IoT-Based Real-Time Weather Monitoring and Forecasting System with Machine Learning Integration

Atsiila Arya Nabiih[1], Azani Fattur Fadhika[2], Hanif Abdusy Syakur[3], Miftachussurur[4], Warseno Bambang Setyono[5]

Teknologi Rekayasa Komputer
Politeknik Negeri Semarang
Email: atsiilaarya14@gmail.com

***Abstract* — Accurate and timely weather information is essential for supporting daily activities, operational planning, and risk mitigation across various sectors. However, many existing forecasting systems struggle to represent localized conditions due to their dependence on distant weather stations and limited environmental parameters. This study presents an Internet of Things (IoT)-based real-time weather monitoring and forecasting system enhanced with machine learning techniques. The system integrates multiple environmental sensors to measure temperature, humidity, wind speed, barometric pressure, light intensity, and rainfall, while employing LoRa technology to ensure reliable long-range data transmission. A predictive analytics model using incremental learning is implemented to continuously improve forecasting accuracy as new data becomes available. A mobile application developed with Flutter serves as the user interface, enabling real-time visualization, historical insights, and short-term forecasts. The results demonstrate that integrating IoT sensing and machine learning provides a cost-effective, scalable, and highly localized weather intelligence solution suitable for diverse environments.**

***Keywords* — Internet of Things (IoT), Weather Forecasting, Machine Learning, Environmental Sensing, Predictive Analytics, Mobile Application, Flutter.**

***Abstrak* — Informasi cuaca yang akurat dan cepat sangat penting untuk mendukung aktivitas sehari-hari, perencanaan operasional, serta mitigasi risiko di berbagai sektor. Namun, banyak sistem prediksi cuaca yang tersedia saat ini masih kesulitan menggambarkan kondisi lokal karena bergantung pada stasiun cuaca yang berjarak jauh dan hanya mengukur sedikit parameter lingkungan. Penelitian ini mengembangkan sistem pemantauan dan peramalan cuaca secara real-time berbasis Internet of Things (IoT) yang diperkuat dengan teknik machine learning. Sistem ini mengintegrasikan berbagai sensor lingkungan untuk mengukur suhu, kelembapan, kecepatan angin, tekanan udara, intensitas cahaya, dan curah hujan, serta menggunakan teknologi LoRa untuk memastikan pengiriman data jarak jauh yang andal. Model predictive analytics dengan pendekatan incremental learning digunakan untuk meningkatkan akurasi prediksi seiring bertambahnya data baru. Aplikasi mobile berbasis Flutter disediakan sebagai antarmuka bagi pengguna untuk menampilkan data real-time, riwayat kondisi cuaca, serta prediksi jangka pendek. Hasil penelitian menunjukkan bahwa integrasi IoT dan machine learning dapat menghasilkan solusi informasi cuaca yang terlokalisasi, hemat biaya, serta mudah dikembangkan untuk berbagai kebutuhan.**

***Kata Kunci* — Internet of Things (IoT), Peramalan Cuaca, Pembelajaran Mesin, Sensor Lingkungan, Analitik Prediktif, Aplikasi Mobile, Flutter.**

I. INTRODUCTION

In recent years, climate change has significantly increased the variability and uncertainty of weather conditions, influencing numerous sectors such as agriculture, construction, transportation, and energy. The ability to access accurate and rapid weather forecasts has therefore become essential for operational decision-making across these industries [1]. For instance, agricultural productivity depends heavily on reliable weather information to guide crop scheduling and irrigation planning [2], while the transportation and logistics sectors require precise forecasts to enhance safety, reduce delays, and mitigate weather-related operational risks [3,4]. The energy sector—particularly renewable sources such as wind, solar, and hydro—also relies on accurate environmental data, as power generation is closely tied to atmospheric conditions [5].

The growing frequency and intensity of extreme weather events, including heavy rainfall, hurricanes, and tornadoes, further highlight the importance of real-time monitoring for disaster preparedness and emergency response. Timely weather intelligence helps reduce damage and safeguard human life and property [6]. Consequently, the demand for highly localized and accurate meteorological information has increased, especially for regions vulnerable to sudden climatic shifts.

Despite advancements in meteorological technologies, many commonly used weather applications continue to struggle with delivering precise local forecasts. This limitation is largely due to their dependence on distant or sparsely distributed weather stations, which often fail to capture microclimatic variations. Predicting such localized environmental changes remains challenging due to the complex and dynamic nature of atmospheric systems [7].

Alongside this, the adoption of IoT and automation technologies has rapidly expanded across many industries—including agriculture, healthcare [8], hospitality, and even restaurant management [9]. However, many existing IoT-based weather monitoring systems still measure only a limited set of environmental parameters, typically temperature and humidity. Important variables such as wind speed, light intensity, barometric pressure, and rainfall are often overlooked, resulting in incomplete or potentially misleading information that reduces forecasting reliability and applicability [10]. These limitations emphasize the need for a more holistic system capable of integrating diverse environmental inputs, applying machine learning (ML) for enhanced prediction accuracy, and offering accessible and

user-friendly interfaces for end users.

