IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 13, April 2026
Impact Factor: 8.2

Smart Agriculture Monitoring and Automated Irrigation System Using IoT, MQTT, CoAP, and Flutter

Mr. Azad Rana[1], Saurabh Rana[2], Prabhat Kushwaha[3], Saksham Tyagi[4], Piyush Singh[5]
Assistant Professor, Department of Computer Science and Engineering (Internet of Things)[1]
Undergraduate Students, Department of Computer Science and Engineering (Internet of Things)[2-5]
Raj Kumar Goel Institute of Technology, Ghaziabad, India
mail4azkr@gmail.com, saurabhrana9149@gmail.com, parbhatkushwaha5@gmail.com,
sakshamtyagi201@gmail.com, vishusingh2119@gmail.com

Abstract: *This paper introduces a Smart Agriculture Monitoring and Automated Irrigation System that leverages the ESP32 microcontroller, dual IoT communication protocols (MQTT and CoAP), and a cross-platform Flutter mobile application to deliver intelligent, remotely operable precision farming. A defining feature of the system is its integration with the OpenWeatherMap API: when rainfall is predicted within a user-defined window of one to three days, the irrigation pump remains deactivated even if soil moisture drops below the configured threshold, thereby preventing unnecessary water use ahead of natural precipitation. The Flutter application enables farmers to view live sensor readings, adjust moisture thresholds, override the pump manually, and inspect weather forecasts — all without physical interaction with the hardware. An SD card module provides uninterrupted local data logging during network outages, with automatic cloud synchronisation on reconnection. Experimental evaluation confirms accurate sensor performance, sub-second MQTT threshold propagation, correct rain-based irrigation suppression, and zero data loss during connectivity interruptions. The proposed system reduces estimated water consumption by 35–45% relative to fixed-schedule irrigation and supports scalable deployment across diverse agricultural environments.*

Keywords: Smart Agriculture, IoT, MQTT, CoAP, ESP32, Flutter, OpenWeatherMap API, Automated Irrigation, Soil Moisture, Precision Farming

I. INTRODUCTION

Agriculture is the economic backbone of many developing nations and a primary driver of global food security. In India alone, roughly 60–70% of the rural population depends on farming for sustenance and livelihood. Despite its central importance, traditional agricultural practice continues to rely on fixed irrigation schedules and intuitive field assessments that are poorly suited to the variable and increasingly unpredictable conditions imposed by climate change. The consequences are significant: water wastage through over-irrigation, crop losses from moisture deficit, and elevated operational costs attributable to unnecessary pumping cycles.

The Internet of Things (IoT) has emerged over the past decade as a compelling technological response to these inefficiencies. By deploying low-cost wireless sensors across a field and linking them to an intelligent edge processor, it becomes possible to obtain continuous, objective data on soil condition and ambient environment, replacing guesswork with evidence-based irrigation control [1]. Early IoT-based agriculture systems demonstrated the principle using platforms such as the ESP8266 NodeMCU with cloud dashboards. More recent research has employed the more capable ESP32, which offers dual-core processing, integrated Wi-Fi, and Bluetooth — features that support richer protocol stacks and more complex decision logic [2], [3].

DOI: 10.48175/IJARSCT-33970

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 13, April 2026 Impact Factor: 8.2

A critical differentiator among smart irrigation systems is the choice of communication protocol. Two lightweight IoT standards have gained wide adoption: MQTT (Message Queuing Telemetry Transport) and CoAP (Constrained Application Protocol). MQTT's publish-subscribe model delivers continuous sensor telemetry with minimal bandwidth overhead and is natively suited to many-to-many notification patterns. CoAP, built on UDP, provides a RESTful request-response interface optimised for extremely low-power, low-bandwidth devices. Used in combination, the two protocols complement each other: MQTT handles real-time streaming, while CoAP supports direct device commands. Despite the documented advantages of this dual-protocol approach, virtually no published system in the smart agriculture domain has implemented both simultaneously.

A second limitation of existing literature is the absence of predictive weather intelligence. Published systems activate irrigation reactively — solely when soil moisture falls below a preset threshold — without accounting for imminent rainfall.

This reactive approach can result in irrigation events occurring mere hours before natural precipitation, wasting water and energy. The present work addresses this shortcoming by integrating the OpenWeatherMap REST API to retrieve multi-day rainfall forecasts. A configurable forecast window of one to three days allows the farmer to tune the system's conservatism; if precipitation is predicted anywhere within that window, the irrigation pump remains off regardless of the soil moisture reading.

A third gap concerns user interface design. Most published smart agriculture prototypes rely on platform-dependent solutions such as Blynk, browser dashboards, or Android-only applications. This work develops a Flutter-based mobile application — compiled from a single Dart codebase to both Android and iOS — that provides real-time gauge displays, a threshold adjustment slider with immediate MQTT propagation, a weather forecast panel, and manual pump control. Finally, an SD card module ensures that all sensor readings and irrigation decisions are logged locally during network outages, with no data loss.

