Offline Rainfall Prediction Using Edge Machine Learning in an IoT-Based Pocket Weather Station

1st Imtiaz Asif
Department of Computer Science and Engineering
Leading University
Sylhet, Bangladesh
asifimtiaz60@gmail.com

2nd Pracurjo Chowdhury
Department of Computer Science and Engineering
Leading University
Sylhet, Bangladesh
pracurjo2003@email.com

3rd Afrin Jahan
Department of Computer Science and Engineering
Leading University
Sylhet, Bangladesh
afrenzhn@gmail.com

4th Sifat Ur Reza
Department of Computer Science and Engineering
Leading University
Sylhet, Bangladesh
sifaturreza@gmail.com

5th Shafiqa Mahzuzah Qureshi
Department of Computer Science and Engineering
Leading University
Sylhet, Bangladesh
qureshishafiqa659@gmail.com

6th Md. Arifuzzaman
Department of Computer Science and Engineering
Leading University
Sylhet, Bangladesh
arif_cse@lus.ac.bd

***Abstract*—Short-term rainfall prediction is very important for agricultural planning, especially for a country like Bangladesh, where most of the areas have limited internet access. This paper presents the design and implementation of a low-cost, portable, fully offline IoT-based rainfall prediction device that is compact in size. We built this system by including an ESP32 microcontroller with several environmental sensors and an embedded machine learning model to predict daily rainfall occurrence. Historical weather data from Bangladesh covering the years 2013 to 2022 was preprocessed and used to train various classification models. Even though the ensemble Random Forest showed higher accuracy, we chose Logistic Regression for its simpler and better work performance on resource-limited embedded devices. The trained model was converted from Python to C++ format and deployed on the ESP32. Our experiments show that the Pocket Weather Station provides reliable rainfall predictions while working entirely offline, making it suitable for smart irrigation and agricultural automation in areas with poor or intermittent connections.**

***Index Terms*—Internet of Things; Rainfall Prediction; Embedded Machine Learning; Smart Irrigation;**

I. INTRODUCTION

Rainfall prediction plays a vital role in terms of agriculture. Farmers are heavily dependent on weather forecasts to apply irrigation to their crop fields.

In recent advances in the Internet of Things (IoT), many researchers have developed low-cost weather monitoring systems using IoT-based technology [12], [14], [15]. Some existing solutions collect data through the device and send it to a server or cloud to store, visualize, or analyze data [7], [13]. Although cloud-based processing offers high computational capability, it makes the system impractical and unrealistic for rural and remote areas [8], [9].

Studies have shown that classical machine learning and deep learning models can effectively forecast rainfall when deployed in server-based environments [10], [11]. Artificial Neural Networks and deep learning techniques outperform traditional statistical methods in rainfall prediction [2]. However, these models require huge memory space and computational resources, which limits their use on resource-constrained embedded system [3], [4].

The emergence of edge computing has made it possible to execute lightweight ML models directly into embedded devices, limiting the dependency on cloud services and enabling local interface [5]. This paradigm has allowed faster response time and improved system reliability in places with poor or no internet connection. Micro-controllers like ESP-32 are widely used in environmental sensing applications as they come with integrated wireless capabilities, affordable and energy efficient [6]. Nevertheless, most of the ESP-32 devices are used as a monitoring device rather than using them directly for device intelligence.

In this work, we propose an IoT-based pocket weather solution that integrates environmental sensors like DHT22, LDR, Digital Barometric Pressure Sensor with an on-device integrated machine learning model to predict the rainfall even without any internet connection. This work aims to design a smart, compact, low-cost, energy-efficient device that predicts rainfall fully offline, and the device is easily extendable to smart irrigation and agricultural automation applications.

A Logistic Regression model is implemented and deployed on an ESP32 microcontroller to demonstrate real-time rainfall prediction under resource-constrained conditions.

II. LITERATURE REVIEW

The fast growth of the Internet of Things (IoT) has allowed for ongoing and real-time weather monitoring using distributed sensor networks and embedded systems. Early IoT weather monitoring solutions mostly focused on measuring environmental factors like temperature, humidity, pressure, and rainfall. They sent this data to remote servers for visualization and analysis [12], [14], [15]. These systems showed that low-cost automated weather stations could work, but they offered little beyond just collecting data.