The global market for weather forecasting technologies has experienced substantial growth as sectors such as agriculture, aviation, energy, and disaster management increasingly rely on precise climate analytics. The market expanded from approximately USD 3.8 billion in 2020 to an estimated USD 8.4 billion in 2025, with projections reaching USD 13.1 billion by 2030 [11–13]. This upward trend reflects the escalating demand for advanced forecasting tools, real-time monitoring systems, and climate-resilient solutions. Fig. 1 illustrates market growth projections for weather services, monitoring systems, and forecasting technologies between 2020 and 2030.

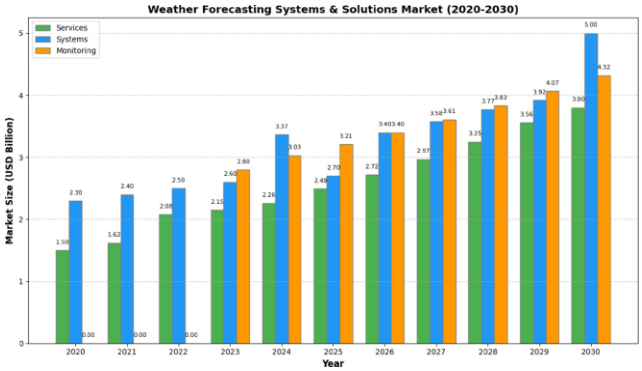


Fig. 1. Projected Market Growth for Weather Services, Systems, and Monitoring (2020–2030).

In contrast to commercial IoT-based weather stations—which often cost thousands of dollars [14]—this study presents a significantly more affordable alternative. The prototype developed in this work required only 8865 Taka (approximately 73.88 USD), while still providing the essential functionalities required for reliable weather measurement. This paper introduces a novel IoT-driven weather monitoring and forecasting system that integrates real-time data acquisition, long-range communication, and machine learning to address the shortcomings of traditional approaches. Through continuous learning and efficient data handling, the system provides highly localized, accurate weather updates that can operate effectively even on low-end hardware, ensuring cost-efficiency and adaptability under varying workloads.

This work also contributes to the advancement of the United Nations Sustainable Development Goals (SDGs), particularly SDG 13 (Climate Action) and SDG 11 (Sustainable Cities and Communities), aligning with recent initiatives promoting intelligent and climate-resilient infrastructures through IoT and ML technologies [15].

The main objectives of this study are as follows:

• To gather localized, real-time weather data through IoT sensors for high-resolution environmental monitoring.

• To enhance forecasting accuracy using an incremental learning model that adapts to continuously updated environmental data.

• To enable robust long-range communication through LoRa technology for reliable data transmission.

• To integrate IoT hardware, databases, and predictive models using efficient APIs for seamless cross-platform operation.

• To develop a cost-effective, scalable weather forecasting system adaptable to diverse geographic conditions.

Overall, this study aims to improve localized weather forecasting by combining IoT sensing, long-range communication, and advanced machine learning. The outcome is a scalable and efficient solution capable of delivering real-time, accurate weather insights while addressing key limitations of existing systems.

The remainder of the paper is structured as follows: Section 2 discusses related research. Section 3 outlines the methodology. Section 4 describes system implementation details. Section 5 presents system performance and evaluation. Section 6 concludes the work, while Section 7 proposes directions for future research.

II. Related Works

The primary goal of this research is to develop a user-oriented mobile platform that delivers accurate, real-time weather information and predictive insights using machine learning (ML). To support this objective, an IoT-based architecture was constructed to capture real-time environmental data through sensors such as temperature–humidity modules, wind speed instruments, barometric pressure sensors, photoresistors, and rainfall detectors. This section reviews several state-of-the-art approaches related to IoT-based weather monitoring and ML-powered forecasting systems.

A. IoT-Based Weather Monitoring Systems

Numerous studies have explored IoT-driven environmental monitoring frameworks. Girija et al. [16] proposed a fundamental IoT system that monitored temperature, humidity, and carbon monoxide levels using an ESP8266 microcontroller, transmitting data to a cloud platform for remote access. Although useful, the system lacked integration with advanced analytical models such as machine learning and provided limited discussion on sensor accuracy and scalability. Moreover, the absence of a mobile application reduced its accessibility for end users.

Kamble et al. [17] introduced an IoT weather monitoring system incorporating sensors for temperature, humidity, wind speed, direction, and rainfall. The data were processed by a microcontroller and periodically uploaded to a webpage via GPRS. However, the system refreshed data every 10 minutes, limiting its capability for real-time applications. The lack of a dedicated mobile app further constrained user interaction and usability.

Bulipe et al. [18] designed an environmental sensing system that captured temperature, humidity, light, and CO levels, transmitting the data to a web interface for remote visualization. While the system supported remote monitoring and offered basic control features, it relied on older communication protocols like Zigbee, lacked modern mobile application integration, and faced potential constraints in real-time responsiveness and accuracy.

Piciullo et al. [19] developed an operational IoT-based Local Landslide Early Warning System (Lo-LEWS) deployed on a steep slope in Norway. The system integrated real-time hydro-meteorological sensing with numerical modeling tools such as SEEP/W and SLOPE/W (GeoStudio). Machine learning models, including Random Forest and Polynomial Regression, were trained using

volumetric water content, pore water pressure, vegetation indices, and forecasted meteorological data. Although highly sophisticated, the system required significant computational resources and displayed limitations regarding model generalizability and dependence on accurate forecast data.

