Hand-Gesture-Controlled Car with GPS System and Object Detection

Shuhurat Fariha*, Bivas Nandan Debnath*, Maisha Sameha*
Department of Computer Science and Engineering,
University of Asia Pacific
Email: 22201252@uap-bd.edu , 22201256@uap-bd.edu, 22201266@uap-bd.edu

***Abstract*—This paper presents the design and implementation of a hand-gesture-controlled robotic vehicle integrating real-time GPS tracking and basic object detection for safe navigation. The system uses an MPU6050 inertial sensor mounted on a glove (or handheld controller) to capture hand gestures, an ESP32 Dev Board as the transmitter, and an ESP32-CAM as the receiver and main controller of the robotic vehicle. Gesture commands are transmitted via ESP-NOW, offering low-latency wireless communication. The vehicle uses DC geared motors and a motor driver (TB6612FNG/DRV8833) to carry out motion commands, while an ultrasonic sensor mounted on a tiny servo detects obstructions to prevent collisions. A NEO-6M GPS module provides real-time coordinates for remote monitoring and emergency location reporting. The prototype demonstrates reliable GPS reporting, straightforward obstacle avoidance, and simple remote control for small-scale rescue, surveillance, and logistical scenarios. We provided important implementation details, test observations, and the reasons behind component selection. Future improvements will include better geolocation, interaction with cloud dashboards for remote monitoring, and sophisticated vision-based object detection.**

***Index Terms*—Hand-gesture control, ESP32, ESP-NOW, MPU6050, ESP32-CAM, NEO-6M GPS, obstacle avoidance, embedded systems.**

I. Introduction

Human-machine interaction for mobile robots is an active research and application domain. Gesture-based control offers an intuitive, low-effort interface for remote operation in environments where traditional control means are impractical. This work documents the design, development, and evaluation of a hand-gesture-operated robotic car with integrated GPS-based tracking and obstacle detection. The primary objectives are (1) to enable real-time wireless robotic control through hand gestures, (2) to allow live location reporting via GPS for monitoring and emergency response, and (3) to implement basic obstacle detection to improve operational safety.

A. Motivation and Problem Statement

In situations like last-mile delivery, surveillance, and search and rescue, robotic vehicles are becoming more and more common. Traditional control interfaces, such as remote consoles and joysticks, can be difficult or confusing. While GPS integration provides situational awareness, gesture control optimizes the interface. The key difficulties addressed are robust gesture identification in noisy sensor data, low-latency wireless command transmission, and steady power and motor control for dependable vehicle operation.

B. Scope

This project targets a prototype-level demonstration using off-the-shelf modules (ESP32, MPU6050, ESP32-CAM, NEO-6M GPS, HC-SR04) and common motor drivers and batteries. Advanced autonomous navigation and machine-learning-based vision are out of scope but we discussed this as future work.

II. Related Work

Numerous tutorials and scholarly studies have presented ESP32-based mobile systems and gesture-controlled robotics. This project was influenced by earlier works such as NEO-6M GPS integration tutorials, ESP32 with MPU6050 gesture mapping instructions, and ESP32-CAM projects for remote vision. This project combines these elements into a single demonstrative prototype, with a focus on low-latency ESP-NOW communication and practical power management for mobile use.

III. System Design and Methodology

A. Overview

The system consists of two main subsystems: a *transmitter* (gesture controller) and a *receiver* (robotic vehicle). After obtaining direction data from the MPU6050, the transmitter generates discrete gesture commands and transmits them to the vehicle's ESP32-CAM via ESP-NOW. The vehicle interprets motor control signals while also monitoring obstacle proximity with an ultrasonic sensor and location data from the NEO-6M GPS module.

B. System Architecture

Figure 1 illustrates the overall architecture of the Hand-Gesture-Controlled Car with GPS and object detection system.It consists of three primary components: the gesture controller, the robotic automobile, and the communication interface. Using the MPU6050 sensor, the gesture controller detects motion and uses ESP-NOW to wirelessly send commands in response. Through the ESP32-CAM, the robotic car receives these signals, interprets them, and adjusts the motors while continuously tracking GPS and distance.

C. Primary Hardware Components

Table I summarizes the key hardware modules used in the prototype, along with their roles and quantities.