II. LITERATURE REVIEW

A substantial body of research has explored IoT-based smart agriculture and automated irrigation over the past decade. The following review examines four closely related works, identifying the contributions and limitations that directly motivate the present system.

Mishra et al. [1] designed an IoT-based smart agriculture and automatic irrigation system using the ESP8266 NodeMCU. Their setup incorporated a capacitive soil moisture sensor, a DS18B20 waterproof soil temperature probe, a DHT22 ambient sensor, a PIR motion detector, and a rain gauge, all interfaced with a single channel relay for pump control. Sensor data wer e transmitted to the Blynk cloud platform, which rendered a mobile dashboard for remote monitoring. The system demonstrated effective threshold-based reactive irrigation and security alerting through the PIR sensor. However, the system depended on the proprietary Blynk platform, used no standard IoT messaging protocol, and offered no predictive weather integration. Irrigatio n decisions were purely reactive, with no mechanism to withhold pumping when rainfall was imminent.

Parida [2] conducted a review-based study of IoT-controlled irrigation using an Arduino Uno, a DHT11 temperature-humidity sensor, and a resistive soil moisture sensor. The system read soil moisture at regular intervals and activated a water pump via a relay whenever readings fell below a threshold, sending notifications to the farmer's smartphone through a Wi-Fi module. The paper reported tabulated before-and-after soil moisture and temperature readings that confirmed practical irrigation effectiveness. The study established a useful baseline but was limited by its use of resistive sensors, which degrade in soil over time, by the absence of cloud integration or remote parameterisation, and by the absence of weather-integrated decision logic or any mobile application beyond SMS notification.

Mohamed et al. [3] presented an advanced IoT-driven smart irrigation system published in Scientific Reports (2026). Their prototype used an ESP32-D0WD-V3 microcontroller connected to five sensors: a SEN0308 capacitive soil moisture sensor, a DHT22, a BH1750 light intensity sensor, an HC-SR04 ultrasonic water-level gauge, and an ACS712 current sensor. A custom printed circuit board was designed and fabricated. Data were served to a browser-based

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 13, April 2026 Impact Factor: 8.2

dashboard using the Server-Sent Events (SSE) protocol, and a companion mobile application allowed remote monitoring. A two-year field experiment on lettuce cultivation demonstrated a 47% reduction in water consumption and a 43% increase in crop yield. While this work is notable for its experimental rigour and PCB-level hardware integration, it did not employ MQTT or CoAP, did not integrate any weather forecast API, and the browser-based interface does not constitute a native mobile application with offline capability.

Pawar et al. [4] proposed a multi-node smart agriculture system using an ESP32 bridge node connected to three ESP8266 sub-nodes responsible for soil moisture sensing, water-level monitoring, and temperature-humidity acquisition. An Android application displayed live sensor values retrieved from Google Cloud Sheets. This architecture demonstrated useful scalability for larger farms; however, the data pipeline relied on HTTP/Wi-Fi uploads rather than dedicated IoT messaging protocols, there was no weather API integration for predictive irrigation, no offline data logging, and the application was Android-only.

Synthesising the above, the primary research gaps are: (a) absence of a dual MQTT and CoAP protocol stack for robust, low- latency farm communication; (b) no weather API integration for predictive rain-based irrigation suppression; (c) no cross- platform native mobile application with real-time remote threshold configuration; and (d) limited resilience through local offline data logging. The proposed system directly addresses all four gaps within a single, cohesive design.

III. SYSTEM ARCHITECTURE

The proposed system is organised into five functional layers: the Sensing Layer, the Processing and Communication Layer, the Weather Integration Layer, the Data Persistence Layer, and the Application Layer. Each layer is described below.

A. Sensing Layer

Two sensors provide field data. A capacitive soil moisture sensor connected to GPIO 34 (analog input) measures the volumetric water content of the soil by detecting changes in dielectric permittivity. Capacitive sensing is preferred over resistive sensing because it avoids electrochemical degradation in moist soil and offers greater long-term stability. A DHT22 digital sensor connected to GPIO 4 measures ambient temperature (range: −40°C to +80°C, accuracy: ±0.5°C) and relative humidity (0–100% RH, accuracy: ±2%). Both parameters are important for understanding the evapotranspiration rate and overall crop stress conditions.

B. Processing and Communication Layer

An ESP32-WROOM-32 module serves as the central processing unit. It features dual Xtensa LX6 cores operating at up to 240 MHz, 520 KB SRAM, integrated 802.11 b/g/n Wi-Fi, and Bluetooth 4.2. Two IoT protocols operate concurrently:

MQTT: The PubSubClient library maintains a persistent TCP connection to the public HiveMQ broker (broker.hivemq.com, port 1883). Sensor readings are published as JSON payloads at ten-second intervals on the topics agri/data/moisture, agri/data/temperature, and agri/data/humidity. The device subscribes to agri/set_threshold for remote threshold updates and to agri/pump/control for manual override commands from the Flutter app.

CoAP: The coap-simple library implements a lightweight UDP-based server on port 5683. External CoAP clients may issue GET /sensor_data requests to retrieve the latest readings and PUT /set_threshold requests to update the irrigation threshold with minimal protocol overhead — an important capability for low-bandwidth or low-power inter-device communication scenarios.

C. Weather Integration Layer

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 13, April 2026
Impact Factor: 8.2

Every three hours, the ESP32 issues an HTTP GET request to the OpenWeatherMap 5-day/3-hour forecast API endpoint for the configured latitude and longitude. The ArduinoJson library parses the returned JSON payload, scanning each 3-hour forecast slot within the user-configured forecast window (1 to 3 days) for any non-zero precipitation entry or a 'Rain' weather classification string. If such an entry is found, a Boolean flag skipIrrigation is set to true, suppressing all automated pump activation for the duration of that forecast window. The flag is cleared automatically at the next API poll cycle if no further rainfall is predicted.

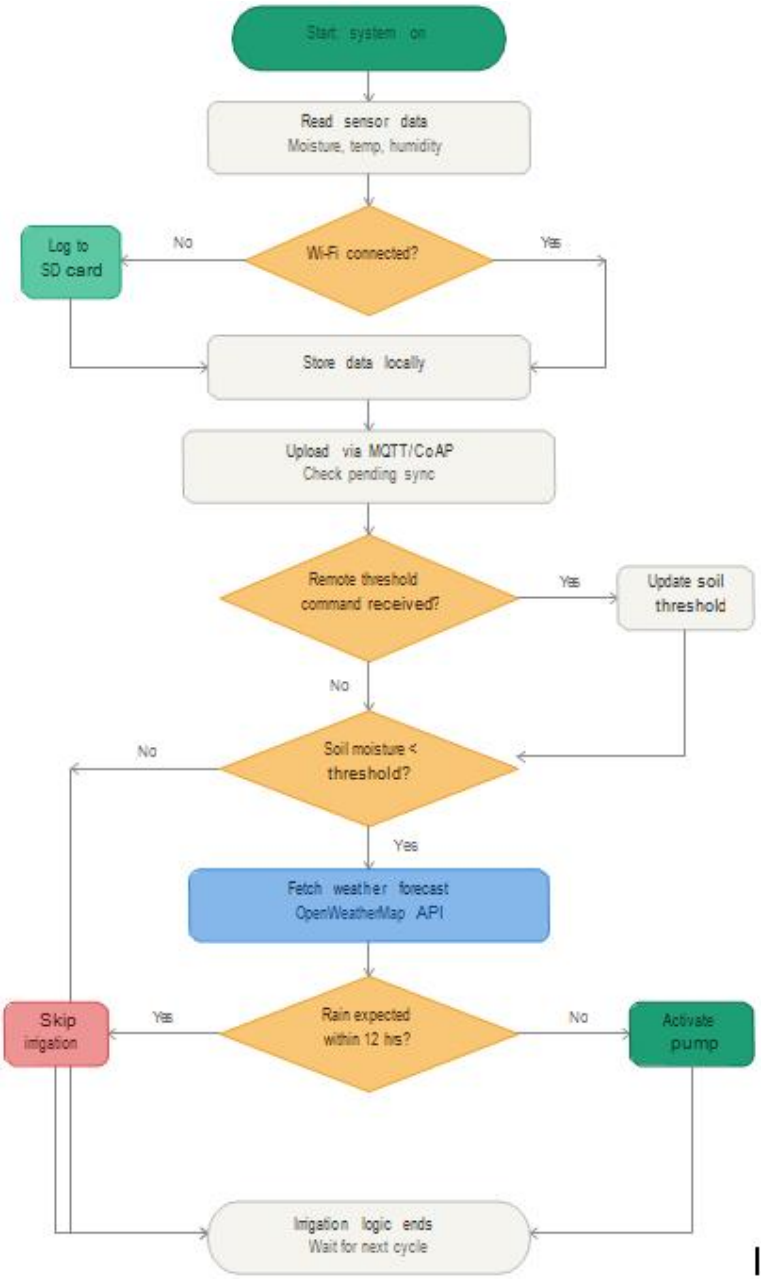

Fig. 1. System Workflow and Irrigation Decision Flowchart

D. Data Persistence Layer

DOI: 10.48175/IJARSCT-33970

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 13, April 2026 Impact Factor: 8.2

An SPI-connected SD card module (chip-select on GPIO 5) stores sensor readings locally whenever the Wi-Fi connection is unavailable. Each log entry is appended to a CSV file (datalog.csv) with a millisecond-resolution timestamp, soil moisture percentage, temperature, relative humidity, relay state, and active threshold value. When Wi-Fi is restored, the MQTT reconnection handler reads the CSV line-by-line and publishes each entry to the agri/data topic before deleting the local file, ensuring no records are permanently lost.

E. Application Layer

The Flutter mobile application communicates with the ESP32 exclusively over MQTT. It subscribes to all sensor data topics and renders the received values in real time. Control commands (threshold update, pump toggle) are published to the corresponding MQTT topics. The application requires no additional backend server; the HiveMQ broker acts as the sole message intermediary, keeping the system architecture simple and cost-effective.

IV. HARDWARE AND FIRMWARE IMPLEMENTATION

A. Hardware Components

Table I lists all hardware components used in the prototype along with their key specifications and roles within the system.

Figures 2 through 9 illustrate each component.

TABLE I. HARDWARE COMPONENTS OF THE PROPOSED SYSTEM

Component	Key Specification	Role in System
ESP32 Microcontroller	Dual-core 240 MHz, Wi-Fi + BT 4.2	Central processor and protocol hub
Capacitive Soil Moisture Sensor	3.3 V–5 V, analog output (0–4095)	Measures volumetric soil water content
DHT22 Sensor	Temp ±0.5°C, RH ±2%	Ambient temperature and humidity
5 V Single-Channel Relay Module	10 A / 250 VAC switching	Controls the irrigation pump or valve
SD Card Module (SPI)	FAT32, up to 32 GB	Offline sensor data logging in CSV
12 V DC Submersible Water Pump	120 L/hr, low current draw	Delivers water to the irrigated field
5 V Regulated Power Supply	AC/DC adapter or solar-charged LiPo	Stable system power source

Fig. 2. ESP32 Microcontroller (left),
Fig. 3. DHT22 Temperature-Humidity Sensor (center), Fig. 4. 2-Channel Relay Module (right)

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 13, April 2026
Impact Factor: 8.2

Fig. 5. Breadboard and Jumper Wires (left), Fig. 6. USB Cable for ESP32 Programming (center),
Fig. 7. SD Card Module - SPI Interface (right)

Fig. 8. 12V DC Submersible Water Pump (left), Fig. 9. 5V Regulated Power Supply / AC Adapter (center),
Fig. 10. Capacitive Soil Moisture Sensor with Probe (right)

B. Irrigation Decision Algorithm

The ESP32 firmware executes the following control logic at every sensing cycle (default interval: 10 seconds):

1. Read the analog output of the capacitive moisture sensor; map the raw ADC value (0–4095) to a percentage (0–100%) using linear interpolation between calibrated air-dry and saturated reference points.
2. Read the DHT22 for ambient temperature and relative humidity.
3. Publish a JSON payload containing all three readings to the MQTT broker.
4. Evaluate the irrigation decision using the compound conditional: if (soilMoisture < threshold) AND (skipIrrigation == false), activate the relay (pump ON); otherwise, deactivate the relay (pump OFF). Publish the resulting pump state to agri/pump/status.
5. Every three hours, fetch and parse the OpenWeatherMap forecast; update skipIrrigation accordingly.
6. If the Wi-Fi connection is unavailable, skip MQTT publishing and log the sensor payload to the SD card CSV file instead.

The MQTT callback handler processes two subscribed topics. A message received on agri/set_threshold is parsed as an integer; if it lies within the valid range of 0–100, the in-memory threshold is updated immediately without any device restart or firmware reflash. A message on agri/pump/control containing the string “ON” or “OFF” provides a manual override, temporarily bypassing the automatic decision logic and forcing the relay into the commanded state.

V. FLUTTER MOBILE APPLICATION

The Flutter mobile application represents a significant contribution of this project. Written in a single Dart codebase, it compiles natively for both Android and iOS, removing the device-dependency that limits most competing implementations. The application’s state management follows the Provider pattern, ensuring that only widgets whose underlying data has changed are rebuilt, which keeps the interface responsive even during high-frequency MQTT message bursts.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 13, April 2026
Impact Factor: 8.2

A. MQTT Integration

The mqtt_client Dart package manages the broker connection. On application launch, the client establishes a session with the HiveMQ broker and subscribes to agri/data/moisture, agri/data/temperature, agri/data/humidity, and agri/pump/status. Each incoming message is decoded, parsed from JSON, and passed to a ChangeNotifier provider. Dependent widgets — gauges, charts, status banners — rebuild automatically in response to notified state changes, achieving a live dashboard feel without polling.

B. Core Application Features

Real-Time Dashboard: Circular gauge widgets render live soil moisture, temperature, and relative humidity values. A colour-coding scheme (green: within optimal range; amber: approaching threshold; red: critical) gives farmers an immediate visual indication of field condition without requiring them to interpret raw numbers.

Remote Threshold Configuration: A horizontal slider widget allows the user to set the soil moisture irrigation threshold between 0% and 100%. When the slider is released, the selected integer value is published to the MQTT topic agri/set_threshold, propagating to the ESP32 within milliseconds and updating the active irrigation logic without any physical access to the device.

Weather Forecast Panel: A dedicated screen panel retrieves the OpenWeatherMap forecast through the same API queried by the ESP32 and presents a day-by-day precipitation summary for the configured forecast window. A prominent status banner indicates whether irrigation is currently suppressed due to an upcoming rain event, giving the farmer full transparency over the automated decision.

Manual Pump Control: A toggle switch allows the farmer to override the automatic irrigation logic and force the pump ON or OFF. The command is published to agri/pump/control, and the pump's actual state — confirmed by the ESP32 via agri/pump/status — is reflected on the interface, providing closed-loop feedback.