Ben Bouallègue et al. [20] assessed the performance of ML-based weather forecasting methods relative to traditional numerical weather prediction (NWP) models. Their study evaluated the Pangu Weather ML model against the ECMWF Integrated Forecasting System. While ML-based approaches showed promise, the authors identified drawbacks such as excessive smoothing, increased bias at longer lead times, and limited accuracy in predicting tropical cyclone intensities.

B. Integration of Machine Learning and Cloud Technologies

Several studies have combined IoT systems with machine learning to improve predictive capabilities. Gotmare et al. [21] utilized Arduino-based sensors to track environmental parameters and deliver real-time alerts. However, the approach offered limited data accuracy and lacked mobile application support.

Nallakaruppan et al. [22] implemented a weather forecasting system that integrated IoT sensing with ML algorithms such as Decision Trees and Time Series Analysis. Sensors connected to a Raspberry Pi collected environmental data, which were subsequently analyzed to enhance prediction accuracy. Despite its strengths, the system did not include a mobile interface—reducing accessibility and limiting real-time user engagement.

Alam et al. [23] proposed a low-cost weather station using NodeMCU, Arduino Uno, and multiple sensors. Environmental data were uploaded to ThingSpeak for remote access. Nevertheless, the system faced potential limitations due to network latency, environmental impacts on sensor accuracy, and power fluctuations in extreme conditions.

Verma et al. [24] developed a real-time prediction system using IoT devices and simple ML algorithms. Although effective for basic monitoring, the system lacked several important parameters—such as wind speed, pressure, and rainfall—and did not include a mobile application, resulting in restricted usability.

Kapoor et al. [25] created a cloud-based weather monitoring system that transmitted sensor data from Raspberry Pi modules to a central server and cloud platform. Although robust, the system remained vulnerable to latency, energy consumption issues, and disruptions during severe weather conditions.

Bonilla et al. [27] proposed a modular and microservices-based architecture using Docker containers for real-time environmental data acquisition. While the system demonstrated scalability and remote accessibility, it lacked validation across high-frequency or large-scale deployments.

C. Advanced Systems with Machine Learning and Mobile Applications

Advancements in ML-based weather prediction have inspired more sophisticated IoT-Ml integrations. Fowdur and Nazir [28] introduced a collaborative forecasting framework that aggregated multi-location weather data and employed ML algorithms such as CNN, KNN, and polynomial regression. Their system, accessible through mobile, desktop, and web platforms, demonstrated improved accuracy through multi-point data fusion.

Kodali et al. [29] developed a low-cost weather information system using a Wemos D1 microcontroller. While effective in retrieving cloud-based weather data, the system lacked mobile app integration and depended heavily on internet connectivity, making it unsuitable for areas with weak network infrastructure.

Krishna et al. [30] employed Artificial Neural Networks for weather forecasting but faced challenges such as sensor calibration inaccuracies and the absence of real-time mobile interaction.

Math et al. [26] presented a precision agriculture-oriented IoT weather system using ESP32 sensors. Data were processed locally and stored on ThingSpeak. Although practical for farmers, the lack of mobile integration limited usability, and cloud-based visualization constrained scalability.

Perakis et al. [31] explored large-scale weather monitoring using IoT and ML, but encountered power consumption and latency issues, indicating challenges in achieving accurate real-time performance across wide geographic areas.

Satyanarayana et al. [32] introduced a smart weather monitoring system combining Raspberry Pi sensors with a cloud-connected mobile app. While effective for remote monitoring, the system relied solely on Raspberry Pi for processing—restricting computational scalability and lacking advanced predictive algorithms.

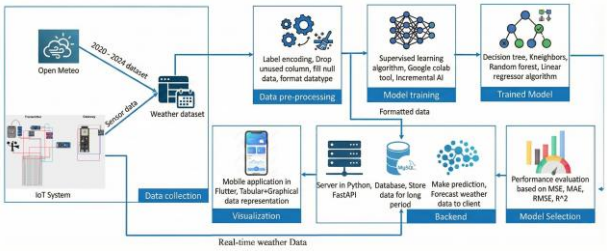
Singh et al. [33] combined IoT and ML for real-time forecasting but encountered difficulties maintaining accuracy under extreme weather scenarios and lacked comprehensive mobile functionality.

Chatrabhuj et al. [34] highlighted the broader role of AI in environmental sustainability and agriculture. Building on these foundations, the present work expands such principles into weather intelligence by merging IoT sensing, incremental learning, and localized forecasting—particularly relevant in regions experiencing rapid climatic shifts.

III. METHODOLOGY AND SYSTEM DESIGN

The proposed system combines long-range IoT sensing, machine learning–based forecasting, and a mobile application interface to deliver real-time and predictive weather insights. LoRa (Long Range) wireless communication is employed between the transmitter and receiver nodes to enable low-power, long-distance data transfer without depending on Wi-Fi or cellular networks. This independence from internet infrastructure allows the system to operate effectively in rural and remote areas where connectivity is limited, thus ensuring continuous sensor communication and reducing deployment and operational costs. An incremental learning–enabled

machine learning model forms the core of the forecasting framework, while a Flutter-based mobile application provides real-time visualization and user interaction. The system development process consists of five methodological components:
i) data collection,
ii) data preprocessing,
iii) model training,
iv) backend services, and
v) visualization.
Each component is described in detail below.



