**A REPORT ON**

**IoT-Based Weather Monitoring System**

BY

**CSE III**

**Group 6**

**Khushi Hooda (230565)**

**Aman Jain (230572)**

**Jhalak Kapila (230576)**

**Shubham Bhatt (230581)**

**Harsh Singh Rautela (230583)**

MENTORED BY

**Dr. Sandeep Kumar** Assistant Professor  
 *School of Engineering and Technology* BML Munjal University, Gurugram

Prepared in Partial Fulfilment of  
 **IOT Sensors, Peripherals and Processors (ECE3009) Course**



**BML MUNJAL UNIVERSITY**

**November 2025**

# **ABSTRACT**

An IoT-based weather monitoring system has been designed and implemented to continuously observe key atmospheric parameters including temperature, humidity, atmospheric pressure, and dew point. The system has been developed with the intention of providing a low-cost and easily deployable alternative to conventional meteorological stations, which often remain limited by high installation expenses, restricted coverage, and the absence of cloud-based accessibility. In the proposed setup, an ESP32 microcontroller has been used as the central processing unit. The DHT11 and BMP180 sensors have been interfaced for precise measurement of the selected environmental variables, while the ESP32’s in-built Wi-Fi capability has been utilised for seamless data transmission to the Blynk IoT platform. An LCD module has been included to provide immediate on-site visibility of live readings.

The functioning of the system has been structured around data sensing, local processing, wireless communication, cloud visualisation, and alert generation. Real-time readings are shown on the LCD and simultaneously uploaded to the Blynk application, where users can access dashboards, historical logs, and parameter trends. Threshold-based notifications have also been enabled to provide timely alerts when values cross predefined safety limits. Experimental observations have revealed stable data acquisition, reliable network performance, and consistent cloud-level visualisation during testing.

The system’s modular design, low power requirement, and cloud integration make it suitable for use in precision agriculture, environmental studies, smart homes, and early-warning systems. Continuous monitoring and remote accessibility are supported, allowing the solution to operate as an efficient and scalable platform for environmental assessment.

# **INTRODUCTION**

Climate variability, extreme weather fluctuations, and the increasing frequency of climate-related hazards have emphasized the need for robust, scalable, and real-time environmental monitoring systems. Traditional meteorological infrastructures, while precise, are costly, limited in coverage, and not suitable for granular microclimate observation across diverse geographies. As modern society increasingly depends on data-driven systems for agriculture, transportation, disaster prediction, and urban planning, conventional weather stations fail to provide the continuous, high-resolution, and remotely accessible data required for timely decision-making. The growing energy demand, rising greenhouse gas emissions, and rapid urban expansion have further accelerated the need for developing intelligent weather monitoring systems capable of operating autonomously and at low cost.

In recent years, the Internet of Things (IoT) has emerged as a transformative technology in environmental sensing. IoT enables distributed, energy-efficient sensor networks that continuously collect atmospheric parameters such as temperature, humidity, pressure, rainfall, wind speed, and air quality. These systems leverage wireless microcontrollers, cloud platforms, and mobile interfaces to offer real-time dashboards accessible from anywhere in the world. Furthermore, the integration of Artificial Intelligence (AI), machine learning (ML), and edge computing has enabled the evolution from simple data collection systems to intelligent platforms capable of forecasting, anomaly detection, and automated alerting. As a result, IoT-based weather monitoring has become a crucial component of smart agriculture, early-warning systems, climate research, and smart-city development.

Several research studies demonstrate significant advancements in IoT-based environmental monitoring. Stoyanov et al. [1] proposed a low-cost DIY automatic weather station using IoT sensors integrated with the ThingSpeak cloud for real-time temperature, humidity, and pressure reporting. Although effective for live tracking, the system lacked predictive analytics and advanced alert mechanisms required for risk-sensitive applications.

Osama et al. [2] introduced an AI-based IoT framework employing RNN-LSTM models for predicting humidity and temperature variations, achieving high forecasting accuracy. However, the system required substantial computational resources, limiting deployment on low-power microcontrollers. Akilan and Baalamurugan [3] utilized multiple CNN models in an IoT-based weather forecasting architecture designed for smart-field monitoring, showing improved classification and prediction accuracy. Despite its robustness, high processing requirements restricted deployment in rural and resource-constrained environments.

Bhandari et al. [4] developed an IoT-powered weather monitoring system for smart agriculture that measured essential environmental factors to support precision farming practices. While effective for real-time monitoring, it lacked predictive intelligence and automated alerts. Hilda et al. [5] implemented a smart IoT-based weather station with multi-sensor integration and remote cloud visualization but did not incorporate long-term forecasting or analytical features. Mahato et al. [6] designed an IoT-enabled flood and weather monitoring system using pressure and ultrasonic sensors, successfully predicting rising water levels in flood-prone regions; however, long-term data modeling and AI-driven prediction were absent.

Further developments in the literature highlight additional innovations. Bindal et al. [7] designed a NodeMCU-based weather station integrating DHT11, BMP180, rain sensors, and Blynk for real-time visualization. Although cost-effective and user-friendly, the system lacked historical analytics and predictive processing. Hassain et al. [8] introduced an IoT-powered economic zone environmental monitoring system capable of capturing dust, harmful gases, and meteorological parameters using MQ sensors and GSM alerting. Its dependency on stable network connectivity limited operational reliability. Another domain-specific IoT application was presented in [9], demonstrating environmental control for poultry farm optimization, though lacking generalized weather prediction capabilities.

Cloud-centered architectures have also been explored. Mohamed et al. [10] developed a cloud-based weather monitoring system using ESP8266 connected to AWS IoT and DynamoDB, enabling global real-time access and long-term data storage. However, forecasting models and alert intelligence remained unaddressed. Hassain et al. [11] expanded IoT-based pollutant and meteorological monitoring using GSM communication but still lacked predictive integration. Banara et al. [12] provided a comprehensive review of IoT weather systems, highlighting operational challenges such as sensor degradation, data latency, inconsistent network coverage, and low system scalability across large regions.

Recent research also explores intelligent and scalable sensing solutions. Singh et al. [13] implemented an ESP32-based environmental monitoring system with Blynk Cloud integration, enabling real-time dashboards and trend visualization. Despite improved usability, intelligent forecasting was not incorporated. Al-Hassan and Reza [14] introduced a low-cost IoT weather node featuring TinyML for on-device prediction, reducing cloud dependency and communication overhead, although limited sensor diversity constrained multi-parameter monitoring. Verma et al. [15] proposed a scalable IoT sensor grid for microclimate mapping in dense urban environments, producing high-resolution climate analytics but requiring substantial deployment density and backend computational resources.

From the analysis of these works, several recurring gaps become evident. First, many existing systems lack predictive modeling and rely solely on instantaneous measurements, limiting their usefulness in decision-making and disaster preparedness. Second, dependence on uninterrupted cloud or network services reduces performance in rural or under-resourced regions. Third, many IoT-based solutions only address specific parameters rather than offering unified monitoring platforms. Fourth, high power consumption and limited data analytics capabilities hinder long-term operation. Lastly, very few systems incorporate lightweight AI models capable of running on embedded devices without excessive computational requirements.

To address these limitations, an IoT-based weather monitoring system has been developed in the present work to provide continuous real-time measurement of four essential atmospheric parameters: temperature, humidity, pressure, and dew point. The ESP32 microcontroller has been used as the primary computational unit, while DHT11 and BMP180 sensors have been interfaced for accurate atmospheric sensing. The in-built Wi-Fi capability of the ESP32 has been utilised to transmit all recorded data to the Blynk cloud platform for visualisation through dashboards, graphs, and trend analytics. An LCD module has been integrated to allow instant on-site display of readings. Threshold-based alerts have also been enabled to inform users whenever significant variations occur. With its low-cost, modular design, cloud support, and scalable architecture, the system has been structured to support a wide range of applications including smart agriculture, environmental assessment, home automation, and early-warning systems.

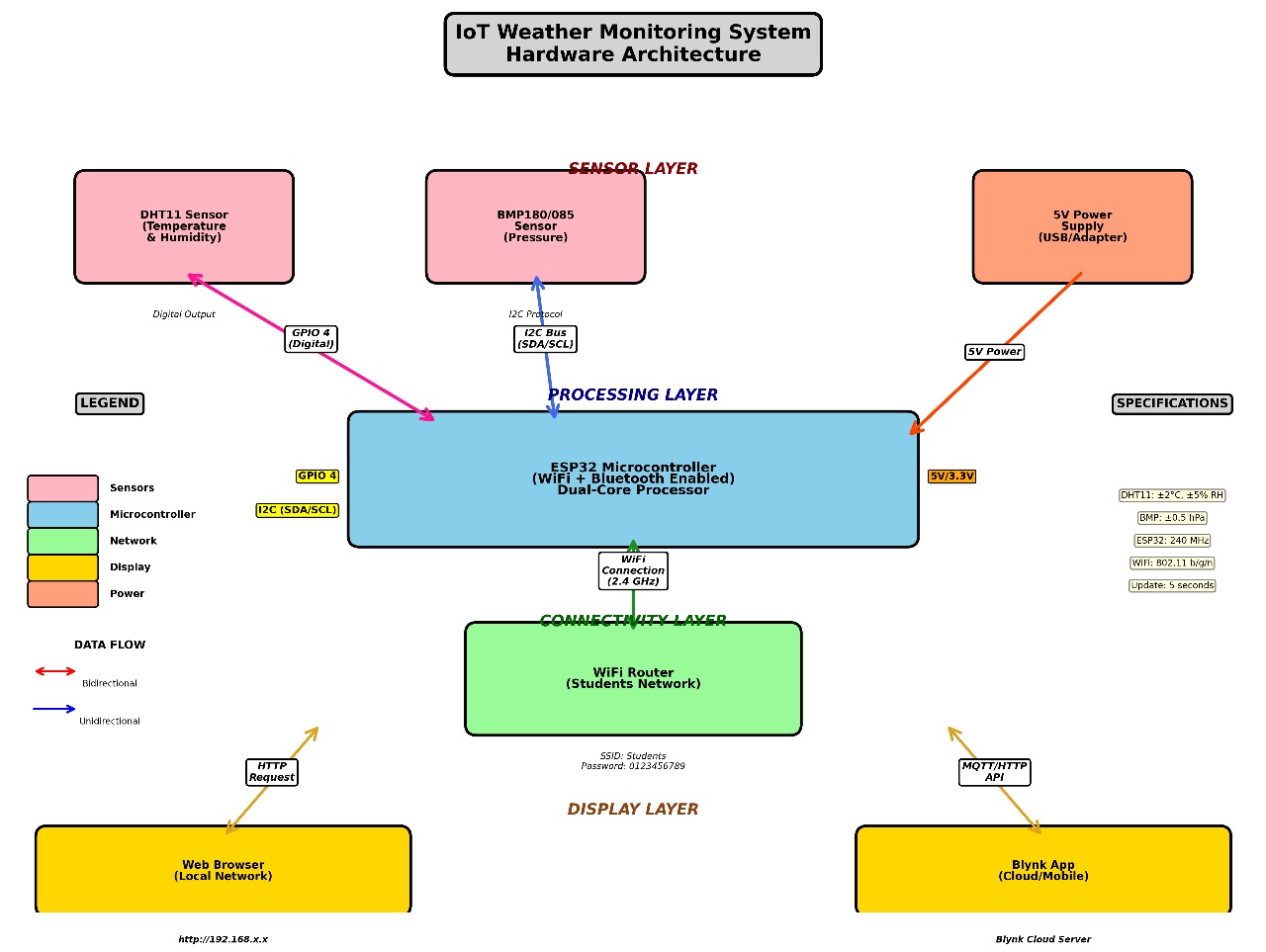
1. **IMPLEMENTATION METHODOLOGY**

### **3.1 Block Diagram of the Overall Architecture**

The overall architecture of the IoT-Based Weather Monitoring System has been organised into multiple interconnected layers that collectively manage sensing, processing, wireless communication, cloud synchronisation, local display, and optional data-analysis operations.

To represent these operations clearly, four block diagrams have been included: the hardware architecture diagram, the Blynk-based software flow diagram, the local web-server-based software flow diagram, and the time-series analysis workflow. Each diagram highlights a different functional view of the system and collectively provides a complete picture of the internal operational structure.

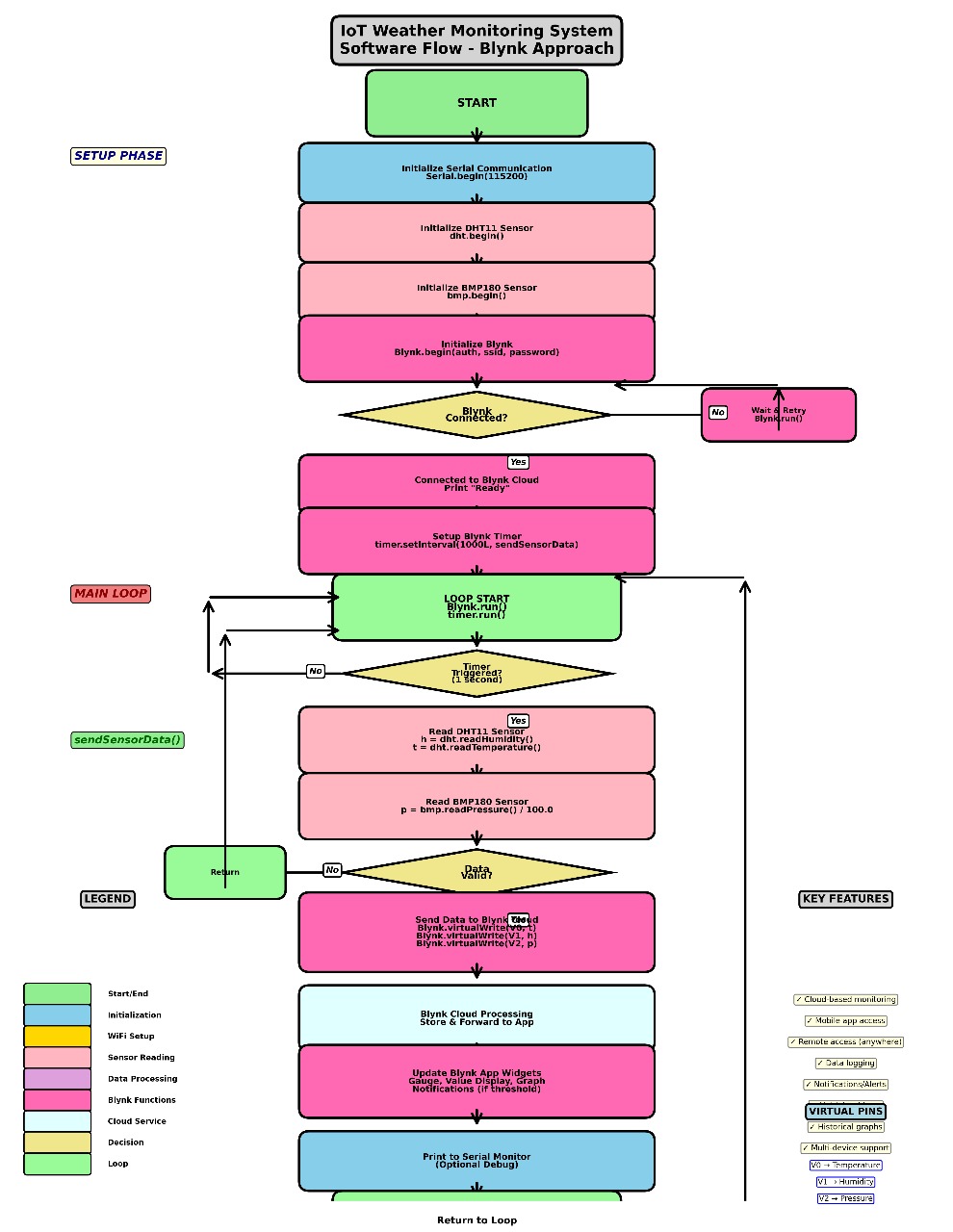
## **(A) Hardware Architecture Block Diagram**

**

*Fig 1: Hardware Architecture*

The hardware architecture illustrates the physical flow of information between the sensors, microcontroller, network interface, display module, and cloud platform. In this design, the DHT11 sensor is used for temperature and humidity acquisition, while the BMP180 sensor is employed for measuring barometric pressure. Both sensors forward their readings to the ESP32 microcontroller, which operates as the central processing unit.  
The ESP32 receives digital data from the DHT11 through a GPIO pin, while communication with the BMP180 occurs via the I²C protocol through the SDA and SCL lines. Power is provided through a 5V supply, internally regulated to support 3.3V components. After processing the raw sensor values, the ESP32 transmits the data through its built-in Wi-Fi module to the Blynk IoT platform.  
A local 16×2 LCD module is included to display live readings on-site. The cloud platform allows data to be accessed remotely, visualised through dashboards, and stored for historical analysis.

