**TITLE**

**Hackathon Project : Gesture – Controlled Car**

**Team Members**

**Rahul, Abhishek, Atul, Purnima**

**Abstract**

The Gesture-Controlled Car is a smart robotic vehicle that can be controlled using just hand movements. It uses a special sensor called the MPU6050 to detect the direction in which your hand moves. This information is sent to a small computer chip (ESP32), which then controls the motors and moves the car in the same direction. This system does not need any remote o

.r mobile app — just your hand gestures.

This type of technology can be very useful in many real-life situations. For example, it can help in dangerous areas where people cannot go, like in rescue operations, military zones, or areas with harmful gases. It can also be helpful for people with disabilities who may find traditional controls difficult to use.

We create this project during hackathon to show how useful and powerful gesture-based technology can be. This project shows how we can make controlling machines easier and more natural using simple hand movements In the future, we plan to improve it by adding features like :

* Obstacle detection
* Camera-based gesture control.

**Problem Statement**

“In dangerous places like war zones or rescue missions, using remotes or phones to control robots can be risky and slow. For people with physical disabilities, these controls may not work at all. We need a smarter, hands-free solution and that’s where gesture control comes in…..”

* To address these challenges, a hands-free system using simple hand gestures offers a safer and more inclusive solution.
* By using sensors and microcontrollers, gestures can be recognized in real time to control the robot’s movement.
* Our project aims to build a gesture-controlled car that can operate safely in risky environments and be accessible to users with limited mobility.

**Why Gestures? Why Hands-Free?**

Gesture control is a natural way for humans to interact with machines. It allows a person to control a system using simple hand movements, without needing to touch any buttons or devices. This makes the process faster, cleaner, and more accessible, especially in situations where physical contact is not ideal or even possible.

Hands-free control is particularly important in environments that are dangerous, high-risk, or hard to reach. For example:

* In **military zones**, soldiers might need to control robots without putting down their weapons or exposing themselves.
* In **chemical plants**, where toxic substances might be present, touching any surface could be harmful.
* In **healthcare or assistive settings**, users with limited mobility may struggle to use standard controllers.

**But What If Someone Can’t Move Their Hands?**

If someone cannot use their hands due to a disability or injury, then gesture control alone won’t be enough. In such cases, the system can be expanded to **alternative input methods** such as:

* **Voice Control :**
* The robot can be programmed to respond to voice commands like “forward,” “stop,” “left,” or “right.”
* Useful for people who can speak clearly but may not have physical mobility.
* **Eye-Tracking :**
* Eye-tracking devices use cameras to detect where the user is looking.
* The direction of gaze can be mapped to movements of the robot.
* Ideal for people with very limited mobility, such as those with ALS or paralysis.

**Objective of the Project**

* The **Main goal** of this project is to :
* Design and develop a gesture-controlled robotic car that can be operated without any physical contact, using hand movements.
* provide an efficient way to control machines in both everyday and critical situations.
* Main key objective is to promote **innovation** by replacing traditional remotes with gesture-based technology.
* This makes the system more modern, user-friendly, and suitable for hands-free operation.
* Another important goal is to improve **accessibility**. People with physical challenges often struggle with regular control systems. Gesture control offers an alternative that is easier to use and more inclusive.
* The project also explores its use in **defense and rescue operations**, where hands-free control allows robots to operate in dangerous or hard-to-reach areas, reducing risks for human operators.

**System Overview**

* The gesture-controlled car system is designed to detect hand movements and convert them into motion commands for a robotic vehicle.
* When a user moves their hand, the MPU6050 sensor captures the motion data (acceleration and orientation). This data is sent to the ESP8266 microcontroller, which processes the information and decides how the car should respond. The ESP8266 then sends signals to the ESP32 and ESP32 sends signal to L298N motor driver, which powers the DC motors to move the car in the desired direction.
* The entire system is powered by a Lead acid battery, and it operates wirelessly, enabling smooth, real-time, hands-free control.

**Block Diagram**

**Hardware Used**

* **ESP32 / ESP8266**

A powerful microcontroller with built-in Wi-Fi, used to process gesture data and control the motors wirelessly. It reads sensor input, executes control logic, and sends appropriate signals to the motor driver.

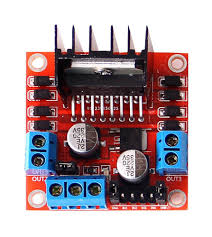
* **MPU6050**

A 6-axis motion tracking sensor that includes a 3-axis accelerometer and a 3-axis gyroscope. It detects the direction and movement of the hand to capture real-time gesture data.



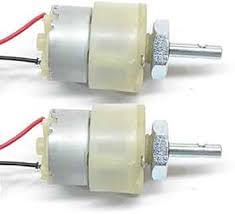
* **Motor Driver (L298N)**

A motor driver module that allows the microcontroller to control the speed and direction of the DC motors. It acts as a bridge between the ESP32 and the motors.