Several cloud-based IoT weather monitoring setups have been suggested to improve scalability and data access. Kapoor et al. [7] introduced a cloud-based weather station that depends on centralized data processing. Meanwhile, Mohapatra et al. [13] highlighted the integration of affordable sensors with cloud analytics. Recent studies have also reported similar real-time monitoring systems that focus on better sensing accuracy and reliability [8], [9], [16].

Early research showed that traditional machine learning methods are effective for weather forecasting [11]. Deep learning techniques have since improved prediction accuracy by capturing complex patterns [10]. Comparative studies reveal that artificial neural networks and deep neural networks can achieve high accuracy in predicting rainfall when used in server-based environments [2]. Likewise, IoT-enabled weather monitoring systems combined with machine learning algorithms have shown promising results for short-term rainfall forecasting [1].

Ohemu et al. [3] pointed out these limitations when trying to implement real-time weather forecasting on microcontroller-based systems. Recent projects like MCUNet have aimed to enable compact deep learning on microcontrollers. However, these solutions need complicated optimization techniques and special deployment processes [4].

Edge computing has recently surfaced as an alternative to lessen reliance on cloud infrastructure. Al-Jame et al. [5] showed that edge-based embedded controllers can effectively support smart irrigation applications without depending on cloud services.

Many studies have confirmed the effectiveness of ESP32-based weather monitoring systems for real-time environmental sensing and data transmission [6], [8], [9].

While there has been a lot of research on IoT-based weather monitoring, cloud-related forecasting, and machine learning-based prediction, a clear research gap still exists.

III. SYSTEM OVERVIEW

A. Hardware Architecture

This proposed system is built around an ESP32 DevKit V1 because of its low power consumption, sufficient computational capability, and embedded wireless support. Accordingly, the following sensors and components are integrated:

- DHT22 for measuring temperature and humidity
- Digital barometric pressure sensor module for atmospheric pressure
- LDR for ambient light intensity
- Potentiometer as a low-cost alternative to an anemometer for the prototype
- 16×2 LCD display with I2C module

This configuration is designed to achieve low cost and easy portability with support for real-time data acquisition and inference.

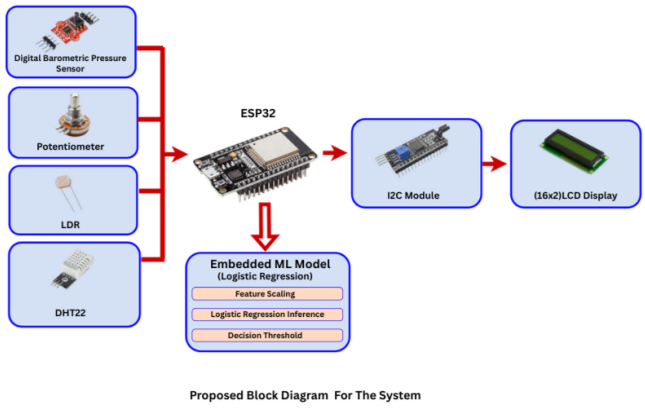

Fig. 1. Proposed Block Diagram for the System

Fig. 1 shows the components that were used for the device and how they communicate with each other as a system.

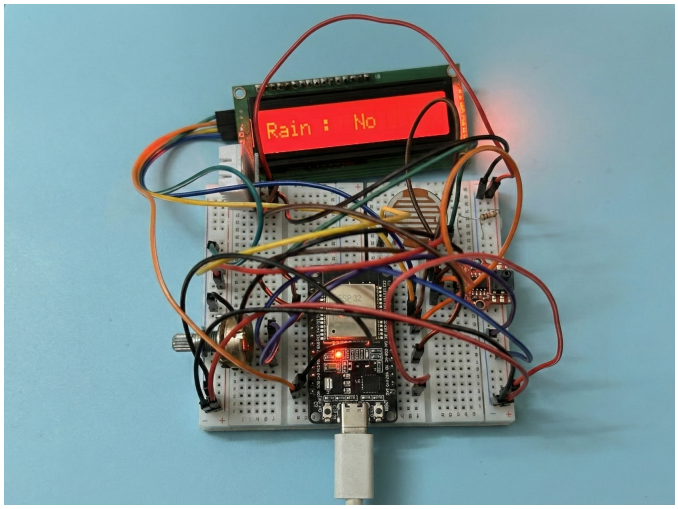

Fig. 2. Hardware Prototype

Fig. 2 is the prototype that was built to implement the system and evaluate its performance in real world scenario.