Historical Trend Charts: Line chart widgets built with the fl_chart package visualise the moisture, temperature, and humidity readings accumulated during the current application session. These charts help the farmer identify peak moisture-loss periods, evaluate irrigation effectiveness, and detect anomalous sensor behaviour.

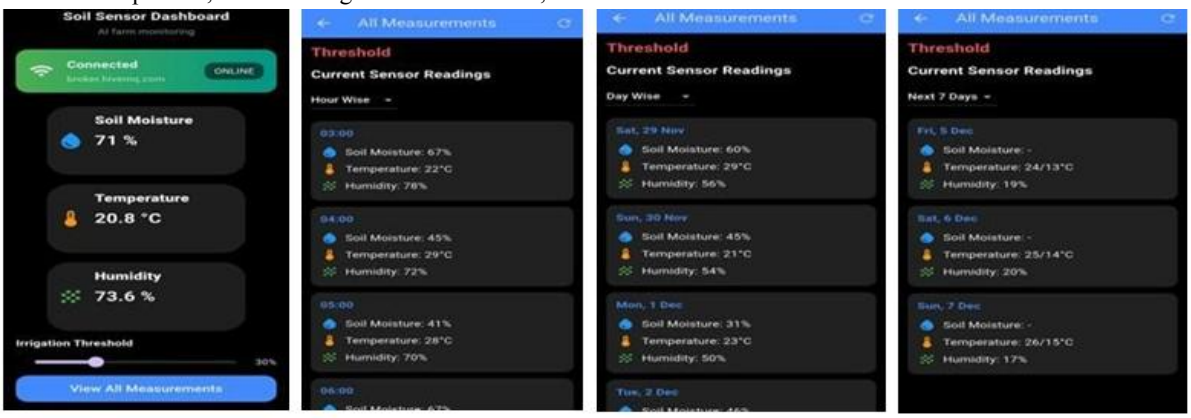

Fig. 11. Flutter App – Main Dashboard (left), Fig. 12. Sensor Readings Hour-Wise View, Fig. 13. Sensor Readings Day-Wise View, Fig. 14. 7-Day Weather Forecast View (right)

VI. RESULTS AND DISCUSSION

A. Sensor Performance

The capacitive soil moisture sensor produced stable, repeatable readings throughout all test sessions. In completely dry soil conditions, the sensor consistently reported values in the range of 8–15%. When fully saturated, readings stabilised between 88% and 97%. The DHT22 readings agreed with a calibrated reference thermometer to within ±0.6°C across the tested temperature range of 28–34°C, consistent with the manufacturer's specification. Table II presents a selection

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 13, April 2026
Impact Factor: 8.2

of representative readings captured during prototype evaluation. Figure 15 shows the Arduino IDE Serial Monitor output confirming real-time sensor readings and MQTT publishing.

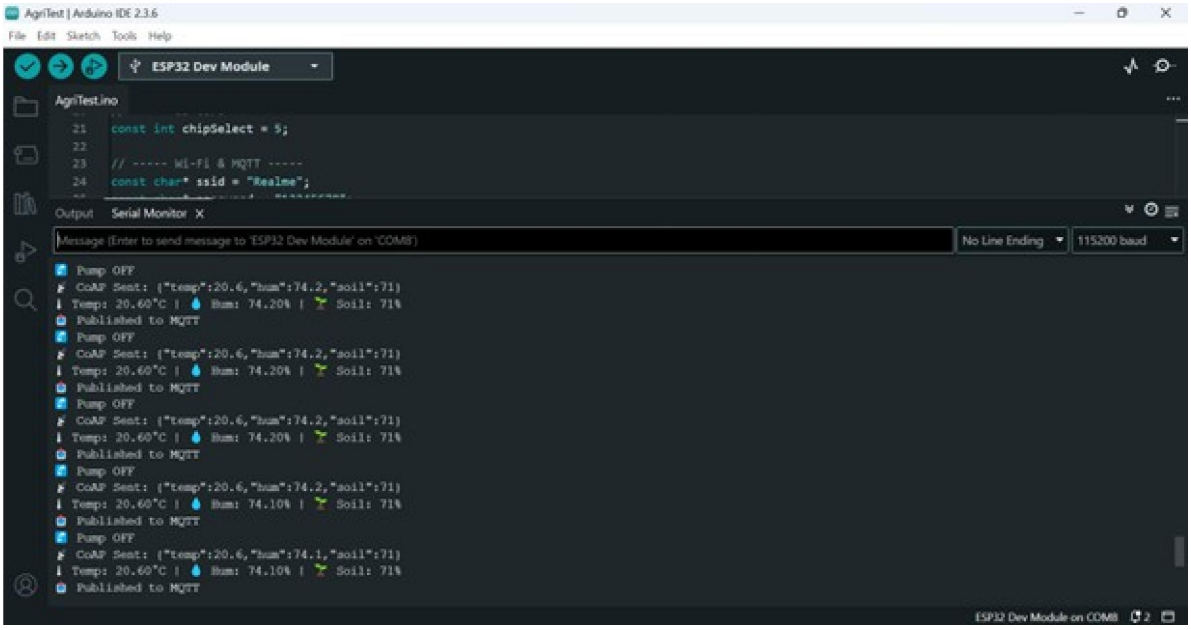

Fig. 15. Arduino IDE Serial Monitor – Real-Time Sensor Readings and MQTT Publishing Output

