iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: https://doi.org/10.22214/ijraset.2026.85528

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

IoT-Based Fall Detection Prototype Using MPU6050 and Threshold-Based Motion Analysis

Dr. N. C. Patil[1], Manoj H[2], Namratha K P[3], Harish D[4], Khushboo D[5]
Dept. of ECE, Sri Krishna Institute of Technology, Bengaluru, India

Abstract: ***Falls can cause serious injury when assistance is delayed. This paper presents a hardware demonstration prototype using an ESP32 DevKit V1 and MPU6050 motion sensor to identify motion patterns associated with a possible fall. The implemented algorithm applies sequential thresholds to resultant acceleration and gyroscope angular-velocity magnitude; no trained machine-learning model is claimed. Hall-effect and capacitive inputs gate monitoring. When the programmed conditions are satisfied, the controller updates an LCD, activates a buzzer, checks for a recent valid GPS location, and attempts SMS transmission through an A7670C cellular modem. Functional development tests observed ESP32 serial output, Wi-Fi connection, GPS data, fall-sequence status, SMS submission, and an SMS-received indication. Detection accuracy, false-alarm rate, and delivery timing have not been quantified through repeated controlled trials.***
Index Terms: Fall detection, ESP32, MPU6050, threshold algorithm, IoT, GPS, emergency alert.

I. INTRODUCTION

Falls are a safety concern for older adults and people who may be unable to request help after an accident. An automated system can identify motion patterns associated with a possible fall and initiate an alert. Such systems must distinguish abrupt normal activities from possible falls and must account for communication availability.

This paper describes a hardware demonstration prototype, not a wearable belt or a clinically validated product. It integrates an ESP32 DevKit V1, MPU6050, Hall-effect sensor, capacitive input, 16×2 I²C LCD, buzzer, NEO-6M GPS receiver, and A7670C cellular modem. The principal software method is a sequential threshold algorithm using acceleration and angular-velocity magnitudes.

The contribution is a documented prototype architecture, an explicit description of the implemented thresholds, and a validation plan for sensor and alert behavior.

II. RELATED WORK

Fall-detection research has explored inertial sensors, ambient sensing, computer vision, threshold rules, and machine-learning methods. Accelerometers describe linear acceleration, while gyroscopes measure angular velocity. Combining both signals provides information about changes in movement and rotation. Threshold methods are computationally simple, but abrupt activities such as jumping, quickly sitting, or handling the device can create similar signals. Machine-learning methods require labeled data and independent evaluation; no trained classifier is implemented in the prototype described here [1]–[3].

Sensor and controller documentation should be used to confirm the selected measurement range, sensitivity, electrical connections, and communication interfaces [4], [5].

III. PROPOSED SYSTEM

The signal path is MPU6050 sensing → ESP32 acquisition and calibration → threshold sequence → fall flag → local indication and remote-alert attempt. The Hall-effect and capacitive inputs act as monitoring gates. When either required condition is absent,the code clears the partial trigger sequence. The assembled system is a demonstration prototype and is not described as a wearable belt.

Monitoring-gate behavior: The firmware checks the Hall-effect and capacitive inputs before continuing the fall-detection sequence. If either required condition is absent, the partial trigger states and counters are reset. When the programmed motion sequence is satisfied, the controller activates the local alarm, checks whether a recent GPS location is available, and attempts SMS transmission. A transmission attempt is not proof that the message was received.

©IJRASET: All Rights are Reserved

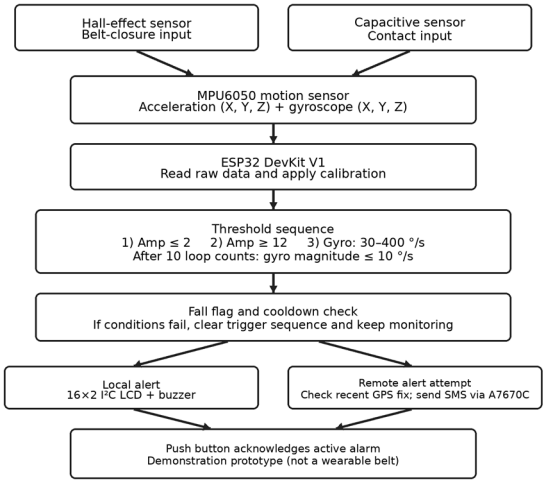


Fig. 1. System architecture of the demonstration prototype.

A. Hardware components

The ESP32 DevKit V1 reads sensor data and controls the response. The MPU6050 provides three-axis acceleration and three-axis gyroscope measurements. The Hall-effect sensor provides the configured belt-state input, while the capacitive sensor provides a contact-related input; their HIGH/LOW polarities must be verified on the actual hardware.

The NEO-6M GPS receiver provides coordinates when a valid satellite fix is available. The A7670C modem is used to attempt SMS transmission through AT commands. The LCD presents status, the buzzer gives a local alert, and the push button acknowledges an active alarm as implemented in firmware.

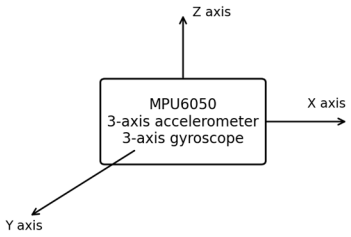


Fig. 2. MPU6050 axes and complementary motion measurements.

IV. THRESHOLD-BASED ALGORITHM

A. Accelerometer calibration

The raw accelerometer readings are converted into approximate acceleration in g using the implementation-specific offsets:

$$a_x = (AcX - 2050) / 16384.00$$

$$a_y = (AcY - 77) / 16384.00$$

$$a_z = (AcZ - 1947) / 16384.00$$

The offsets must be verified by calibration on the actual sensor. The divisor 16384 counts/g assumes the accelerometer is configured for the ±2 g range.

Why these offsets are used: the raw MPU6050 outputs include sensor bias as well as motion-related values. The implementation subtracts the measured accelerometer baseline (2050 counts on X, 77 counts on Y, and 1947 counts on Z) before dividing by 16384. These values are not constants specified for every MPU6050; they are calibration values selected for the sensor used when the code was prepared. If the board, mounting orientation, temperature, or sensor changes, the baseline can change and the offsets should be measured again.

A practical calibration procedure is to place the sensor motionless on a stable surface, collect multiple raw samples from each axis, and average them to reduce random noise. The expected gravity contribution depends on the sensor orientation: one axis may be close to +1 g or −1 g while the other axes are near zero. Therefore, the calibration should account for gravity rather than blindly

subtracting every resting reading to zero. The values 2050, 77, and 1947 should be retained only if repeat measurements show that they are suitable for this exact setup.

The divisor 16384.00 represents the approximate sensitivity of 16384 raw counts per g when the accelerometer full-scale range is configured to ±2 g. If the configured range is ±4 g, ±8 g, or ±16 g, the counts-per-g value changes and the conversion must be updated. A mismatch between the configured range and divisor causes incorrect acceleration magnitudes and can make the thresholds trigger too early or fail to trigger.

B. Resultant acceleration

$$\text{Raw_Amp} = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

$$\text{Amp} = 10 \times \text{Raw_Amp}$$

For illustration, if $a_x = 0$ g, $a_\gamma = 0$ g, and a_z = 1 g, then Raw_Amp = 1 g and Amp = 10. This is an example, not a measured prototype result.

C. Gyroscope calibration

$$g_x = (\text{GyX} + 270) / 131.07$$

$$g_\gamma = (\text{GyY} - 351) / 131.07$$

$$g_z = (\text{GyZ} + 136) / 131.07$$

$$G = \sqrt{g_x^2 + g_y^2 + g_z^2}$$

The offsets are implementation-specific. The factor 131.07 is consistent with a sensitivity near 131 counts/(°/s), generally associated with the ±250°/s range. Verify the configured range and calibrate the sensor before reporting final results.

The gyroscope offsets (270, 351, and 136) compensate for the raw sensor bias observed on the X, Y, and Z axes in the implementation. The code adds or subtracts each offset according to the measured sign. To obtain replacement offsets, keep the module stationary, read a sufficiently large sample set for GyX, GyY, and GyZ, calculate each axis average, and use those averages as the zero-rate bias values. Do not copy these numbers as universal MPU6050 constants.