B. System Workflow

Fig. 3 depicts the overall system architecture. Environmental parameters collected from the sensor nodes are processed locally on the ESP32. Feature scaling and inference are

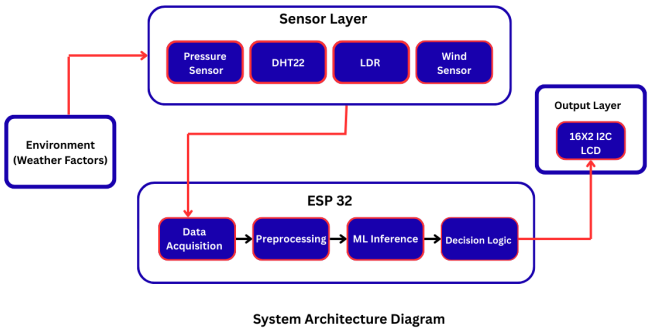

Fig. 3. System architecture diagram

performed directly on the device without requiring internet connectivity. The predicted rainfall status and probability are displayed on the LCD, allowing the user to easily interpret the result.

IV. DATASET AND PREPROCESSING

A. Dataset Description

The dataset used in this study consists of historical weather data collected in Bangladesh from 2013 to 2022 [17]. Each record corresponds to a single day and includes several meteorological variables, namely temperature, humidity, atmospheric pressure, wind speed, wind gust, wind direction at two separate times of a day, and hours of sunshine. The target variable is the occurrence of rainfall, which is binary and indicates whether rainfall occurred on a given day.

B. Data Preprocessing

The following preprocessing steps have been taken to prepare the data for the model training and embedded deployment:

There were some inconsistencies in the format of the date feature, which were handled by converting all the dates into a common format during the preprocessing phase. Later, features like DayOfYear and Month were extracted from the date feature.

The purpose of our work was to implement our trained model on a low-cost embedded system, so we had limitations in terms of which features to pick and which data could be collected by various sensors. With that in mind, we had to drop some features like the portion of sky covered by clouds, or the evaporation during a particular day (in millimeters), which could not be attained by our selected sensors.

Furhthermore, important and meaningful features were engineered from the existing features, such as the average temperature, temperature range, humidity range, pressure trend, etc.

One major challenge we faced was the high target class imbalance present in the dataset. Methods like SMOTE were not helpful because of the nature of the dataset. Generating synthetic data for weather factors lead to poor performance of a model. So, we had to use undersampling of the majority class, resulting in a smaller dataset, which actually helped develop a model that was lightweight and perfect for implementing on an embedded system.

As for categorical features like wind direction, one-hot encoding was used. Extracted features like day of the year and month are not linear, but periodic. Which is why we used cyclical encoding technique to preserve the meaning of those features. Cyclical encoding maps the feature onto a circle, preserving continuity.

Finally, before training the model, the features were scaled using Standard Scaler, except for the features that were already encoded.

These steps significantly increased the model performance by reducing computational complexity and made the model suitable for deployment on resource-constrained embedded systems.

V. METHODOLOGY

This section outlines how we designed and implemented the Pocket Weather Station. The focus is on system workflow, acquiring sensor data, preparing features for embedded use, and predicting rainfall on the device using a simple machine learning model.

A. Overall System Workflow

The system follows a step-by-step process. It starts with collecting environmental data, then preparing features, running the model on the device, and finally visualizing the results. All computations happen locally on the ESP32 microcontroller, allowing it to work offline without depending on external servers or cloud services.

B. Sensor Data Acquisition

We gather environmental data using affordable sensors connected to the ESP32 microcontroller. The system measures temperature, relative humidity, atmospheric pressure, ambient light intensity, and wind-related values. We sample sensor readings at fixed intervals and check them for accuracy before further processing.

The collected data is temporarily stored in memory and arranged into a structured feature vector suitable for running on the device.