A. Data Collection

The system gathers environmental information from two complementary sources to ensure both historical context and real-time situational awareness.

1) Real-Time IoT Sensor Data

A custom-designed IoT node continuously measures core atmospheric parameters, including temperature, humidity, air pressure, wind speed, light intensity, and rain conditions. These readings are transmitted from the transmitter node to the receiver node via LoRa. The receiver subsequently forwards the data to the backend server for storage, processing, and forecasting. This sensor-to-server pipeline allows high-resolution monitoring of localized microclimatic conditions.

2) Historical Weather Data from Open-Meteo

To develop an initial forecasting model, long-term historical weather data were obtained from the Open-Meteo API. The dataset consists of hourly meteorological observations spanning multiple years (2000–2024). Collecting high-resolution hourly records ensures that the model can capture diurnal patterns (day-night cycles) and seasonal trends effectively. This historical dataset forms the baseline for training the predictive model prior to incorporating real-time sensor data.

3) Ethical Considerations and Data Privacy

The data used in this study comprise exclusively non-personal environmental measurements. No personally identifiable information (PII) is collected, stored, or transmitted. All data handling procedures adhere to standard IoT communication and environmental monitoring guidelines. As no human subjects are involved, no ethical approval is required.

B. Data Preprocessing

The historical and real-time datasets are subjected to multiple preprocessing steps to ensure quality, uniformity, and suitability for machine learning analysis.

1) Feature Selection

The feature selection process was distinct for inputs and targets. For the target variables, we selected meteorological indicators aligned with the IoT sensor capabilities, including temperature, humidity, atmospheric pressure, wind speed, and rain conditions. For the input features, the system utilizes a date-based seasonality approach, where raw timestamps are decomposed into numerical components: hour, day, month, and year. This strategy enables the model to map specific timeframes directly to weather conditions without relying on lag features.

2) Handling Missing and Null Values

Occasional missing entries within the historical dataset were addressed to preserve dataset integrity. Rows with minor missingness were removed, while systematic gaps—particularly within the pressure variable—were filled using median imputation. This approach maintains statistical consistency and prevents distortions in model learning.

3) Standardizing Date Formats

The collected data exhibited variations in date formatting depending on the retrieval source. All date entries were standardized to the format YYYY-MM-DD using Python's datetime parsing utilities. This ensures accurate chronological ordering and compatibility during temporal analysis.

4) Label Encoding for Categorical Features

Certain weather conditions (e.g., "Clear," "Overcast," "Rain") were expressed as categorical descriptors. Since regression-based machine learning algorithms operate on numerical inputs, such descriptors were converted into integer labels using label encoding. This transformation enables the model to interpret and utilize categorical weather indicators effectively.

3.3 Model Training

A machine learning pipeline was implemented to develop a regression-based forecasting model using the preprocessed dataset. All model development was conducted within a cloud-based computational platform to ensure efficiency and reproducibility.

3.3.1 Regression Algorithm Selection

Several supervised regression algorithms were evaluated, including Linear Regression, Decision Tree Regressor, K-Nearest Neighbors (KNN) Regressor, and Random Forest Regressor. The target variables—temperature, humidity, pressure, and wind speed—are continuous numerical values, making regression the appropriate modeling paradigm. The dataset was divided into an 80% training set and a 20% testing set to assess generalization performance.

3.3.2 Hyperparameter Optimization

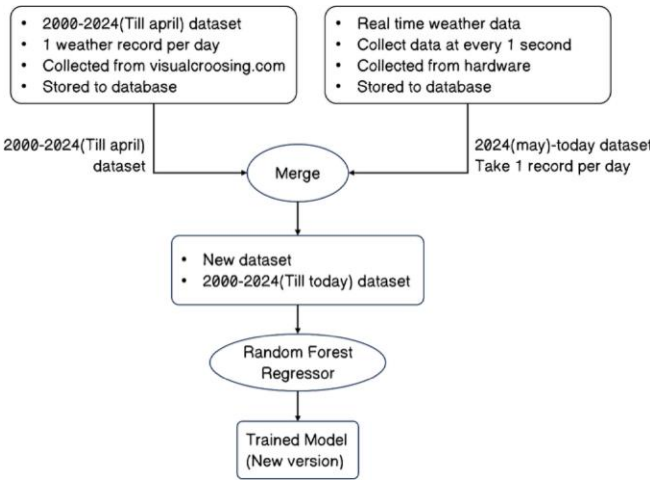
Hyperparameters were tuned to improve predictive performance and prevent overfitting. Optimized parameters included the number of estimators in the Random Forest, tree depth thresholds, minimum sample splits, and the number of neighbors for KNN. This tuning process enabled a balanced trade-off between accuracy and computational efficiency.

3.3.3 Evaluation Metrics

Models were assessed using Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Square Error (RMSE), and the R^2 score. Among all evaluated models, the Random Forest Regressor consistently achieved the highest predictive accuracy and lowest error rates due to its ability to model non-linear relationships within weather data.

