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 executes motion commands through a motor driver (TB6612FNG/DRV8833) and DC geared motors while an ultrasonic sensor mounted on a micro servo scans for obstacles to enable collision avoidance. A NEO-6M GPS module provides live coordinates for remote monitoring and emergency location reporting. The prototype demonstrates intuitive remote control, basic obstacle avoidance, and reliable GPS reporting suitable for small-scale rescue, surveillance, and logistics scenarios. Key implementation details, component selection rationale, and test observations are reported. Future enhancements include advanced vision-based object detection, improved localization, and integration with cloud dashboards for remote monitoring.**

***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

Robotic vehicles are increasingly used in scenarios such as search-and-rescue, surveillance, and last-mile delivery. Conventional control interfaces (joysticks, remote consoles) may be bulky or unintuitive. Gesture control simplifies the interface while GPS integration ensures situational awareness. The main challenges addressed are robust gesture recognition under noisy sensor data, low-latency wireless command transmission, and stable power and motor control for reliable 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 discussed as future work.

II. Related Work

Gesture-controlled robots and ESP32-based mobile platforms have been demonstrated in multiple tutorials and academic works. Prior works that informed this project included ESP32 with MPU6050 gesture mapping guides, ESP32-CAM projects for remote vision, and NEO-6M GPS integration tutorials. This project synthesizes these components into a single demonstrable prototype with a focus on low-latency ESP-NOW communication and practical power management for mobile operation.

III. System Design and Methodology

A. Overview

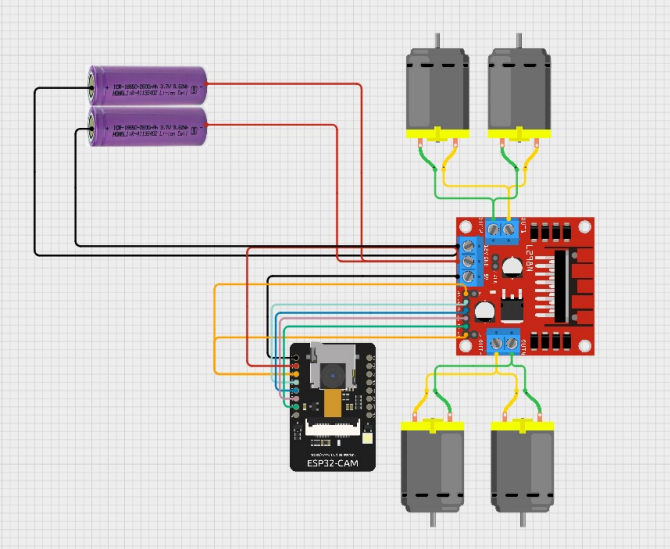
The system consists of two main subsystems: a *transmitter* (gesture controller) and a *receiver* (robotic vehicle). The transmitter reads orientation from the MPU6050, computes discrete gesture commands, and sends them via ESP-NOW to the ESP32-CAM on the vehicle. The vehicle interprets commands to drive motors and concurrently monitors obstacle proximity via an ultrasonic sensor and positional 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 comprises three main parts: the gesture controller, the robotic car, and the communication interface. The gesture controller captures motion through the MPU6050 sensor and transmits corresponding commands wirelessly using ESP-NOW. The robotic car receives these signals through the ESP32-CAM, processes them, and actuates the motors accordingly while continuously monitoring distance and GPS data.

C. Primary Hardware Components

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



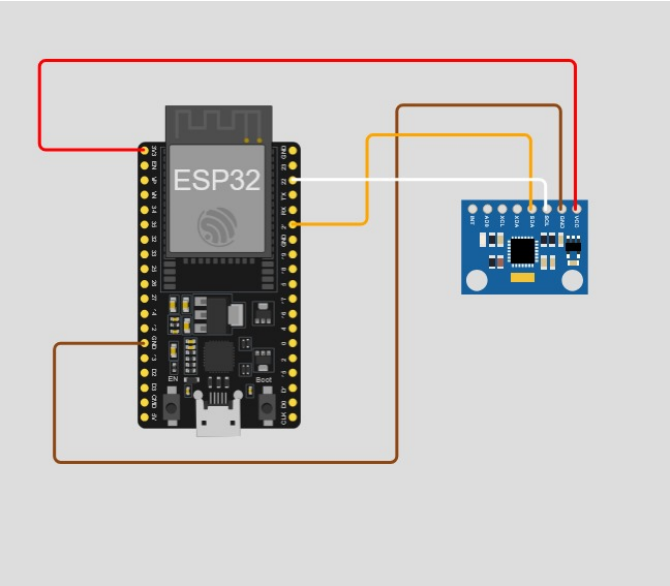


Fig. 1: System block diagram of the Hand-Gesture-Controlled Car.