C. Feature Preparation for Embedded Inference

Before running the model, we scale the raw sensor values using normalization parameters defined during model training. This step ensures that the real-time sensor data matches the trained machine learning model.

We apply categorical and temporal changes during training, which are included in the embedded implementation through fixed mappings. This allows the ESP32 to create feature representations that are the same as those used during offline training.

D. Embedded Model Integration

We convert the trained Logistic Regression model into embedded C++ code to run on the ESP32. The model parameters, such as weights and bias, are kept as constant values in the firmware. The inference logic calculates a linear combination of input features and then uses a sigmoid activation function to estimate the likelihood of rainfall.

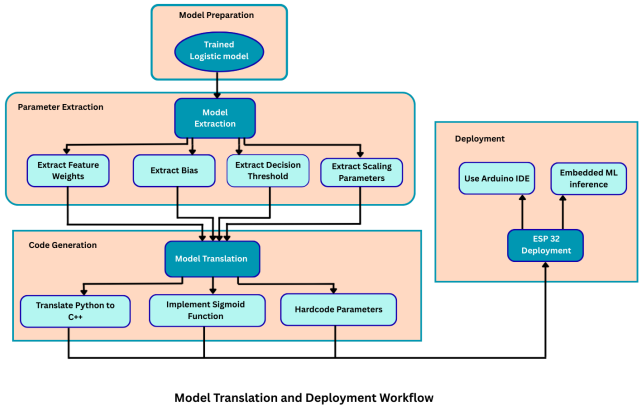

Fig. 4. Model translation and deployment workflow for embedded logistic regression inference on ESP32

Fig. 4 shows the workflow for turning the trained logistic regression model into embedded C++ code for use on the ESP32 platform. This process includes preparing the model, extracting the learned parameters, developing the code, and deploying it for on-device inference.

A predefined decision threshold is used to change the probability output into a binary rainfall prediction. The implementation is designed to reduce memory use and processing load.

E. On-Device Rainfall Prediction and Output

While in use, the ESP32 constantly collects sensor data, prepares features, and makes real-time rainfall predictions. The predicted rainfall status and probability appear on a 16×2 LCD module, which offers a simple user interface.

All prediction tasks happen entirely on the device. This setup ensures low latency, lower power consumption, and dependable performance in areas with little or no internet access.

VI. MACHINE LEARNING MODEL DEVELOPMENT

A. Model Selection

As our target output was binary, we needed a binary classification model that was strong and significantly lightweight at the same time. A number of classification techniques were investigated, like Logistic Regression, Random Forest Classification, Linear Support Vector Machine, Single Decision Tree, Tiny Neural Network, Gaussian and Bernoulli Naive Bayes, and Gradient Boosting.

The best accuracy was attained by the Random Forest Classifier, which was about 82 percent accurate. But because the model is based on a large number of decision trees, which makes it memory-intensive as well as a heavy computational process, it was not suitable for use on the ESP32. Among the others, all were suitable for deployment on an ESP32 except Gradient Boosting, but the performance of the model was not up to the level of the others.

Logistic Regression proved to be the most feasible option with a balance between accuracy and resource utilization. Finally, the accuracy of the Logistic Regression model was approximated to be around 79 percent with good and balanced precision and recall scores.

TABLE I
PERFORMANCE COMPARISON OF EVALUATED MACHINE LEARNING MODELS

Model	Accuracy	Precision	Recall	F-1
Logistic Regression	79	79	79	79
Decision Tree	76	76	76	76
Linear SVM	77	77	77	77
Tiny Neural Network	74	74	74	74
Gradient Boosting	76	76	76	76

Table I summarizes the performance of different machine learning models evaluated for rainfall prediction. Although the Random Forest Classifier achieved the highest accuracy, Logistic Regression was selected due to its lower computational complexity and suitability for embedded deployment.

B. Model Training and Evaluation

The selected Logistic Regression model was trained using the preprocessed dataset with an 80:20 train-test split approach. Model performance was evaluated using commonly used classification metrics, including accuracy, precision, recall, F1-score, and confusion matrix analysis.

These evaluation metrics indicate that the Logistic Regression model is capable of providing reliable rainfall predictions while maintaining a lightweight computational footprint, making it suitable for deployment on resource-constrained embedded systems such as the ESP32.

