WWW.IJRPR.COM

**International Journal of Research Publication and Reviews**

Journal Homepage: www.ijrpr.com ISSN 2582-7421

# IoT-Based Smart Earthquake Monitoring System

***Dr. Mrs. V. B. Bagde, Prajwal Ahirkar, Rushikesh Junghare, Kunal Hingmire, and Monal Bansinge***

Department of Electronics and Communication Engineering, Priyadarshini College of Engineering, Nagpur, Maharashtra, India
Email: prajwalahirkar@gmail.com

**A B S T R A C T**

This paper presents a low-cost Internet of Things (IoT) prototype for monitoring vibration-related activity using an ESP32 microcontroller and an MPU6050 inertial measurement unit. The proposed architecture combines motion sensing, Wi-Fi-based data transfer, data logging, a Random Forest classification workflow, and a dashboard for viewing observations. The supplied dataset contains 1,500 synthetic records divided equally among five labels: NORMAL, WALKING, VEHICLE, CONSTRUCTION, and IMPACT. A stratified 80:20 train–test split was used, with 1,200 records for training and 300 for testing. The reported model correctly classified 218 test records, corresponding to an accuracy of 72.67%. Reported feature-importance scores were 72.20% for vibration, 14.19% for humidity, and 13.61% for temperature. These results describe classification on the supplied synthetic dataset only. They do not establish real-world earthquake-detection accuracy, earthquake-prediction capability, or structural safety. The paper therefore describes the system as an educational vibration-monitoring prototype and outlines the additional sensor calibration, field-data collection, and independent validation required before any seismic-monitoring claim could be supported.

**Keywords:** Internet of Things; vibration monitoring; ESP32; MPU6050; Random Forest; synthetic data.

## 1. Introduction

Earthquakes and other sources of ground motion generate vibrations that can be measured using inertial sensors. Continuous sensing and remote access to measurements can support research, education, and experimental monitoring. Conventional seismic instruments are designed for calibrated measurements, while low-cost microcontrollers and micro-electromechanical-system (MEMS) sensors provide accessible platforms for developing and testing monitoring concepts.

This work adapts an ESP32-based vibration-monitoring prototype into an IoT-oriented study of motion classification. An ESP32 development board interfaces with an MPU6050 accelerometer/gyroscope module and communicates over Wi-Fi. A software pipeline logs observations, evaluates a supervised classifier, and presents outputs through a dashboard or local display. The goal is to describe a low-cost sensing and data-processing workflow rather than to claim a validated earthquake-warning instrument.

A critical distinction is necessary: detecting vibration or assigning a sample to a class such as IMPACT is not equivalent to confirming an earthquake. The system does not predict future earthquakes or estimate earthquake magnitude. The reported classification results use synthetic data and must not be interpreted as validated seismic performance. The principal contributions are: (1) a layered IoT monitoring architecture based on ESP32 and MPU6050; (2) a documented machine-learning evaluation workflow using five activity labels; (3) a summary of the supplied synthetic-data evaluation; and (4) a validation plan for future testing with measured sensor data.

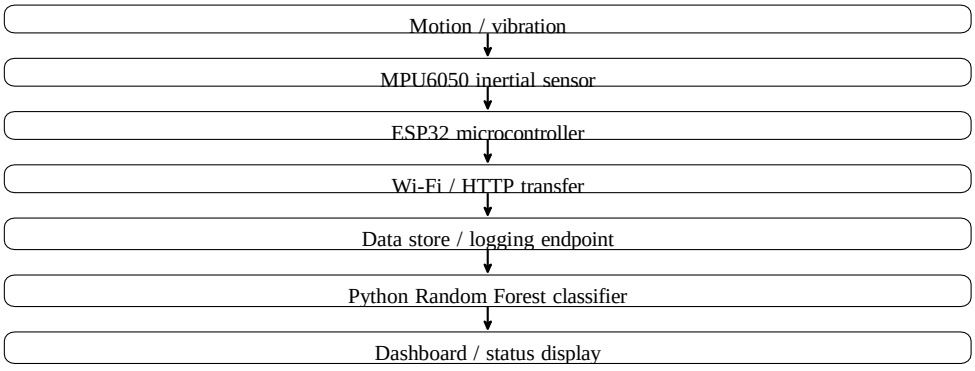



Figure 1. Layered architecture of the proposed IoT vibration-monitoring prototype.