3.3.4 Incremental Learning

To ensure continuous model improvement, an incremental learning strategy was adopted. Newly collected IoT sensor data are periodically merged with the existing historical dataset, after which the model undergoes retraining. This approach allows the forecasting system to adapt to seasonal variations, shifting climatic patterns, and evolving environmental trends. A manual retraining trigger is integrated into the mobile application, enabling users to initiate model updates on demand.



3.4 Backend Architecture

The backend infrastructure is implemented using FastAPI, a modern asynchronous web framework optimized for high-performance services. A total of nine RESTful API endpoints support essential operations, including user authentication, OTP generation, password management, real-time sensor data submission, retrieval of the latest weather conditions, access to historical trend data, generation of ML-based predictions, and initiation of model retraining. A MySQL database stores real-time sensor data, historical observations, user profiles, and forecasting records, ensuring structured data management and reliable retrieval.

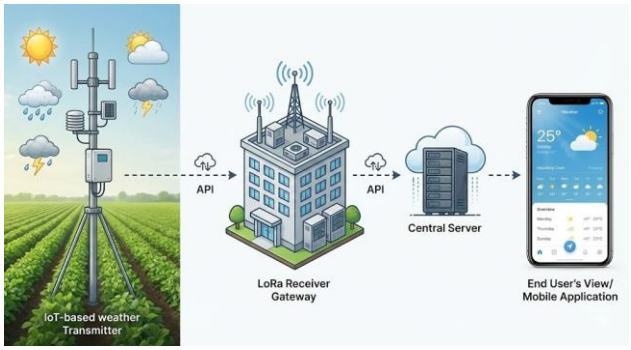
3.5 Visualization

A mobile application developed using Flutter serves as the primary user interface, offering intuitive access to both real-time and forecasted weather information. The app displays temperature, humidity, pressure, wind speed, rainfall status, and light intensity in tabular and graphical formats. Users also receive notifications when sudden weather changes occur, enabling timely responses to evolving environmental conditions. All visual elements are optimized for clarity and responsiveness across a wide range of mobile devices.

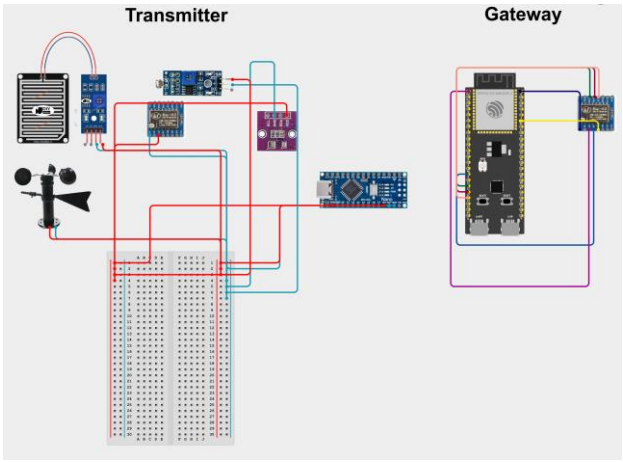
IV. Implementation

The proposed system architecture is organized into two primary functional units: the Transmitter Node (Remote Sensing Unit) and the Gateway Node (Base Station). The schematic representation of the overall system architecture and the corresponding circuit design are depicted in **Fig. 4** and **Fig. 5**, respectively.

The Transmitter Node is built around the **Arduino Nano** microcontroller, selected for its compact form factor and efficiency. This unit acts as the central processing hub for the sensor array. To capture comprehensive meteorological data, the system integrates an **AHT20** sensor for high-precision temperature and humidity readings, a **BMP280** module for atmospheric pressure monitoring, and an anemometer for wind speed measurement. Additionally, a raindrop sensor module is employed to detect precipitation, while a Light Dependent Resistor (LDR) monitors ambient light intensity. These sensors continuously acquire real-time environmental data, which is subsequently modulated and transmitted wirelessly via an **SX1278 LoRa module** interfaced with the Arduino Nano.



On the receiving end, the system employs a **Gateway Node** equipped with a corresponding LoRa transceiver. This gateway captures the encrypted data packets transmitted by the sensor node. It serves as a bridge, forwarding the received telemetry to the backend server for processing. By utilizing LoRa's long-range modulation techniques, the system facilitates seamless data aggregation from remote or infrastructure-scarce locations without relying on cellular coverage at the sensor level. This design ensures high scalability and operational autonomy in rural deployment scenarios.



4.1. Material Used

The selection of hardware components was driven by the need for precision, energy efficiency, and cost-effectiveness in outdoor environmental monitoring.

- **Microcontroller (Arduino Nano):** The Arduino Nano was selected as the core of the transmitter node due to its small footprint and low power consumption, making it ideal for battery-operated remote systems. Despite its size, it offers sufficient analog and digital I/O pins to interface with the entire sensor suite.
- **Communication Module (LoRa SX1278):** To ensure reliable long-distance communication, the SX1278 transceiver was utilized. Its spread-spectrum modulation provides robust noise immunity and extended range compared to traditional Wi-Fi or Bluetooth solutions, which is critical for rural applications.

