

IOT-Enabled Local Weather Analysis and Classification Using XGBoost Machine Learning Algorithm

Seerapu Swaroop Rani,

PG Scholar, Department of Computer Science and Systems Engineering, Andhra University College of Engineering (A), Visakhapatnam, Andhra Pradesh, India

Abstract—This project presents an IoT-enabled weather monitoring and forecasting system that integrates real-time environmental sensing with machine-learning-based weather classification. The sensing unit incorporates a DHT11 sensor to obtain temperature and humidity, an MQ-7 sensor to monitor carbon monoxide (CO), an MQ-135 sensor to measure carbon dioxide (CO₂) and related air-quality conditions, an LDR to measure ambient light intensity, and a rain sensor to identify precipitation conditions. The collected measurements are transmitted to the ThingSpeak cloud platform for remote monitoring and storage. In addition, the ESP32 sends real-time sensor observations to a local computer through serial communication. A pre-trained XGBoost machine-learning model processes these observations and determines the prevailing weather category, including Sunny, Cloudy, Rainy, Foggy, and DustStorm conditions. The experimental evaluation, using an 80:20 stratified dataset split, achieved an accuracy of 98.90%, with only seven misclassified samples among 637 test observations. The proposed system demonstrates the feasibility of combining low-cost IoT sensors, cloud connectivity, and machine learning for real-time localized weather monitoring and intelligent weather classification. The predicted weather information is presented through a web-based interface that provides real-time sensor values, weather classification, and notification alerts to the user.

Index Terms—Internet of Things (IoT), ESP32, Weather Monitoring, Weather Classification, Machine Learning, XGBoost, DHT11, MQ-7, MQ-135, Rain Sensor, ThingSpeak, Real-Time Data.

I. INTRODUCTION

Weather conditions influence agriculture, transportation, public infrastructure, environmental management, and emergency preparedness. Conventional meteorological stations provide high-quality observations, but dense deployment can be

costly and may not provide sufficiently localized information for every application. This work therefore considers an IoT-based alternative in which inexpensive sensors continuously capture environmental variables and a computational model interprets the observations to provide hyper-local, real-time weather intelligence.

The proposed system uses an ESP32 as the sensing and communication controller. The DHT11 sensor supplies temperature and humidity, the MQ-7 sensor measures carbon monoxide, the MQ-135 sensor is used for carbon-dioxide and general air-quality-related measurements, an LDR represents ambient light intensity, and a rain sensor provides precipitation-related information. The architecture also includes ThingSpeak for remote visualization and storage. Sensor observations are simultaneously forwarded through serial communication to a Python environment, where a trained classifier produces a weather label that is served through a lightweight web application.

The main contribution of this work is the development of an end-to-end IoT-based system that combines real-time environmental sensing with machine learning for weather classification. The proposed system covers the complete pipeline, starting from sensor data collection and pre-processing to weather prediction and presentation of results through a web-based interface. XGBoost is used as the machine-learning classifier, and its performance is evaluated using accuracy, precision, recall, F1-score, and the confusion matrix.

II. LITERATURE REVIEW

Recent studies have used IoT and Machine Learning (ML) techniques for weather and environmental monitoring. Different ML models, including Deep

Learning, Random Forest, SVM, Naive Bayes, and XGBoost, have been applied to analyze environmental data and predict weather conditions [1]–[10].

Many IoT-based systems use low-cost sensors such as DHT, MQ, LDR, and rain sensors along with microcontrollers such as ESP32 and Arduino. The sensor data can be stored and monitored remotely using cloud platforms such as ThingSpeak [2]–[4], [6]. Deep learning models can provide good prediction results, but they generally require more computational resources. In comparison, Random Forest and XGBoost are simpler and work effectively with sensor-based data while requiring less processing. Therefore, these ML models are suitable for real-time weather monitoring applications where computing resources are limited [5], [7]–[10].

Research Gap

Although the above studies demonstrate the effectiveness of IoT-based environmental monitoring and machine-learning techniques, many systems focus primarily on data collection and visualization or on individual forecasting variables. Some employ relatively complex forecasting architectures, while others use only a limited number of environmental parameters. The proposed work combines ESP32-based multi-sensor acquisition, ThingSpeak cloud visualization, serial communication with a local ML application, XGBoost-based weather classification, and web-based notification within a single framework. This combination is intended to provide a practical and low-cost solution for localized real-time weather monitoring.

Project Description

The proposed project delivers a complete, low-cost, end-to-end pipeline for hyper-local weather sensing and classification, built around commodity IoT hardware and a lightweight machine-learning model. An ESP32 microcontroller is interfaced with five sensors — a DHT11 for temperature and humidity, an MQ-7 for carbon monoxide, an MQ-135 for carbon dioxide and general air quality, an LDR for ambient light intensity, and a rain sensor for precipitation detection — to continuously capture a compact, six-dimensional environmental feature vector at approximately 2-second intervals. Each reading is transmitted along two parallel paths. First, it is pushed over Wi-Fi to a ThingSpeak cloud

channel, giving the user a browser-accessible, historical view of every sensor field for remote monitoring. Second, the same reading is streamed over a USB serial link to a host PC, where a Python application parses the incoming data and feeds it to a pre-trained XGBoost classifier. The classifier assigns the reading to one of five weather categories — Sunny, Cloudy, Rainy, Foggy, or DustStorm — in real time. A Flask-based web server then exposes the raw sensor values, the derived air-quality and thermal-comfort indicators, and the predicted weather label through a browser dashboard that also raises simple notifications/alerts for the end user.

The project was implemented and validated on physical hardware: the sensing unit was assembled and deployed to collect a labelled dataset of real sensor observations, the ThingSpeak channel was configured and populated with live data, the XGBoost model was trained and evaluated on an 80:20 stratified split of the collected dataset, and the web dashboard was built and tested against the live ESP32 stream. The resulting system therefore demonstrates, on real hardware and real data rather than in simulation alone, that a low-cost multi-sensor IoT front end combined with a lightweight gradient-boosted classifier can deliver accurate, real-time, localized weather intelligence together with cloud-based historical visualization and web-based user notification..

III. PROPOSED SYSTEM AND METHODOLOGY

The proposed system consists of four main components that work together for real-time weather monitoring and prediction, as shown in Fig. 1: (i) the sensing and edge layer, (ii) the cloud visualization layer, (iii) the machine-learning and web-server layer, and (iv) the web notification layer.

A. Sensing and Edge Layer

The sensing unit uses an ESP32 microcontroller, which offers integrated Wi-Fi connectivity and sufficient analog/digital I/O to interface with multiple low-cost sensors. The ESP32 periodically samples all connected sensors, formats the readings, and forwards them both to the cloud (via Wi-Fi) and to a host computer (via USB serial communication) at a fixed sampling interval of approximately 2 seconds. Fig. 2 shows the physical hardware prototype assembled on a breadboard, with the ESP32 at the center interfaced

to the DHT11, MQ-7, MQ-135, LDR, and rain-sensor modules. Table I summarizes the sensors used and the environmental parameters they capture.

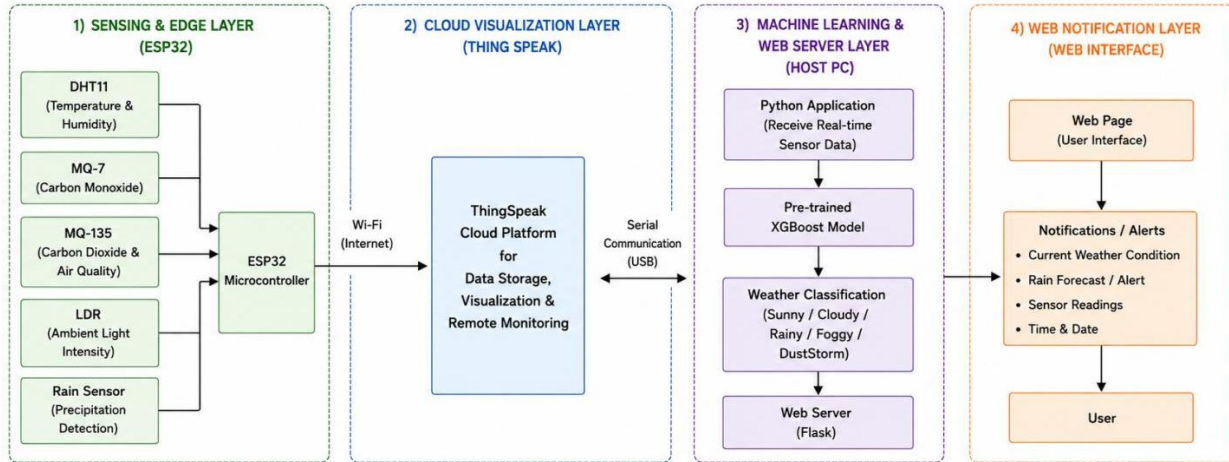


Fig. 1. Block diagram of the proposed IoT-based weather monitoring and forecasting system with web notification.

TABLE I. Sensors Used in the Proposed System

Sensor	Parameter Measured	Interface	Role in Classification
DHT11	Temperature (°C) and Relative Humidity (%)	Digital (1-Wire)	Distinguishes Sunny / Cloudy / Rainy thermal-humidity signatures
MQ-7	Carbon Monoxide (CO) concentration	Analog	Indicates air-quality degradation, e.g. during DustStorm/Foggy events
MQ-135	Carbon Dioxide (CO ₂) and general air quality	Analog	Supports air-quality-based discrimination between classes
LDR	Ambient light intensity	Analog	Differentiates Sunny/Cloudy/Foggy conditions from ambient brightness
Rain Sensor	Precipitation / surface wetness	Analog/Digital	Primary indicator for the Rainy class

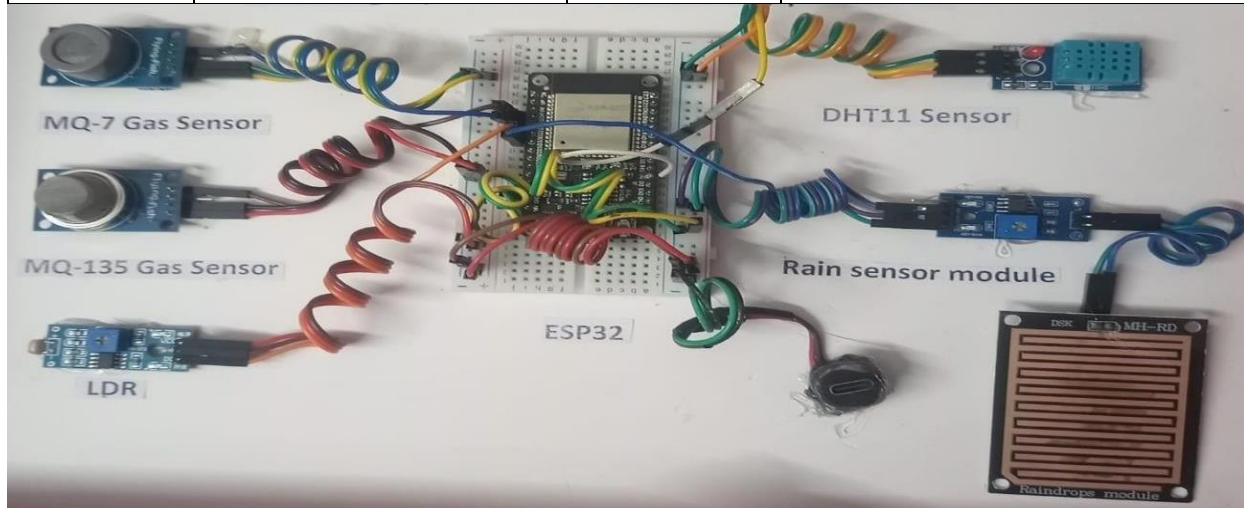


Fig. 2. Hardware prototype of the sensing and edge layer, showing the ESP32 microcontroller interfaced with the DHT11, MQ-7, MQ-135, LDR, and rain-sensor modules.

B. Cloud Visualization Layer

Sensor readings transmitted over Wi-Fi are pushed to a dedicated ThingSpeak channel, with one field allocated to each sensor parameter (temperature, humidity, CO, CO₂, rain, and light intensity). ThingSpeak provides remote, browser-accessible time-series charts for each field, enabling continuous monitoring and historical trend analysis without requiring local storage on the microcontroller. Figs. 5–6 show representative field charts captured from the deployed ThingSpeak channel.

C. Data Acquisition for Machine Learning

In parallel with the cloud upload, the ESP32 streams the same sensor readings over a USB serial connection to a host PC. A Python application listens on the serial port, parses each incoming line, and reconstructs a structured feature vector consisting of temperature, humidity, CO, CO₂, light intensity, and rain-sensor

value. Fig. 3 shows a representative snapshot of the real-time serial output received on the host computer, and Fig. 4 shows a sample of the resulting structured dataset records together with their assigned weather-class labels (Sunny, Cloudy, Rainy, Foggy, and DustStorm).

```

---- Sensor Data ----
Temp:      27.70
Humidity:  62.00
MQ135:     3229
MQ7:       1085
LDR:       496
Rain:      4095
DATA,27.7,62.0,1085,3229,496,4095

```

Fig. 3. Real-time sensor data received on the host PC via serial communication from the ESP32.

#	Temp (°C)	Humidity (%)	CO (MQ7)	CO ₂ (MQ135)	Light (LDR)	Rain	Label
1	27.6	60.5	1023	2948	499	4095	Sunny
2	27.6	60.5	1023	2923	500	4095	Sunny
3	27.6	60.6	1018	2931	496	4095	Sunny
4	27.6	60.6	1023	2934	499	4095	Sunny
5	27.6	60.6	1021	2929	496	4095	Sunny
6	27.6	60.6	1020	2931	499	4095	Sunny
7	27.6	60.6	1022	2938	497	4095	Sunny
8	27.6	60.6	1020	2925	501	4095	Sunny
9	27.6	60.7	1018	2908	497	4095	Sunny
10	27.6	60.6	1018	2903	501	4095	Sunny

Fig. 4. Sample dataset records showing sensor readings and the corresponding weather-class label.

The labelled dataset assembled from these readings was split using an 80:20 stratified train–test split, preserving the relative proportion of each of the five weather classes in both partitions. This yielded 637 samples in the held-out test set, which was used exclusively for the final performance evaluation reported in Section IV.

D. Machine Learning Model — XGBoost

XGBoost (Extreme Gradient Boosting) is an ensemble learning algorithm that builds a sequence of decision trees, where each new tree is trained to correct the residual errors of the trees built before it. It incorporates regularization terms to control model

complexity, uses second-order gradient information for more accurate split decisions, and supports efficient handling of tabular, structured sensor data. Because it does not require the large training datasets or heavy computational resources associated with deep-learning models, XGBoost is well suited to a resource-constrained, real-time embedded-sensing application such as the one proposed here, consistent with the observations made in the literature review.

The classifier takes the six-dimensional sensor feature vector (temperature, humidity, CO, CO₂, light intensity, and rain value) as input and outputs one of five weather categories: Sunny, Cloudy, Rainy, Foggy, or DustStorm. The trained model is serialized

(as `weather_model.joblib`) so that it can be loaded directly by the web-server process without retraining, enabling low-latency inference on each new sensor reading.

E. Software Architecture and Web Application

On the host computer, a Python application continuously receives the incoming serial stream and feeds each new observation to the pre-trained XGBoost model to obtain a predicted weather label. A Flask-based web server exposes this prediction, together with the raw sensor readings, model-confidence information, and derived indicators such as air-quality status and thermal comfort, through a browser-based dashboard. The web notification module presents the current weather condition, sensor readings, and relevant alerts to the end user, and can be refreshed on demand or polled periodically for near-real-time updates.

The overall processing sequence proceeds as follows:

- 1) Sensors continuously acquire temperature, humidity, CO, CO₂, light, and rain data.
- 2) The ESP32 processes and timestamps the readings and transmits them over Wi-Fi and USB serial simultaneously.
- 3) ThingSpeak stores and visualizes the incoming data for remote monitoring.

4) The host-side Python application receives the serial data and applies the pre-trained XGBoost model to predict the weather condition.

5) The Flask web interface displays the sensor readings, predicted class, and any relevant notifications.

6) The end user views the current weather status and alerts through the web dashboard.

IV. RESULTS AND DISCUSSION

This section presents the experimental outcomes obtained from the deployed hardware prototype, the cloud dashboard, and the trained XGBoost classifier.

A. Cloud-Based Data Visualization

The ThingSpeak channel continuously logged sensor readings over the monitoring period. Fig. 5 shows the temperature and humidity field charts, Fig. 6 shows the CO and CO₂ field charts, and Fig. 7 shows the rain and light-intensity field charts, all captured from the live channel dashboard. These visualizations confirm that the multi-sensor data pipeline reliably delivers time-stamped readings to the cloud for remote monitoring.



Fig. 5. ThingSpeak channel charts for temperature (Field 1) and humidity (Field 2).



Fig. 6. ThingSpeak channel charts for CO (Field 3) and CO₂ (Field 4).



Fig. 7. ThingSpeak channel charts for the rain sensor (Field 5) and light intensity (Field 6).

B. Model Training and Overall Accuracy

The XGBoost classifier was trained on the assembled sensor dataset and evaluated on the held-out 20% stratified test split. Fig. 8 shows the console output produced when running the training script, reporting an overall accuracy of 98.90% together with per-class precision, recall, and F1-score for the initial classes shown. Only seven samples out of 637 test observations were misclassified, corresponding to a test-set error rate of approximately 1.1%.

```
Next step: Run 'python train_model.py' to train your model on this hardware dataset!
PS C:\Users\Anand\Downloads\IoT-Enabled_Weather_Analysis_using_ML_Algorithms-main\ml> python train_model.py
Accuracy: 0.989010989010989
precision    recall  f1-score   support
Cloudy       0.98      0.98      0.98      133
DustStorm    1.00      1.00      1.00      120
Foggy        1.00      1.00      1.00      124
```

Fig. 8. Model training console output showing overall accuracy and the per-class classification report.

C. Confusion Matrix Analysis

Fig. 9 shows the confusion matrix obtained on the test set, with rows corresponding to the true weather class and columns corresponding to the predicted class, in the order Cloudy, DustStorm, Foggy, Rainy, and

Sunny. The matrix is strongly diagonal, indicating that the overwhelming majority of samples in every class were classified correctly. The small number of off-diagonal entries occurs mainly between classes that share overlapping environmental signatures — for example, Cloudy and Sunny conditions can present similar light-intensity and humidity readings under transitional weather — which is consistent with the very small number of total misclassifications observed.

```
Confusion matrix (rows=true, cols=pred):
[[131  0  0  0  2]
 [  0 120  0  0  0]
 [  0  0 124  0  0]
 [  2  0  0 124  2]
 [  1  0  0  0 131]]
Class order: ['Cloudy', 'DustStorm', 'Foggy', 'Rainy', 'Sunny']
Saved weather_model.joblib
```

Fig. 9. Confusion matrix of the XGBoost weather classifier on the held-out test set.