## 2. Related Work

Earthquake early-warning research has examined low-cost sensing, embedded processing, distributed sensor networks, and rapid transmission of seismic information. Kim et al. (2021) studied an earthquake alert device using a low-cost accelerometer and considered the effect of signal-window characteristics on false alarms in noisy environments. Their work illustrates a central challenge for low-cost deployments: everyday structural and environmental vibrations may resemble earthquake-related motion.

Pierleoni et al. (2023) compared communication approaches for earthquake early-warning data transmission, emphasizing that communication delay and detection performance both affect the usefulness of a monitoring system. Packet formation, publication rate, connection recovery, and receiver-side processing are therefore important parts of the end-to-end design. Other work has explored smart sensors and accelerometer-based warning architectures, including an IEEE 1451-compliant sensor approach (Carratu et al., 2023) and an early-warning system using an ADXL335 accelerometer (Husni and Laumal, 2018). Recent research has also investigated AIoT-based early-warning methods for resource-constrained devices (Esposito et al., 2025).

The present study differs in its evidence base. It documents an ESP32–MPU6050 educational workflow and evaluates a Random Forest model on the supplied synthetic tabular dataset. It does not reproduce the real seismic validation of the cited studies, and it does not claim calibrated earthquake-detection performance.

## 3. Technical Background

Vibration signals depend on the source of motion, propagation path, mounting surface, sensor orientation, sampling rate, and signal-processing method. Walking, traffic, machinery, construction, and impacts can produce overlapping measurements. A single threshold can indicate that a signal has exceeded a chosen level, but it cannot reliably identify the cause of that motion on its own.

Window-based signal features such as root-mean-square (RMS) amplitude, standard deviation, peak-to-peak amplitude, and frequency-domain power can describe a signal more fully than one scalar measurement. The MPU6050 integrates a three-axis accelerometer and a three-axis gyroscope and communicates through the I2C bus (TDK InvenSense, n.d.). The ESP32 provides processing capability and wireless connectivity (Espressif Systems, n.d.). In an IoT system, measurements are acquired at the sensing node and transmitted to a remote endpoint for storage, analysis, and display.

Random Forest is an ensemble-learning method that combines predictions from multiple decision trees (Breiman, 2001). It can model nonlinear relationships in tabular data and provides feature-importance estimates. However, evaluation is meaningful only relative to the data distribution used. A random split of synthetic records cannot demonstrate generalization to a new building, sensor installation, or real earthquake.

## 4. Proposed System Architecture

The system is organized into six logica layers. (1) The sensing layer uses the MPU6050 to measure acceleration and angular velocity associated with motion. (2) The embedded layer initializes the sensor, reads samples, validates readings, and computes or formats the vibration feature. (3) The communication layer transfers records through Wi-Fi and an HTTP request to a configured endpoint. (4) The storage layer records observations for inspection and export. (5) The analysis layer prepares the input features and applies the trained Random Forest model. (6) The presentation layer displays recent measurements, timestamps, connection status, and the predicted activity class.

The data store may retain timestamps, vibration values, temperature, humidity, and contextual metadata. The reported classifier uses three inputs: vibration, humidity, and temperature. The physical source of temperature and humidity values must be stated accurately in the final implementation record; if those values are simulated or supplied by the dataset rather than measured by installed sensors, they must not be described as live sensor measurements.

Table 1. Main System Components

| Component | Role |
|---|---|
| ESP32 DevKit | Embedded controller and Wi-Fi communication |
| MPU6050 | Three-axis acceleration and angular-rate sensing |
| OLED (optional) | Local measurement and status display |
| Spreadsheet endpoint | Prototype data logging and record inspection |
| Python / scikit-learn | Model training, prediction, and evaluation |
| Dashboard | Display of recent values and predicted class |

## 5. Hardware and Software

The core hardware consists of an ESP32 development board and an MPU6050 motion sensor connected through I2C. An OLED display may be used to show local status if it is installed and tested. A buzzer, LED, or cellular module should be described as an implemented component only if it was physically integrated and verified; otherwise, it remains a possible extension.

The described software workflow uses an Arduino-compatible environment for firmware and Python for data processing and model evaluation. The project materials identify pandas for tabular data handling, scikit-learn for Random Forest training and evaluation, joblib for model persistence, Flask for a possible local dashboard, and Google Apps Script for HTTP-based spreadsheet logging. The final implementation should keep feature names, column order, endpoint configuration, model version, and record format consistent across all components. Optional or proposed components should not be presented as completed and tested features.