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 selected for wireless communication due to its low latency and peer-to-peer operation without the need for an access point. The transmitter registers the vehicle's MAC address as a peer and sends small packets containing mapped x/y/z gesture values (0–254 range). The receiver runs an ESP-NOW receive callback to immediately actuate motors upon packet reception.

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 defined for discrete actions (forward, backward, left, right, stop). A short delay and smoothing in the transmitter help mitigate jitter from the inertial data.

F. Motor Control

Motor direction is controlled through digital direction pins (IN1-4) and PWM for speed on enable pins. The receiver maintains a safety timeout: if no packet is received for a defined interval (e.g., 1000 ms), it stops motors to prevent runaway behavior.

G. Obstacle Detection and Safety

A micro servo rotates the ultrasonic sensor (or a small scanning routine is used) to cover forward arc. If the measured distance is below a threshold, the vehicle can stop or reroute by commanding differential wheel motions. An SOS button on the vehicle triggers an emergency routine that stops the vehicle, sounds a buzzer, and can request GPS coordinates for rescue.

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. Gesture data from the MPU6050 is processed into discrete movement commands. These commands are sent via ESP-NOW to the receiver, which interprets them to drive the motors. Concurrently, the GPS module provides coordinate data, and the ultrasonic sensor ensures obstacle avoidance. If any obstacle is detected within a predefined range, the system stops movement and takes a safety response action. Data flow and decision logic are explained with a flowchart shown in Figure 2.

IV. Implementation

A. Transmitter (Gesture Controller)

The transmitter uses the MPU6050 DMP implementation to retrieve stable orientation data. After calibration and DMP setup, the system computes roll/pitch/yaw and maps them to bytes for transmission. The code periodically packages these into a small struct and uses `esp_now_send()` to transmit to the receiver's MAC address. The transmitter operates from a LiPo cell with a TP4056 for charging and an on/off switch and status LED.

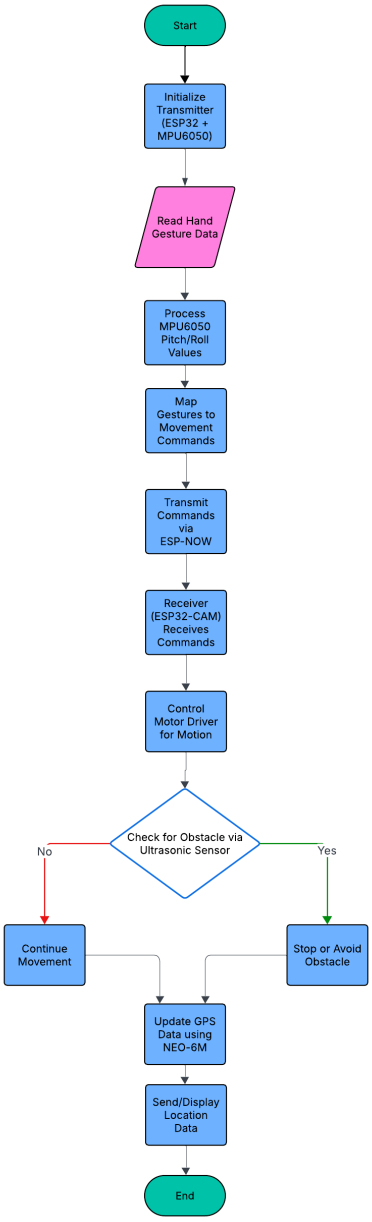


Fig. 2: Hand-Gesture Controlled Car Control Flow & Decision-Making Logic

B. Receiver (Robotic Vehicle)

The receiver (ESP32-CAM) initializes ESP-NOW in station mode, registers the receive callback, and sets up motor driver control pins and PWM channels. In the receive callback, the packet is parsed and `processMovement()` is invoked to translate the gesture bytes into left/right motor PWM values and directions. GPS parsing from the NEO-6M (via serial) runs concurrently to collect coordinates and lock status. Obstacle checks are performed in a periodic loop.