D. Class-Wise Performance

Table II summarizes the class-wise precision, recall, and F1-score reported by the deployed classification-performance module, which corroborates the confusion-matrix analysis above.

TABLE II. Class-Wise Classification Performance

Weather Class	Precision	Recall	F1-Score
Sunny	0.99	1.00	0.99
Cloudy	0.99	0.99	0.99
Rainy	1.00	1.00	1.00
Foggy	0.99	0.98	0.99
DustStorm	1.00	0.99	1.00

The uniformly high precision and recall values (≥ 0.98) across all five classes indicate that the model does not favor any particular class and generalizes well across the range of environmental conditions represented in the dataset. The Rainy class achieves perfect precision and recall, which is expected given the strong and largely unambiguous signal provided by the dedicated rain sensor.

E. Web-Based Monitoring and Notification Interface

Fig. 11 shows the ESP32 hardware-telemetry summary panel of the web dashboard, which reports the derived air-quality status, thermal-comfort assessment, sampling rate, and the live model accuracy. Fig. 12 shows the main IoT Weather Analysis dashboard, which displays the ESP32 connection status and the machine-learning-predicted weather condition (e.g., “Sunny — Clear skies, great weather ahead”) in a user-friendly format, along with quick-refresh and demo-preset controls.

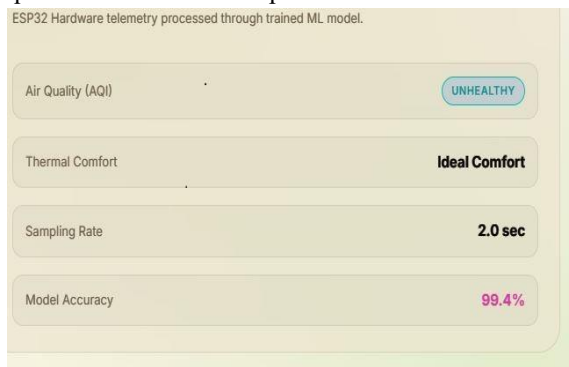


Fig. 11. Web dashboard panel showing derived air-quality status, thermal comfort, sampling rate, and live model accuracy.

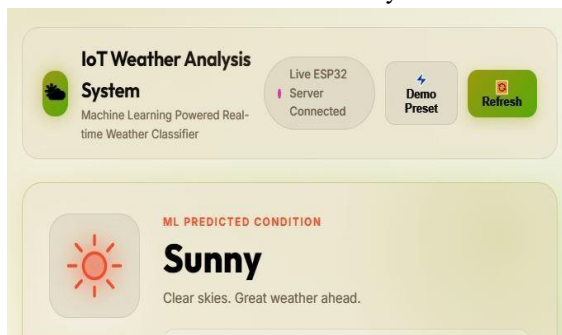


Fig. 12. IoT Weather Analysis web dashboard showing the live ESP32 connection status and the ML-predicted weather condition.

F. Discussion

The results demonstrate that a low-cost combination of DHT11, MQ-7, MQ-135, LDR, and rain-sensor readings, when processed by a gradient-boosted tree ensemble such as XGBoost, is sufficient to reliably discriminate between five common weather categories. Compared with the deep-learning-based approaches surveyed in Section II, which typically demand greater computational and memory resources, the proposed XGBoost-based pipeline offers a

lightweight alternative that is well matched to a host-PC or edge-server deployment while still achieving accuracy above 98%. The dual-path data flow — simultaneous cloud upload to ThingSpeak and local serial streaming for inference — further ensures that the system retains the benefits of remote, historical cloud visualization highlighted in prior IoT weather-station studies, while adding the real-time predictive and notification capability that many of those systems lack.

A limitation of the present evaluation is that the dataset was collected from a single deployment site over a limited monitoring period; broader validation across multiple geographic locations and seasons would help confirm the generalizability of the reported accuracy. Nonetheless, the consistently high per-class precision and recall values, together with the small absolute number of misclassifications (seven out of 637 test samples), support the practical feasibility of the proposed architecture for localized, real-time weather monitoring and alerting.

