INDIAN INSTITUTE OF TECHNOLOGY ROPAR
RUPNAGAR 140001, INDIA

GE107 - TINKERING LAB PROJECT
(WRITE-UP)

Group - G14_WED

Our Team Members -

NAME	ENTRY NO.
HARSH SHARMA	2023MEB1346
BARINDER SINGH	2023MEB1336
BEWLE SUYOG	2023MEB1337
HARSH ALOK SHAH	2023MEB1344
KAPISH MINA	2023MEB1353
AYUSH KUMAR SAW	2023MEB1331

Smart Gesture Controlled Robot

Introduction to the Problem

With the rapid advancement of technology, traditional methods of controlling machines and devices are being replaced by more intuitive and natural interactions. Gesture recognition has emerged as a promising Lield, allowing users to interact with devices using simple hand movements. The goal of this project is to design and develop a gesture-controlled robot that can be operated using hand gestures, eliminating the need for conventional remote controls or wired systems.

Background & Context

Gesture control technology has seen increasing use in various applications, including gaming, virtual reality, robotics, and assistive technology. The idea of controlling a robot using gestures can signiLicantly enhance usability, making robotics more accessible to individuals who may struggle with traditional controllers. This project explores the implementation of a gesture-controlled car using an MPU6050 accelerometer, Arduino, and NRF24L01 transceivers, providing a seamless and wireless control experience.

Importance & Need for the Project

The need for gesture-controlled systems stems from various real-world applications, such as -

- **Accessibility -** Helping individuals with disabilities control devices without requiring complex physical inputs.
- **Industrial Automation -** Offering touch-free control in hazardous environments.
- **Military & Defense -** Providing an alternative method for controlling unmanned ground vehicles (UGVs).
- **Healthcare Integration** - Hands-free wheelchair control for quadriplegic patients and surgical assistant navigation in sterile environments

---

Challenges in Addressing the Problem

Several challenges need to be addressed to make the system efLicient and reliable -

- **Accurate Gesture Recognition -** Ensuring the MPU6050 precisely detects hand movements without false positives.
- **Wireless Communication Stability -** The NRF24L01 module must ensure a lag-free and interference-resistant connection.
- **Power Management -** Optimizing power consumption for longer operation.
- **Synchronization Issues -** Smooth mapping of hand gestures to motor actions.
- **Hardware Reliability -** Ensuring motors, sensors, and electronics function correctly under different conditions.

Objective & Goals of the Project

- Design a gesture-controlled robotic car that responds accurately to hand movements.
- Implement wireless communication using NRF24L01 transceivers.
- Develop a gesture recognition algorithm that translates hand movements into directional commands.
- Ensure real-time response with minimal latency.
- Create a user-friendly and ergonomic glove-based controller.

Scope of the Project

- **Hardware Development -** Designing and assembling the robot, integrating motors, sensors, and wireless modules.
- **Software Development -** Writing Arduino programs to process sensor data and control the motors accordingly.
- **Testing & Optimization -** Improving accuracy, responsiveness, and stability of the gesture recognition system.
- **Potential Future Enhancements -** Adding features such as voice control, obstacle detection, or AI-based gesture learning.

Potential Advantages

Unlike conventional joystick-controlled robots, our solution -

- Eliminates physical controller dependence.
- Reduces cognitive load through natural gestures.
- Implements adaptive Liltering for <0.5° drift error.

Components Required

1. Core Electronics

Component	Qty	Technical Specs	Purpose
Arduino UNO	1	ATmega328P, 16MHz	Main robot controller
Arduino Nano	1	ATmega328P, 16MHz	Glove-mounted controller
NRF24L01 Transceiver	2	2.4GHz, 250kbps-2Mbps	Wireless gesture data transmission
HC-05 Bluetooth Module	1	Class 2, 2.4GHz	Backup control channel
MPU6050 (6-axis IMU)	1	±2g/±4g/±8g/±16g range	Hand gesture detection
L298N Motor Driver	1	2A per channel, 5-35V	DC motor control
16x2 LCD Display	1	HD44780, I^2C or parallel interface	Real-time status monitoring

2. Power Systems

Component	Qty	Specifications
9V Battery + Clamp	1	Alkaline/Lithium
2-Cell Battery Holder	1	7.4V output (for motors)
18650/AA Battery Cells	2	3.7V (Li-ion) or 1.5V (Alkaline)
On/Off Switch	1	SPDT, 5A rating