The factor 131.07 counts per degree per second is approximately appropriate for the ±250°/s gyroscope range. If the firmware configures a different full-scale range, its sensitivity changes. The selected accelerometer and gyroscope ranges should therefore be confirmed in the MPU6050 configuration before interpreting threshold values or comparing results between devices.

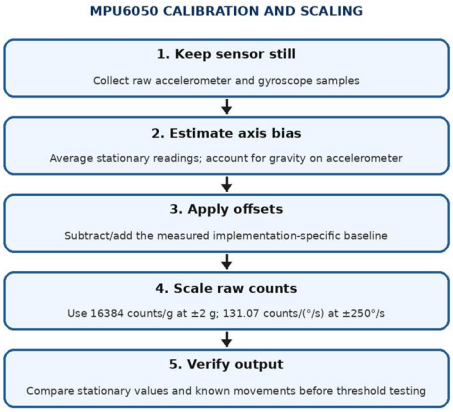


Fig. 4. Calibration and scaling procedure for the MPU6050.

D. Trigger sequence

Trigger 2 is evaluated only after Trigger 1; Trigger 3 is evaluated after Trigger 2. The conditions implemented in the code are:

Trigger 1: Amp ≤ 2

Trigger 2: Amp ≥ 12, after Trigger 1

Trigger 3: 30 ≤ G ≤ 400 °/s, after Trigger 2

After ten Trigger 3 loop counts, the program checks:

$$0 \leq \text{angleChange} \leq 10 \text{ °/s}$$

Here, angleChange is the gyroscope magnitude, not a physical angle. Ten loop counts nominally correspond to about one second only if each loop takes 100 ms; actual elapsed time may be longer. If the final condition is satisfied, the code sets the fall flag.

Threshold rationale: the first threshold looks for a low resultant acceleration that may occur during a brief unloading or free-fall-like phase. The second threshold requires a subsequent increase in acceleration that may be associated with impact. The third threshold checks for rotation using the combined angular-velocity magnitude. Finally, the program checks whether the angular-velocity magnitude has become low after ten loop iterations. This sequence uses the order of events as well as their magnitudes; a single high reading does not by itself satisfy all stages.

The thresholds Amp ≤ 2, Amp ≥ 12, $30 \leq G \leq 400$°/s, and final $G \leq 10$°/s are implementation-selected values. They should be treated as starting points for controlled testing, not universal clinical thresholds. Normal actions such as quickly sitting, turning, dropping the device, or handling the prototype may create similar signals. A more reliable future design could add a final posture estimate, additional temporal features, and a larger labeled dataset, but those improvements are not claimed as implemented here.

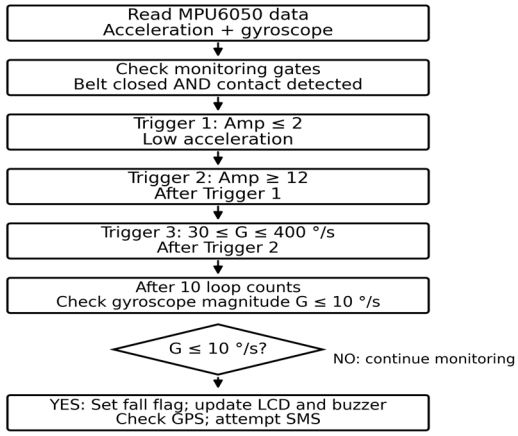


Fig. 3. Threshold sequence implemented in the prototype.

E. Alert process

The configured logic treats LOW from the Hall sensor as belt closed and HIGH from the capacitive input as contact detected; these polarities should be verified on the actual modules. After fall confirmation, the ESP32 updates the LCD, activates the buzzer, checks whether GPS location is valid and recent, and submits an SMS through the A7670C. The project screenshots/logs include both an "SMS Submitted" status and an "SMS Received" indication, so message reception was observed in the reported demonstration test. This observation confirms that at least one message-reception event was recorded; it does not establish delivery reliability across repeated trials.

V. IMPLEMENTATION AND TESTING

The firmware initializes the controller and modules, reads the motion and digital inputs, evaluates the trigger sequence, and manages the alert path. GPS parsing, AT-command handling, SMS submission, LCD updates, buzzer control, button handling, and trigger reset are separate functions in the implementation.