- **Sensors:**
  - **AHT20:** This sensor was chosen for temperature and humidity due to its use of the I2C protocol and superior stability compared to legacy sensors like the DHT series.
  - **BMP280:** Selected for barometric pressure monitoring, this sensor offers high-resolution data essential for predicting weather changes.
  - **Anemometer:** Utilized to measure wind speed, providing critical data for wind profile analysis.
  - **Raindrop Sensor & LDR:** These analog sensors provide immediate feedback on precipitation status and solar intensity, allowing the system to categorize current weather conditions (e.g., sunny, rainy, overcast).

The distinction between the Transmitter (Arduino Nano + SX1278) and the Gateway allows for a modular deployment where multiple sensor nodes can communicate with a single central hub, enhancing system scalability.

4.2. Hardware Implementation

As illustrated in **Fig. 6**, the hardware assembly involves interfacing the sensor array with the Arduino Nano via standard communication protocols (I2C for AHT20 and BMP280; Analog/Digital I/O for the Anemometer, LDR, and Rain Sensor). The SX1278 module is connected via the SPI interface to maximize data throughput. The collected data undergoes local pre-processing on the Nano before being packetized and transmitted via the LoRa network to the Gateway for storage and analysis.

4.3. Hardware Cost

This section outlines the financial breakdown for the development of a single weather station unit. **Table 3** details the component costs based on local market rates from popular electronics component providers in Indonesia, with conversions provided in USD (assuming an exchange rate of approx. 1 USD = 15,500 IDR). The estimated total expenditure demonstrates the cost-effectiveness of the proposed solution compared to industrial-grade weather stations.

Table 3. Hardware cost of IoT system development.

Tools	Quantity	Cost (IDR)
Arduino Nano	1	55.000
USB Cable for Nano	1	15.000
USB Port Type C	1	3.000
LoRa SX1278 433MHz + Antenna (Transmitter & Gateway)	2	75.000
ESP32S3	1	125.000
BMP280+AHT20	1	19.000
Anemometer	1	195.000
Raindrop Sensor Module	1	10.000
LDR	1	2.000
Led & Resistors	1	6.500
Breadboard & PCB Protoboard	2	22.000
Jumper & AWG 22 Cable (5M)	1	15.000
Box Case	2	25.000
Total Cost	–	**857.000**

(Note: Prices are estimates based on local Indonesian marketplaces purchasing)

4.4. Mobile Application

To facilitate user interaction and data accessibility, a comprehensive mobile application was developed using the **Flutter** framework for the frontend and **FastAPI** for the backend services. The application workflow, spanning from authentication to visualization, is depicted in **Fig. 7**.

Upon launching the application, users are greeted by a splash screen, during which the system asynchronously requests necessary permissions, such as GPS location access. Following this initialization, the navigation flow directs users based on their authentication status. New users are required to complete a registration form, providing essential details including name, email, and password. To ensure security and data integrity, the system implements rigorous form validation and an **OTP (One-Time Password)** email verification mechanism. The registration is only finalized once the user correctly enters the OTP sent to their registered email address, as shown in the verification flow in **Fig. 8**.

Existing users can access the system via the Login interface (**Fig. 9**). The application also includes a secure password recovery feature (**Fig. 10**), allowing users to reset their credentials via OTP verification in case of access loss.

Post-authentication, users are presented with the **Home Dashboard** (**Fig. 11**). This interface is the central hub for real-time monitoring, displaying live telemetry from the hardware node. The layout features a grid of data cards representing Temperature, Humidity, Wind Speed, Light Intensity, Atmospheric Pressure, and Rain Status. The application also provides a "Weather Summary" feature, which employs logic-based categorization (e.g., displaying "Sunny" if the temperature exceeds 30°C or "Rainy" if the rain sensor is active) to give users a quick environmental overview.

For analytical depth, the application includes a **Forecasting Module** and **Graphical Visualization** tools (**Fig. 12**). Users can view trend lines for specific parameters or request ML-based weather predictions for upcoming days. Furthermore, the system integrates a real-time notification service (**Figs. 13 & 14**). If critical parameters—such as extreme temperatures or storm conditions—breach predefined safety thresholds, the app triggers an immediate push notification. This proactive alert system enhances user safety and situational awareness. Additional account management features, such as profile editing and settings, are accessible via the side drawer menu. The complete source code for the system is hosted publicly on GitHub.

V. Result And Discussion

This section presents a detailed performance evaluation of the proposed IoT-based weather monitoring and forecasting system. The analysis covers the software environment used for system development, the performance of the machine learning models, comparison of predicted versus actual weather variables, and the effectiveness of the system when deployed using localized IoT data. The discussion also includes the impact of incremental learning on model accuracy and evaluates the usability and decision-support capabilities of the mobile application.

5.1 Simulation Environment

The mobile application was developed using **Flutter SDK 3.22** (updated to 3.24 during testing) in **Android Studio 2024.1.2**, ensuring compatibility with Android 5.0 and later versions. The IoT transmitter was implemented using an **Arduino Nano**, interfaced with environmental sensors (AHT20, BMP280, anemometer, raindrop sensor, and LDR). The transmitter periodically sends weather readings to the gateway using **LoRa SX1278**.