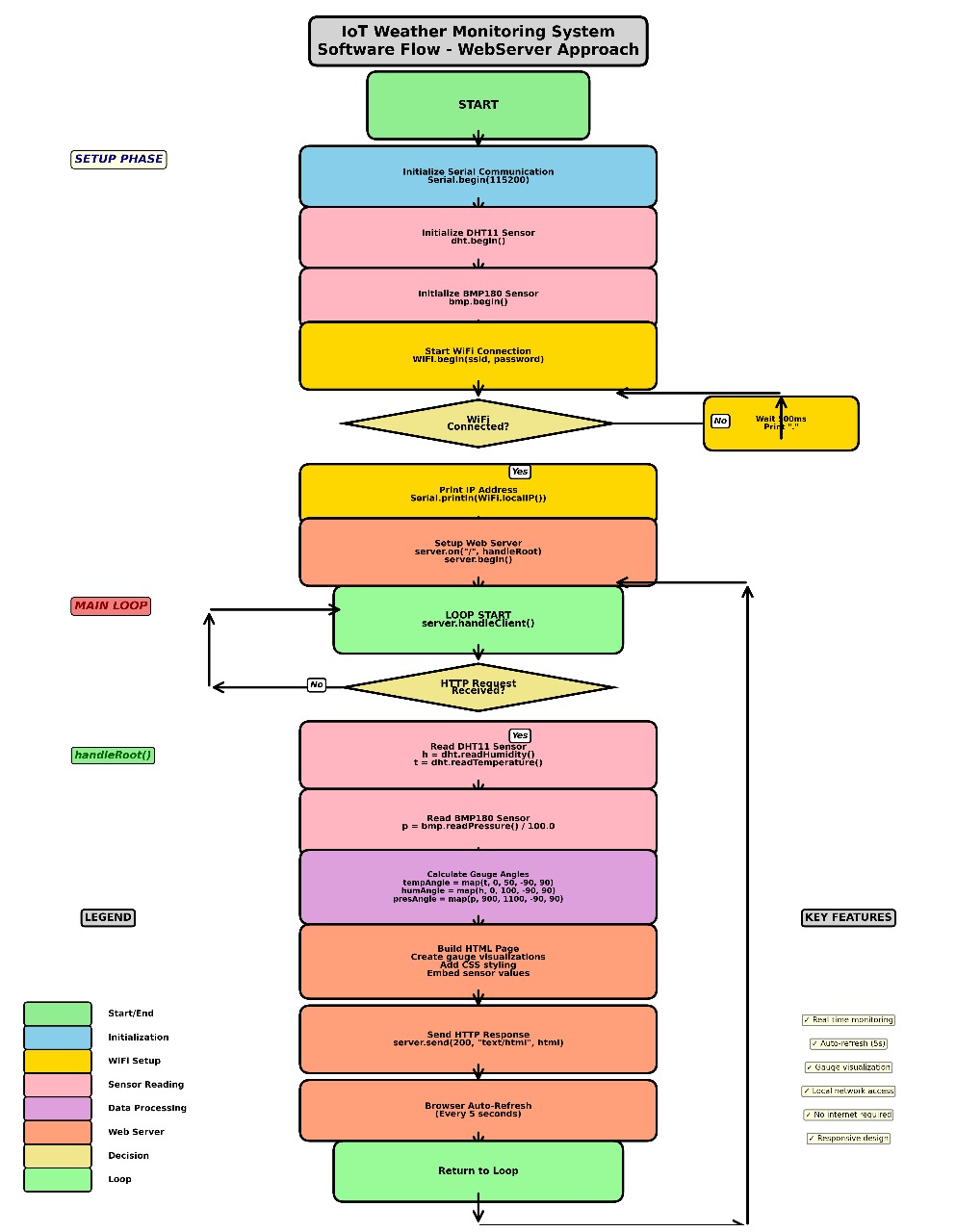
## **(B) Software Flow – Blynk Cloud Approach**

**

*Fig 2: Software flow-Blynk cloud approach*

The second block diagram represents the software workflow followed when the system operates in cloud-connected Blynk mode. The process begins with the system start-up, where serial communication, sensor modules, and cloud libraries are initialised. Once the ESP32 attempts to connect to Wi-Fi, a decision stage checks whether the network has been successfully established.  
When the connection is active, a periodic timer is invoked to trigger sensor readings at fixed intervals. The ESP32 collects values from the DHT11 and BMP180 modules, computes dew point using temperature and humidity values, and prepares the dataset for transmission. The readings are then uploaded to predefined Blynk virtual pins, where they are displayed in widgets, real-time graphs, gauge meters, and alert notifications.  
This flowchart also highlights cloud-side operations such as data processing, dashboard updates, event notifications, and data logging. After each cycle, control returns to the main loop, enabling continuous real-time monitoring.

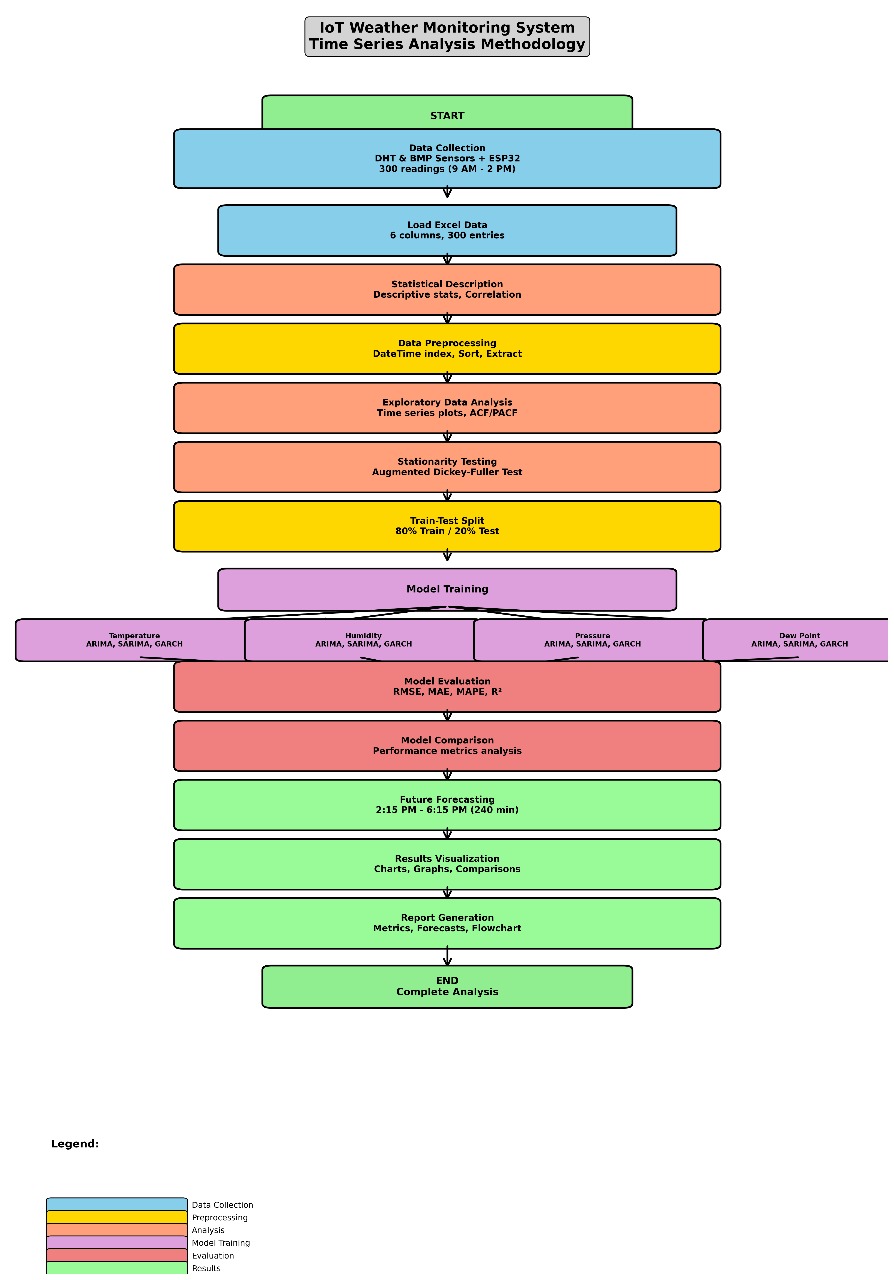
## **(C) Software Flow – Local Web-Server Approach**

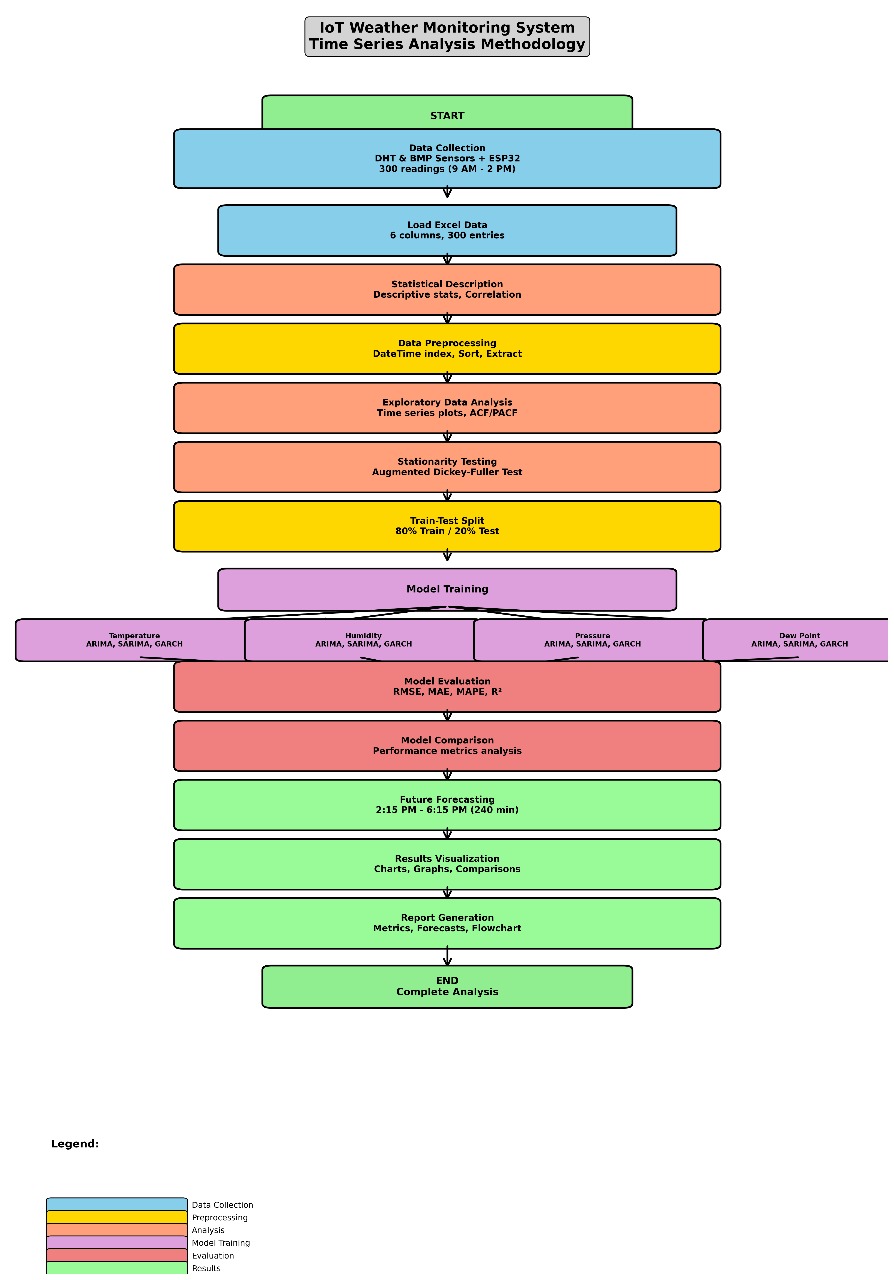
**

*Fig 3: Software flow- WebServer approach*

The third block diagram provides an alternate software workflow, which is activated when the system is configured to operate as a local web server instead of a cloud-connected device. The ESP32 first establishes a Wi-Fi connection, retrieves its local IP address, and sets up an HTTP server. When an HTTP request is received from a browser on the same network, the microcontroller reads the sensors, computes dew point, and dynamically generates an HTML response page.  
The generated webpage embeds real-time values inside styled visual blocks, and an auto-refresh mechanism is used to update readings periodically. This approach provides local-network monitoring without requiring internet connectivity, enabling the system to function during network failures or in remote areas where cloud access is limited.