Testing should verify I²C sensor communication, digital input polarity, UART communication with GPS and modem, buzzer operation, and button behavior separately before integrated trials. A GPS fix alone does not demonstrate successful SMS delivery or detection accuracy.

VI. EXPERIMENTAL EVALUATION

The prototype was evaluated through functional checks of the controller, motion-sensor path, monitoring inputs, GPS output, and cellular alert path. The project report and captured serial/status screens document the observations summarized below. These are functional demonstrations rather than a statistically complete fall-detection evaluation; no numerical accuracy or response-time claims are made.

A. *Observed functional results*

Controller and serial output: the ESP32 firmware started and produced status information in the Serial Monitor. The development log also records a Wi-Fi connection and the IP address 10.150.121.66. This demonstrates that the controller and the tested Wi-Fi connection were operational during that session; Wi-Fi is not the same path used by the A7670C cellular SMS modem.

Motion and monitoring gates: the project screens show MPU6050 initialization/ready status and changes in belt and human-contact indications. The logs also show Trigger 3 activity and fall-detection status. These observations demonstrate that the firmware reached the corresponding states during the recorded tests, but they do not establish how often real falls are correctly detected or how often normal activities cause false alarms.

GPS: GPS NMEA sentences were observed during development, and a valid latitude/longitude fix was obtained in subsequent outdoor testing. A valid fix confirms that the receiver obtained location data under those test conditions. GPS availability indoors, acquisition time, and coordinate error were not quantified in the available results.

Cellular message: the captured project results show an “SMS Submitted” status and an “SMS Received” indication. Accordingly, the paper reports an observed SMS reception during demonstration testing rather than describing SMS delivery as unavailable. The available record does not establish a repeated-trial success rate, average delivery delay, or performance under weak cellular coverage.

Local alert and user interaction: the documented fall-response sequence includes LCD status updates and buzzer activation, while the firmware provides a push-button acknowledgement path. These functions should be recorded as separate pass/fail checks in future repeated tests, including whether the alarm stops and monitoring resets as intended.

Interpretation of the demonstration: the observed outputs confirm that the main software and hardware paths can be exercised in sequence under the recorded test conditions. The Wi-Fi connection is a controller connectivity check; it does not prove that the A7670C cellular path is operating. Likewise, receiving GPS coordinates confirms location acquisition for that test, but it does not establish a quantified position error. The “SMS Received” indication is evidence of at least one successful message-reception event, while repeatability, delivery delay, and success rate remain unmeasured. These distinctions are important when reporting a prototype: each subsystem should be evaluated separately and then as an integrated system, with dated logs and the exact hardware and firmware configuration retained for each trial.

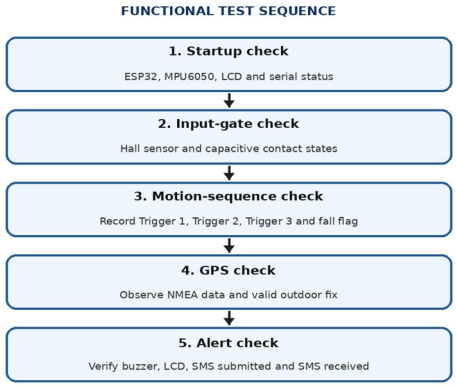


Fig. 5. Functional test workflow used to organize prototype checks.

B. *Limits of the available results*

The available evidence is qualitative and does not include a complete labeled set of fall and normal-activity trials. Consequently, true-positive (TP), true-negative (TN), false-positive (FP), and false-negative (FN) counts cannot be calculated. Accuracy, sensitivity, precision, recall, F1-score, false-alarm rate, mean detection delay, GPS position error, and SMS delivery time are therefore not reported. The successful SMS-received indication is a valid observed functional result, but it must not be converted into a 100% delivery claim without repeated trials.

For a publishable evaluation, run a defined number of safe, supervised simulated-fall tests and representative normal-activity tests. Record each trial’s ground-truth activity, sensor thresholds reached, final fall flag, local alarm response, GPS fix status, SMS submission, actual receipt, and elapsed times. Report the total number of trials and the conditions under which they were conducted. The prototype should not be tested by deliberately falling without appropriate supervision and safety precautions.