V. CONCLUSION AND FUTURE SCOPE

This paper presented an IoT-enabled local weather analysis and classification system that integrates an ESP32-based multi-sensor front end, ThingSpeak cloud visualization, a pre-trained XGBoost machine-learning classifier, and a Flask-based web notification interface. The system continuously acquires temperature, humidity, carbon monoxide, carbon dioxide, ambient light, and rainfall data, and classifies the prevailing weather condition into one of five categories — Sunny, Cloudy, Rainy, Foggy, or DustStorm — with an experimentally verified accuracy of 98.90% and only seven misclassifications across 637 test observations. The class-wise precision, recall, and F1-score results confirm that the model performs consistently well across all weather categories, and the web-based dashboard successfully delivers real-time sensor data, predictions, and alerts to the end user.

Future work will focus on extending the sensor suite (e.g., barometric pressure and wind-speed sensors) to capture additional weather categories and improve discrimination between visually and thermally similar conditions such as Cloudy and Foggy. Deploying multiple sensor nodes across different geographic

locations would enable spatial weather mapping and cross-site model validation. Additional directions include incorporating solar-powered, battery-backed operation for fully autonomous outdoor deployment, extending the web application into a native mobile notification service, and exploring ensemble or deep-learning hybrid models to further improve robustness under noisy or transitional weather conditions.

REFERENCES

- [1] G. Verma, P. Mittal, and S. Farheen, "Real Time Weather Prediction System Using IOT and Machine Learning," 2020 International Conference on Smart Computing and Informatics, 2020, doi: 10.1109/ICSC48311.2020.9182766.
- [2] K. Ioannou, D. Karampatzakis, P. Amanatidis, V. Aggelopoulos, et al., "Low-Cost Automatic Weather Stations in the Internet of Things," *Information*, vol. 12, no. 4, p. 146, 2021.
- [3] D. Parida, A. Behera, J. K. Naik, S. Pattanaik, R. S. Nanda, et al., "Real-time Environment Monitoring System using ESP8266 and ThingSpeak on Internet of Things Platform," 2019 International Conference on Communication and Signal Processing, 2019, doi: 10.1109/ICCS45141.2019.9065451.
- [4] M. S. Chinnathampy, J. C. Jobin, J. T., and V. P. M. B. Aarthi, "Cloud Computing's IoT based ThingSpeak, NodeMCU for Weather Monitoring," 2024 5th International Conference on Electronics and Sustainable Communication Systems (ICESC), 2024, doi: 10.1109/ICESC60852.2024.10689978.
- [5] I. Ioannou et al., "An Intelligent Anomaly Detection Approach for Accurate and Reliable Weather Forecasting at IoT Edges: A Case Study," *Sensors*, vol. 23, no. 5, p. 2426, 2023.
- [6] J. R. K. K. Dabbakuti, A. Jacob, V. R. Veeravalli, and R. K. Kallakunta, "Implementation of IoT analytics ionospheric forecasting system based on machine learning and ThingSpeak," *IET Radar, Sonar & Navigation*, vol. 14, pp. 341–347, 2020, doi: 10.1049/iet-rsn.2019.0394.
- [7] "IoT-driven real-time weather measurement and forecasting mobile application with machine learning integration," *Array*, vol. 27, p. 100474, 2025, doi: 10.1016/j.array.2025.100474.
- [8] "An Arduino-Based, Portable Weather Monitoring System, Remotely Usable Through the Mobile Telephony Network," *Electronics*, 2025.
- [9] "Real-Time AIoT-Driven Weather Forecasting on the Edge for Off-Grid Settings," *Network*, vol. 6, no. 2, p. 34, 2026, doi: 10.3390/network6020034.
- [10] T. Sarkar and P. Mog, "Smart IoT-centric Meteorological Monitoring Station Using Low-cost Arduino Nodes," *Research & Review: Electronics and Communication Engineering*, 2026.