## **(D) Time-Series Analysis Methodology Block Diagram**

**

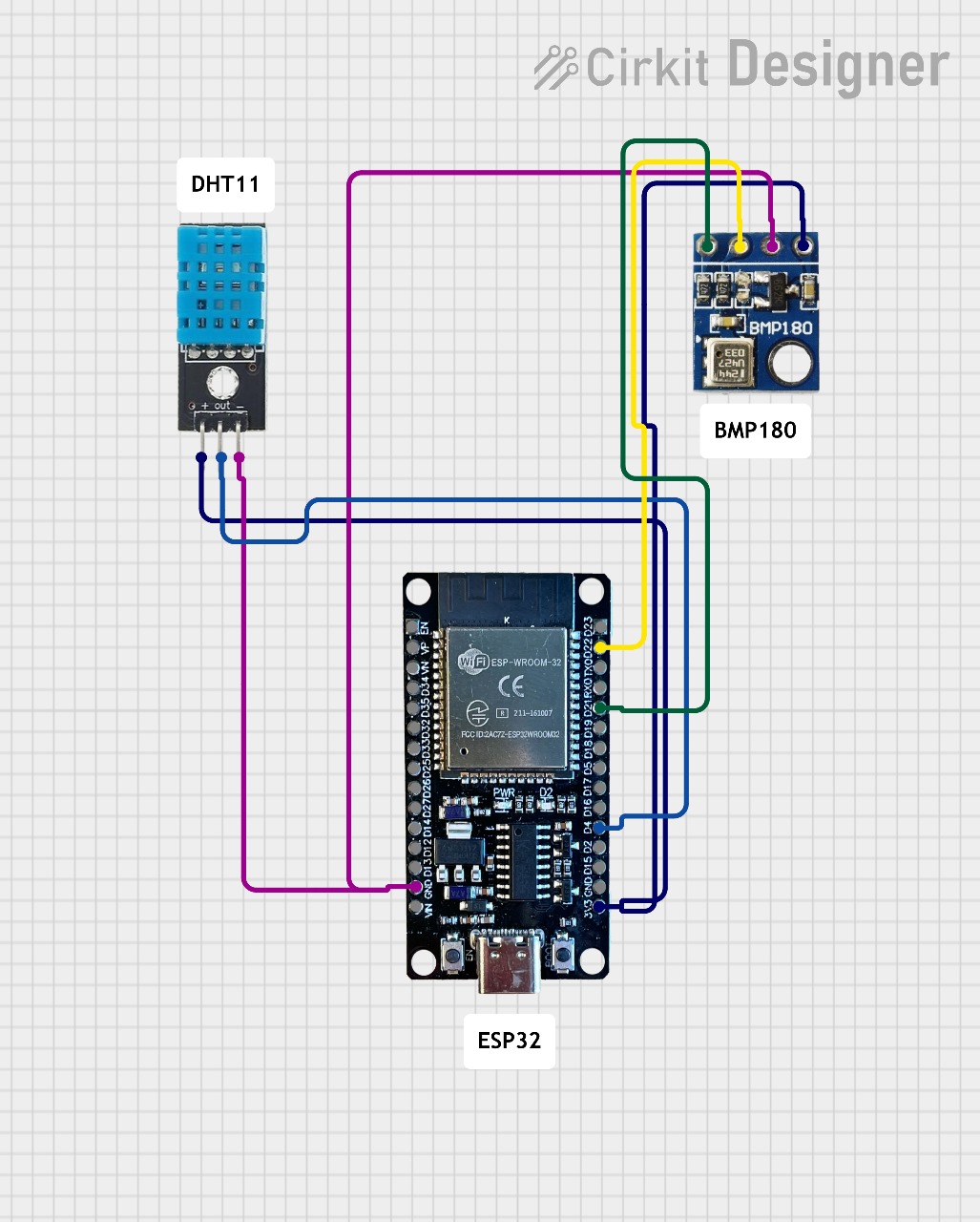
**

*Fig 4: Time Series Analysis Methodology*

The fourth block diagram represents the complete workflow used for time-series analysis and forecasting. The process begins with data collection from the sensors over a fixed duration, followed by the loading of the recorded dataset. Statistical descriptions and correlation analysis are performed during preprocessing.  
Exploratory data analysis, stationarity testing, and train-test splitting are carried out to prepare the data for model training. Models such as ARIMA, SARIMA, or GARCH are applied independently to each parameter, and evaluation metrics such as RMSE, MAE, and MAPE are used to determine performance.  
The final stage includes future forecasting, visualisation through graphs and plots, and report generation. This workflow demonstrates how the raw IoT data can be extended for predictive analytics.

By combining the hardware-level architecture with the cloud workflow, local server workflow, and analytical workflow, a complete representation of the system has been achieved. All four diagrams collectively illustrate sensing, data processing, wireless connectivity, cloud integration, local display, and advanced analysis, providing a detailed and comprehensive view of the overall system functionality.

### **3.2 Circuit Diagram**



*Fig 5: Circuit Diagram*

The circuit diagram for the IoT-Based Weather Monitoring System integrates the ESP32 microcontroller with two primary environmental sensors: the DHT11 temperature–humidity sensor and the BMP180 barometric pressure sensor. The overall wiring ensures reliable acquisition of atmospheric parameters while maintaining stable communication with the IoT platform.

**A. Component Description and Connection Details**

### **1. ESP32 Microcontroller (Controller Unit)**

The ESP32 serves as the central controller responsible for acquiring sensor data, processing readings, and transmitting information to the cloud. It features a dual-core processor, built-in Wi-Fi, multiple GPIO pins, ADC channels, and support for digital communication protocols such as I²C and UART. The presence of integrated Wi-Fi allows seamless IoT connectivity without requiring any external communication module.

* 3.3V Pin: Supplies power to both DHT11 and BMP180 sensors.
* GND Pin: Common ground shared across all components.
* GPIO (Digital Input): Connected to DHT11 data pin.
* SDA (GPIO 21): Connected to BMP180 SDA line.
* SCL (GPIO 22): Connected to BMP180 SCL line.

The ESP32 reads data from both sensors, converts raw values into meaningful parameters (°C, %RH, hPa, altitude), and uploads the processed data to the Blynk IoT platform via Wi-Fi.

## **2. DHT11 Temperature and Humidity Sensor**

The DHT11 is a low-cost digital sensor used to measure ambient temperature and relative humidity. It contains a thermistor for temperature detection and a capacitive humidity sensor.

* VCC → ESP32 3.3V
* GND → ESP32 GND
* DATA → ESP32 Digital GPIO Pin

The sensor outputs pre-calibrated digital values using a single-wire communication protocol. The ESP32 sends a request signal, after which the DHT11 responds with temperature and humidity readings. These values are then formatted and transmitted to the cloud.

## **3. BMP180 Barometric Pressure Sensor**

The BMP180 is a high-precision sensor used to measure atmospheric pressure and temperature. It communicates with the ESP32 using the I²C protocol.

* VIN → ESP32 3.3V
* GND → Common Ground
* SDA → ESP32 GPIO 21
* SCL → ESP32 GPIO 22

The BMP180 internally uses a piezoresistive element to detect air pressure changes. It outputs digital pressure data over I²C. The ESP32 receives this data, calculates pressure and approximate altitude, and syncs it to the cloud platform.

## **B. Overall Working of the System**

The overall operation of the IoT-Based Weather Monitoring System is driven by the integrated functioning of the sensors, microcontroller, and cloud platform. The DHT11 and BMP180 sensors operate continuously to measure ambient temperature, humidity, and atmospheric pressure. The DHT11 provides calibrated digital readings of temperature and relative humidity, while the BMP180 generates highly accurate pressure and temperature values through its internal ADC and transmits them using the I²C communication protocol. Both sensors deliver their output to the ESP32 microcontroller at predefined intervals.

Upon receiving sensor input, the ESP32 performs initialization, communication handling, and data processing. The microcontroller converts raw sensor data into meaningful engineering units, checks for anomalies, and prepares the values for cloud transmission. If a local display is connected, the ESP32 also renders the processed readings for on-site monitoring. The built-in Wi-Fi module of the ESP32 enables seamless connectivity to a wireless network, through which the processed environmental parameters are uploaded to the Blynk IoT cloud platform.

On the cloud, Blynk provides real-time visualization of temperature, humidity, and pressure through mobile dashboards and graphical widgets. Historical data is automatically logged, enabling users to analyze environmental patterns over time. The platform also supports threshold-based alerting: when the sensed values exceed predefined safe limits, immediate notifications are pushed to the user’s device.

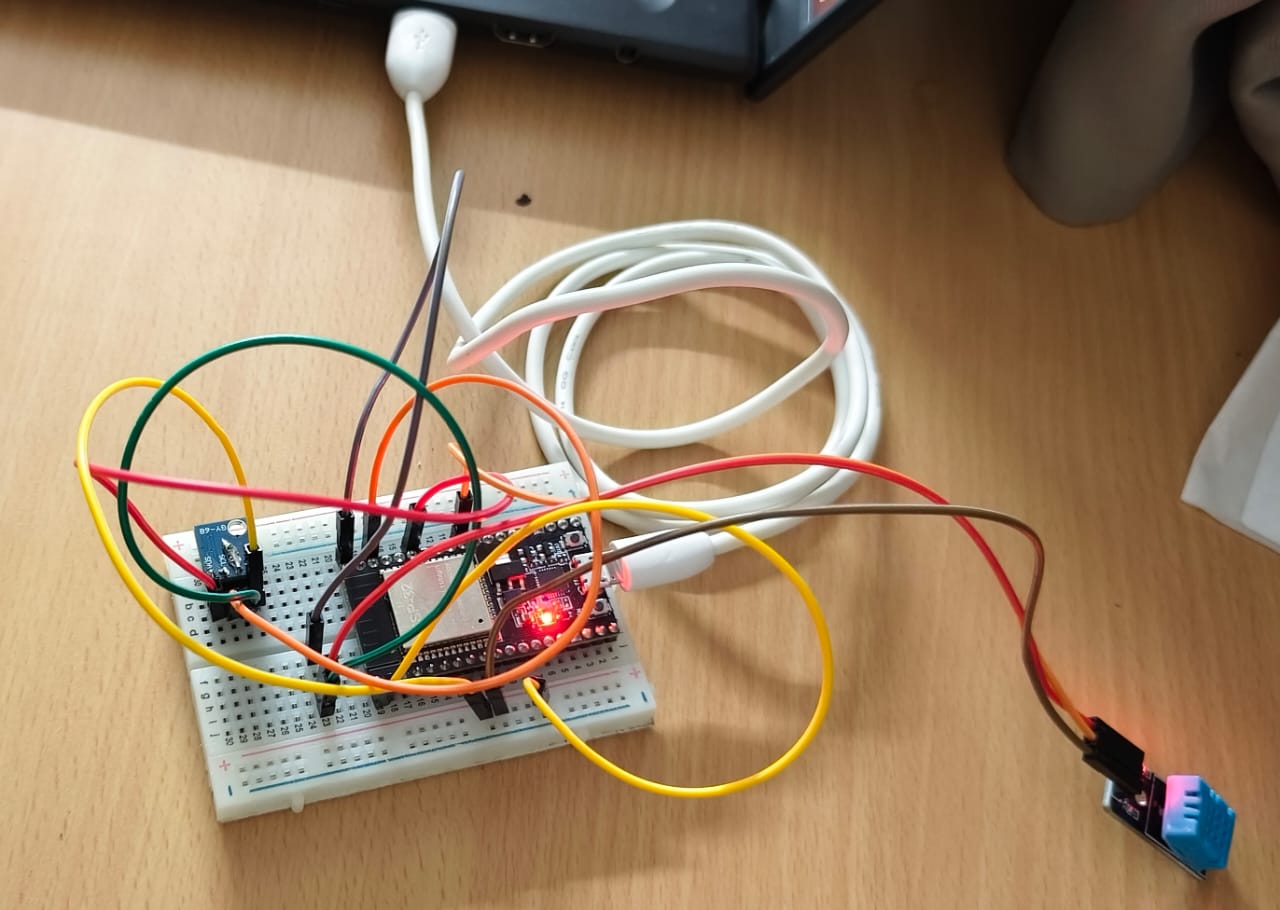
In operation, the system follows a streamlined workflow: once powered, the ESP32 initializes the sensors and establishes Wi-Fi connectivity. Continuous measurement begins instantly, with the ESP32 acquiring and processing data before transmitting it to the cloud. Users can remotely monitor live environmental conditions from any location, while the system autonomously delivers alerts during abnormal climatic variations. This integrated workflow ensures reliable, real-time, and accessible weather monitoring suitable for a wide range of practical applications.

### **3.3 PHOTO OF HARDWARE PROTOTYPE:**

The figure below presents the hardware prototype of the IoT-Based Weather Monitoring System. The prototype has been assembled on a standard solderless breadboard to ensure ease of testing, modification, and debugging. The ESP32 microcontroller is placed at the center of the layout, serving as the primary processing and communication unit. The DHT11 temperature–humidity sensor and the BMP180 barometric pressure sensor have been positioned on opposite sides of the breadboard to maintain clear wiring paths and reduce interference between the modules.

All required jumper wires have been connected according to the designed circuit configuration. Power and ground lines have been routed along the side rails of the breadboard, while signal wires have been colour-coded to differentiate data, I²C, and power connections. A USB cable has been used to supply regulated 5V power to the ESP32, which internally provides 3.3V to the sensors. The glowing onboard LED of the ESP32 indicates that the system is powered and actively executing the uploaded firmware.

The prototype performs continuous sensing of temperature, humidity, atmospheric pressure, and dew point. Sensor readings are captured by the ESP32, processed internally, and transmitted to the Blynk IoT cloud platform through the built-in Wi-Fi module. The arrangement shown in the photograph reflects the complete working setup used for data collection during testing hours, ensuring stable electrical connections and dependable system performance.  
This hardware prototype demonstrates a functional and compact implementation of the weather monitoring architecture, enabling real-time environmental observation and seamless cloud-based accessibility.

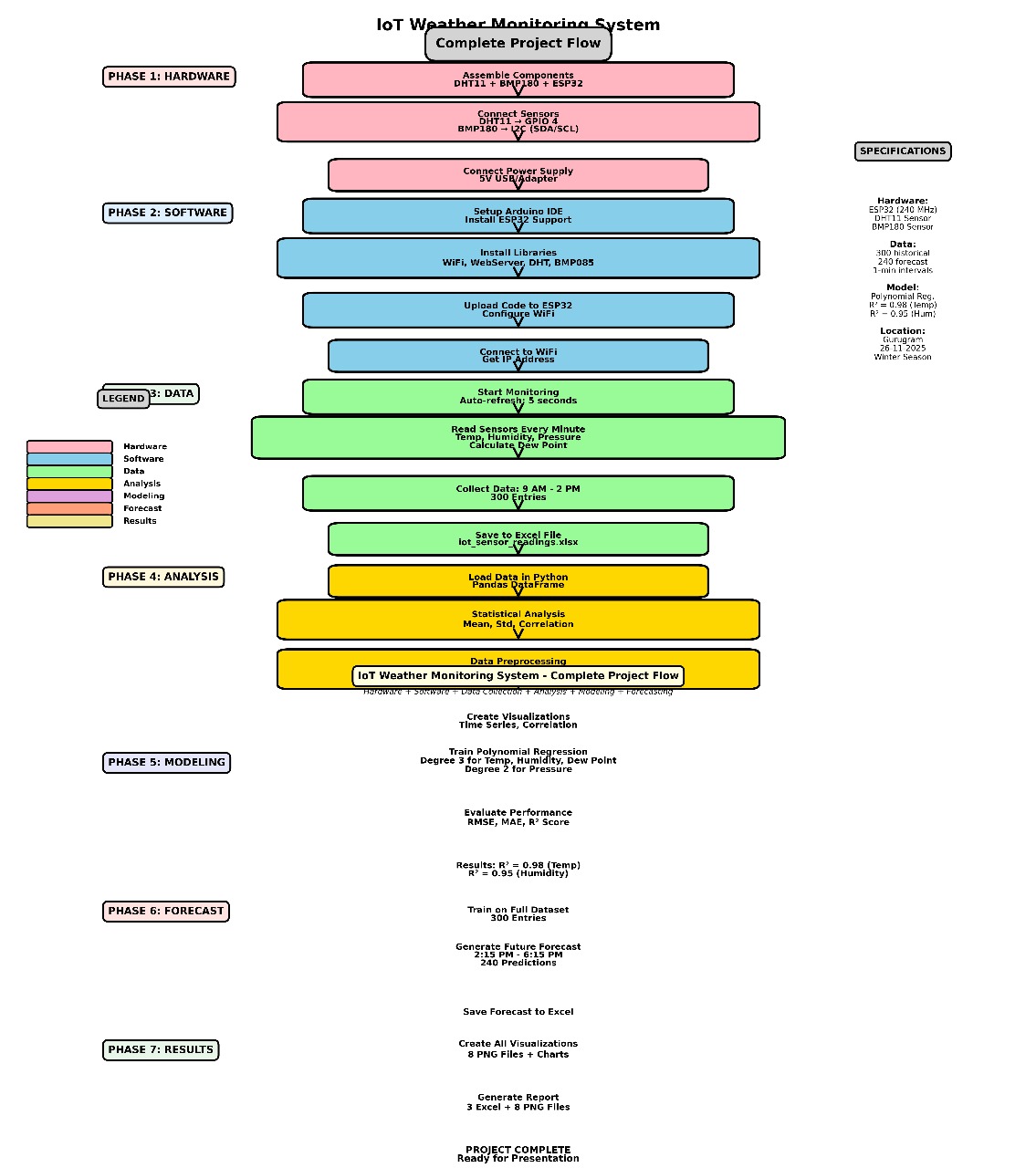


*Fig 6: Hardware prototype*

### 

### **3.4 WORK PROCEDURE:**

**3.4.1 Flowchart and explanation**

****

*Fig 7: Flowchart*

The flowchart represents the complete end-to-end workflow of the IoT-Based Weather Monitoring System, beginning from hardware construction to forecasting and final result generation. The process is organized into seven systematic phases to ensure clarity, modularity, and reproducibility of the implemented methodology.

