International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

IOT-Based Head Controlled Smart Assistive Robot with Safety Intelligence

Nandana S[1], Navyashree V R[2], Pooja[3] P, Priyanka D N[4], Dr.Sundaresha M P[5]
Department of Electronics and Communication Engineering
Kalpataru Institute Of Technology, Tiptur, India

Abstract—People with physical disabilities may experience difficulties in controlling conventional mobility systems. This paper presents an IoT-based head-controlled smart assistive robot designed to support hands-free movement through head gestures. The system uses an MPU6050 sensor to detect head movements and an ESP32 microcontroller to process movement commands. Commands are transmitted wirelessly to a second ESP32 using ESP-NOW communication. The receiver controls the robot's motors through an L298N motor driver. The robot supports forward, backward, left, and right movement. An HCSR04 ultrasonic sensor detects nearby obstacles, and the robot is configured to stop when an obstacle is detected within a threshold distance of 20 cm. A communication timeout mechanism provides an additional stopping condition when valid control commands are no longer received. The Blynk platform is used for monitoring system information. The proposed prototype integrates gesturebased control, wireless communication, obstacle detection, and fail-safe stopping.

Index Terms—Assistive Robot, Head Gesture Control, ESP32, MPU6050, ESP-NOW, Obstacle Detection, IoT, Blynk.

I. Introduction

Mobility is an important part of independent living. People with limited movement in their hands or arms may find it difficult to operate conventional mobility systems. Assistive technology can provide alternative methods of interaction and control.

Head movement is one possible input method because it can be detected using inertial sensors. A head-controlled robot can translate selected head gestures into movement commands without requiring continuous manual operation of buttons or joysticks.

This paper presents an IoT-based head-controlled assistive robot prototype. The transmitter uses an MPU6050 sensor to measure head movement and an ESP32 to process and transmit commands. A receiver ESP32 controls the robot through an L298N motor driver. The robot supports four directional movements: forward, backward, left, and right. An HC-SR04 ultrasonic sensor detects nearby obstacles and stops the robot when the measured distance falls below the configured safety threshold. A communication timeout provides an additional stopping mechanism when valid commands are not received.

The main objective is to demonstrate a low-cost prototype integrating head gesture recognition, wireless control, obstacle detection, and remote monitoring.

II. Literature Survey

Assistive mobility systems have been developed using joysticks, voice commands, eye tracking, and body gestures. Joystick-based systems may not be suitable for people who cannot operate hand controls. Voice-controlled systems provide hands-free interaction, although their performance can be affected by background noise and speech limitations.

Inertial measurement units such as the MPU6050 combine an accelerometer and gyroscope to measure movement and orientation-related changes. These sensors are used in gesture recognition and motion-control applications. Microcontrollers such as the ESP32 support sensor processing and wireless communication in embedded systems.

Ultrasonic sensors are commonly used for short-range distance measurement and obstacle detection in mobile robots. Combining obstacle detection with gesture-based control can help reduce the risk of movement toward nearby objects.

The proposed prototype combines these approaches by using head gestures for movement commands, ESP-NOW for wireless communication, ultrasonic sensing for obstacle detection, and a timeout mechanism for stopping the robot when communication is interrupted.

III. Problem Statement

Conventional robot-control systems often require manual interaction through buttons, joysticks, or touch interfaces. These methods may be difficult for users with restricted hand movement. Wireless mobile robots also require mechanisms to respond when an obstacle is detected or when control communication is interrupted.

Therefore, a hands-free robot-control prototype is proposed using head gestures, wireless communication, and safetyrelated stopping conditions.

IV. Objectives

The main objectives of the proposed system are:

- To detect head movements using an MPU6050 sensor.
- To convert head gestures into movement commands.
- To support forward, backward, left, and right movement.
- To transmit commands wirelessly using ESP-NOW.
- To control the robot motors through an L298N motor driver.
- To stop the robot when an obstacle is detected within 20 cm.
- To stop the robot when valid commands are not received within the configured communication timeout.
- To monitor system information through the Blynk platform.

V. Proposed System

The proposed system consists of a transmitter unit and a robot receiver unit.

The transmitter contains an MPU6050 sensor and an ESP32 microcontroller. The sensor measures head movement, and the microcontroller interprets the measurements to generate movement commands. These commands are sent to the receiver using ESP-NOW.

The receiver contains another ESP32, an HC-SR04 ultrasonic sensor, an L298N motor driver, and DC motors. The receiver evaluates incoming commands and checks the obstacle distance before allowing movement. The motor driver operates the motors according to the received direction command.