C. Software Components and Libraries

Implementation relies on:

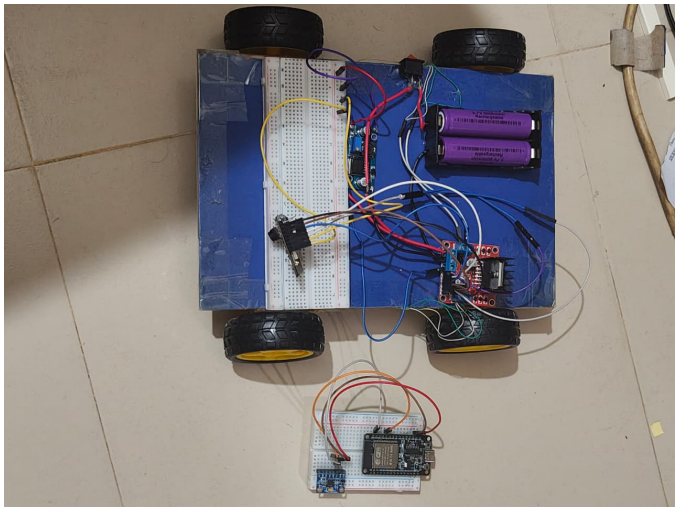


Fig. 3: System Prototype Image of Hand-gesture Controlled Car.

- ESP32 Arduino core and ESP-NOW APIs.
- MPU6050 libraries(I2Cdev,MPU6050_6Axis_MotionApps20) for DMP support.
- TinyGPS or direct NMEA parsing for GPS data extraction (optional).
- Standard servo and ultrasonic control routines.

D. Testing Procedure

Testing focused on:

1) Gesture-to-motion latency and reliability using ESP-NOW.
2) Motor control accuracy for forward/backward and turns.
3) Ultrasonic detection and reaction at various speeds.
4) GPS lock time and coordinate reporting during motion.
5) Safety behavior when communication is lost (signal timeout).

V. Results and Discussion

A. Functional Outcomes

The prototype demonstrated:

- Reliable low-latency command transmission using ESP-NOW at short ranges (household/lab scale).
- Stable gesture recognition after calibration and simple smoothing.
- Effective motor actuation and emergency stop on communication loss.
- Basic obstacle detection and avoidance using ultrasonic scanning and small reactive maneuvers.
- GPS coordinate reporting sufficient for coarse location monitoring (typical NEO-6M accuracy).

B. Limitations and Challenges

- GPS accuracy is limited in indoor or obstructed environments; NEO-6M requires an antenna and view of the sky for best fixes.

- Gesture recognition based on simple thresholds can misinterpret ambiguous gestures; more robust filtering or ML-based classification would improve reliability.
- Battery management requires careful balancing between motor demands and controller/sensor needs; capacitors near the ESP32-CAM help mitigate Wi-Fi/camera current spikes.
- ESP32-CAM has constrained GPIO availability; careful pin planning is essential for expansion.

C. Lessons Learned

Careful calibration of sensors, peer registration in ESP-NOW, and conservative safety timeouts are critical to produce a usable prototype. Testing should include signal loss scenarios and realistic obstacle arrangements to validate safety logic.

VI. Conclusion and Future Work

We presented a gesture-controlled robotic vehicle combining ESP-NOW communication, inertial-sensor-based gesture capture, ultrasonic-based obstacle avoidance, and a GPS module for live location tracking. The prototype meets the core objectives for intuitive control and basic safety features. Future improvements include adding robust vision-based object detection with the ESP32-CAM, sensor fusion for better localization, cloud-based dashboards for remote telemetry, and energy-optimized motor control for longer operational 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] J. Jeff Rowberg, "I2Cdevlib: MPU6050 and DMP examples", (Library).