## 6. Design Requirements and Signal Representation

A monitoring node should collect samples at a documented sampling rate and retain their timing information. The feature calculation used by the deployed device must match the calculation used to prepare the training data. Each remote record should include a timestamp and device identifier so that readings can be associated with a node and reviewed chronologically. Temporary network loss should not silently stop sensing; buffering and a visible connection status are recommended. Finally, the system must not treat a class label as a safety-critical earthquake warning without validation against appropriate ground-truth measurements.

For a three-axis accelerometer, let the sampled acceleration components be $ax\,[n]$, $ay\,[n]$, and $az\,[n]$. The acceleration magnitude can be expressed as

$$a_m[n] = \sqrt{a_x[n]^2 + a_y[n]^2 + a_z[n]^2}. \quad (1)$$

If gravity is included in the measured signal, the magnitude contains a static component. Gravity compensation or high-pass filtering may be appropriate for some applications, but the selected method must be documented and validated rather than assumed.

For a window containing $N$ samples, the RMS magnitude after subtracting the window mean is

$$A_{\mathrm{RMS}} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(a_m[n] - \bar{a}_m)^2}, \quad (2)$$

where $\bar{a}_m$ is the mean magnitude within that window. The peak-to-peak amplitude is

$$A_{\mathrm{pp}} = \max(a_m) - \min(a_m). \quad (3)$$

Standard deviation similarly describes variation around the window mean. These are candidate signal features; they must not be

described as inputs to the current classifier unless they were actually used in model training.

If the sampling rate is $f_s$ samples per second and the selected window duration is $T_w$, the approximate number of samples per window is

$$N = f_s T_w. \quad (4)$$

Longer windows may stabilize the feature estimate but increase the time before a decision can be produced. Shorter windows respond more quickly but may be more sensitive to noise. Window duration should be selected through controlled experiments and evaluated alongside end-to-end latency. A vibration alert is not automatically an earthquake early warning.

## 7. Methodology

### *7.1 Dataset and Labels*

The supplied dataset contains 1,500 synthetic records, with 300 records assigned to each of five labels: NORMAL, WALKING, VEHICLE, CONSTRUCTION, and IMPACT. The reported input features are temperature, humidity, and vibration. Because the dataset is synthetic, these labels represent generated classes rather than independently verified observations of seismic events. The source and generation rules for each feature should be documented when the dataset is prepared for reproducibility.

### *7.2 Training and Evaluation*

A stratified 80:20 split assigned 1,200 records to training and 300 records to testing, with 60 test records per class. The reported Random Forest configuration uses 200 trees, a fixed random seed of 42, balanced class weights, and a minimum leaf size of 2, using the RandomForestClassifier

implementation in scikit-learn (Scikit-learn Developers, n.d.). These settings should be checked against the original training script before submission. The model's reported accuracy is calculated as

$$\text{Accuracy} = \frac{N_{\text{correct}}}{N_{\text{test}}} \times 100\%. \quad (5)$$

Precision, recall, and the F1-score are also useful for assessing class-specific performance. For a given class, precision and recall are

$$\text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN}, \quad (6)$$

where $TP$, $FP$, and $FN$ denote true positives, false positives, and false negatives. The F1-score is the harmonic mean of precision and recall. These metrics should be calculated from the original predicted labels and the complete confusion matrix before being reported as experimental results.

Feature importance is a model-specific score, not evidence of physical causation. A random split may place highly similar synthetic records in both training and test sets and may therefore overestimate performance on new physical conditions. A stronger future protocol should hold out complete recording sessions, sensor nodes, locations, or dates. Feature windows and hyperparameters should be selected using training data, preferably with cross-validation, and a final independent dataset should be reserved for evaluation. The fixed random seed supports repeatability of the split and estimator, but it does not make synthetic records representative of natural seismic signals.

### *7.3 Operational Workflow*

At startup, the ESP32 initializes the sensor and configures the communication interface. It reads motion samples, checks their validity, calculates the agreed feature, creates a record, and attempts to transmit it when a connection is available. The endpoint stores incoming records. The Python process loads the model and uses the same feature names, units, and ordering as those used during training. The dashboard presents recent records and predicted classes. If sensing, communication, or inference fails, the interface should show an explicit error or stale-data state rather than silently presenting an old result as current.

A key implementation requirement is feature compatibility. Firmware-derived vibration values and synthetic training values may use different scales or calculation rules. They are not interchangeable. The same documented feature definition, units, sampling assumptions, and preprocessing must be used for training and live inference. If the deployed feature differs from the synthetic feature, live predictions cannot be assumed reliable.