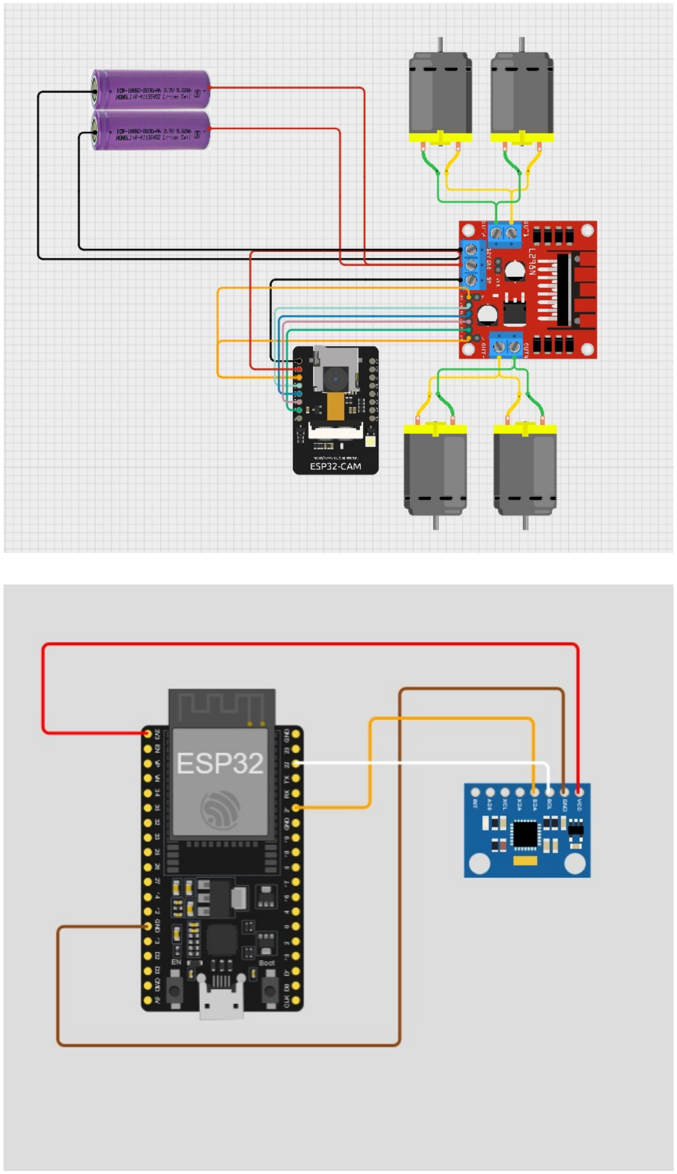

Fig. 1: System block diagram of the Hand-Gesture-Controlled Car.

TABLE I: Primary Hardware Components

Component	**Role**	**Qty**
ESP32 Dev Board	Transmitter(gesture processing)	1
ESP32-CAM	Main controller	1
MPU6050	Hand gesture(accelerometer + gyro)	1
HC-SR04	Obstacle detection	1
NEO-6M GPS	Location tracking	1
Motor Driver	Drive control	1
DC Motors	Vehicle motion (left/right)	2
LiPo Battery Pack	Power supply	1

D. Communication Protocol

ESP-NOW was chosen for wireless communication because of its low latency and peer-to-peer operation that doesn't require an access point. The transmitter recognizes the vehicle's MAC address as a peer and sends small data packets with mapped x/y/z gesture values (0–254 range). After receiving a packet, the receiver instantly activates motors by executing an ESP-NOW receive callback.

E. Gesture Mapping and Processing

The MPU6050 DMP (Digital Motion Processor) outputs yaw/pitch/roll values. These are converted to degrees and mapped into a byte range using a linear mapping (e.g., map from $[-90, 90]$ to $[0, 254]$). Thresholds are set for specific actions (forward, backward, left, right, stop). Short delays and smoothing in the transmitter help to reduce jitter from inertial data.

F. Motor Control

Digital direction pins (IN1–IN4) and PWM for speed on enable pins are used to control motor direction. The receiver maintains a safety timeout: if no packet is received for a certain interval (e.g., 1000 ms), the motors are turned off to prevent runaway behavior.

G. Obstacle Detection and Safety

A micro servo rotates the ultrasonic sensor (or a small scanning method is used) to cover the forward arc. The car can stop or redirect by using differential wheel motions if the measured distance is less than a certain threshold. The car's SOS button initiates an emergency routine that can request GPS coordinates for rescue, stop the car, and sound a buzzer.

H. System Control Flow

The overall control logic is summarized in the flowchart shown in Figure 2. The system begins with initialization of sensors and network setup. Sensor initialization and network configuration are the first steps in the system's operation. Gesture data from the MPU6050 is processed into discrete movements. The receiver receives these commands via ESP-NOW and decodes them to drive the motors. At the same time, the ultrasonic sensor ensures obstacle avoidance, and the GPS module supplies coordinate data. The system stops moving and initiates a safety response if it detects an obstruction within a predefined range. Data flow and decision logic are explained with a flowchart shown in Figure 2.

IV. Implementation

A. Prototype Setup

The prototype was built on a four-wheel robotic chassis and powered by two 18650 Li-ion batteries. An ESP32 development board was put on a breadboard and linked to an L298N motor driver to operate the four DC motors. The motor driver received power directly from the battery pack, while the ESP32 sent directional control signals via its GPIO pins.

The hand-gesture controller was implemented using a separate ESP32 module equipped with an MPU6050 sensor. The MPU6050 was connected to the ESP32 via the I2C interface (SDA and SCL), and this unit wirelessly communicated gesture-based commands to the automobile. The physical

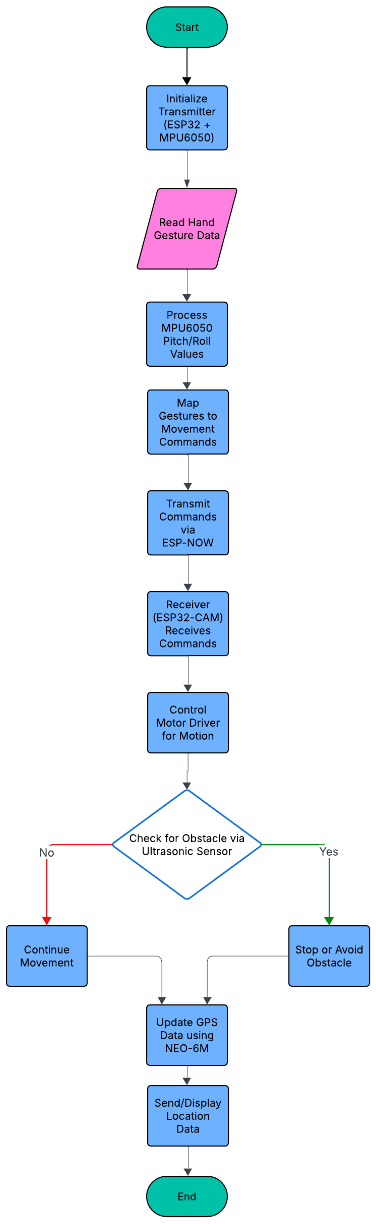
Start
Initialize Transmitter (ESP32 + MPU6050)
Read Hand Gesture Data
Process MPU6050 Pitch/Roll Values
Map Gestures to Movement Commands
Transmit Commands via ESP-NOW
Receiver (ESP32-CAM) Receives Commands
Control Motor Driver for Motion
Check for Obstacle via Ultrasonic Sensor
No
Yes
Continue Movement
Stop or Avoid Obstacle
Update GPS Data using NEO-6M
Send/Display Location Data
End

Fig. 2: Hand-Gesture Controlled Car Control Flow & Decision-Making Logic

configuration consists of the chassis, battery holder, motor driver, ESP32 controller board, and gesture controller module, as shown in the Figure 3.

B. Testing Procedure

The testing focused on.

1) Gesture-to-motion latency and reliability using ESP-NOW.
2) Accurate motor control for forward/backward movement and turns.
3) Safety behavior when communication is lost (signal timeout).