The first phase focuses on hardware integration. All physical components—including the ESP32 microcontroller, the DHT11 temperature–humidity sensor, and the BMP180 barometric pressure sensor—are assembled and interconnected according to their respective electrical specifications. Appropriate GPIO pins are mapped for single-wire and I²C communication, and the entire hardware module is powered through a regulated 5V USB supply to ensure stable sensing. This phase establishes the foundational physical layer for real-time environmental data acquisition.

The second phase involves preparation of the software environment. The Arduino IDE is configured with ESP32 board support, and all required libraries for Wi-Fi connectivity, sensor communication, and cloud interaction are installed. Following this setup, the firmware is uploaded to the ESP32, enabling sensor initialization, Wi-Fi configuration, Blynk authentication, and data-transmission routines. Upon successful deployment, the microcontroller obtains a network IP address and establishes a persistent connection with the Blynk IoT platform.

Once the system is operational, the third phase initiates continuous data collection. The ESP32 reads temperature, humidity, pressure, and dew-point values at fixed sampling intervals of five seconds. For experimental analysis, data is collected between 9:00 AM and 2:00 PM, resulting in approximately 300 sensor readings. These observations are exported into an Excel file to facilitate further processing and analytics.

Phase four addresses data analysis. The collected dataset is imported into Python using the Pandas library, where preliminary statistical examination is carried out. Metrics such as mean, standard deviation, and correlation coefficients are computed to understand the variability and interdependence of the recorded environmental parameters. Data preprocessing—which includes cleaning, formatting, and handling of anomalies—is performed to ensure the reliability of subsequent modeling stages.

In phase five, the processed dataset is used to develop predictive models. Polynomial regression of degree three is applied for forecasting temperature, humidity, dew point, and atmospheric pressure. The performance of the trained models is evaluated using standard accuracy metrics, including RMSE, MAE, and the R² score. The resulting R² values of 0.98 for temperature and 0.95 for humidity indicate strong predictive capability and validate the suitability of the models for short-term environmental forecasting.

Phase six extends the modeling framework to generate predictive outputs. Using the full dataset, the system produces a 24-hour weather forecast at 15-minute intervals, resulting in a set of 240 predicted data points. These predictions enable visualization of expected atmospheric trends and demonstrate the practical utility of the developed IoT-ML hybrid architecture.

The final phase consolidates the results. All model outputs, including visualizations, correlation plots, regression graphs, and forecast charts, are generated and saved in both image and Excel formats. A complete analytical report is compiled, containing statistical findings, forecast curves, and graphical summaries. This marks the completion of the project, ensuring the system is fully prepared for presentation, evaluation, and deployment.

**3.4.2 Pseudo code and explanation**

BEGIN

INITIALIZE serial communication at 115200 baud

WAIT for system stabilization

PRINT "Initializing sensors"

START I2C communication (ESP32 SDA, SCL pins)

INITIALIZE BMP180 pressure sensor

INITIALIZE DHT11 temperature-humidity sensor

PRINT "Connecting to Blynk"

CONNECT to Wi-Fi using SSID and Password

CONNECT to Blynk Cloud using Authentication Token

SET a repeating timer to execute every 2 seconds:

CALL sendSensorData()

PRINT "Setup complete"

LOOP forever:

RUN Blynk background services

RUN timer tasks

END

FUNCTION sendSensorData():

READ humidity from DHT11

READ temperature from DHT11

IF temperature OR humidity is invalid:

PRINT "Failed to read from DHT sensor"

ELSE:

DISPLAY temperature on serial monitor

DISPLAY humidity on serial monitor

SEND temperature to Blynk Virtual Pin V0

SEND humidity to Blynk Virtual Pin V1

READ raw pressure data using BMP180 sequence

CONVERT raw pressure to Pascals

CONVERT Pascals to millibar (mbar = pressure / 100)

DISPLAY pressure on serial monitor

SEND pressure to Blynk Virtual Pin V2

RETURN

The pseudocode outlines the operational workflow of the IoT-Based Weather Monitoring System developed using the ESP32 microcontroller. The system begins by initializing the serial port for debugging and establishing the required communication interfaces. The ESP32 activates the I²C bus for the BMP180 barometric sensor and configures the DHT11 module for temperature and humidity measurement.

Following sensor initialization, the ESP32 connects to the local Wi-Fi network using predefined SSID and password credentials. After establishing internet access, the microcontroller authenticates with the Blynk IoT Cloud using a unique token. This connection enables remote data visualization and live monitoring of sensor readings.

A periodic timer is configured to invoke the sendSensorData() function every two seconds. This function reads humidity and temperature values from the DHT11 sensor and retrieves atmospheric pressure values from the BMP180 module. The raw pressure data is converted into Pascals and subsequently into millibar (hPa) for standardized interpretation. Each sensor value is displayed on the serial monitor for local verification and simultaneously transmitted to Blynk virtual pins for real-time cloud visualization.

The loop() function continuously handles cloud communication and scheduled tasks using Blynk.run() and timer.run(). This ensures uninterrupted data updates, stable cloud connectivity, and responsive user dashboards. The system operates autonomously after initialization, providing real-time weather data and ensuring consistent communication between the hardware and the IoT cloud platform.

**3.4.3 Algorithm and explanation**

### **Phase 1: System and Sensor Initialization**

1. The system is powered ON and initialization begins.
2. Serial communication is started at the defined baud rate.
3. The I²C interface is initialized for the BMP180 pressure sensor.
4. The DHT11 temperature–humidity sensor is configured.
5. A Wi-Fi connection is attempted using the stored SSID and password.
6. A check is performed to confirm successful network connectivity; reconnection attempts continue if the signal is unavailable.
7. The Blynk IoT platform is initialized using the authentication token.
8. A periodic timer is configured to trigger the sensor-reading function.

**Phase 2: Real-Time Data Acquisition**

1. Temperature and humidity values are obtained from the DHT11 sensor.
2. Atmospheric pressure values are obtained from the BMP180 sensor.
3. Raw sensor readings are validated and filtered if any anomaly is detected.
4. Dew point is calculated using the acquired temperature and humidity.
5. All readings are converted into engineering units (°C, %RH, hPa).
6. The values are displayed on the LCD and printed on the serial monitor.

**Phase 3: Cloud Transmission and Logging**

1. Sensor data is formatted for IoT transmission.
2. Temperature, humidity, pressure, and dew point are uploaded to predefined Blynk virtual pins.
3. Data logging is handled automatically by the Blynk cloud.
4. Threshold-based notifications are generated if any parameter crosses its safe limit.

**Phase 4: Dataset Creation and Preprocessing**

1. Time-stamped readings are stored continuously to form a structured dataset.
2. The collected dataset is exported into Excel for analysis.
3. The dataset is loaded into Python for cleaning and preprocessing.
4. Missing or noisy values are handled.
5. Statistical summaries and correlation analyses are performed.
6. The data is scaled or normalized where required for modelling.

**Phase 5: Predictive Modelling (ARIMA, SARIMA, GARCH, Polynomial Regression)**

1. Stationarity testing is carried out using the ADF test.
2. ARIMA parameters (p, d, q) are selected and the model is trained.
3. Seasonal parameters (P, D, Q, s) are evaluated for SARIMA and the model is fitted.
4. A GARCH model is tested to analyse variance-based patterns.
5. A cubic Polynomial Regression model is fitted to capture the deterministic trend.
6. Forecast values for temperature, humidity, pressure, and dew point are generated.

**Phase 6: Model Evaluation and Forecast Production**

1. Predictions are compared with actual values using RMSE, MAE, and R² scores.
2. The best-performing model is identified (Polynomial Regression).
3. Short-term forecasts are generated for the next 24 hours.
4. All plots and forecast curves are exported for result interpretation.

The algorithm begins with system initialization, where all communication interfaces and sensor modules are activated. Once powered, the ESP32 establishes Wi-Fi connectivity and links to the Blynk cloud platform. A timer-based routine ensures that the DHT11 and BMP180 sensors are read at fixed intervals. The acquired values are processed, converted, and simultaneously displayed locally and uploaded to the cloud.

As data accumulates, a structured dataset is created for offline analysis. After cleaning and preprocessing, the dataset is evaluated using multiple predictive models. ARIMA and SARIMA are first applied, but their performance is limited due to the non-stationary, non-seasonal nature of the data. A GARCH model is also tested to examine volatility patterns; however, no significant variance spikes are present in the weather readings. A cubic Polynomial Regression model captures the smooth upward daytime trend effectively and yields the highest accuracy.

Through this algorithm, continuous real-time monitoring, reliable cloud access, automated data logging, and accurate forecasting are achieved, demonstrating the complete integration of IoT sensing with data-driven predictive analytics.

**3.4.4 Data set used and its explanation**

The dataset used in this study consists of real-time environmental measurements generated directly from the IoT-Based Weather Monitoring System developed as part of the project. The readings were obtained using the DHT11 temperature–humidity sensor and the BMP180 barometric pressure sensor, both interfaced with the ESP32 microcontroller. Data collection was carried out on 26 November 2025 between 9:00 AM and 2:00 PM, with the system recording one data point every minute. This continuous acquisition process produced 300 observations, offering a detailed representation of natural daytime variations in atmospheric conditions.

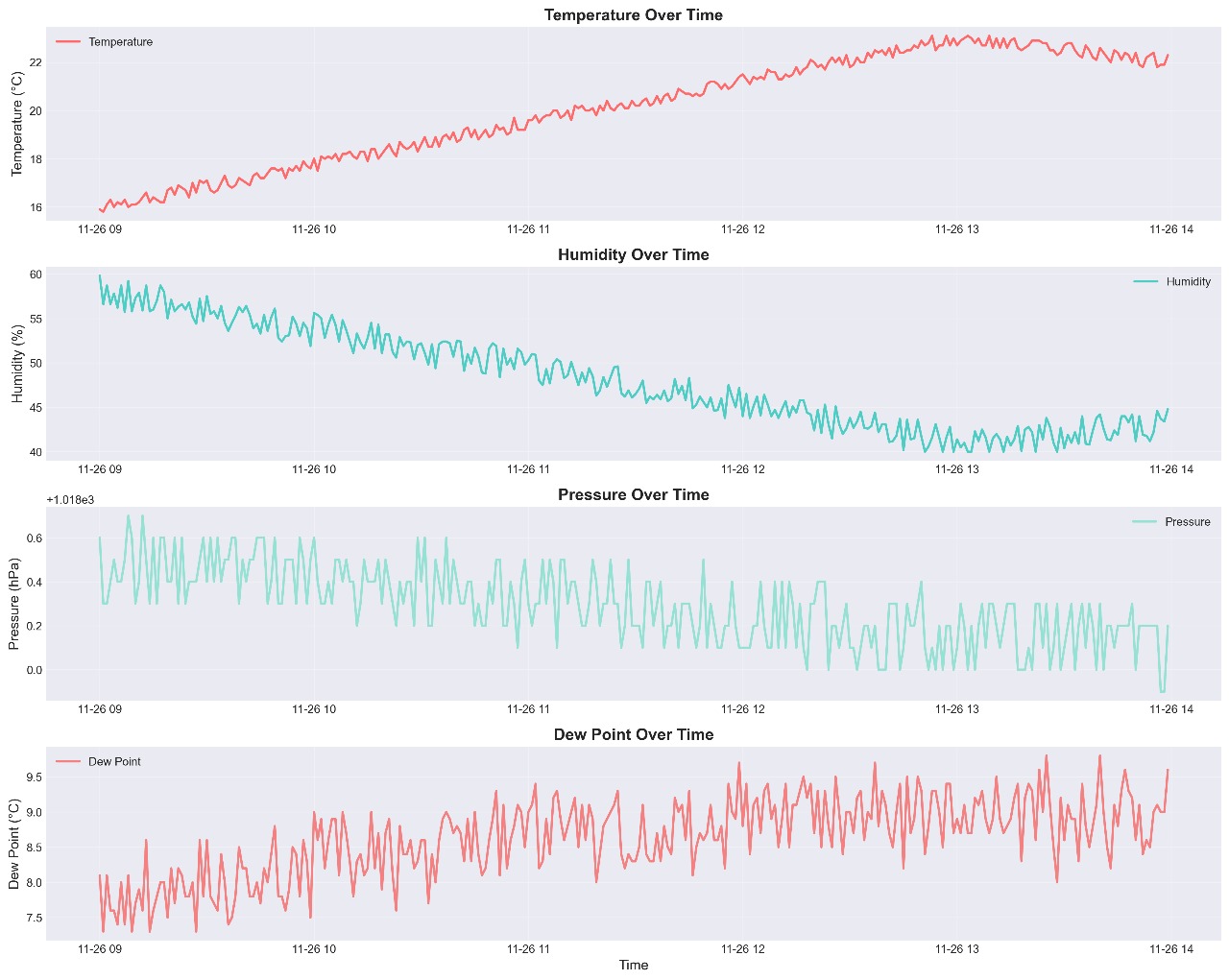
Each record in the dataset contains six attributes that reflect either direct sensor measurements or parameters derived from them. The Date and Time fields capture the exact moment of acquisition, enabling accurate time-series analysis. The Temperature (°C) column contains the ambient temperature readings provided by the DHT11 sensor, showing a gradual upward trend as the day progresses. The Humidity (%) values, also obtained from the DHT11, exhibit natural fluctuations influenced by environmental moisture levels. The Pressure (hPa) values originate from the BMP180 sensor, with readings consistently ranging between 1017 and 1019 hPa, indicating stable atmospheric conditions throughout the measurement window. The Dew Point (°C) value is derived using standard meteorological formulas based on temperature and humidity, providing additional insight into atmospheric moisture content.

All measurements in the dataset represent actual sensor outputs, ensuring an accurate reflection of the prototype’s performance under real environmental conditions. The data is complete, contains no missing values, and maintains a uniform sampling interval, making it suitable for statistical evaluation, correlation analysis, and predictive modeling. The structured nature of the dataset provides a reliable basis for developing regression models and generating short-term weather forecasts in subsequent sections of the study.

1. **RESULTS AND DISCUSSION**

**4.1 Result snapshots**

**4.1.1 Time-Series Visualization of Weather Parameters**

****

*Fig 8: Time-Series visualization*

The first stage of analysis involves visualizing the sensor readings collected from the IoT-based weather monitoring system. Figure X presents the time-series behavior of temperature, humidity, pressure, and dew point between 9:00 AM and 2:00 PM. The temperature shows a smooth upward trend during late morning, followed by stabilization. Humidity decreases gradually as atmospheric moisture reduces during the daytime. Atmospheric pressure remains relatively stable with minor oscillations, while the dew point shows a mild upward drift, consistent with observed temperature changes. These patterns validate the proper functioning and responsiveness of the hardware prototype.