### *7.4 Data Transfer and Dashboard Logic*

The described communication layer uses Wi-Fi and an HTTP-based endpoint for logging. A robust record should include a timestamp, sensor features, derived values, device identity, and status information. The receiver should reject incomplete or malformed payloads and return a success or error status that the ESP32 can interpret. If the endpoint is unavailable, the node should retry with a bounded backoff and, where feasible, buffer records for later upload. Duplicate submissions should be detected using a unique record identifier or a combination of device identifier and timestamp.

The dashboard should distinguish fresh observations from stale records and show the timestamp, connection status, predicted class, and relevant measured values. Any alert should clearly identify the reported class and should state that the model has not been validated for earthquake detection. A buzzer or SMS function that was not physically tested must be described as a proposed extension, not as a completed feature.

### *7.5 Reliability, Safety, and Maintenance*

Monitoring nodes may fail because of sensor disconnection, power interruption, unstable Wi-Fi, server unavailability, clock errors, or software exceptions. The firmware should check sensor initialization and reading validity, while the server should log malformed records and endpoint failures. Periodic health messages can help identify nodes that have power but are no longer sending useful measurements. Where time synchronization is unavailable, the timestamp convention should be documented.

Table 2. Overall Model Evaluation on Synthetic Data

| **Metric** | **Reported value** |
|---|---|
| Total records | 1,500 |
| Training records | 1,200 |
| Test records | 300 |
| Correct predictions | 218 |

| Metric | Reported value |
|---|---|
| Incorrect predictions | 82 |
| Accuracy | 72.67% |

API credentials and endpoint tokens should not be hard-coded into public source code. Authentication, restricted API permissions, and access controls for dashboards reduce the risk of unauthorized data injection or disclosure. These reliability and security measures are recommendations unless their implementation has been verified in the prototype.

## 8. Results and Discussion

The following results are those reported for the supplied synthetic dataset; they are not measurements of physical earthquake-monitoring performance. The model correctly classified 218 of 300 test records and misclassified 82 records. The resulting accuracy was 72.67

Table 3. Class-Wise Correct Counts and Recall Derived from Reported Counts

| Class | Test records | Correct | Recall |
|---|---|---|---|
| NORMAL | 60 | 59 | 0.98 |
| WALKING | 60 | 58 | 0.97 |
| VEHICLE | 60 | 37 | 0.62 |
| CONSTRUCTION | 60 | 21 | 0.35 |
| IMPACT | 60 | 43 | 0.72 |

The reported feature-importance scores are 72.20% for vibration, 14.19% for humidity, and 13.61% for temperature. Vibration is therefore the most influential feature in the fitted model according to the reported output. This does not mean that the physical sensor measures ground motion with 72.20% accuracy, nor does it show that the classifier can detect a real earthquake. Feature importance depends on the dataset and the fitted model.

The difference between the relatively high NORMAL score and the lower CONSTRUCTION score shows that overall accuracy can conceal class-specific errors. A classifier that recognizes ordinary activity but confuses more energetic classes may be useful for exploring generated labels, yet it cannot be assumed suitable for safety-critical detection. False positives and false negatives also have different consequences. Any operational threshold must therefore be based on a documented use case and validated measurements rather than overall accuracy alone.

The class-wise recall values in Table 3 are derived from the reported correct counts divided by 60 test records per class. The full confusion matrix and original prediction file were not available for this revision; therefore, class-wise precision and F1-scores are not reported as verified results. In addition, a mismatch between synthetic training values and ESP32-derived live measurements may make real-time predictions unreliable.

## 9. Limitations and Safety Considerations

The main limitation is the use of synthetic activity data. The reported evaluation does not establish sensitivity to actual ground motion, false-alarm rate, earthquake-detection accuracy, magnitude estimation, or warning lead time. The physical end-to-end system also requires verification of sensor mounting, sampling rate, feature units, network transfer, data storage, and dashboard updates.

The prototype must not be presented as a certified seismic instrument, an earthquake predictor, or a replacement for official warning systems and established safety procedures. An IMPACT prediction means only that a sample resembles that generated class under the fitted model; it does not confirm an earthquake. The system should not be used as the basis for evacuation, structural inspection, or other emergency decisions. Data-transfer failures should be visible, and credentials should be protected.

### *9.1 Recommended Physical Validation Protocol*

A stepwise validation plan is recommended. First, verify the sensor interface and record raw three-axis measurements while the device is stationary; examine noise, bias, saturation, and repeatability. Second, conduct controlled non-seismic tests involving walking, nearby traffic, tapping on the mounting surface, and safe, authorized construction-like vibration. Label these as vibration-source observations, not earthquakes. Third, compare the low-cost sensor with a reference accelerometer or calibrated instrument under the same conditions. Fourth, test communication interruption, device restart, missing readings, and timestamp continuity.

