      digitalWrite(MotorA1, LOW);
      digitalWrite(MotorA2, HIGH);
      digitalWrite(MotorB1, LOW);
      digitalWrite(MotorB2, HIGH);
      analogWrite(ENA, 150);
      analogWrite(ENB, 150);
    } else {
      digitalWrite(MotorA1, LOW);
      digitalWrite(MotorA2, LOW);
      digitalWrite(MotorB1, LOW);
      digitalWrite(MotorB2, LOW);
      analogWrite(ENA, 0);
      analogWrite(ENB, 0);
    }
  }
}
```

---

3. Experimental Results

3.1 System Performance

The robot responds effectively to various hand gestures. The delay between gesture execution and robot movement is minimal, demonstrating robust communication and processing.

3.2 Applications

1. **Healthcare**: Assistive robots for physically disabled individuals.
2. **Industrial Automation**: Gesture-controlled machinery.
3. **Education**: Hands-on learning for robotics and wireless communication.

---

4. Conclusion and Future Work

GestureBot successfully demonstrates real-time gesture-based control using wireless communication. Future enhancements may include:

- Integration of additional sensors for advanced control.
- Development of a smartphone application for gesture simulation.
- Implementation of machine learning for dynamic gesture recognition.

References

1. Arduino Documentation. https://www.arduino.cc
2. NRF24L01 Datasheet. Nordic Semiconductor.
3. ADXL335 Datasheet. Analog Devices.