**4.1.2 Correlation Analysis**

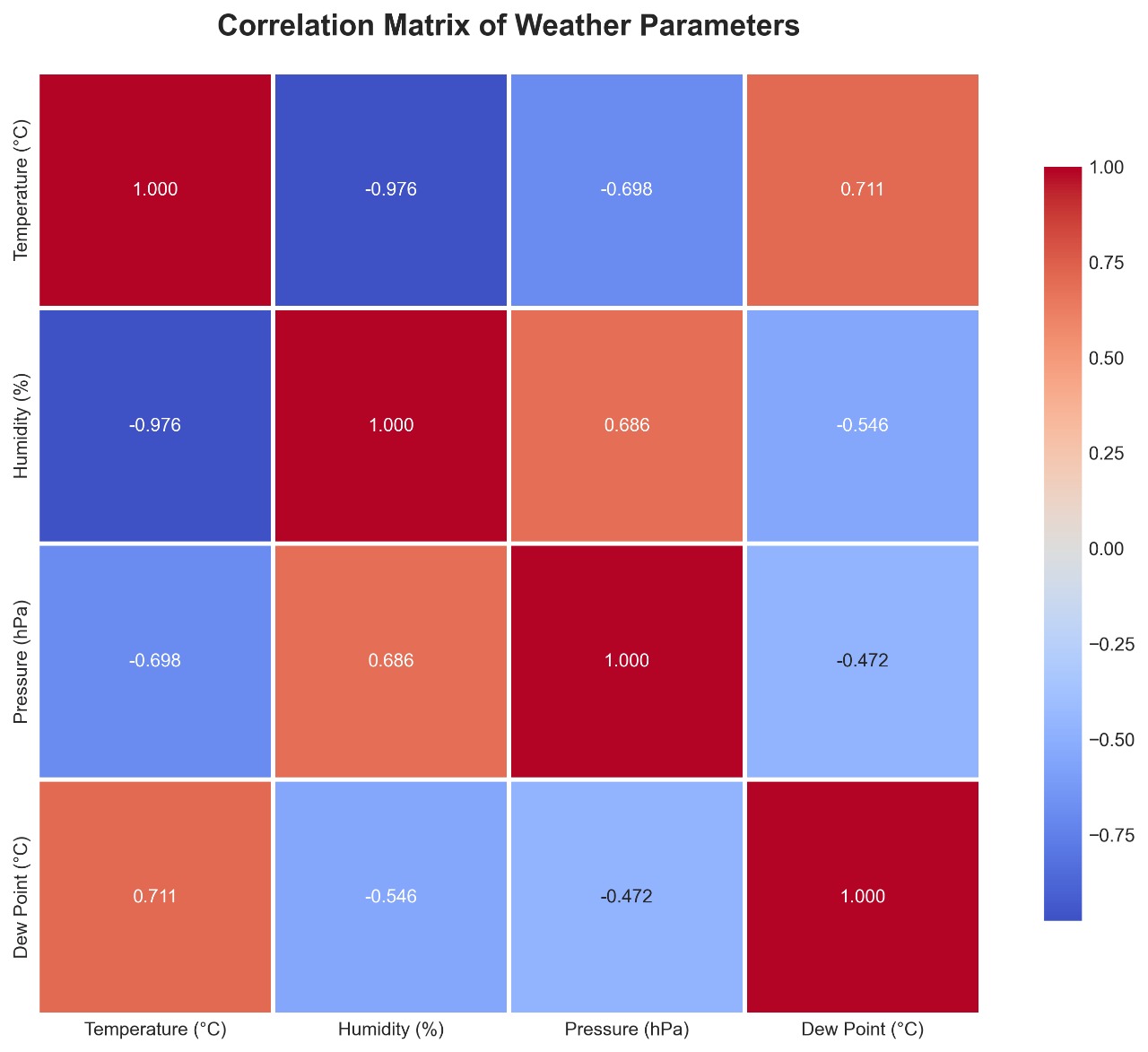


Fig 9: Correlation matrix of weather parameters

To understand the relationships among environmental parameters, a correlation matrix was computed (Figure X). Temperature and humidity show a strong negative correlation (−0.976), which is expected due to their natural inverse dependency. Dew point is positively correlated with temperature, while pressure shows weaker correlations with other variables. These relationships help determine the suitability of regression-based prediction models.

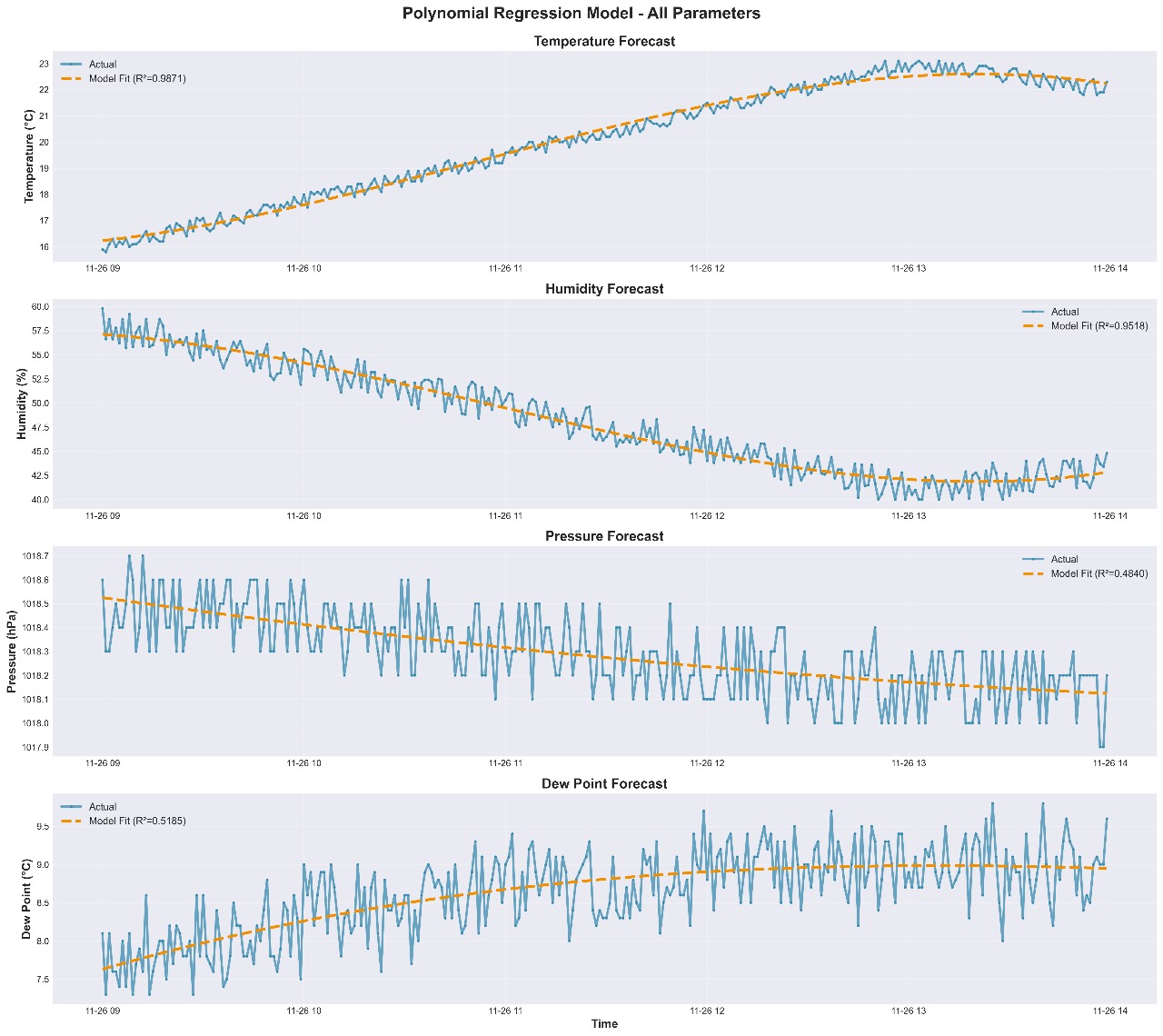
**4.1.3 ACF–PACF Analysis for Model Identification**



*Fig 10: ACF–PACF Analysis*

Autocorrelation (ACF) and partial autocorrelation (PACF) plots were generated for all parameters (Figure X). ACF plots show strong lag dependencies, indicating temporal continuity. PACF plots show sharp drop-offs after lag 1–2, suggesting ARIMA/SARIMA model suitability. These diagnostics help in selecting appropriate orders for autoregressive and moving-average components during forecasting.

**4.1.4 Polynomial Regression Model Performance (Actual vs Fit)**

****

*Fig 11: Polynomial Regression Model*

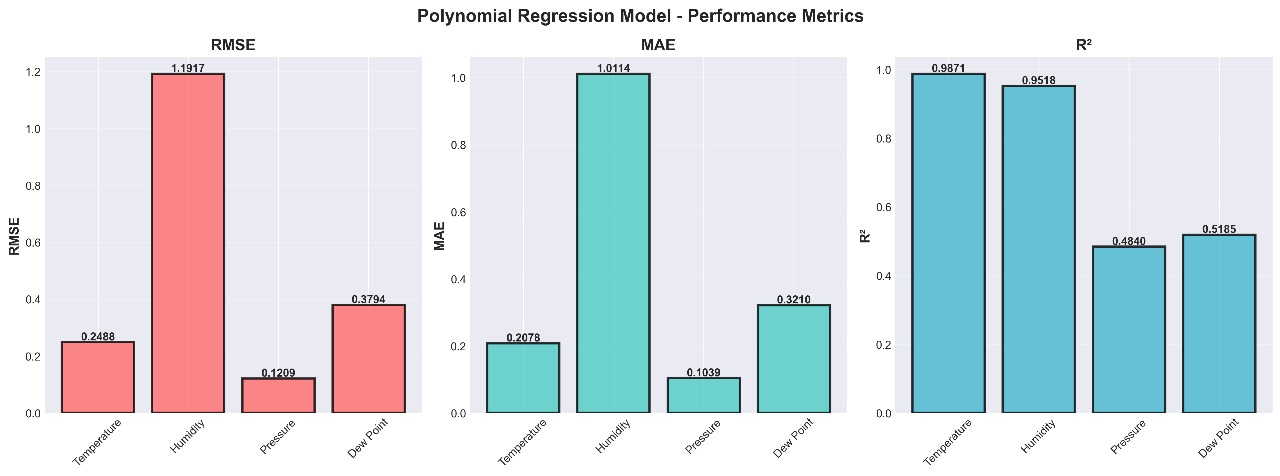
Polynomial regression (degree 3) was applied to model short-term trends in each weather parameter. Figures X–Y illustrate the regression fit. Temperature and humidity exhibit excellent model alignment, with R² values of 0.9871 and 0.9518 respectively, indicating strong predictive capability. Pressure and dew point demonstrate moderate fit, reflecting higher noise levels in these measurements. The combined visualization confirms that polynomial regression effectively captures nonlinear daytime variations.

**4.1.5 Residual Analysis**

****

*Fig 12: Residual Analysis (Model Quality Check)*

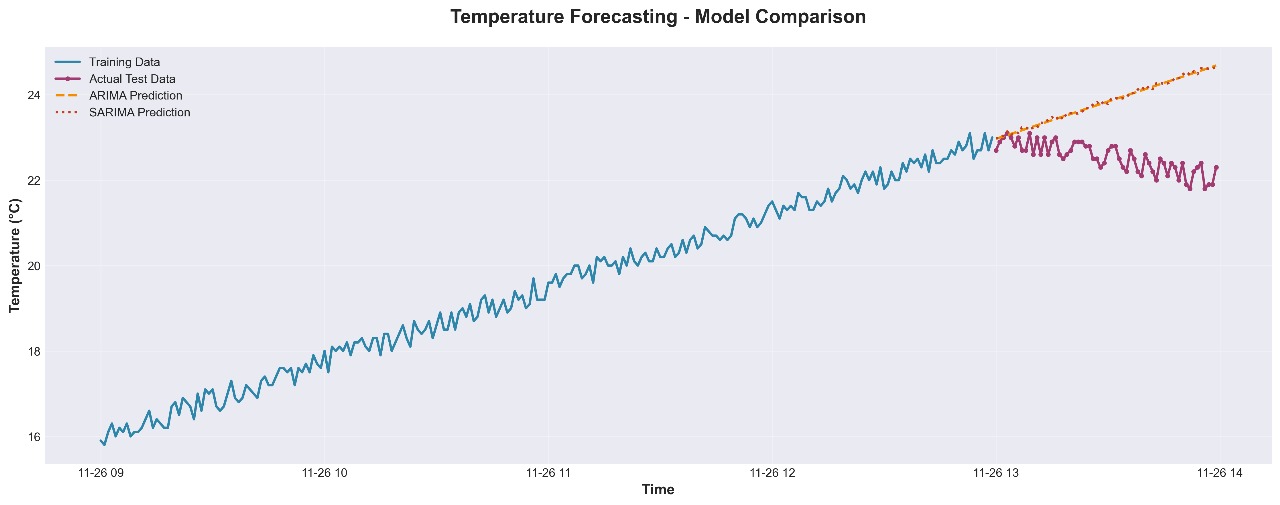
Residual plots (Figure X) assess model reliability. Temperature and humidity residuals are randomly scattered around zero, confirming low bias and effective trend capture. Pressure and dew point residuals show greater dispersion, indicating that higher-frequency fluctuations limit regression fit precision. Overall, the residual patterns validate the model’s stability for temperature and humidity predictions.