3. Mechanical Assembly

Component	Qty	Details
DC Gear Motors	4	100-300 RPM, 6-12V
Rubber Tires	4	60-80mm diameter, 6mm shaft
4WD Robot Chassis	1	Acrylic/metal, pre-drilled holes
M3 Nuts & Bolts Kit	10	10mm length

4. Wiring & Prototyping

Component	Qty	Type
Small Breadboard (400pt)	1	For glove circuitry
Large Breadboard (830pt)	1	For robot base
Male-to-Male Jumpers	20	10cm length, 22AWG
Male-to-Female Jumpers	10	Sensor connections
Female-to-Female Jumpers	10	Module interlinks

Expected Timeline

PHASE	TASKS	DURATION
Research & Planning	Study gesture recognition, Linalise components.	1 Week
Hardware Assembly	Assemble the robot, attach sensors and motors.	1 Week
Software Development	Writing Arduino code for sensor and motor control.	1 Week
Testing & Debugging	Calibrate gestures, optimise response time.	1 Week
Final Integration	Combine hardware and software, ensure stability.	1 Week

---

Conclusion

This gesture control system represents a paradigm shift in human-robot interaction, combining proven components with innovative signal processing techniques. The project establishes a framework for -

- Expandable gesture vocabulary through machine learning.
- Mesh networking for multi-robot coordination.
- Haptic feedback integration for improved UX.

Next Development Phase: Integration with ROS for advanced robotic applications

WEEK-2

What can we do additionally?

1. Speed Control and Speed Display via LCD

Hardware Required:

- **Microcontroller**: Arduino Uno / ESP32 / Raspberry Pi
- **Motor Driver**: L298N / TB6612FNG
- **Motors**: DC Motors with encoders for feedback
- **Speed Sensor**: Hall-effect sensor / Rotary Encoder
- **Display**: 16x2 LCD / OLED display
- **Power Source**: Li-Po or Lithium-ion battery

Implementation Steps:

1. **Speed Control Using PWM**:
    a. PWM (Pulse Width Modulation) controls motor speed by varying duty cycles.
    b. Example code snippet for PWM control:

```
int motorPin = 9; // PWM pin
int speed = 150;  // Adjust speed (0-255)
analogWrite(motorPin, speed);
```

2. **Speed Display on LCD**:

- Read encoder values to calculate speed.
- Display the real-time speed on an LCD.
- Example code:

```
lcd.setCursor(0, 0);
lcd.print("Speed: ");
lcd.print(speed);
lcd.print(" RPM");
```

2. Drift Mode

Hardware Required:

- High-speed **DC motors** with independent wheel control.
- **IMU (Inertial Measurement Unit) sensor** (MPU6050) for drift detection.
- **Motor Driver** that supports variable torque.

Implementation Steps:

1. **Steering Angle Control**:
    a. Modify wheel alignment dynamically to achieve drifting.
2. **Torque Control**:
    a. Increase rear-wheel power for controlled skidding.
3. **Code Logic:**

```
if (driftMode == true) {
leftMotorSpeed = baseSpeed - 30;
rightMotorSpeed = baseSpeed + 30;
analogWrite(leftMotorPin, leftMotorSpeed);
analogWrite(rightMotorPin, rightMotorSpeed);
}
```

3. Self-Parking

Hardware Required:

- **Ultrasonic Sensors (HC-SR04) / LiDAR**
- **Camera (for AI-based detection)**
- **Servo motors** for steering control

Implementation Steps:

1. **Detect Empty Parking Spots**:
    a. Use an **ultrasonic sensor** to measure distances from obstacles.
    b. Camera-based detection using OpenCV (for AI-based parking).

2. **Maneuver the Car Automatically**:
    a. Implement parallel parking logic using stepper/servo motors.
3. **Basic Code Logic (Ultrasonic-based detection)**:

```
if (distance < 10) {
  // Stop car and start parking sequence
  parkCar();
}
```

4. Additional Controls

A. Voice Commands

- **Hardware**: Voice recognition module (Elechouse V3) / Google Assistant API
- **Implementation**: Use predefined commands like "Forward," "Stop," "Left," and "Right."

B. Remote App Control

- **Hardware**: Bluetooth (HC-05) / Wi-Fi (ESP8266)
- **Implementation**:
    - Create an Android app to send movement commands.
    - Example **Bluetooth Code**:

```
if (command == 'F') { moveForward(); }
else if (command == 'L') { turnLeft(); }
if (command == 'F') { moveForward(); }
else if (command == 'L') { turnLeft(); }
```