If an obstacle is detected within the configured safety threshold, the movement command is overridden and the robot stops. A communication timeout is used as an additional failsafe. If a valid command is not received within 500 ms, the receiver stops the robot. Blynk is used for monitoring and is not the primary motor-control mechanism.

VI. System Architecture

Fig. 1 illustrates the architecture of the proposed system.

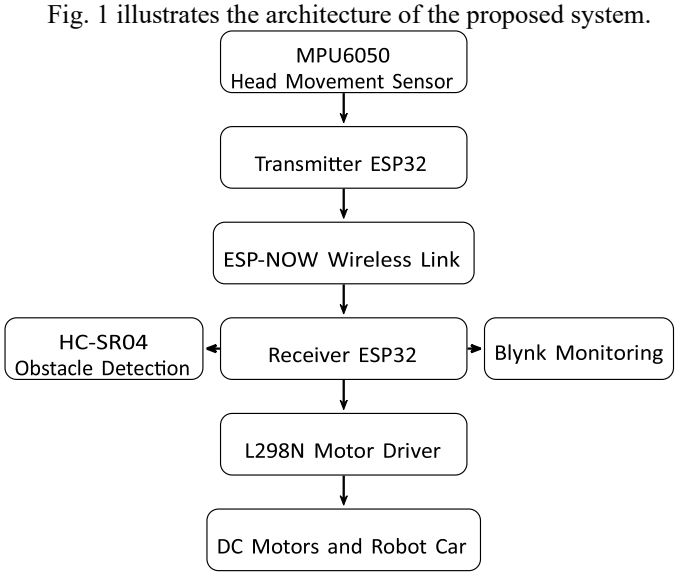


Fig. 1. Architecture of the head-controlled assistive robot

VII. Hardware Components

A. ESP32 Microcontroller

The ESP32 is used in both the transmitter and receiver units. In the transmitter, it processes sensor measurements and sends movement commands. In the receiver, it evaluates received commands, applies safety conditions, and controls the motor driver.

B. MPU6050 Sensor

The MPU6050 combines a three-axis accelerometer and a three-axis gyroscope. It measures changes in movement and orientation. These measurements are used to identify the selected head gestures.

C. ESP-NOW Communication

ESP-NOW is a wireless communication protocol supported by ESP32 devices. It allows the transmitter to send control messages directly to the receiver without requiring a conventional Wi-Fi access point. Reliable operation depends on valid packet reception and suitable handling of communication failures.

D. HC-SR04 Ultrasonic Sensor

The HC-SR04 measures the distance between the robot and an object using ultrasonic pulses. The receiver uses the measured distance to determine whether the robot should continue moving or stop.

The configured obstacle threshold is 20 cm. When a valid distance measurement is below this threshold, the system stops the motors.

E. L298N Motor Driver

The L298N motor driver provides the interface between the ESP32 control signals and the DC motors. It enables the robot to move forward, backward, turn left, turn right, and stop, depending on the motor wiring and control logic.

F. Blynk Monitoring

Blynk is used to monitor selected system information through a dashboard. Motor movement commands are processed by the ESP32 receiver rather than being dependent on Blynk dashboard controls.

VIII. Software Design

The software is divided into transmitter and receiver operations.

A. Transmitter Operation

The transmitter reads data from the MPU6050 sensor. The measured values are compared with configured gesture conditions. Based on the identified gesture, the transmitter generates a command such as forward, backward, left, right, or stop.

The command is sent to the receiver using ESP-NOW. Sensor calibration and suitable gesture thresholds are important for reducing unintended commands.

B. Receiver Operation

The receiver waits for incoming commands and checks whether they are valid. It also evaluates the latest obstacledistance measurement and the time elapsed since the last valid command.

The receiver gives priority to safety conditions. If an obstacle is too close or communication has timed out, it stops the motors. Otherwise, it executes the valid directional command.

IX. Gesture Classification

The MPU6050 sensor detects head movements by measuring changes in orientation. The ESP32 transmitter processes the sensor readings and generates movement commands. The robot supports four directional movements: forward, backward, left, and right.

Let θ represent the measured head tilt and θ_0 the neutral reference angle. Let ϕ represent the measured head rotation and ϕ_0 its neutral reference angle. A conceptual gesture classification is given by:

$$C(\theta,\phi) = \begin{cases} \text{Forward}, & \theta > \theta_0 + \Delta_f \\ \text{Backward}, & \theta < \theta_0 - \Delta_b \\ \text{Left}, & \phi < \phi_0 - \Delta_l \\ \text{Right}, & \phi > \phi_0 + \Delta_r \\ \text{Stop}, & \text{otherwise} \end{cases} \quad (1)$$

where Δ_f, Δ_b, Δ_l, and Δ_r are the configured gesture thresholds.