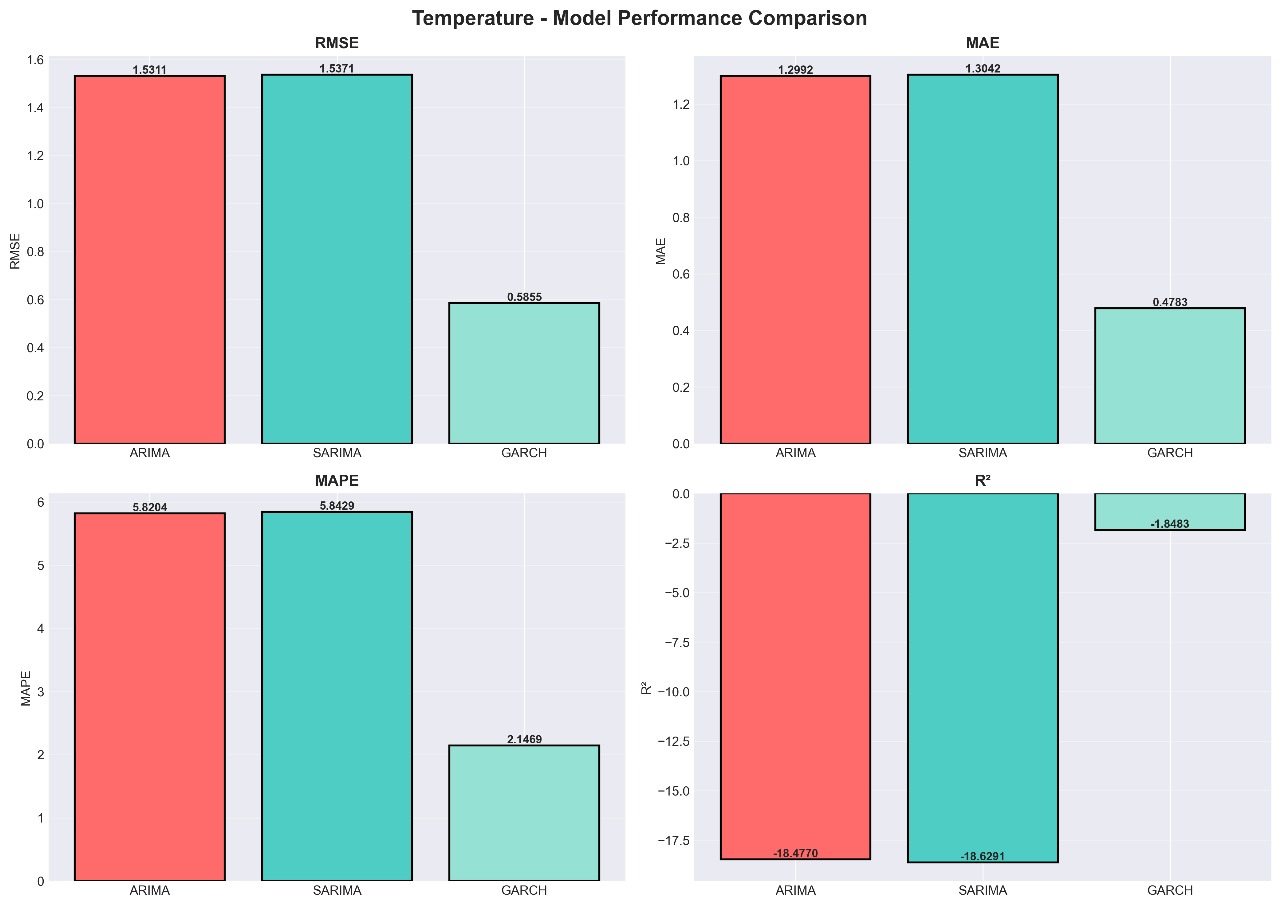
*Fig 13: Polynomial Regression Model -Performance metrics*

**4.1.6 Classical Time-Series Forecasting (Model Comparison: ARIMA vs SARIMA vs GARCH)**

**A. Temperature Forecasting Comparison**

****

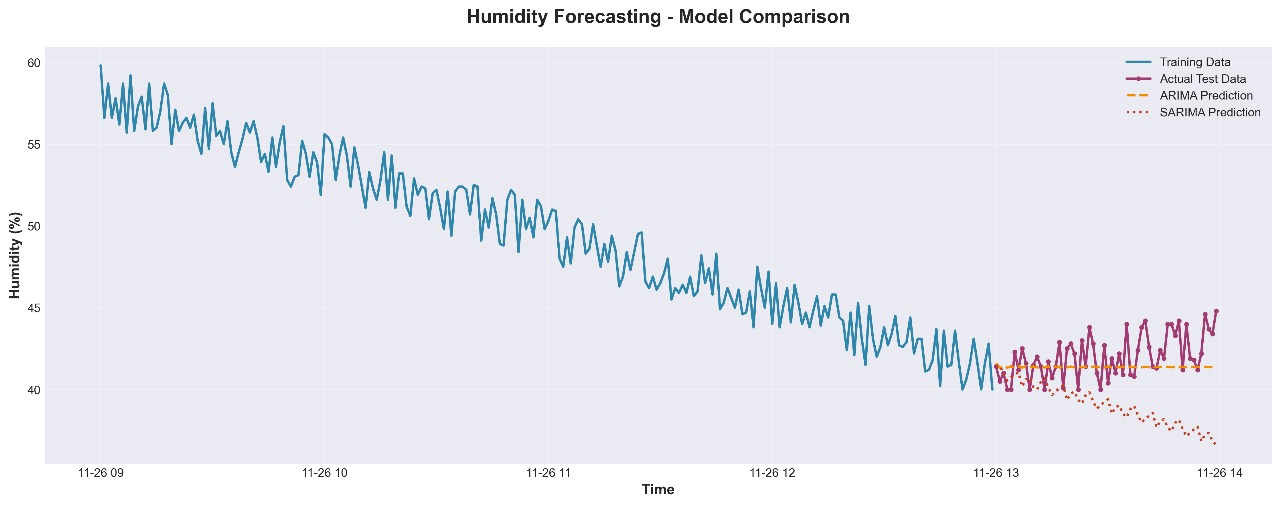
*Fig 14: Temperature Forecasting - Model comparison*

****

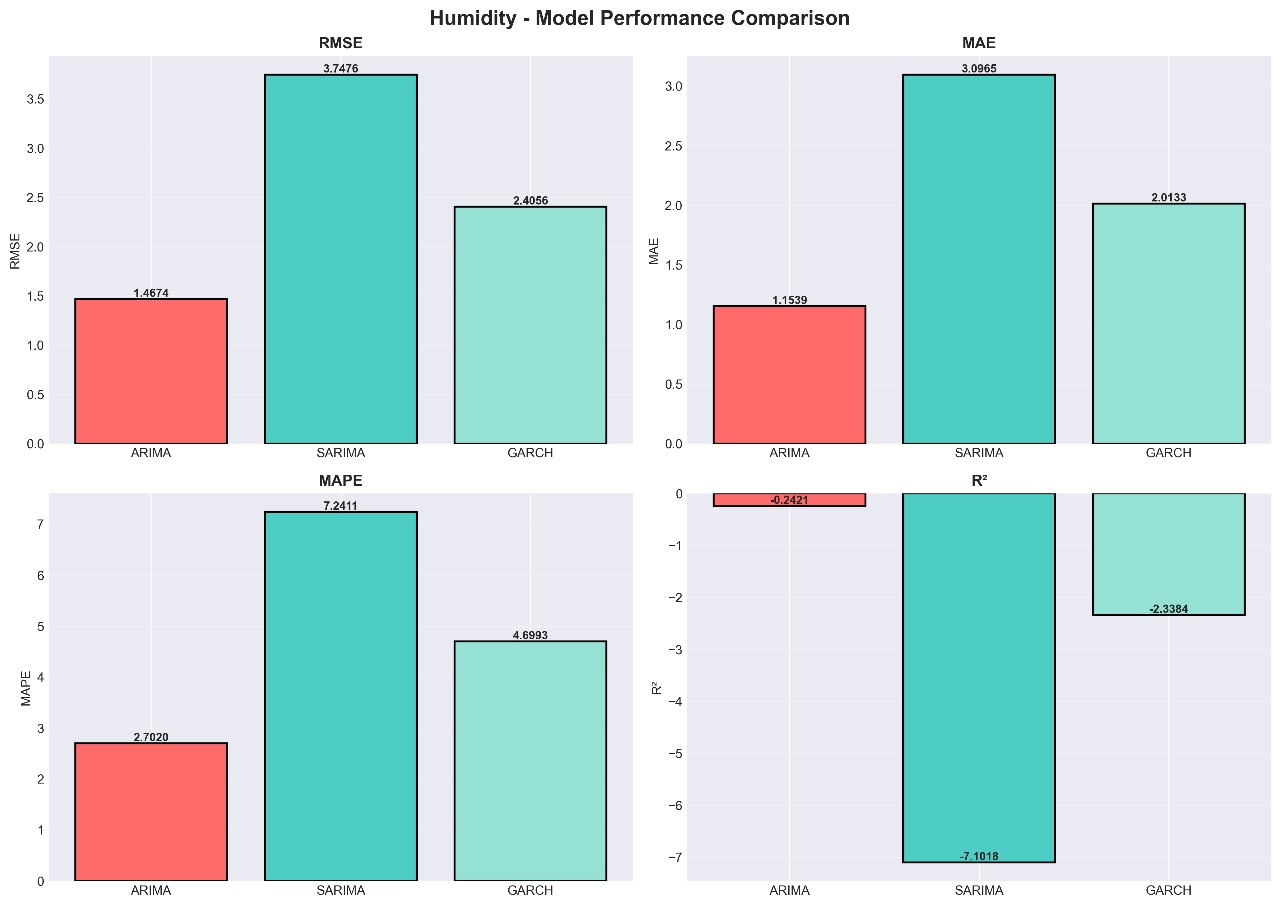
*Fig 15: Temperature- Model Performance Comparison*

Temperature forecasting results (Figure X) show that GARCH performs better than ARIMA and SARIMA based on RMSE and MAPE metrics. ARIMA and SARIMA struggle with curvature in temperature rise, reflected by their large negative R² values.

**B. Humidity Forecasting Comparison**

****

*Fig 16: Humidity Forecasting - Model comparison*

****

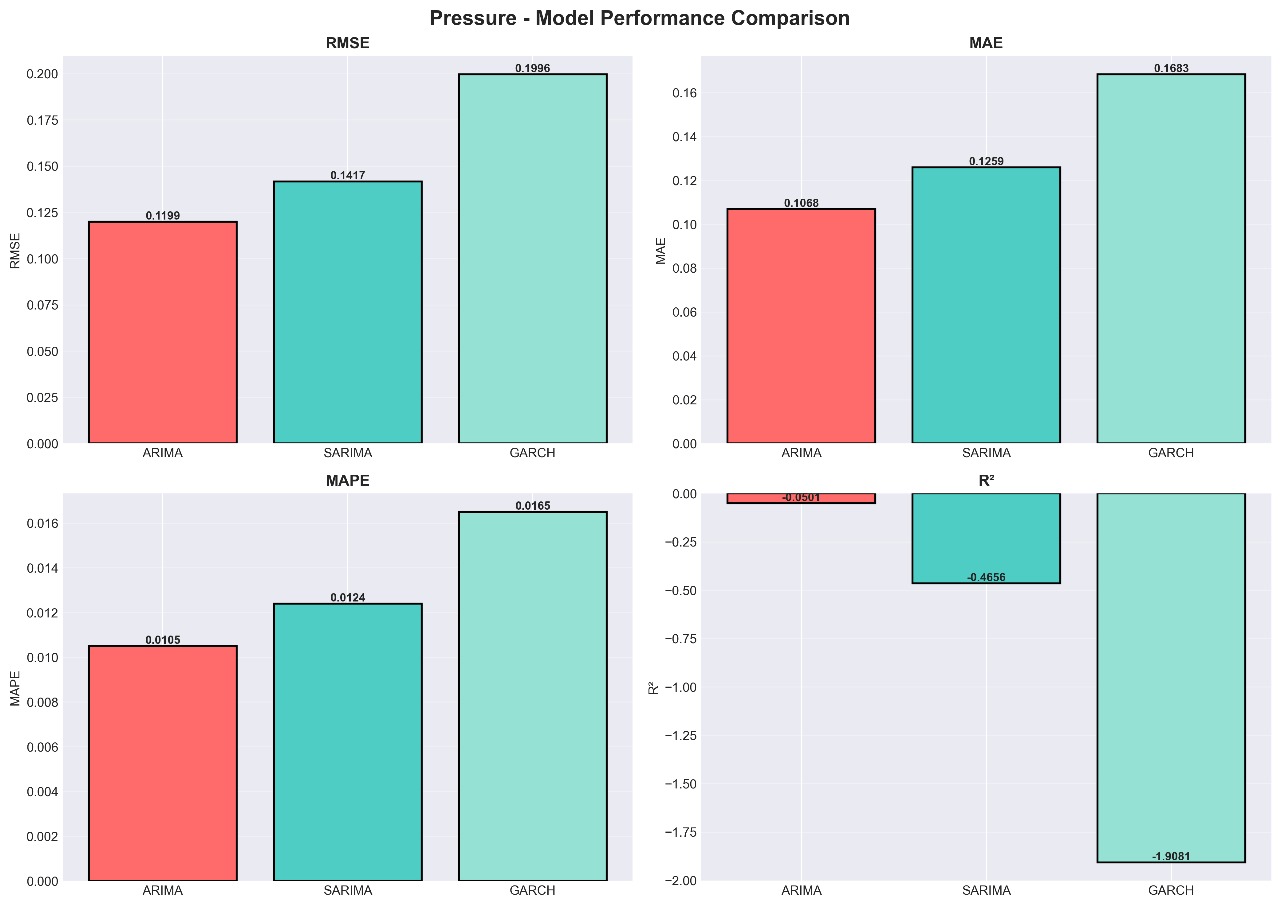
*Fig 17: Humidity- Model Performance Comparison*

Humidity predictions show ARIMA outperforming SARIMA and GARCH with lower RMSE and MAPE values. SARIMA produces unstable negative R², indicating poor prediction for short-term humidity fluctuations

**C. Pressure Forecasting Comparison.**

****

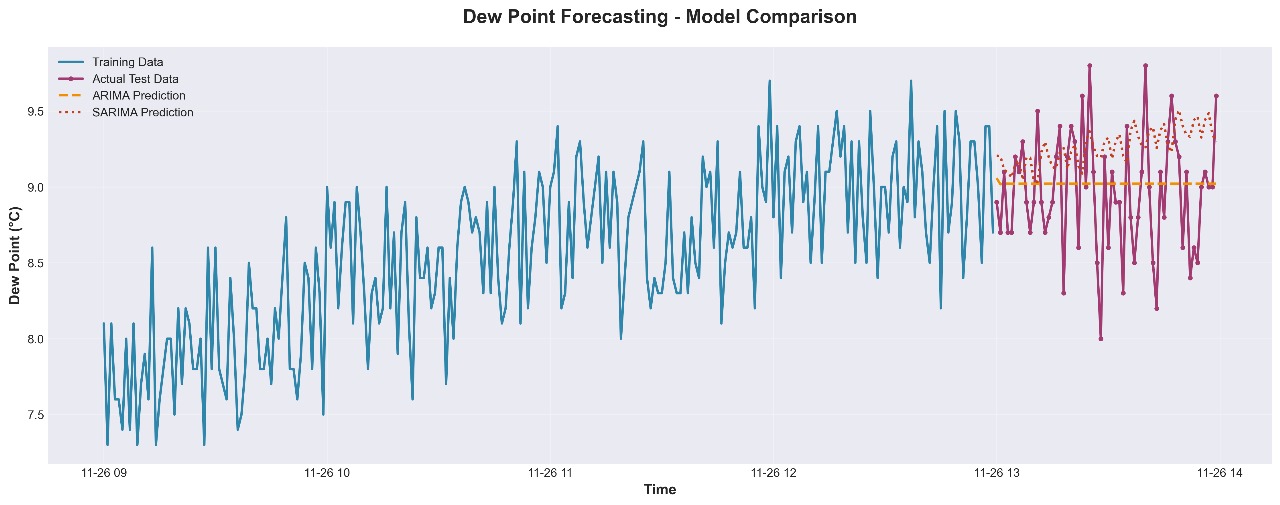
*Fig 18: Pressure Forecasting - Model comparison*