* **Chassis, DC (Gears) Motors and Wheels**

The physical structure of the robotic car, equipped with BO (Battery Operated) motors for motion. The chassis holds all components together and supports the car’s movement.

* **Power Supply**

The system is powered by rechargeable Lead acid battery , regulated to provide the correct voltage to different components. Stable power supply ensures smooth and safe operation of the car.



**Software Architecture**

The software forms the brain of the gesture-controlled car system. It is responsible for reading sensor data, interpreting gestures, and translating them into motor control commands in real time.

* **Development Platform**
* **Arduino IDE** was used for writing, compiling, and uploading the code to the ESP32.
* Open-source libraries were used for handling the MPU6050 sensor and motor control.
* **Code Logic Overview**

**1. Initialization:**

* Set up I2C communication between the ESP8266 and the MPU6050.
* Calibrate the MPU6050 to reduce noise.

**2. Data Collection:**

* Continuously read acceleration and gyroscope data from the sensor.

**3. Gesture Detection:**

* Analyze real-time sensor data to detect tilt and directional changes.
* Map specific gesture patterns to movement commands:
  + Forward Tilt → Move Forward
  + Backward Tilt → Move Backward
  + Left Tilt → Turn Left
  + Right Tilt → Turn Right

**4. Motor Control:**

* Send PWM signals from ESP32 to L298N motor driver based on recognized gestures.
* Control the direction and speed of motors accordingly.

**5. Looping & Adjustment:**

* Continuously monitor hand movement for dynamic updates.
* Apply thresholds and filters to minimize unintentional movement.
* **Gesture Recognition Logic**
* Gestures are recognized based on threshold values of X, Y, and Z-axis data from the MPU6050.
* For example, a strong forward tilt is detected when the Y-axis value exceeds a certain range.
* Simple IF-ELSE conditions are used for decision-making based on sensor data.
* **Communication Protocols**
* **I2C**: Used for communication between ESP32 and MPU6050 sensor.
* **PWM (Pulse Width Modulation)**: Used to control the speed and direction of DC motors via L298N.
* **Code for Transmitter** :

#include <Wire.h>

#include <ESP8266WiFi.h>

#include <MPU6050.h>

const char\* ssid = "YOUR\_SSID"; // Same as used in ESP32

const char\* password = "YOUR\_PASSWORD"; // Same as used in ESP32

const char\* host = "192.168.4.1"; // Default IP of ESP32 in AP mode

MPU6050 mpu;

WiFiClient client;

void setup() {

Serial.begin(115200);

Wire.begin();

mpu.initialize();

// Connect to ESP32 AP

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("Connected to ESP32");

}

void loop() {

int16\_t ax, ay, az;

mpu.getAcceleration(&ax, &ay, &az);

String command = "";

if (ay > 10000) command = "F"; // Tilt forward

else if (ay < -10000) command = "B"; // Tilt backward

else if (ax > 10000) command = "L"; // Tilt left

else if (ax < -10000) command = "R"; // Tilt right

else command = "S"; // Stay still / Stop

Serial.println("Gesture: " + command);

if (client.connect(host, 80)) {

client.print(command + "\n");

client.stop();

}

delay(200); // Adjust based on responsiveness

}

* **Code for Receiver :**

#include <WiFi.h>

const char\* ssid = "YOUR\_SSID";

const char\* password = "YOUR\_PASSWORD";

// Motor A - Left motor

#define IN1 18

#define IN2 19

#define ENA 5

// Motor B - Right motor

#define IN3 16

#define IN4 17

#define ENB 4

WiFiServer server(80);

void setup() {

Serial.begin(115200);

// Motor pins setup

pinMode(IN1, OUTPUT);

pinMode(IN2, OUTPUT);

pinMode(ENA, OUTPUT);

pinMode(IN3, OUTPUT);

pinMode(IN4, OUTPUT);

pinMode(ENB, OUTPUT);

// Start Access Point

WiFi.softAP(ssid, password);

server.begin();

Serial.println("ESP32 ready and waiting for gesture data...");

}

void loop() {

WiFiClient client = server.available();

if (client) {

String command = client.readStringUntil('\n');

command.trim();

Serial.println("Command: " + command);

if (command == "F") moveForward();

else if (command == "B") moveBackward();

else if (command == "L") turnLeft();

else if (command == "R") turnRight();

else stopMotors();

client.stop();

}

}

// Both motors forward

void moveForward() {

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

analogWrite(ENA, 200);

analogWrite(ENB, 200);

}

// Both motors backward

void moveBackward() {

digitalWrite(IN1, LOW);

digitalWrite(IN2, HIGH);

digitalWrite(IN3, LOW);

digitalWrite(IN4, HIGH);

analogWrite(ENA, 200);

analogWrite(ENB, 200);