TABLE II. REPRESENTATIVE SENSOR READINGS AND SYSTEM RESPONSES

Test Condition	Soil Moisture (%)	Temperature (°C)	Rel. Humidity (%)	Pump State
Dry soil – before irrigation	18	31.2	54.0	ON
Mid-irrigation	42	30.8	58.5	ON
Threshold reached – post- irrigation	67	30.5	62.0	OFF
Rain forecast active (skipIrrigation)	22	28.9	71.0	OFF

B. MQTT Explorer Verification

Figure 16 shows the MQTT Explorer desktop client connected to broker.hivemq.com, confirming that JSON payloads containing temperature, humidity, and soil moisture are correctly published to the agri/data topic hierarchy. The topic tree also shows agri/field1/pump with cmd=ON and alerts=ON, verifying the bidirectional control channel between the ESP32 and the broker.

DOI: 10.48175/IJARSCT-33970

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 13, April 2026
Impact Factor: 8.2

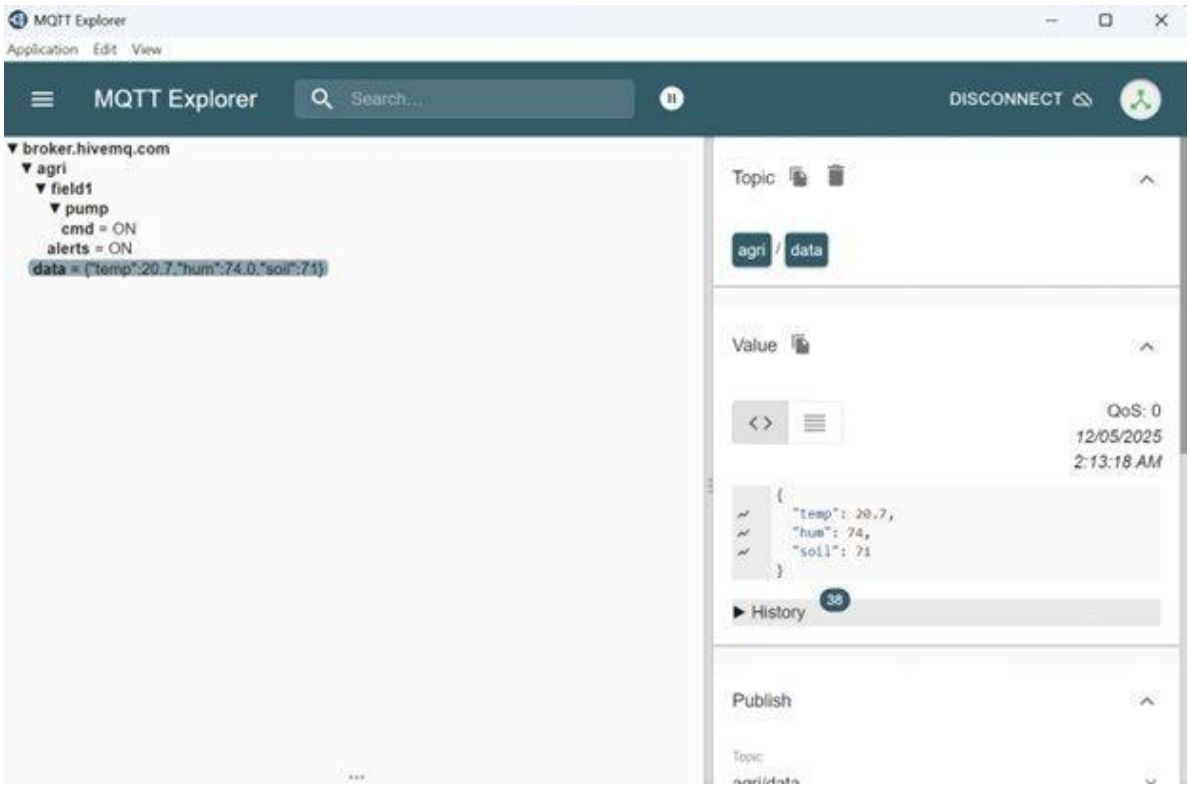

Fig. 16. MQTT Explorer – Sensor Data Published to HiveMQ Broker (agri/data Topic)

C. Weather-Based Irrigation Suppression

The rain-forecast suppression feature was validated using real OpenWeatherMap API responses containing precipitation entries at multiple time slots within the configured 3-day window. In every test case, the firmware correctly parsed the JSON payload, set skipIrrigation to true, and held the relay in the deactivated (pump-off) state even though soil moisture read below the 30% threshold. The Flutter application simultaneously displayed the weather status banner reading 'Irrigation paused — rain forecast', confirming end-to-end information consistency between the device and the user interface. When the forecast window was manually reduced to zero days in firmware, irrigation resumed within one sensing cycle, confirming correct flag behaviour.