TABLE II
CLASS-WISE PERFORMANCE OF LOGISTIC REGRESSION

Class	Precision (%)	Recall (%)	F-1 (%)
0	0.79	0.79	0. 78
1	0.80	0.79	0.79

While training our model, we noticed that tuning the probability threshold of the model impacted its performance. To find the sweet spot for the threshold, we plotted a graph that shows the probability distribution for both classes. In the graph, some overlapping of the classes is seen in the middle region, which is because of ambiguous weather conditions, and is quite normal for real-world weather data. Errors like false positives and false negatives mostly occur here.

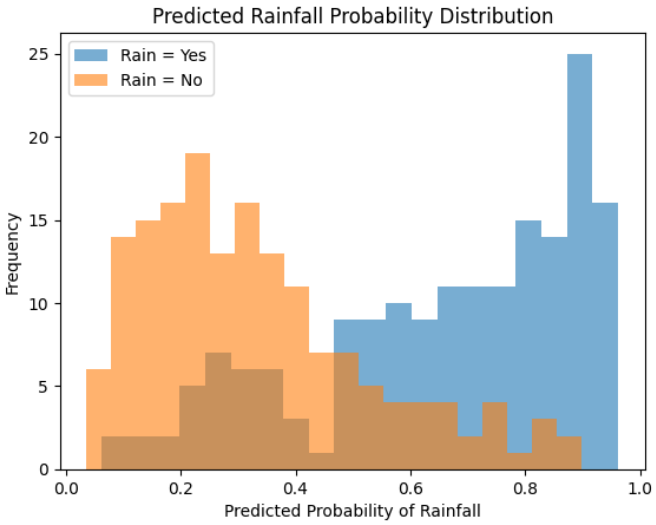

Fig. 5. Probability Distribution

From the Fig. 5, a clear separation of classes can be noticed at around 0.47 of the X-axis, which we used as our threshold to get the highest accuracy for our model.

C. Mathematical Formulation

Given an input feature vector

$$\mathbf{x} = [x_1, x_2, \ldots, x_n],$$

Logistic Regression computes a linear combination of features as

$$z = \sum_{i=1}^{n} w_i x_i + b,$$

where w_i represents the model weights and b is the bias term. The probability of rainfall occurrence is obtained using the sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

A threshold τ is applied to generate the final binary prediction:

$$\hat{y} = \begin{cases} 1, & \text{if } \sigma(z) \geq \tau, \\ 0, & \text{otherwise.} \end{cases}$$

Here, $\hat{y} = 1$ indicates rainfall occurrence.

VII. EMBEDDED IMPLEMENTATION

The trained Logistic Regression model was converted from Python to C++ manually, as the ESP32 does not support direct execution of Python code. Such conversion basically contains learned weight extraction, bias extraction, and the decision threshold extraction, along with the implementation of the sigmoid function.

Real sensor readings are scaled using the same parameters applied during training and fed to the embedded inference function. Due to cost issues, the team decided to use a potentiometer rather than an anemometer to input manual values for wind speed. The final rainfall prediction and probability are displayed on the LCD.

Completely offline, it minimizes latency, reduces power consumption, and does not rely on any external connectivity.

VIII. RESULTS AND DISCUSSION

This section presents the experimental results and performance analysis for the developed Pocket Weather Station system. The system works perfectly in real-time for the prediction of rainfall on the device without the need for internet access.

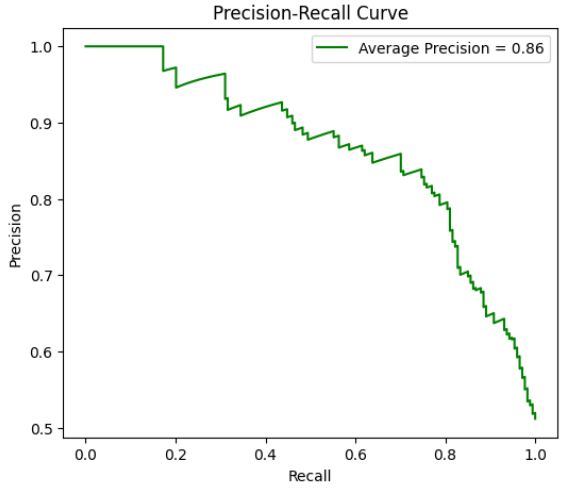

Fig. 6. Precision–Recall curve of the Logistic Regression model used for rainfall prediction.