}

// Left turn: right motor ON only

void turnLeft() {

digitalWrite(IN1, LOW);

digitalWrite(IN2, LOW);

digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

analogWrite(ENA, 0);

analogWrite(ENB, 200);

}

// Right turn: left motor ON only

void turnRight() {

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

digitalWrite(IN3, LOW);

digitalWrite(IN4, LOW);

analogWrite(ENA, 200);

analogWrite(ENB, 0);

}

// Stop all motors

void stopMotors() {

digitalWrite(IN1, LOW);

digitalWrite(IN2, LOW);

digitalWrite(IN3, LOW);

digitalWrite(IN4, LOW);

analogWrite(ENA, 0);

analogWrite(ENB, 0);

}

**Circuit Design & Soldering**

The circuit for the gesture-controlled car was designed to connect all key hardware components in a compact and organized layout. Initially tested on a **breadboard**, the final version was shifted to a **custom PCB (Printed Circuit Board)** for better stability and portability during movement.

* **Connection Overview:**
* **MPU6050 to ESP8266:**
  + Connected via I2C (SCL & SDA lines)
  + Powered using 3.3V from the ESP8266
  + Ground shared with ESP8266
* **ESP32 to Motor Driver (L298N):**
  + GPIO pins connected to IN1, IN2, IN3, IN4 on L298N
  + PWM pins used for speed control
  + Powered by 5V line
  + Ground connected for common reference
* **L298N to Motors:**
  + Output terminals connected to left and right DC motors
  + Motor power supply input fed from battery (12V)
* **Power Supply:**
  + A Lead acid battery powers the motors through the L298N and also powers the ESP32 (through voltage regulation if needed)

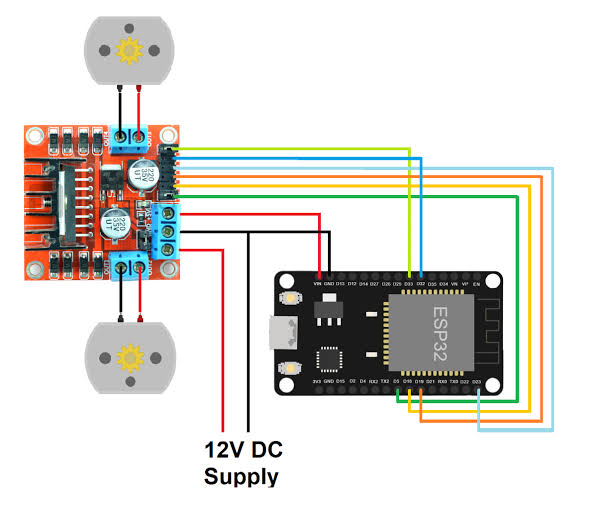
**Soldering Process:**

All components were carefully soldered onto a custom PCB. The board layout was designed to minimize wire clutter and prevent short circuits. Special attention was given to:

* I2C line separation to reduce interference
* Mounting headers for easy replacement of modules
* Proper insulation and heat dissipation zones around the motor driver

**Diagram Description**

*The diagram would show:*



**Power System Design**

The power system is a crucial part of the gesture-controlled car, ensuring that all components receive the correct and stable voltage to operate efficiently and safely.

* **Voltage Requirements**
* **ESP32 Microcontroller**: Operates on 3.3V
* **MPU6050 Sensor**: Requires 3.3V
* **L298N Motor Driver**: Typically requires 5V for logic and up to 12V for motor input
* **DC Motors**: Operate smoothly between 6V to 12V depending on torque requirements
* **Battery Setup**
* The system uses a **7.4V Li-ion battery pack** (typically 2-cell 18650 setup), chosen for its compact size, rechargeability, and sufficient output.
* The battery connects to both the motor driver (for motor power) and a voltage regulator to supply the ESP32 and MPU6050 with a safe 3.3V.
* **Voltage Regulation & Safety**
* A **buck converter** is used to step down the battery voltage to 3.3V for the ESP32 and MPU6050.
* Capacitors are added to filter out voltage spikes and noise, protecting sensitive electronics.
* Heat sinks are used on the motor driver (L298N) to prevent overheating during continuous operation.
* Proper grounding is ensured across all components to maintain signal stability and avoid short circuits.

**Team Contributions**

Each team member played a key role in the successful development of the gesture-controlled car. The responsibilities were divided based on skillsets to ensure smooth collaboration and progress throughout the project.