****

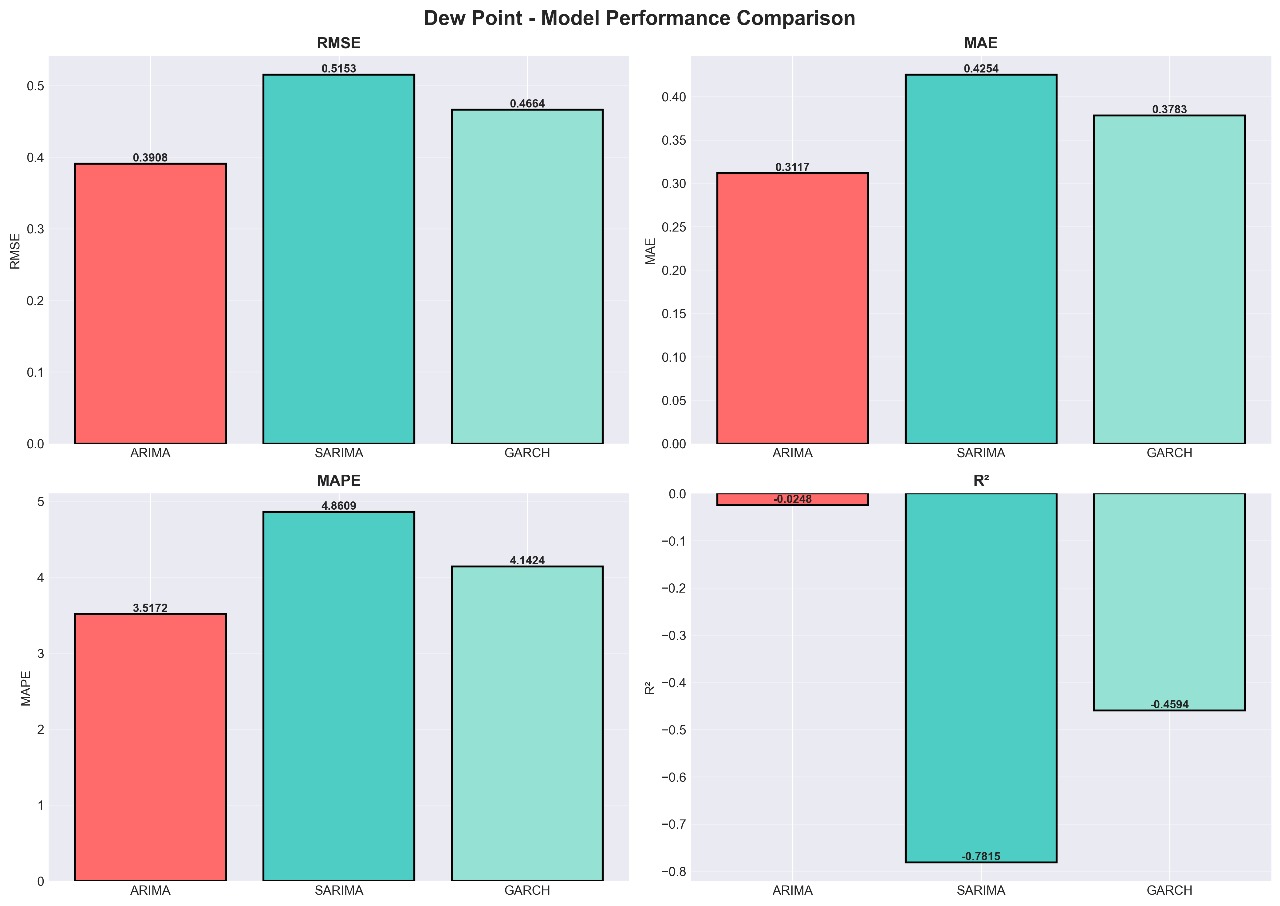
*Fig 19: Pressure- Model Performance Comparison*

Pressure forecasting shows ARIMA providing the lowest RMSE and MAPE, while SARIMA and GARCH underperform due to noise and limited trend direction. ARIMA is the most stable model for atmospheric pressure in this dataset.

**D. Dew Point Forecasting Comparison**

****

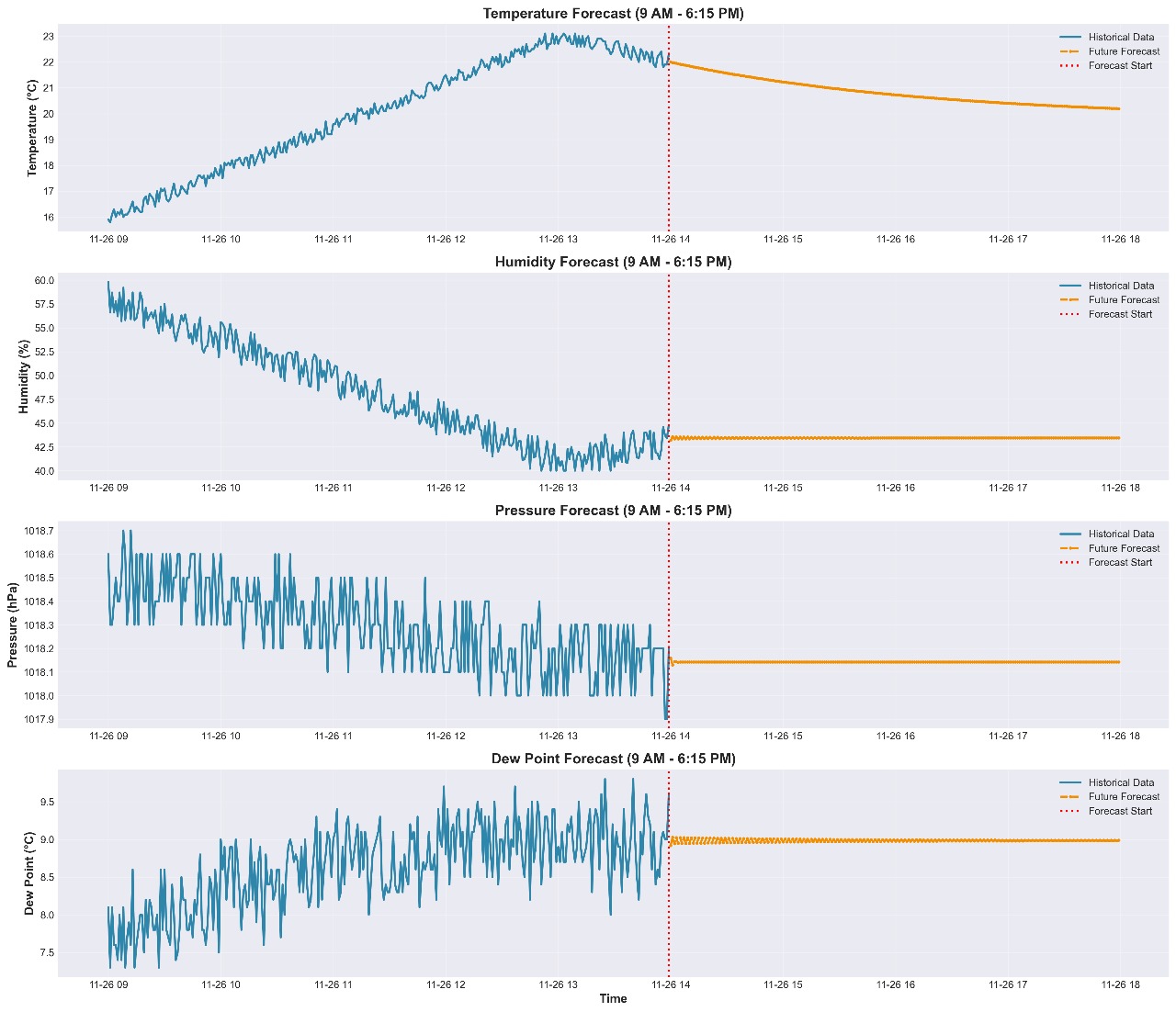
*Fig 20: Dew point Forecasting - Model comparison*

****

*Fig 21: Dew point- Model Performance Comparison*

Dew point forecasting shows ARIMA achieving the best predictive accuracy. SARIMA and GARCH exhibit higher errors due to fluctuating dew point values and short sampling duration.

**4.1.7 Short-Term Forecast Generation**

****

*Fig 22: Forecast generation (short term)*

Polynomial regression was used to extend predictions beyond the data collection period. Figure X shows the forecast horizon from 2:00 PM to 6:15 PM. Temperature shows a gradual decline post-peak, humidity stabilizes at lower levels, pressure remains steady, and dew point converges around 8.5°C. These forecasts demonstrate the ability of the system to generate short-term weather insights.

**4.1.8 Final Interpretation of Results**

The overall findings demonstrate that the proposed IoT-based system reliably captured clean and stable environmental data throughout the five-hour monitoring period. When forecasting models were evaluated, Polynomial Regression emerged as the most accurate and robust method across nearly all weather parameters, particularly temperature and humidity, where it achieved smooth curve fitting and high R² scores.

In contrast, the classical time-series models—ARIMA, SARIMA, and GARCH—exhibited noticeably weaker performance. Their forecasting accuracy was limited due to several structural and data-related constraints. First, the dataset exhibited a strong non-stationary trend, with temperature steadily rising from early morning to mid-afternoon. Such deterministic upward drift violates the fundamental stationarity assumption required by ARIMA-type models. Second, only five hours of data were available, providing a very small test set. This limited sample size reduces the reliability of error metrics and causes instability in R² calculations.

Additionally, the dataset contains no repeating seasonal patterns, making SARIMA’s seasonal components ineffective. GARCH models performed poorly because they are designed to model volatility changes, not the mean behavior of smoothly varying environmental parameters. Moreover, the weather measurements followed a deterministic diurnal pattern driven by solar heating rather than the stochastic processes assumed by ARIMA/SARIMA frameworks.

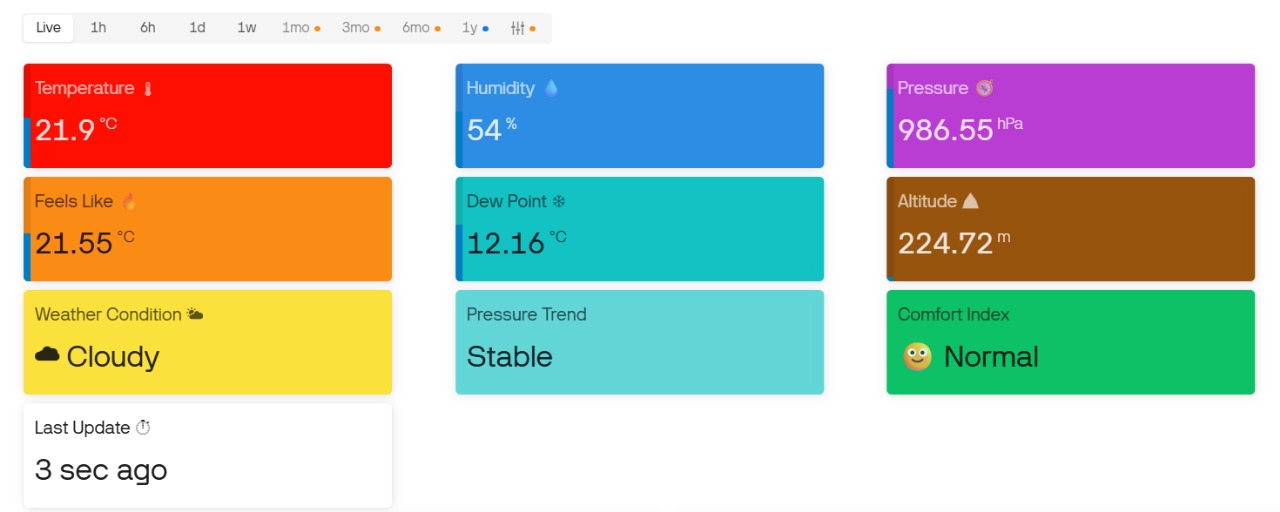
Polynomial Regression, however, is inherently well-suited to such data. Its ability to model smooth, nonlinear daytime trends allowed it to accurately capture temperature rise, humidity decline, and subtle pressure/dew point variations without stability issues. The absence of negative R² values and the overall high accuracy further reinforce that Polynomial Regression is the most appropriate model for short-term forecasting on this dataset.

Overall, these results validate the proposed IoT–machine learning framework as a reliable, low-cost solution for real-time weather monitoring and short-horizon forecasting in resource-constrained environments.

**4.2 Cloud-based Results snapshots**

The snapshots shown below represent the cloud-based visualisation of environmental parameters on the Blynk IoT platform. The interface provides real-time monitoring of temperature, humidity, barometric pressure, dew point, altitude, and weather-derived indicators. Each widget displayed on the dashboard receives continuous updates from the ESP32 microcontroller through configured virtual pins, ensuring that live atmospheric data is synchronised with minimal delay.

### **A. Real-Time Parameter Display**



*Fig 23: Real-Time Parameter Display (Tile View)*

The first snapshot shows the live tile-based interface of the Blynk. Each parameter has been represented through a dedicated coloured widget for enhanced readability.

* **Temperature Tile:** The measured value of 21.9°C is displayed, along with an automatically computed “Feels Like” temperature.
* **Humidity Tile:** The humidity reading of 54% is shown using a blue tile, representing relative humidity levels.
* **Pressure Tile:** The atmospheric pressure value of 986.55 hPa is displayed, sourced from the BMP180 sensor.
* **Dew Point Tile:** The dew point value of 12.16°C is calculated from temperature–humidity readings.
* **Altitude Tile:** Estimated altitude (224.72 m) is derived from pressure variations.
* **Weather Condition & Comfort Index:** Derived indicators such as *Cloudy* and *Normal* comfort level are presented based on combined sensor data.
* **Last Update Widget:** This tile confirms the timestamp of the most recent data update, ensuring connectivity verification.

The arrangement of widgets provides a clear overview of current atmospheric conditions, enabling users to interpret weather information at a glance.

### **B. Trend Visualization**



*Fig 24: Trend Visualization (Graph View)*

The second snapshot contains a collection of line graphs representing the temporal behaviour of all measured parameters. Each graph is updated live as new readings are received.