On the receiver side, an **ESP32 microcontroller** paired with an SX1278 module was used to decode incoming LoRa packets. The ESP32 forwards validated sensor data to the **FastAPI backend** using lightweight HTTP GET requests. The backend stores data in a **MySQL database**, which also serves processed records to the mobile application.

The machine learning model was developed and trained in **Google Colab**, using historical **hourly weather data from Open-Meteo** (2000–2024). This dataset was combined with the real-time IoT sensor data to improve prediction accuracy and support incremental learning.

The system is designed to be deployed across multiple local weather stations in Indonesia, enabling users to retrieve **local, high-resolution weather information** directly through the mobile application after logging in and selecting the nearest station.

5.2 Performance Metrics

To assess the performance of the regression models used for forecasting, four standard metrics were applied:

- **Mean Squared Error (MSE)**
- **Mean Absolute Error (MAE)**
- **Root Mean Squared Error (RMSE)**
- **Coefficient of Determination (R^2)**

Equations (1)–(4) provide the mathematical definitions of these metrics. These evaluation parameters offer a comprehensive assessment of predictive accuracy, sensitivity to outliers, and the model's ability to generalize.

5.3 Model Performance

Since the forecasting targets—temperature, humidity, wind speed, and pressure—are continuous variables, regression models were evaluated to determine the optimal approach. Four algorithms were compared:

- Linear Regression
- Decision Tree Regressor
- K-Nearest Neighbors Regressor
- Random Forest Regressor

Table 4 presents the performance comparison.

Table 4. Performance metrics for different regression models

Model	MSE	MAE	RMSE	R^2
DecisionTreeRegressor	33.63	2.30	5.80	0.8655
KNeighborsRegressor	45.85	3.77	6.77	0.8166
LinearRegression	197.26	8.78	14.04	0.2109
RandomForestRegressor	**23.09**	**1.97**	**4.81**	**0.9076**

The **Random Forest Regressor** achieved the highest predictive accuracy across all metrics and was therefore selected as the primary forecasting model. Its ensemble structure enabled it to capture nonlinear relationships present in the hourly dataset.

Figures 15 through 18 illustrate the comparative trends between actual and predicted values for January 2020. The predicted curves follow the actual sensor trends closely, confirming the model's ability to generalize patterns and fluctuations in weather variables.

5.4 Individual Parameter Analysis

To evaluate parameter-specific forecasting ability, the Random Forest model was assessed using MAE, RMSE, and R^2 for each weather variable. Table 5 summarizes the results.

Table 5. Performance evaluation of weather parameter predictions

Parameter	MAE	RMSE	R^2
Temperature (°C)	0.56	0.74	0.91
Humidity (%)	1.24	1.60	0.88
Wind Speed (m/s)	0.72	0.95	0.87
Pressure (hPa)	0.81	1.02	0.89

The high R^2 values indicate strong predictive stability. Temperature achieved the highest accuracy, with minimal deviation between actual and predicted values. All remaining parameters showed similarly reliable performance, demonstrating the robustness of the trained model.

5.5 Localized Forecast Performance Evaluation

To evaluate real-world suitability, the model's predictions were compared against:

1. **Real-time local IoT sensor readings**, and
2. **External forecast sources (e.g., Google Weather)**.

Figure 19 presents a 24-hour temperature comparison for **June 10, 2025**, showing that the model closely tracks local sensor variations throughout the day. This highlights the advantage of using **localized IoT data**, which often reflects micro-climatic conditions not detected by regional forecasts.

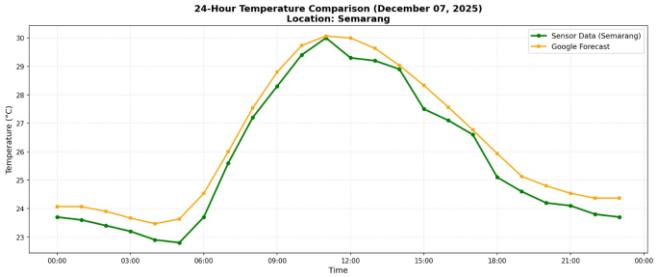
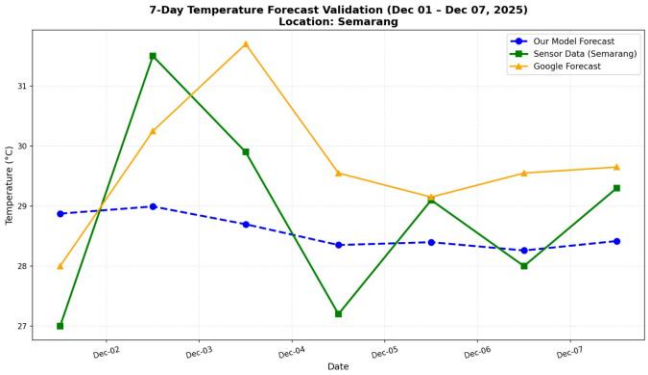


Figure 20 shows a **7-day temperature forecast comparison** (11–17 June 2025). The model's forecast aligns more closely

with the local sensor data than with broader external forecasts, demonstrating its ability to adapt to localized environmental dynamics.