The exact thresholds and axis directions depend on the sensor orientation, calibration, and implemented gesture-detection logic. The detected command is transmitted wirelessly to the receiver ESP32 using ESP-NOW. The receiver controls the robot's motors through the L298N motor driver.

X. Safety Intelligence

A. Obstacle Detection

Let d be the distance measured by the ultrasonic sensor. The configured minimum permitted distance is 20 cm.

The obstacle safety condition is:

$$S_o = \begin{cases} 1, & d < 20 \text{ cm} \\ 0, & d \geq 20 \text{ cm} \end{cases} \quad (2)$$

When $S_o = 1$, the receiver overrides the requested movement command and stops the motors. The implementation should also define how invalid or missing distance readings are handled.

B. Communication Timeout

Let t_{now} be the current receiver time and t_{last} be the time at which the last valid command was received.

The communication timeout condition is:

$$S_c = \begin{cases} 1, & t_{now} - t_{last} > 500 \text{ ms} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

When $S_c = 1$, the receiver stops the motors until valid communication resumes according to the implemented commandhandling logic.

C. Combined Safety Decision

The overall safety decision is expressed as:

$$S = S_o \vee S_c \quad (4)$$

The final motor command is:

$$M_{final} = \begin{cases} \text{STOP}, & S = 1 \\ M_{requested}, & S = 0 \end{cases} \quad (5)$$

This decision gives obstacle detection and communication timeout priority over ordinary movement commands.

XI. Control Algorithm

The main control procedure is as follows:

1) Initialize the MPU6050, ESP32, wireless communication, ultrasonic sensor, and motor driver.
2) Read head movement data from the MPU6050.
3) Identify the corresponding head gesture.
4) Generate a forward, backward, left, right, or stop command.
5) Transmit the command using ESP-NOW.
6) Receive and validate the command on the robot.
7) Measure the obstacle distance using the HC-SR04.
8) Check whether the obstacle is closer than 20 cm.
9) Check whether the communication timeout has expired.
10) Stop the robot if either safety condition is triggered.
11) Otherwise, execute the requested motor movement.
12) Update available monitoring information through Blynk.
13) Repeat the process continuously.

XII. Safety Conditions

TABLE I
Safety Conditions and Expected Actions

Condition	Expected Action
Obstacle distance below 20 cm	Stop motors
No valid command for 500 ms	Stop motors
Valid command and safe distance	Execute command
Stop command received	Stop motors

XIII. Results and Discussion

The prototype is designed to demonstrate wireless headgesture control with obstacle detection and communication timeout protection. The robot supports four directional commands: forward, backward, left, and right. Its safety logic is intended to stop the motors when an obstacle is detected within the configured threshold or when valid commands are no longer received.

Fig. 2 shows the combined prototype image containing the robot car and transmitter unit. The image is placed within one column of the IEEE two-column layout.

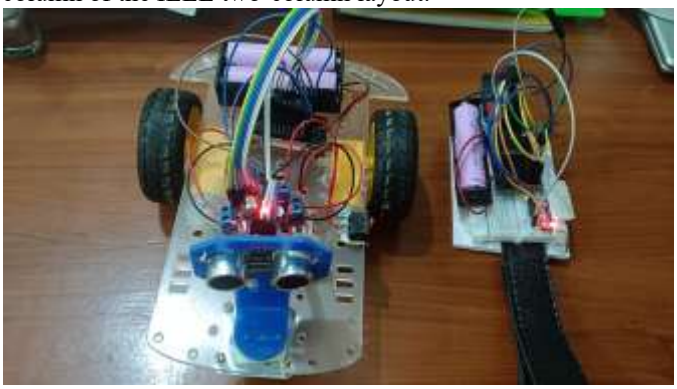


Fig. 2. Prototype showing the robot car and head-gesture transmitter unit

TABLE II
Functional Test Cases

Test	Input Condition	Expected Result
T1	Forward head gesture	Robot moves forward
T2	Backward head gesture	Robot moves backward
T3	Left head gesture	Robot turns left
T4	Right head ges-ture	Robot turns right
T5	Obstacle below 20 cm	Robot stops
T6	Communication timeout	Robot stops
T7	Stop command	Motors stop

A. Functional Testing

The following tests can be performed to evaluate the system. The expected results in Table II must be verified during physical testing. Actual outcomes should be recorded before claiming that a test has passed. Performance values such as recognition accuracy, response time, and wireless range should be measured rather than assumed.