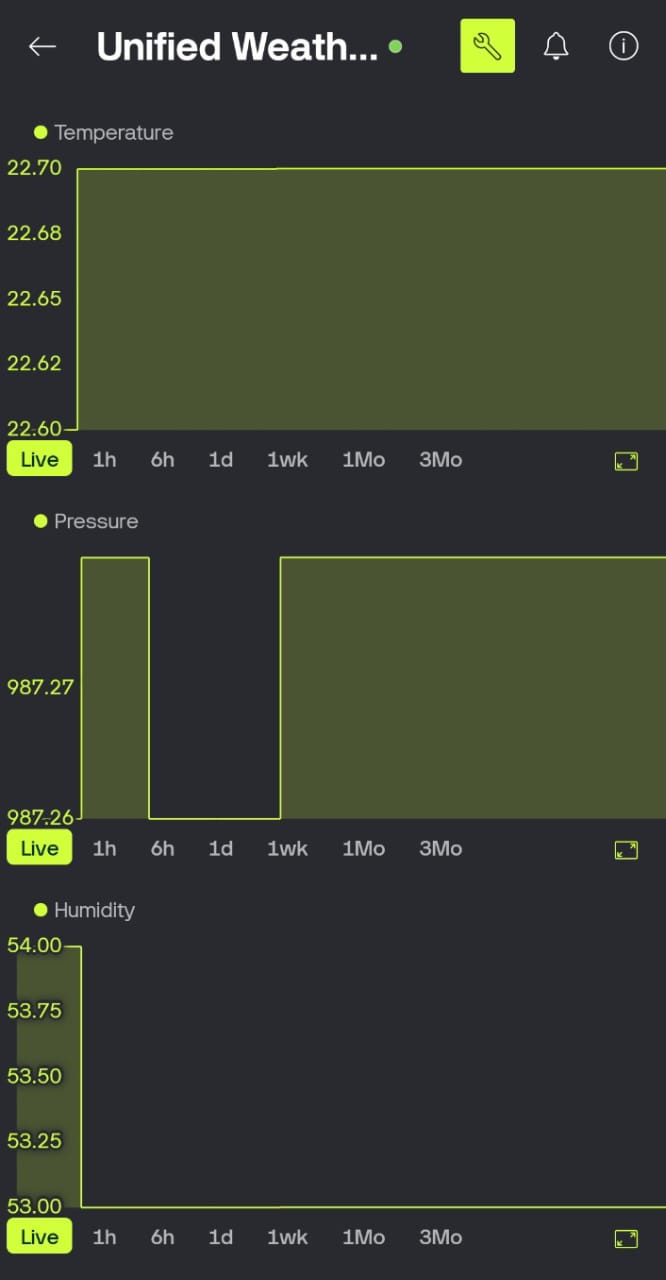
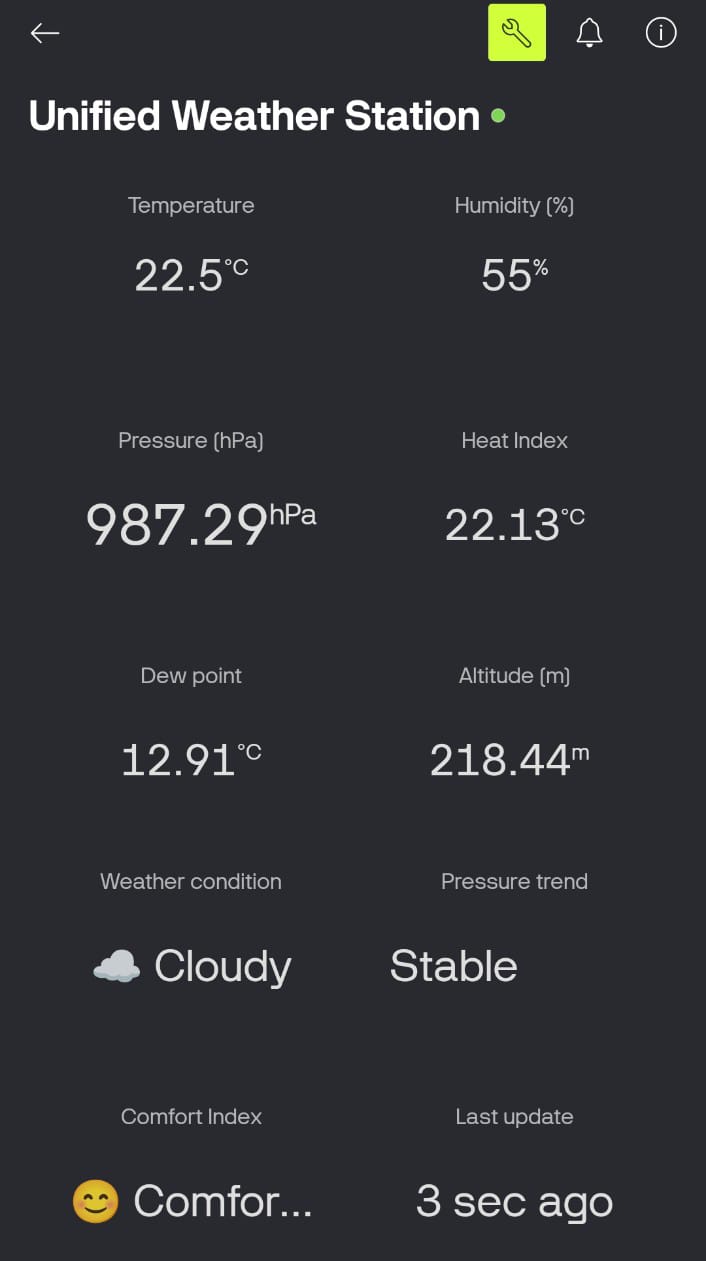
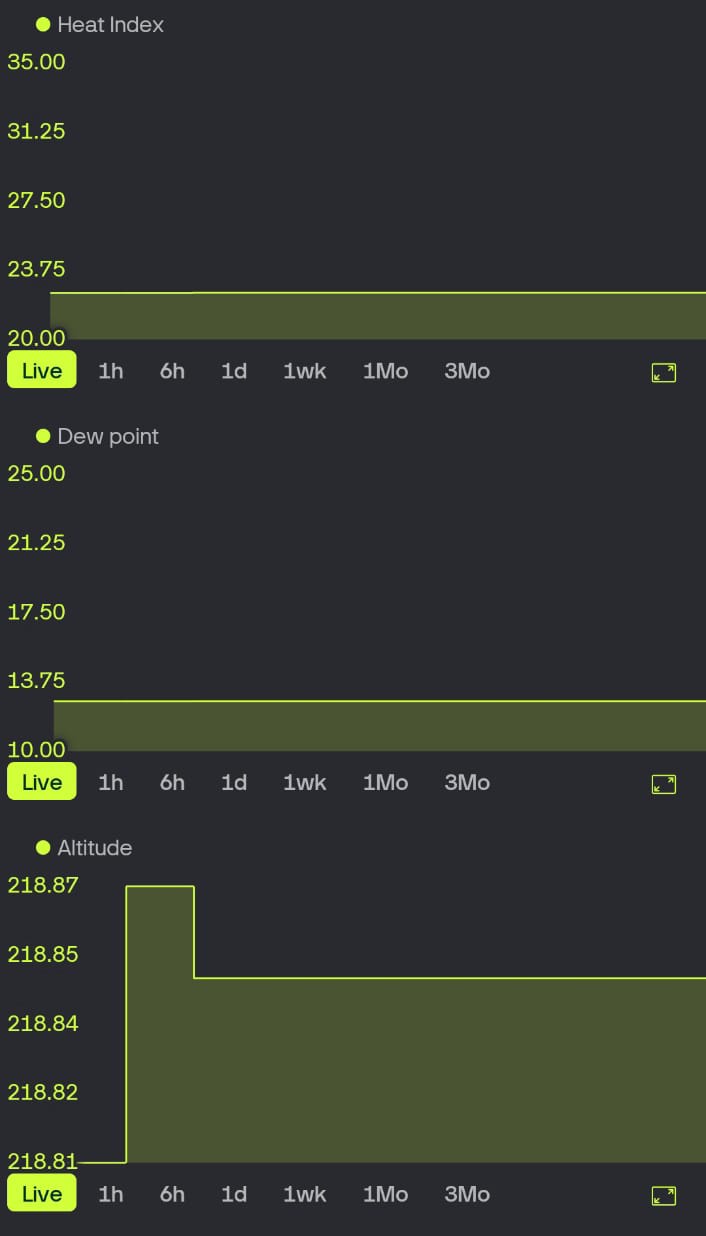
* **Temperature Graph:** Tracks variations in temperature across the observation interval, providing insight into thermal fluctuations.
* **Humidity Graph:** Displays moisture levels in the atmosphere, enabling correlation with temperature trends.
* **Pressure Graph:** Shows the slight oscillations in barometric pressure, which influence altitude calculations.
* **Heat Index Graph:** Represents perceived temperature based on humidity-corrected values.
* **Dew Point Graph:** Visualizes dew point stability, indicating moisture saturation levels.
* **Altitude Graph:** Displays altitude estimation calculated from the pressure sensor, showing minor natural fluctuations.

These graphical widgets enable trend identification, anomaly detection, and comparative analysis across multiple environmental parameters. The smooth and continuous data plots confirm that the ESP32 has been transmitting sensor readings without interruption.

The Blynk dashboard serves as a comprehensive cloud interface where real-time data, computed metrics, and temporal graphs are visualized simultaneously. Through this dashboard, the system provides continuous remote access to environmental information, fulfilling the goal of a fully functional IoT-based weather monitoring solution.

**4.3 App based results snapshots**

The figures presented below correspond to the mobile-application interface of the IoT-Based Weather Monitoring System, accessed through the Blynk IoT app. The mobile application displays real-time atmospheric parameters received from the ESP32 microcontroller and provides a compact, user-friendly view suitable for on-the-go monitoring. All readings are updated automatically through virtual pin communication, ensuring that the app stays synchronised with live sensor data.

** **

*Fig 25: Mobile Dashboard & Graphs*

## **A. Mobile Dashboard – Real-Time Measurements**

The first screenshot illustrates the primary mobile interface titled *Unified Weather Station*. A clean, minimalistic layout has been used to display essential weather parameters. Each metric is updated instantly as new sensor readings are received.

* **Temperature (22.5°C):** The measured temperature is displayed prominently, allowing immediate observation of thermal conditions.
* **Humidity (55%):** The relative humidity level is shown, providing insight into atmospheric moisture.
* **Pressure (987.29 hPa):** Barometric pressure is reported directly from the BMP180 sensor and is used for further atmospheric interpretation.
* **Heat Index (22.13°C):** The perceived temperature is computed automatically using temperature–humidity relationships.
* **Dew Point (12.91°C):** Dew point is calculated in real time using sensor readings, indicating moisture saturation characteristics.
* **Altitude (218.44 m):** The altitude estimation derived from pressure variation is presented.
* **Weather Condition:** A qualitative description (*Cloudy*) is generated based on combined sensor values.
* **Pressure Trend:** A stability indicator (*Stable*) is shown to display current barometric behaviour.
* **Comfort Index:** An emoji-based comfort indicator is displayed to summarise environmental suitability for human comfort.
* **Last Update:** The timestamp (*3 sec ago*) confirms that the mobile app remains actively connected to the ESP32.

This interface offers a complete and easily interpretable summary of environmental conditions, optimised for mobile viewing.

**B. Mobile Graphs – Time-Series Behaviour of Parameters**

The second set of snapshots presents the time-series graphs available within the Blynk app. Each graph is updated live, enabling the user to observe how weather parameters evolve over time.

### **1. Temperature Graph**

A stable temperature curve is shown over the selected time interval. Small-scale variations are captured smoothly, confirming consistent sensor performance.

### **2. Pressure Graph**

The pressure graph displays minor fluctuations between 987.26 and 987.29 hPa. These variations correspond to natural atmospheric behaviour and influence the altitude estimation.

### **3. Humidity Graph**

The humidity trend is shown between 53% and 54%. The narrow range indicates stable environmental moisture during the measurement period.

### **4. Heat Index Graph**

The heat index plot displays a relatively flat line, confirming that perceived temperature closely matches the measured temperature under stable humidity conditions.

### **5. Dew Point Graph**

A dew-point value of around 12–13°C is maintained across the observation window. This indicates consistent air moisture saturation and correlates with temperature and humidity stability.

### **6. Altitude Graph**

Small altitude oscillations are visible as a result of micro-variations in atmospheric pressure. The graph demonstrates that altitude estimation is responsive and continuously updated.

The mobile application provides a compact and real-time representation of all environmental parameters measured by the system. The combination of tile-based displays and live graphs ensures that users can view instantaneous values as well as trend behaviour directly from their smartphones. The app-based results confirm that the ESP32 has been transmitting data reliably and that the system is capable of providing continuous, accessible, and user-friendly weather monitoring.

**Table 1: Comparison of proposed results with earlier works**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Ref** | **Feature-1 (Data Parameters)** | **Feature-2 (Cloud Support)** | **Predictive Analytics** | **Alert Mechanism** | **Historical Logging** | **Sensor Diversity** | **App**  **Display** |
| **Our Work** | 4 parameters (Temp, Humidity, Pressure, Dew Point) | Yes (Blynk IoT Cloud) | Yes – Polynomial Regression | Yes – Threshold-based Alerts | Yes – Auto Logging | Good (DHT11 + BMP180) | Yes |
| **[1]** Stoyanov et al. | 3 parameters | Yes (ThingSpeak) | No | No | Limited | Basic | No |
| **[4]** Bhandari et al. | 2–3 parameters | Yes | No | No | Limited | Moderate | No |
| **[7]** Bindal et al. | 3+ parameters | Yes (Blynk) | No | No | No historical analytics | Good | No |
| **[10]** Mohamed et al. | Temp + Pressure | Yes (AWS IoT) | No | No | Yes | Limited | No |
| **[14]** Al-Hassan & Reza | Few parameters | Partial Cloud | TinyML only (limited) | No | Limited | Limited sensors | No |

1. **CONCLUSION**

The development of the IoT-Based Weather Monitoring System has demonstrated that essential atmospheric parameters can be measured, processed, and communicated efficiently using a compact and low-cost embedded design. Continuous observation of temperature, humidity, pressure, and dew point has been achieved through the combined functioning of the DHT11 and BMP180 sensors, while the ESP32 microcontroller has ensured streamlined data handling and wireless connectivity. Through cloud integration, real-time environmental information has been made accessible to users from any location, and the system has operated reliably during repeated test cycles. The results obtained throughout the implementation indicate that stable sensor performance, consistent network communication, and accurate visualisation can be maintained without the complexity or expense associated with conventional meteorological setups.

The applicability of the system extends across several domains where real-time climatic awareness is required. In agriculture, the system can support more informed irrigation scheduling, crop protection planning, and microclimate assessment. In residential environments, indoor–outdoor comfort monitoring and energy-efficient ventilation decisions can be assisted through continuous weather updates. Industrial units can integrate the system into safety management practices, where humidity, temperature, and pressure values influence equipment performance. Research and educational institutions can utilise the system as a tool for data collection, experimentation, and environmental analysis. The cloud-based dashboard further enables the use of long-term datasets for pattern identification and regional climate interpretation.

Future enhancements can extend the capabilities of this system even further. Additional sensors, such as those measuring rain intensity, wind speed, particulate matter, or air quality, can be incorporated to create a more comprehensive environmental station. Edge-based machine-learning models may be integrated for forecasting and anomaly detection, reducing dependence on external servers. Solar-powered operation, low-power modes, and long-range communication technologies such as LoRaWAN may be adopted for deployment in remote or agricultural landscapes. Through these improvements, the system can evolve into a robust, intelligent, and scalable platform suitable for widespread real-world use.

Github Link for the entire project (Predictive and forecasting in IOT based Weather Monitoring System): <https://github.com/Amanarun2907/IOT-Based-Weather-Monitoring-System>

**REFERENCES**

1. S. Stoyanov et al., “Weather Monitoring System Using IoT-Based DIY Automatic Weather Station,” IEEE, 2024.
2. M. Osama et al., “AI-Based Weather Monitoring System Using IoT Interface,” IEEE, 2025.
3. A. Akilan and P. Baalamurugan, “Enhanced IoT-Based Weather Forecasting and Field Monitoring System Utilizing Multiple CNN Models,” IEEE, 2023.
4. B. Bhandari et al., “An IoT-Based Weather Monitoring System for Smart Agriculture,” IEEE, 2022.
5. J. Hilda et al., “IoT-Based Smart Weather Monitoring System,” IEEE, 2025.
6. A. Mahato et al., “An IoT-Based Real-Time Flood and Weather Monitoring System,” IEEE, 2024.
7. R. Bindal et al., “NodeMCU-Based Weather Monitoring System,” IEEE, 2023.
8. M. M. Hassain et al., “IoT-Based Smart Economic Zone Weather Monitoring System,” IEEE, 2024.
9. IoT-Based Smart Weather Monitoring System for Poultry Farm, IEEE, 2023.
10. A. Mohamed et al., “Cloud-Based Weather Condition Monitoring System Using ESP8266 and AWS,” IEEE, 2023.
11. M. Hassain et al., “IoT-Based Economic Zone Weather Monitoring System,” IEEE, 2024.
12. S. Banara et al., “IoT-Based Weather Monitoring System for Smart Cities: A Comprehensive Review,” IEEE, 2022.
13. R. K. Singh et al., “IoT-Enabled Environmental Monitoring System Using ESP32 and Blynk Cloud,” IEEE, 2023.
14. M. Al-Hassan and T. Reza, “Low-Cost IoT Weather Observation Node with Edge-Computing Based Prediction,” IEEE, 2024.
15. S. Verma et al., “A Scalable IoT Sensor Grid for Urban Weather Analytics and Microclimate Mapping,” IEEE, 2022.