The validation plan should include stationary sensor calibration; normal activities such as walking, sitting, bending, and lying down; safe, supervised simulated-fall movements; and trials with monitoring gates enabled and disabled. Record the ground-truth activity, trigger sequence, alarm outcome, detection delay, GPS availability, and actual SMS receipt.

Calibration and repeatability procedure: before each trial session, allow the MPU6050 to initialize and keep the board motionless on a stable surface. Record multiple raw AcX, AcY, AcZ, GyX, GyY, and GyZ samples and calculate their averages. Compare the accelerometer averages with the expected gravity direction for the board orientation, and use the gyroscope averages to estimate zero-rate bias. Confirm the configured full-scale ranges before applying the 16384 counts/g and 131.07 counts/(°/s) factors. Repeat the readings after repositioning or replacing the sensor; do not assume the current offsets apply to another board. For every activity trial, record whether each trigger occurred, whether the fall flag was set, whether the buzzer activated, whether GPS data was available, and whether the SMS was actually received. These records support reproducibility but do not establish accuracy until a sufficiently large, labeled trial set is evaluated.

A. Evaluation metrics

Sensitivity = TP / (TP + FN) × 100%

Precision = TP / (TP + FP) × 100%

Accuracy = (TP + TN) / (TP + TN + FP + FN) × 100%

TP denotes a correctly detected simulated fall; TN denotes a correctly rejected normal activity; FP denotes a normal activity incorrectly flagged as a fall; and FN denotes a missed simulated fall. Report trial counts and test conditions alongside every metric.

VII. DISCUSSION AND LIMITATIONS

The fixed thresholds may not generalize across sensor orientations, mounting positions, movement types, or users. The algorithm does not explicitly estimate final body orientation; a low angular-velocity magnitude does not prove that a person is lying down. False alerts and missed falls are possible. The implementation-specific offsets must be checked on the actual sensor, and trigger timing should be measured with elapsed-time timestamps rather than inferred only from loop counts. The device described here is a demonstration prototype and is not established as a wearable belt or a clinically validated safety product.

Reproducibility checklist: record the MPU6050 range configuration; raw stationary axis readings; the offset values used; the firmware version; supply voltage; sensor orientation; Hall/capacitive input polarity; trigger thresholds; GPS fix availability; cellular signal conditions; and the exact message-reception evidence. This makes it possible for another group to repeat the functional test without assuming that the current offsets or thresholds will transfer unchanged to another board.

Remote alerting depends on a valid GPS fix, modem configuration, cellular coverage, SIM service, and stable power. The prototype is a demonstration system and has not been established as a medically validated or safety-certified device. Its results should be limited to the conditions actually tested.