* **Rahul – Software Development & Gesture Logic**
* Wrote and tested the code for the ESP32 microcontroller
* Implemented gesture recognition using MPU6050 sensor data
* Handled integration of sensor inputs with motor outputs
* Designed the overall software flow and debugging
* **Purnima – Circuit Design & Soldering**
* Designed the circuit layout for all hardware components
* Created and soldered the PCB for a clean, compact assembly
* Ensured all connections were stable and correctly regulated
* Worked on I2C communication stability between sensor and ESP32
* **Atul – Power System Design**
* Managed the battery selection and voltage regulation design
* Installed and tested voltage converters and safety components
* Ensured safe distribution of power across all modules
* Troubleshot issues related to motor current and overheating
* **Abhishek – Documentation & Presentation**
* Prepared the project documentation and technical report
* Designed presentation slides and visuals
* Managed outreach content for demonstrating project use cases
* Helped summarize technical concepts in easy-to-understand language.

**Challenges Faced & Solutions**

During the development of the gesture-controlled car, the team faced several hardware, software, and coordination challenges. These obstacles helped us learn and improve the system through practical problem-solving.

**1. Sensor Data Instability**

* **Challenge:** The MPU6050 sensor gave noisy and inconsistent readings due to hand tremors and motion sensitivity.
* **Solution:** Implemented calibration routines and applied digital filtering techniques to smooth out the sensor data.

**2. Gesture Recognition Accuracy**

* **Challenge:** The car occasionally responded incorrectly due to similar hand movements being misclassified.
* **Solution:** Added clear threshold values and delays between readings to improve gesture detection accuracy and reduce false triggers.

**3. Power Supply Fluctuations**

* **Challenge:** Motors caused voltage drops that affected the ESP32 and sensor performance.
* **Solution:** Used capacitors for power smoothing and isolated power lines for motors and logic components.

**4. Soldering and Loose Connections**

* **Challenge:** Initial breadboard setup had frequent disconnections, affecting the car’s stability.
* **Solution:** Shifted to a custom PCB with soldered joints to make the setup more compact and reliable.

**5. Team Coordination**

* **Challenge:** Synchronizing hardware and software tasks between team members was difficult at first.
* **Solution:** Created a clear task list with roles, set deadlines, and held short progress meetings to stay on track.

**Defense & Real-World Applications**

The gesture-controlled car system is not just a technical prototype — it has the potential for meaningful real-world use, especially in critical or sensitive environments. Its hands-free nature and remote functionality make it suitable for various applications, including:

**1. Defense and Surveillance**

* The car can be used in military zones to navigate through risky areas without exposing soldiers to danger.
* By attaching a camera module, it can act as a surveillance robot to monitor enemy territory or inspect confined zones.

**2. Hazardous Area Inspection**

* Ideal for exploring environments where human presence is unsafe — such as areas with chemical leaks, fire, or structural damage.
* The user can control the robot from a safe distance using simple hand gestures, avoiding direct contact or entry.

**3. Assistive Technology**

* People with physical disabilities who have difficulty using traditional remotes or controllers can operate the system more easily using hand gestures.
* With future upgrades like voice or eye-tracking, it could become a more inclusive solution for mobility and control assistance.

**Future Enhancements**

While the current gesture-controlled car successfully operates with basic hand gestures, there are several opportunities to enhance its performance, intelligence, and adaptability in real-world environments:

**1. LIDAR Integration**

* Adding a LIDAR sensor would allow the car to scan its surroundings in real time.
* This would enable precise **autonomous navigation** and **obstacle mapping**, making the system safer in unknown or complex terrains.

**2. Camera or AI-Based Gesture Recognition**

* Instead of using motion sensors alone, a camera with AI (e.g., computer vision models) could recognize hand shapes and signs.
* This would allow for **more complex gestures**, **distance-based detection**, and **higher accuracy** even in dynamic environments.

**3. Obstacle Avoidance**

* Implementing ultrasonic or infrared sensors would allow the car to automatically detect and avoid obstacles.
* This feature would act as a safety layer, preventing collisions even if a user gesture accidentally leads the car toward an object.

**Conclusion**

The gesture-controlled car project successfully demonstrates how modern sensors and microcontrollers can work together to create an intuitive, hands-free robotic system. Through the use of the MPU6050 sensor and ESP32 microcontroller, we were able to translate simple hand movements into real-time motor commands, eliminating the need for traditional control devices.

* This project highlights key strengths such as **wireless control**, **ease of use**, and **potential for real-world applications** in defence, rescue, and assistive technology.
* It also reflects the importance of teamwork, problem-solving, and creativity in engineering design.

**Key Takeways**

* Gesture control can be a reliable alternative to physical controllers.
* Embedded systems and sensor fusion enable responsive and smart machines.
* Safety, accessibility, and innovation can be combined in a low-cost prototype.
* **Next Steps:**
* Add LIDAR or camera-based gesture recognition.
* Improve gesture detection using AI and machine learning.
* Develop a more robust chassis for rough terrain or outdoor navigation.