Fig. 6 represents the Precision-Recall curve for the Logistic Regression model. The Precision-Recall curve shows that there is an optimal precision and recall for every index threshold.

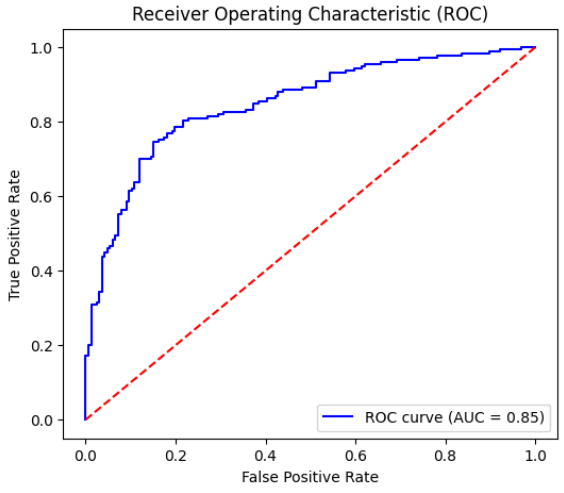

Fig. 7. Receiver Operating Characteristic (ROC) curve of the Logistic Regression model with an AUC of 0.85.

In Figure 7, the Receiver Operating Characteristic (ROC) Curve with an Area Under Curve (AUC) of 0.85 shows effective discriminative power of this model.

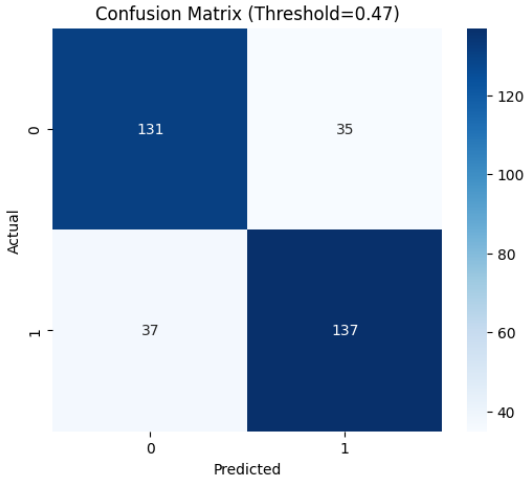

Fig. 8. Confusion matrix of the Logistic Regression model at a decision threshold of 0.47.

Fig. 8 illustrates the confusion matrix with a threshold value of 0.47. This indicates that the model has been equally efficient for the classification of true positives and true negatives, confirming the effectiveness of the model for embedded deployment.

Though it has slightly less accuracy in prediction compared to cloud or ensemble models, it has numerous advantages in terms of mobility, cost-effectiveness, and offline capabilities that make it ideal for smart agriculture and irrigation systems.

IX. LIMITATIONS AND FUTURE WORK

The present prototype employs a potentiometer instead of measuring the wind speed, thereby restricting the process of automation. The future improvements could be:

Integration of a low-cost anemometer.

Deployment of more advanced lightweight ML models.

Solar-powered operation.

Fully automatic irrigation management on a predicted rainfall basis.

Conducting field testing in agricultural settings involving farmers.

X. CONCLUSION

This paper discusses the design and creation of a portable pocket weather station. This is an affordable IoT solution that can make offline rainfall predictions using edge machine learning. The solution uses a Logistic Regression algorithm directly on the ESP32 microcontroller. This setup allows the system to function without cloud computing while still providing reliable predictions. By removing the requirement for a constant internet connection, the system is suitable for use in remote agricultural areas with limited resources.

Additionally, the device's compact design and low power usage allow for easy scaling and long-term operation in the field. The experimental results show that edge-based inference can work well on low-cost microcontrollers without a significant loss in accuracy. This solution can lay the groundwork for smarter, automated irrigation systems. It may help improve water management, lower operational costs, and boost crop productivity. Future research could aim to include more environmental factors, adaptive learning methods, and real-time action mechanisms to improve system performance and user experience.