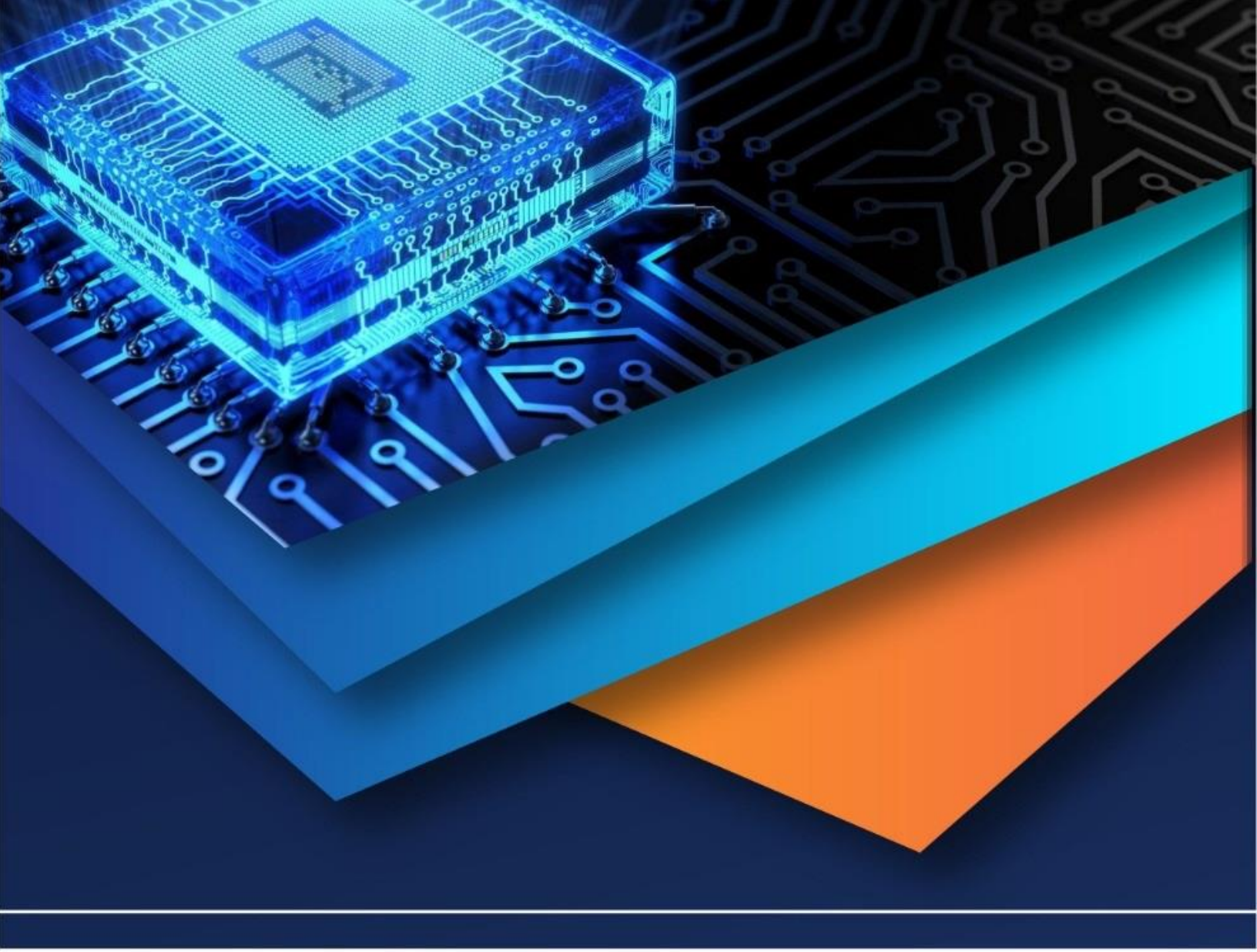
VIII. CONCLUSION AND FUTURE WORK

This paper describes an IoT-based fall-detection demonstration prototype using an ESP32 DevKit V1, MPU6050, Hall-effect and capacitive inputs, LCD, buzzer, GPS receiver, and A7670C modem. The implemented logic uses sequential acceleration and angular-velocity thresholds rather than a trained machine-learning model. The reported development tests observed controller/serial output, GPS data and an outdoor location fix, trigger/fall status, SMS submission, and an SMS-received indication. These results demonstrate operation of selected system functions under the recorded conditions, but do not establish detection accuracy or communication reliability.

Future work includes repeating the calibration procedure for each sensor, confirming full-scale register settings, tuning thresholds with labeled trials, adding an explicit final-posture check, measuring false alarms and missed falls, and quantifying GPS and SMS timing over repeated tests. A machine-learning method may be investigated later only after collecting a representative dataset and independently evaluating a trained model. Any future paper should report the actual test protocol and measured results, not assumed performance values.

REFERENCES

[1] M. Mubashir, L. Shao, and L. Seed, “A survey on fall detection: Principles and approaches,” Neurocomputing, vol. 100, pp. 144–152, 2013.
[2] A. Igual, C. Medrano, and I. Plaza, “Challenges, issues and trends in fall detection systems,” BioMedical Engineering OnLine, vol. 12, 2013.
[3] A. K. Bourke, J. V. O’Brien, and G. M. Lyons, “Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm,” Gait & Posture, vol. 26, no. 2, pp. 194–199, 2007.
[4] TDK InvenSense, “MPU-6000 and MPU-6050 Product Specification,” product documentation.
[5] Espressif Systems, “ESP32 Series Datasheet,” product documentation.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)