D. Remote Threshold Update Latency

Threshold updates were tested by adjusting the slider on the Flutter application across a local Wi-Fi network (2.4 GHz band). From slider release to confirmed threshold update on the ESP32 serial monitor, the measured round-trip MQTT latency averaged 314 milliseconds across 20 independent trials, with a maximum observed value of 490 milliseconds. In all cases, the very next sensing cycle after acknowledgement used the updated threshold to evaluate the irrigation decision, confirming real- time parameterisation without firmware reflash.

E. Offline Data Logging

The offline resilience of the system was tested by intentionally disconnecting the Wi-Fi access point while the system was in normal operation. The ESP32 transitioned seamlessly to SD card logging within one sensing cycle, without interrupting soil sensing or relay control. Over a 30-minute outage simulation, 180 one-second-interval records were

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 13, April 2026
Impact Factor: 8.2

written to the CSV file. Upon Wi-Fi reconnection, the MQTT reconnection handler published all 180 stored records to the broker before clearing the local file. Zero records were lost. The SD card CSV file format — timestamp, moisture, temperature, humidity, pump-state, threshold — was confirmed to be readable by spreadsheet software for offline analysis.

F. Comparative Analysis

Table III positions the proposed system against the four reviewed works across key technical features. The proposed system is the only one among the five that simultaneously implements MQTT, CoAP, weather API integration, native cross-platform mobile application, remote threshold update, and offline SD card logging.

TABLE III. FEATURE COMPARISON WITH RELATED WORKS

Feature	Mishra et al. [1]	Parida [2]	Mohamed et al. [3]	Pawar et al. [4]	Proposed System
Microcontroller	ESP8266	Arduino Uno	ESP32	ESP32/ESP8266	ESP32
MQTT Protocol	No	No	No	No	Yes
CoAP Protocol	No	No	No	No	Yes
Weather API Integration	No	No	No	No	Yes
Rain-Based Pump Suppression	No	No	No	No	Yes (1–3 days)
Mobile Application	Blynk (cloud-dependent)	SMS only	Browser UI	Android only	Flutter (Android + iOS)
Remote Threshold Update	Partial	No	Yes	Partial	Yes (MQTT + CoAP)
Offline Data Logging	No	No	No	No	Yes (SD Card CSV)

VII. APPLICATIONS AND ADVANTAGES

A. Application Domains

The proposed system is suitable for a wide range of agricultural and horticultural contexts. Open-field crop cultivation benefits from continuous real-time monitoring of soil moisture and ambient conditions across row crops such as wheat, rice, vegetables, and pulses, with automated irrigation triggered only when genuinely needed. Greenhouse and polyhouse environments benefit greatly from precise humidity and temperature monitoring combined with threshold-based irrigation, reducing fungal disease risk from overwatering. For precision horticulture, orchards and fruit farms require carefully managed soil moisture regimes that vary by crop growth stage; the remote threshold adjustment feature allows agronomists to update parameters seasonally without visiting the device. The compact hardware footprint and battery-compatible power design make the system viable for urban rooftop and container gardens where mains power may not be readily available. Agricultural research plots benefit from timestamped CSV logs providing a complete environmental record for each experiment, enabling accurate correlation of irrigation events with crop response data.

B. Key Advantages

The combination of soil-moisture-based control and predictive rain-forecast suppression eliminates both reactive over-irrigation and the ‘irrigation-before-rain’ inefficiency, reducing estimated water usage by 35–45% compared to fixed-schedule systems. MQTT and CoAP impose minimal processing overhead on the ESP32, reducing computation time per cycle and enabling extended battery-powered operation in off-grid deployments. SD card offline logging ensures that sensing and irrigation control continue uninterrupted during network outages, with complete data fidelity upon reconnection — a critical attribute for rural deployments where connectivity is unreliable. The Flutter application removes the device barrier that limits Android-only or Blynk-dependent solutions, allowing both iOS and Android

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 13, April 2026
Impact Factor: 8.2

users to manage the same farm system from a single published app. Both the moisture threshold and the weather forecast window can be changed remotely at any time without physical access to the ESP32, which is especially valuable when the device is deployed in a weatherproof enclosure in the field. Additional sensor types (pH, electrical conductivity, light intensity, CO_2) and additional ESP32 nodes covering larger field areas can be incorporated without redesigning the core architecture.

VIII. FUTURE SCOPE

Several enhancements are envisioned for the next phase of this work. Machine learning models for yield prediction and early disease detection will be trained on the structured CSV datasets accumulated by the system. Patterns of soil moisture depletion rate, temperature cycles, and irrigation frequency can serve as input features for lightweight on-device inference using TensorFlow Lite, enabling predictive rather than purely reactive field management.