REFERENCES

[1] S. Indhumathi, S. Aghalya, J. A. Smitha, and M. P. Aarthi, "IoT-Enabled Weather Monitoring and Rainfall Prediction Using Machine Learning Algorithm," in *Proc. Second Int. Conf. on Augmented Intelligence and Sustainable Systems (ICAISS)*, Trichy, India, Aug. 2023, pp. 1491–1495.

[2] D. V. Rayudu and J. F. Roseline, "Accurate Weather Forecasting for Rainfall Prediction Using Artificial Neural Network Compared with Deep Learning Neural Network," in *Proc. Int. Conf. on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF)*, Chennai, India, Jan. 2023, pp. 1–6.

[3] O. M. F. Ohemu, Z. Ohemu, A. J. Owolabi, and H. Ali, "Real-Time Weather Forecasting Using BME280 and STM32 Microcontroller," *Asian Journal of Computer Science and Technology*, vol. 14, no. 1, pp. 47–56, 2025.

[4] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, "MCUNet: Tiny Deep Learning on IoT Devices," *arXiv preprint arXiv:2007.10319*, July 2020.

[5] F. Al-Jame and M. Perera, "AI-Driven Smart Irrigation System Using Edge-Based Embedded Controllers," *Progress in Electronics and Communication Engineering*, vol. 3, no. 2, 2025.

[6] S. Jagtap, S. Yadav, S. Yewale, D. Dudhane, and P. Kumar, "Smart Weather Monitoring System Using ESP32 and IoT," *International Journal of Electrical and Communication Engineering Technology*, vol. 3, no. 2, 2025.

[7] P. Kapoor and F. A. Barbhuiya, "Cloud Based Weather Station Using IoT Devices," in *Proc. IEEE Region 10 Conf. (TENCON)*, Kochi, India, Oct. 2019, pp. 2357–2362,

[8] S. S. V, S. S. Kumar, G. Mohandas, K. Padmakumar, and M. Belwal, "Real-Time Weather Monitoring System," in *Proc. Int. Conf. on Advancements in Smart, Secure and Intelligent Computing (ASSIC)*, Bhubaneswar, India, 2025, pp. 1–7,

[9] J. J. Hilda, S. V. Jinny, P. M. Srilekha, and V. Selvakasi, "IoT-Based Smart Weather Monitoring System," in *Proc. Int. Conf. on Innovative Trends in Information Technology (ICITIIT)*, 2025, pp. 1–6,

[10] A. G. Salman, B. Kanigoro, and Y. Heryadi, "Weather Forecasting Using Deep Learning Techniques," in *Proc. Int. Conf. on Advanced Computer Science and Information Systems (ICACSIS)*, 2015, pp. 281–285,

[11] S. Singh, M. Kaushik, A. Gupta, and A. K. Malviya, "Weather Forecasting Using Machine Learning Techniques," in *Proc. 2nd Int. Conf. on Advanced Computing and Software Engineering (ICACSE)*, 2019.

[12] F. J. J. Joseph, "IoT-Based Weather Monitoring System for Effective Analytics," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 4, pp. 311–315, 2019.

[13] D. Mohapatra and B. Subudhi, "Development of a Cost-Effective IoT-Based Weather Monitoring System," *IEEE Consumer Electronics Magazine*, vol. 11, no. 5, pp. 81–86, 2022,

[14] B. S. Rao, K. S. Rao, and N. Ome, "Internet of Things (IoT)-Based Weather Monitoring System," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, no. 9, pp. 312–319, 2016.

[15] A. Admane, S. Shinde, T. Supe, and V. Tuwar, "IoT-Based Weather Monitoring System," *JournalNX*, pp. 352–357, 2017, Novateur Publication.

[16] N. Kumar, M. J. Haque, A. Mittal, A. Singh, A. Rayal, and S. Oli, "IoT-Based Smart Weather Monitoring System," in *Proc. 2023 Annual Int. Conf. on Emerging Research Areas: Int. Conf. on Intelligent Systems (AICERA/ICIS)*, 2023, pp. 1–6.

[17] A. S. Shawon, "Weather Data Bangladesh," Kaggle Dataset, 2021. [Online]. Available: www.kaggle.com/datasets/apurboshahidshawon/weatherdatabangladesh. Accessed: Oct. 24, 2025.