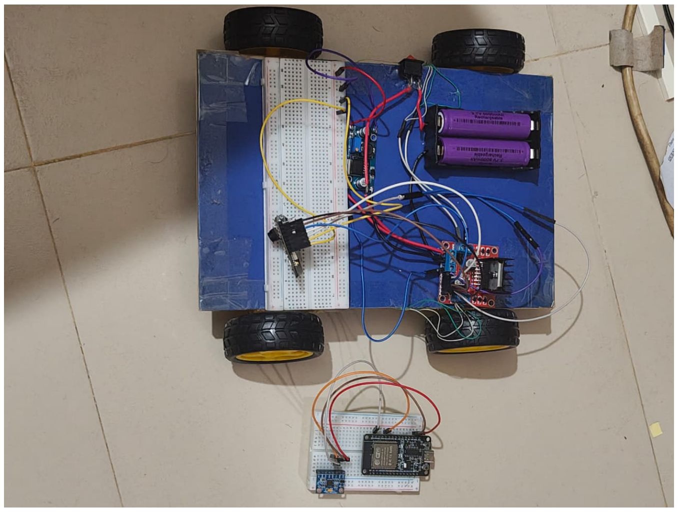

Fig. 3: System Prototype Image of Hand-Gesture-Controlled Car.

C. Key Results

To ensure its correct operation, the system was tested at various functional levels. The ESP32 communicated successfully with the L298N motor driver, and all four DC motors responded correctly to movement commands. The circuit remained stable with no power interruptions during operation. The MPU6050 provided smooth and accurate tilt readings, allowing the hand-gesture controller to send commands wirelessly with very low delay. During testing, the car immediately responded to forward, backward, left, and right gestures. Prototype image 3 shows the complete hardware setup, including the chassis, battery pack, ESP32 modules, motor driver, and sensors. Overall, the results confirm that the system couldn't achieve its key objectives, which are obstacle avoidance and real-time GPS monitoring with object detection.

V. Results and Discussion

A. Project Outcomes

The prototype was successful in showing the following.

- low-latency command transmission by using ESP-NOW at short distances (household/lab scale).
- Improved gesture detection through calibration and smoothing.
- Effective motor actuation and emergency stop after communication loss.

B. Limitations and Challenges

- Failed to implement the NEO-6M GPS module and ultrasonic sensor.
- Due to insufficient pins, sensors were not properly connected, which resulted in failure in some cases.
- GPS accuracy is limited in indoor or obstructed environments; NEO-6M requires an antenna and view of the sky for best fixes.
- Capacitors close to the ESP32-CAM help reduce Wi-Fi/camera current spikes; battery management requires

careful balancing between motor demands and controller/sensor requirements.
- The ESP32-CAM has limited GPIO availability; careful pin setup is required for expansion.

C. Lessons Learned

To create an effective prototype, sensors must be calibrated carefully, peers registered in ESP-NOW, and safety timeouts set cautiously. To validate safety logic, testing needs to include signal loss scenarios as well as realistic obstacle setups. The proper count of pins and the proper knowledge of pin configuration are required to make it a success.

VI. Conclusion and Future Work

Our idea presented a gesture-controlled robotic vehicle that utilized ESP-NOW communication, inertial-sensor-based gesture capture, ultrasonic-based obstacle avoidance, and a GPS module for live location tracking. The prototype achieves the key objectives of intuitive control and basic safety features. But the main important feature is something we couldn't achieve due to limited knowledge and expertise. For future work, it can surely be said that the first thing that should be implemented is the NEO-6M GPS module. Future enhancements will include cloud-based dashboards for remote communication, sensor fusion for improved localization, robust vision-based object detection with the ESP32-CAM, and energy-optimized motor control for longer operating time.

Acknowledgment

The authors acknowledge the Microprocessors and Microcontrollers Lab (CSE 316) at the University of Asia Pacific for guidance and access to laboratory resources.

References

[1] ESP32-WROOM-32 30-pin configuration. (Online). Available:https://esp32.com/viewtopic.php?t=9875
[2] Random Nerd Tutorials, ”ESP32 with NEO-6M GPS Module” (Online). Available:https://randomnerdtutorials.com/esp32-neo-6m-gps-module-arduino/
[3] Random Nerd Tutorials, ”Setting up ESP32 CAM” (Online). Available:https://randomnerdtutorials.com/program-upload-code-esp32-cam/
[4] Random Nerd Tutorials, ”ESP32 with MPU6050” (Online). Available:https://randomnerdtutorials.com/esp32-mpu-6050-accelerometer-gyroscope-arduino/
[5] Gesture-Controlled Robot Car. Available:https://www.ijraset.com/research-paper/gesture-controlled-robot-car?utm_source=chatgpt.com