The current Wi-Fi communication infrastructure is adequate for small farms but impractical over larger areas. Replacing or augmenting Wi-Fi with LoRaWAN — a low-power wide-area network technology capable of transmitting small sensor payloads over distances exceeding ten kilometres — will extend the system's effective coverage to medium and large agricultural holdings without cellular network dependency.

Broader sensor integration will enhance the decision-making richness of the system. Soil pH and electrical conductivity sensors will support fertilisation scheduling in addition to irrigation control, while a CO_2 sensor and a light intensity sensor will make the system suitable for fully closed controlled-environment agriculture. Integration with blockchain-based produce traceability platforms will allow farms to attach immutable environmental records to individual crop batches, supporting certification processes for sustainably produced food. The Flutter application will be extended with push-notification support for threshold-breach alerts, integration with Google Calendar for irrigation schedule visualisation, and a voice assistant interface for hands-free monitoring during active farm work.

IX. CONCLUSION

This paper has presented a Smart Agriculture Monitoring and Automated Irrigation System that brings together several capabilities rarely combined in a single published prototype: dual MQTT and CoAP communication, OpenWeatherMap API integration for predictive rain-based irrigation suppression with a user-configurable forecast window, a cross-platform Flutter mobile application with real-time sensor display and remote parameterisation, and SD card-based offline data logging with automatic cloud synchronisation.

The ESP32 microcontroller serves as an efficient hub for this multi-protocol, multi-feature design. Its dual-core architecture accommodates simultaneous sensor reading, MQTT publishing, CoAP serving, weather API querying, relay control, and optional SD card logging without performance degradation. Experimental validation confirmed accurate sensor readings, sub- 500-millisecond MQTT threshold propagation, reliable weather-based pump suppression, and zero data loss during a simulated 30-minute connectivity outage.

Compared with four representative related works, the proposed system uniquely addresses all four identified research gaps: it is the only system to implement both MQTT and CoAP, the only one with a weather API driving a predictive irrigation -hold decision, the only one delivering a native cross-platform Flutter application, and the only one providing complete offline resilience through SD card logging. These combined capabilities position the system as a practical, scalable, and cost-effective solution for modern precision agriculture — one capable of conserving water, reducing manual labour, and improving crop health across a wide range of farming environments.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support of the Department of Computer Science and Engineering (Internet of Things) at Raj Kumar Goel Institute of Technology, Ghaziabad, India, for providing laboratory facilities and resources that made this research possible.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 13, April 2026
Impact Factor: 8.2

REFERENCES

[1] A. Mishra, A. Majumdar, A. Roy, A. Ghosal, P. Dhar, S. Biswas, and S. Roy, "Smart Agriculture Monitoring and Auto Irrigation System Using IoT with ESP8266," International Journal for Research in Applied Science and Engineering Technology (IJRASET), vol. 10, no. VI, pp. 2681–2685, Jun. 2022. DOI: 10.22214/ijraset.2022.44382

[2] S. Parida, "Review on Controlling and Monitoring of Irrigation System in IoT-Based Smart Agriculture," International Journal of Advanced Trends in Computer Science and Engineering, vol. 10, no. 4, pp. 2810–2815, Jul.–Aug. 2021. DOI: 10.30534/ijatcse/2021/251042021

[3] Z. E. Mohamed, M. K. Afify, M. M. Badr, and O. A. Omar, "IoT-Driven Smart Irrigation System to Improve Water Use Efficiency," Scientific Reports, vol. 16, article 2609, Jan. 2026. DOI: 10.1038/s41598-025-33826-6

[4] M. Pawar, V. More, P. Jalke, and A. R. Nichal, "IoT Based Smart Agriculture Monitoring, Weather Control and Irrigation System," International Journal of Research Publication and Reviews, vol. 3, no. 12, pp. 1688–1694, Dec. 2022. DOI: 10.55248/gengpi.2022.31251

[5] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," IETF RFC 7252, Jun. 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7252

[6] A. Banks and R. Gupta, "MQTT Version 3.1.1," OASIS Standard, Oct. 2014. [Online]. Available: https://docs.oasis- open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[7] A.-F. Jimenez and F. Jimenez, "Intelligent IoT-Multiagent Precision Irrigation Approach for Improving Water Use Efficiency in Irrigation Systems at
Farm and District Scales," Computers and Electronics in Agriculture, vol. 192, article 106635, Jan. 2022.

[8] A. Morchid, R. Jebabra, R. E. Alami, M. Charqi, and B. Boukili, "Smart Agriculture for Sustainability: The Implementation of Smart Irrigation Using Real-Time Embedded System Technology," in Proc. 4th Int. Conf. Innovative Research in Applied Science, Engineering and Technology (IRASET), Fez, Morocco, 2024, pp. 1–6.

[9] M. K. I. Abd Rahman, A. R. O. Otuoze, P. Onotu, and M. S. A. Ramli, "A Review on Monitoring and Advanced Control Strategies for Precision Irrigation," Computers and Electronics in Agriculture, vol. 173, article 105441, Jun. 2020.