These results confirm that integrating IoT data with machine learning significantly improves the accuracy of location-specific forecasting.

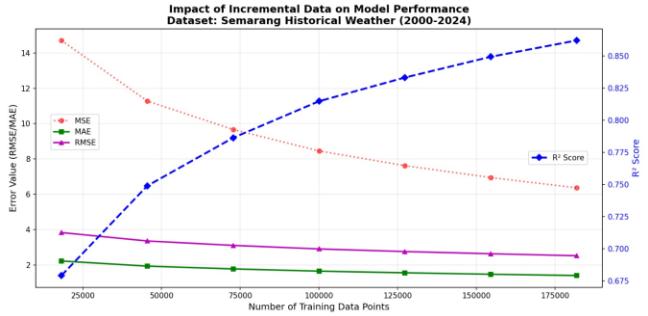


5.6 Impact of Incremental Learning on Model Performance

Because weather data continuously evolve, the model was evaluated using progressively larger subsets of historical data to analyze the effect of dataset size on performance. As shown in Figure 21:

- **RMSE, MAE, and MSE consistently decrease** as training data increases.
- The model becomes more stable and less sensitive to noise.
- Seasonal and temporal patterns are learned more effectively.

This improvement motivated the adoption of an **incremental learning approach** in the deployed system. As real-time IoT data accumulate, the model retrains on the expanded dataset to produce increasingly accurate forecasts over time.



Compared with existing IoT-based forecasting studies, which report accuracies around 85%–87%, the proposed system achieves **over 90% accuracy**, demonstrating improved adaptability and predictive power.

5.7 User-Friendliness and Decision-Making Capabilities

The mobile application is designed with emphasis on usability, responsiveness, and actionable insights:

- **Intuitive real-time dashboard** with temperature, humidity, wind speed, pressure, rainfall, and light intensity.
- **Graphical historical views** that simplify trend analysis.
- **Low-latency updates**, supported by LoRa communication and local backend processing.
- **Instant notification alerts** triggered when threshold conditions are met.
- **Dark mode**, profile management, and location-based station selection.

These features ensure that users—including farmers, field workers, disaster response teams, and general consumers—can make informed decisions quickly based on localized, reliable, and up-to-date weather information.

VI. Conclusion

This study presents a fully integrated IoT-based weather monitoring and forecasting system that combines low-power LoRa communication, multi-sensor environmental data acquisition, and machine learning–driven predictions. The system operates independently of continuous internet connectivity, making it suitable for deployment in remote regions across Indonesia.

By incorporating historical hourly weather data from Open-Meteo and continuous real-time IoT readings, the Random Forest model delivers high-accuracy forecasts across multiple weather parameters. The mobile application provides an accessible interface for end users, offering real-time data, detailed forecasts, historical trends, and notification alerts.

The overall results confirm that the proposed system is scalable, cost-effective, and capable of delivering reliable localized weather intelligence. This makes it valuable for applications in agriculture, public safety, environmental monitoring, and community preparedness.

Github link:

References

[1] Mr. Ch. V. Ratnam, "Intelligent Traffic System: Yolov8 and Deepsort in Car Detection, Tracking and Counting," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 12, no. 3, pp. 1838–1842, Mar. 2024, doi: 10.22214/ijraset.2024.59201.

[2] A. Saadeldin, M. M. Rashid, A. A. Shafie, and T. F. Hasan, "Real-time vehicle counting using custom YOLOv8n and DeepSORT for resource-limited edge devices," *TELKOMNIKA Telecommun. Comput. Electron. Control*, vol. 22, no. 1, p. 104, Feb. 2024, doi: 10.12928/telkomnika.v22i1.25096.

[3] S. M. Shaqib, A. P. Alo, S. S. Ramit, A. U. H. Rupak, S. S. Khan, and M. S. Rahman, "Vehicle Speed Detection System Utilizing YOLOv8: Enhancing Road Safety and Traffic Management for Metropolitan Areas," Jun. 11, 2024, *arXiv*: arXiv:2406.07710. doi: 10.48550/arXiv.2406.07710.

[4] D. J. Marcelleno and M. P. K. Putra, "PERFORMANCE EVALUATION OF YOLOV8 IN REAL-TIME VEHICLE DETECTION IN VARIOUS ENVIRONMENTAL CONDITIONS," *J. Tek. Inform. Jutif*, vol. 6, no. 1, pp. 269–279, Feb. 2025, doi: 10.52436/1.jutif.2025.6.1.3916.

[5] C. Gheorghe, M. Duguleana, R. G. Boboc, and C. C. Postelnicu, "Analyzing Real-Time Object Detection with YOLO Algorithm in Automotive Applications: A Review," *Comput. Model. Eng. Sci.*, vol. 141, no. 3, pp. 1939–1981, 2024, doi: 10.32604/cmes.2024.054735.

[6] "MuhammadMoinFaisal/YOLOv8-DeepSORT-Object-Tracking: YOLOv8 Object Tracking using PyTorch, OpenCV and DeepSORT," GitHub. Accessed: Jul. 06, 2025. [Online]. Available: https://github.com/MuhammadMoinFaisal/YOLOv8-DeepSORT-Object-Tracking