XIV. Performance Evaluation

The system can be evaluated using gesture recognition accuracy, command response time, wireless communication reliability, and obstacle-stop behavior.

Gesture recognition accuracy can be calculated as:

$$\text{Accuracy} = \frac{\text{Correctly Classified Gestures}}{\text{Total Tested Gestures}} \times 100 \quad (6)$$

The communication success rate can be calculated as:

$$\text{Success Rate} = \frac{\text{Commands Received Correctly}}{\text{Commands Transmitted}} \times 100 \quad (7)$$

Response time can be measured as the interval between generating a gesture command and observing the corresponding robot response. These measurements should be collected through repeated experiments under defined testing conditions. No numerical performance result is claimed here because measured experimental data has not been provided.

XV. Applications

Potential applications of the proposed prototype include:

- Demonstration of hands-free robotic movement.
- Assistive technology research and development.
- Educational projects involving embedded systems and IoT.
- Gesture-based control for small mobile robots.
- Research into alternative human-machine interfaces.

The prototype is an experimental platform and is not a certified mobility aid for transporting people.

XVI. Limitations and Future Work

The system's performance depends on sensor calibration, gesture thresholds, communication reliability, motor characteristics, and ultrasonic sensor readings. Sudden head movements may produce unintended commands if classification thresholds are not properly configured. Ultrasonic sensing can also be affected by object shape, surface characteristics, and sensor placement.

Future work may include improved gesture filtering, userspecific calibration, more robust obstacle sensing, battery-level monitoring, additional safety verification, and quantitative testing under different operating conditions. Further evaluation would be required before adapting the prototype for real-world assistive mobility.

XVII. Conclusion

This paper presents an IoT-based head-controlled smart assistive robot prototype that integrates head gesture sensing, ESP32 microcontrollers, ESP-NOW wireless communication, motor control, obstacle detection, and communication timeout protection. The MPU6050 provides the input for gesture-based commands, while the receiver controls the robot's movement through an L298N motor driver. The robot supports forward, backward, left, and right movement. The HC-SR04 provides obstacle distance measurements, and the configured 20 cm threshold is used to trigger a stop. A 500 ms communication timeout provides an additional stopping condition when valid commands are not received. Blynk is included for monitoring system information.

The proposed design provides a foundation for further research into hands-free robotic control. Physical testing and measured performance data are required to establish its reliability and suitability for specific assistive applications.

REFERENCES

[1] InvenSense, MPU-6000 and MPU-6050 Product Specification. Available: https://invensense.tdk.com/

[2] Espressif Systems, ESP32 Series Datasheet. Available: https://www.espressif.com/

[3] Espressif Systems, ESP-NOW Documentation. Available: https://docs.espressif.com/

[4] SparkFun Electronics, HC-SR04 Ultrasonic Sensor Documentation. Available: https://www.sparkfun.com/

[5] STMicroelectronics, L298 Dual Full-Bridge Driver Datasheet. Available: https://www.st.com/

[6] Blynk, Blynk Platform Documentation. Available: https://docs.blynk.io/

[7] Espressif Systems, ESP-IDF Programming Guide. Available: https://docs.espressif.com/projects/esp-idf/

[8] NXP Semiconductors, I²C-Bus Specification and User Manual. Available: https://www.nxp.com/

[9] A. Jackowski, M. Gebhard, and R. Thietje, Head Motion and Head esture-Based Robot Control: A Usability Study, 2018.

[10] N. Rudigkeit and M. Gebhard, AMiCUS—A Head Motion-Based Interface for Control of an Assistive Robot, 2019.

[11] W. Rahmaniar, A. Ma'arif, and T. L. Lin, Touchless Head-Control (THC): Head Gesture Recognition for Cursor and Orientation Control, 2022.

[12] Arduino, Arduino Documentation and Programming Guide. Available: https://docs.arduino.cc/

[13] Espressif Systems, ESP32 Hardware Design Guidelines.Available: https://docs.espressif.com/projects/esp-hardware-design-guidelines/

[14] Arduino, Arduino Programming Documentation. Available: https://docs.arduino.cc/

[15] TDK InvenSense, MPU-6050 Six-Axis Motion Tracking Device Documentation.Available: https://invensense.tdk.com/