For each test condition, collect repeated recordings across multiple sessions and, where possible, different mounting surfaces and sensor orientations. Define the labeling procedure before model evaluation and separate training and test records by session to reduce leakage. Report the confusion matrix, per-class precision and recall, macro-F1, false-alarm rate, missed-event rate, detection latency, transmission delay, and the proportion of missing or

corrupted records. If real earthquake records are used, obtain them from a traceable seismic data source and preserve metadata such as sampling rate and location. Any claim of earthquake detection or early warning requires a separately validated method and suitable ground truth.

## 10. Future Work

Future work should collect labeled measurements using the final sensor and mounting arrangement. Each recording should document the activity label, duration, surface, sensor orientation, sampling rate, and repetitions before evaluation. Data from entire sessions or locations should be held out for independent testing. A reference accelerometer or calibrated instrument is recommended when making quantitative claims about vibration amplitude.

Signal processing could use fixed-duration windows and evaluate RMS, standard deviation, peak-to-peak amplitude, and frequency-domain features. Random Forest could be compared with support vector machines, gradient-boosting methods, and lightweight neural networks using identical data splits. Evaluation should include macro-F1, balanced accuracy, confusion matrices, latency, and robustness to unseen conditions. Further engineering work may include a time-series database, local buffering during network loss, authenticated requests, retry logic, clearer error states, and optional multi-node monitoring.

A multi-node experiment could compare measurements from spatially separated devices to determine whether a recorded event is local to one node or appears at several locations. Such a study would require synchronized clocks, known node locations, reliable networking, and carefully evaluated decision rules. Edge inference on the ESP32 may reduce dependence on a remote server and lower notification latency, but it also introduces constraints on memory, processing time, model size, and power consumption. Any comparison between edge and server-side inference should use the same labeled test set and report latency under comparable conditions. Seismic interpretation ultimately requires appropriate instrumentation, independent ground-truth labels, and validation by qualified professionals. These activities are proposed future work and should not be described as completed experiments.

## 11. Conclusion

This paper described an IoT-oriented vibration-monitoring prototype based on an ESP32, MPU6050, Wi-Fi data transfer, data logging, and Random Forest classification. On the supplied synthetic dataset of 1,500 records, the model achieved a reported accuracy of 72.67% on a 300-record test split. Vibration accounted for 72.20% of the reported feature importance, while CONSTRUCTION had the lowest correct-count result among the five classes. These results demonstrate a data-processing and classification workflow on synthetic records; they do not validate real earthquake detection. The next steps are to standardize feature extraction, collect labeled measurements from the physical sensor, conduct independent tests, and verify the complete data path. Until that validation is completed, the device should be treated as an educational vibration-monitoring prototype rather than a protective earthquake-warning system.

**References**

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

Carratu, M., Dello Iacono, S., Paciello, V., Espirito Santo, A., & Monte, G. (2023). An IEEE 1451-001 compli-ant smart sensor for early earthquake detection. IEEE Open Journal of Instrumentation and Measurement, 2, 1–11. https://doi.org/10.1109/OJIM.2023.3311049

Esposito, M., Belli, A., Falaschetti, L., Palma, L., & Pierleoni, P. (2025). An AIoT system for earthquake early warning on resource constrained devices. IEEE Internet of Things Journal, 12(11), 15101–15113. https://doi.org/10.1109/JIOT.2025.3527750

Espressif Systems. (n.d.). ESP32 series datasheet. https://documentation.espressif.com/esp32_datasheet_en.html

Husni, E., & Laumal, F. E. (2018). The development of an earthquake early warning system using an ADXL335 accelerometer. In Proceedings of the 21st Saudi Computer Society National Computer Conference (NCC). https://doi.org/10.1109/NCG.2018.8593038

Kim, S., Khan, I., Choi, S., & Kwon, Y.-W. (2021). Earthquake alert device using a low-cost accelerometer and its services. IEEE Access, 9, 121964–121974. https://doi.org/10.1109/ACCESS.2021.3103505

Pierleoni, P., Concetti, R., Marzorati, S., Belli, A., & Palma, L. (2023). Internet of Things for earthquake early warning systems: A performance comparison between communication protocols. IEEE Access, 11, 43183–43194. https://doi.org/10.1109/ACCESS.2023.3271773

Scikit-learn Developers. (n.d.). Random Forest Classifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

TDK Inven Sense. (n.d.). MPU-6000 and MPU-6050 product specification and register map. https://invensense.tdk.com/ wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf