Sensors and Materials, Vol. 38, No. 11 (2026) 6415–6436
MYU Tokyo

S & M 4688

Internet of Things-based ESP32 Mamdani Fuzzy Rainfall-risk Alert System with MATLAB Validation and Multichannel Alert Evaluation

Rizky Rahmatullah* and Hidetaka Nambo

Kanazawa University, Kakuma-machi, Kanazawa, Ishikawa 920-1192, Japan

(Received July 18, 2026; accepted September 14, 2026)

Keywords: Internet of Things, ESP32, Mamdani fuzzy inference system, rainfall risk, WGAN-GP, MQTT

Rainfall-related risks can disrupt agricultural activities, while conventional threshold-based warning systems may produce abrupt decisions near predefined boundaries and inadequately represent uncertainty among weather variables. In this study, we developed an Internet of Things (IoT)-based rainfall-risk alert system using a Mamdani fuzzy inference system implemented on a low-cost ESP32 platform. Hourly rainfall, relative humidity, and hourly atmospheric pressure change were used to classify conditions into Safe, Warning, and Danger levels. A publicly available weather dataset recorded at 10 min intervals was aggregated into 8780 complete hourly samples. Because only 67 samples belonged to the Danger class, a conditional Wasserstein generative adversarial network with gradient penalty (WGAN-GP) generated additional 6220 samples for rare-class and boundary-condition stress testing, producing a balanced dataset of 15000 samples. The ESP32 implementation reproduced the MATLAB reference without implementation-induced label differences across the evaluated datasets, indicating the consistent transfer of the Mamdani fuzzy model to a resource-constrained embedded platform. For the combined dataset, the defuzzified outputs produced a mean absolute error of 7.2458×10^{-5} and a root mean square error of 1.3630×10^{-4}. A conventional threshold-based baseline achieved 69.987% agreement and a *Macro-F*1 score of 69.801%. The active buzzer, Message Queuing Telemetry Transport (MQTT), and Telegram channels achieved 100% success during controlled experiments. These findings confirm the implementation consistency and communication feasibility of the proposed prototype, although field-level predictive validity requires further evaluation.

1. Introduction

Environmental variability, especially of rainfall, relative humidity, and atmospheric pressure, has a strong effect on agricultural activities. Heavy or untimely rains may disrupt irrigation scheduling, field operations, crop protection, harvesting, and access to cultivated areas.

*Corresponding author: e-mail: rizk032@stu.kanazawa-u.ac.jp
https://doi.org/10.18494/SAM6535

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

Therefore, continuous environmental monitoring and timely warnings are important to support agricultural decision-making. In this context, smart farming merges environmental sensing, information processing, automation, and decision support to enhance agricultural management and resource efficiency.[1,2]

IoT allows sensors, embedded processors, communication networks and user applications to operate as a single monitoring system. IoT platforms have been employed in irrigation management, greenhouse monitoring, soil assessment, and environmental forecasting.[3–7] Edge processing is particularly suitable for early-warning applications where environmental data can be processed near the monitoring site without relying completely on continuous cloud computation. Moreover, wireless technologies such as Wi-Fi and Long Range Wide Area Network (LoRaWAN) enable measurements and alerts to be transmitted to remote platforms and end users. These functions indicate that an IoT warning system should integrate sensing, decision-making, communication, and information delivery rather than operate only as a data-acquisition device.

Many environmental monitoring systems use fixed thresholds to classify conditions as normal or hazardous. Although threshold-based rules are computationally simple, they produce abrupt transitions near predefined boundaries. Consequently, two observations with only a small numerical difference may be assigned to different warning levels. This limitation becomes more important when several environmental variables simultaneously affect the decision. In contrast, fuzzy inference represents variables using overlapping membership functions and combines them through interpretable linguistic rules. Therefore, fuzzy methods have been applied to agricultural monitoring, irrigation scheduling, and precision water management to represent gradual relationships among environmental inputs.[8–10] More recently, fuzzy inference has also been implemented on IoT and embedded platforms to support real-time agricultural decision-making.[11,12]

Nevertheless, transferring a fuzzy inference system from a software environment to a microcontroller requires careful validation. Differences in floating-point precision, membership-function evaluation, rule aggregation, and centroid discretization may produce small numerical differences between embedded and reference implementations. Therefore, functional operation alone is insufficient to demonstrate implementation consistency. Instead, identical input samples should be evaluated using both implementations, and class agreement should be complemented by numerical error metrics for the defuzzified outputs.

Another challenge is the limited availability of observations representing hazardous environmental conditions. Weather datasets generally contain more normal and moderate observations than extreme cases, which can restrict the evaluation of minority risk classes. Data availability and model generalization remain important concerns in intelligent agricultural systems.[13] Class imbalance can further restrict the evaluation of minority conditions, motivating the use of agricultural generative adversarial network (GAN)-based augmentation.[14,15] However, synthetic observations should not be interpreted as replacements for independent field measurements. Accordingly, the generated samples in this study are used only for rare-class and boundary-condition stress testing, while observational and synthetic subsets are evaluated separately.

Communication performance is also essential in an IoT warning system. Successful message publication does not necessarily guarantee that an alert is received without loss, duplication, or excessive delay. Thus, IoT architectures should be evaluated not only in terms of connectivity but also through delivery and network-performance metrics. In previous studies, integrated sensing and communication for agricultural resource management have been emphasized.[16] Communication performance has also been evaluated using packet reception, packet loss, delay, throughput, jitter, and received signal strength in LoRa–Message Queuing Telemetry Transport (MQTT) networks.[17]

Despite these advances, several problems remain. First, embedded fuzzy systems are often demonstrated functionally without detailed numerical validation against their original software reference. Second, fuzzy approaches are not always compared with conventional threshold-based methods using identical input samples. Third, rare hazardous conditions are frequently underrepresented in observational datasets, limiting the evaluation of minority classes. Finally, alert communication is commonly presented as a system feature without the quantitative assessment of message reception, duplication, and latency. Taken together, the reviewed literature suggests that fuzzy decision-making, embedded validation, imbalanced-data treatment, and communication analysis have generally been investigated as separate components.

To address these problems, in this study, we propose an IoT-based rainfall-risk alert system using a Mamdani fuzzy inference system implemented on an ESP32. Hourly rainfall, relative humidity, and 1 h atmospheric pressure change are used to classify conditions into Safe, Warning, and Danger levels. A publicly available weather dataset recorded at 10 min intervals is processed into complete hourly observations, and conditional GAN-based augmentation is used to supplement the rare Danger class for stress testing. The ESP32 implementation is evaluated row by row against a MATLAB reference to verify that the Mamdani fuzzy model can be reproduced consistently on a low-cost embedded platform with limited RAM and Flash memory. Class agreement, *Macro-F*1, mean absolute error (*MAE*), and root mean square error (*RMSE*) are used to identify possible implementation-induced differences. In addition, a conventional threshold-based classifier is evaluated using identical input samples. Finally, the active buzzer, MQTT, and Telegram channels are quantitatively assessed using execution success, message-delivery rate, duplication, and latency metrics. Thus, the principal contribution of this study is an integrated experimental framework for evaluating fuzzy decision consistency, embedded implementation, rare-class robustness, and multichannel IoT alert delivery. The specific objectives are therefore twofold: (i) to verify that the ESP32 implementation reproduces the MATLAB Mamdani fuzzy model with negligible numerical deviation and (ii) to characterize the feasibility of the multichannel alert communication under controlled conditions. Establishing the independently validated, field-level predictive accuracy of the rainfall-risk classification itself is treated as a distinct objective for future work rather than a claim of the present study.

2. Data, Materials, and Methods

2.1 System architecture and weather data preprocessing

Figure 1 shows the complete processing flow of the proposed IoT-based rainfall-risk alert system. In a practical deployment, rainfall, relative humidity, and atmospheric pressure are measured by environmental sensors. In this study, the sensor data stream was represented by the publicly available Weather Long-term Time Series Forecasting dataset obtained through Kaggle.[18] The raw dataset contained 52696 records. After chronological sorting, one duplicate timestamp and incomplete hourly windows caused by a 100 min recording gap were excluded. Only hours containing six complete 10 min observations and a complete preceding hour were retained. Hourly rainfall was calculated as the sum, relative humidity as the mean, and P_t as the final pressure measurement within each hour. This preprocessing produced 8780 hourly samples. In practical deployment, rainfall, relative humidity, and atmospheric pressure are measured as standard meteorological variables using environmental instruments.[19]

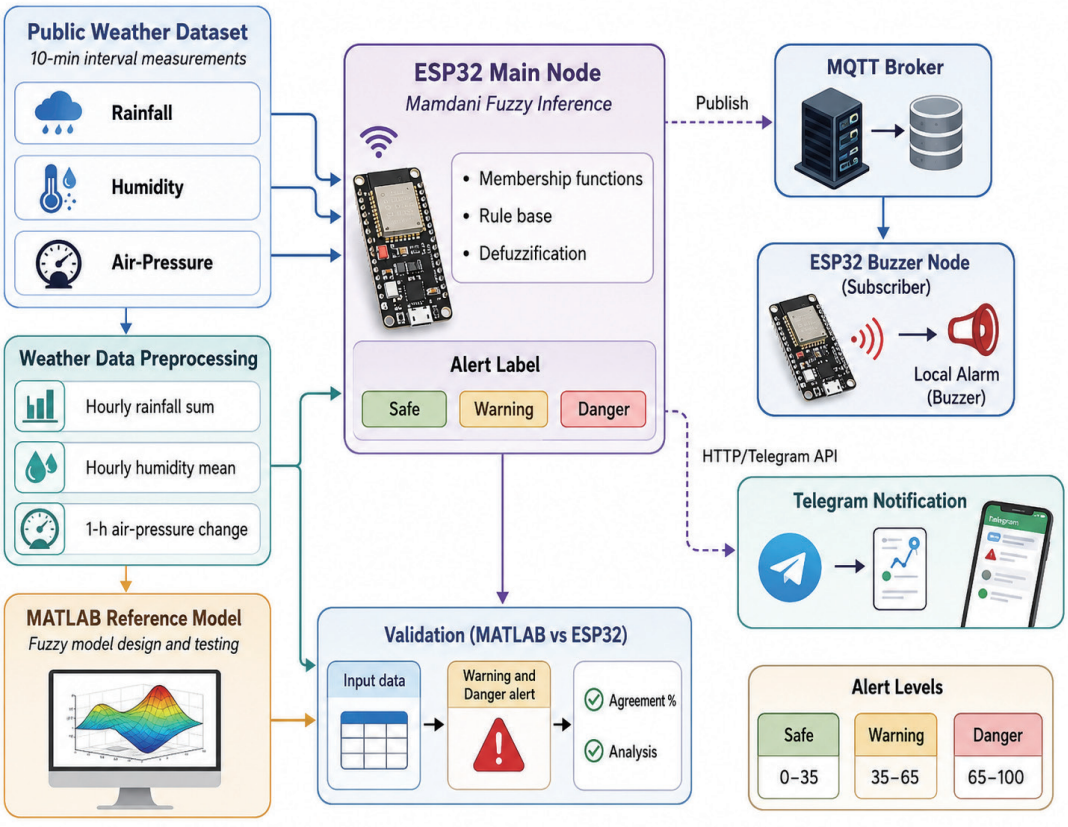


Fig. 1. (Color online) Overall system architecture and weather-data preprocessing flow of the proposed IoT-based rainfall-risk alert system.

The sensor measurements were first converted into hourly input variables. Hourly rainfall was obtained by summing six consecutive rainfall measurements,

$$R_t = \sum_{i=1}^{6} r_{t,i} \ , \tag{1}$$

whereas hourly relative humidity was calculated as their mean:

$$H_t = \frac{1}{6} \sum_{i=1}^{6} h_{t,i} \ . \tag{2}$$

The 1 h atmospheric pressure change was calculated from two consecutive hourly pressure values as

$$\Delta P_t = P_t - P_{t-1} \ . \tag{3}$$

Here, R_t, H_t, and ΔP_t represent hourly rainfall, mean relative humidity, and 1 h atmospheric pressure change, respectively. Only hours containing six complete 10 min observations were retained, resulting in 8780 hourly samples.

Each hourly input vector (R_t, H_t, ΔP_t) was then processed by the Mamdani fuzzy inference system implemented on the ESP32. The ESP32 evaluated the membership functions, activated the fuzzy rules, aggregated the rule outputs, and performed centroid defuzzification. The resulting crisp value was classified as Safe, Warning, or Danger.

The ESP32 generated local and remote warnings when a Warning or Danger condition was detected. The alert was published via an MQTT broker to an ESP32 subscriber connected to an active buzzer. A remote notification was sent via Telegram Bot API. At the same time, the MATLAB reference model was fed with the same hourly inputs and the results from the ESP32 were compared to assess the implementation consistency.

2.2 Mamdani fuzzy inference system

Previous fuzzy weather studies have used rainfall together with relative humidity, atmospheric pressure, temperature, and other meteorological variables for rainfall or flood classification.[20–22] Related fuzzy-IoT agricultural research also supports interpretable embedded decision-making.[23] Therefore, the membership-function breakpoints were determined by considering the literature, the observed data ranges, and the need for overlap between adjacent linguistic terms. These values were used as operational model parameters rather than universal meteorological thresholds.

The Mamdani fuzzy inference system (FIS) used hourly rainfall R_t, hourly mean relative humidity H_t, and 1 h atmospheric pressure change ΔP_t as inputs. Rainfall and humidity were divided into Low, Medium, and High sets, whereas pressure change was divided into HighDrop, Normal, and LowChange sets. The output alert score was represented by Safe, Warning, and Danger sets. Table 1 shows the membership-function parameters used identically in MATLAB and the ESP32. The same parameter values and 11-rule structure were retained in both implementations.

Table 1 presents the membership-function parameters used for the three input variables and the output alert score. Trapezoidal functions were assigned to the boundary linguistic terms to maintain full membership near the limits of each variable, whereas triangular functions were used for the intermediate terms. The overlapping regions allow an input value to belong partially to two adjacent categories, thereby producing gradual transitions instead of abrupt threshold changes.

The rainfall, relative humidity, atmospheric pressure change, and output thresholds were adopted from previous fuzzy rainfall and meteorological studies.[19–22] These thresholds were retained in the present study to enable a consistent evaluation using the new observational dataset. Because these parameters and rule thresholds were adopted directly from the cited literature and were not tuned or re-estimated using the present dataset, the observational data were not required for parameter or rule development, and the full dataset could be reserved for implementation validation and testing.

Algorithm 1 shows the hourly rainfall-risk alert generation procedure implemented in both MATLAB and the ESP32. The procedure receives hourly rainfall, relative humidity, and air pressure measurements, applies the Mamdani fuzzy rules, performs centroid defuzzification, and assigns the resulting output to the Safe, Warning, or Danger class.

The same membership functions, rule base, inference operators, output discretization, and alert boundaries were used in both implementations to ensure a consistent comparison between MATLAB and the ESP32.

Table 1
Membership-function parameters.

Variable	Linguistic term	Type	Parameters
Rainfall, (R_t) (mm/h)	Low	Trapezoidal	([0,0,1,2])
	Medium	Triangular	([1.5,3,4.5])
	High	Trapezoidal	([4,5,50,50])
Humidity, (H_t) (%)	Low	Trapezoidal	([0,0,55,65])
	Medium	Triangular	([60,70,80])
	High	Trapezoidal	([75,85,100,100])
Pressure change, (ΔP_t) (hPa/h)	HighDrop	Trapezoidal	([−6,−6,−4.2,−3])
	Normal	Triangular	([−4,−2.5,−0.8])
	LowChange	Trapezoidal	([−1.5,−0.5,10,10])
Alert score, (y)	Safe	Trapezoidal	([0,0,25,45])
	Warning	Triangular	([35,55,75])
	Danger	Trapezoidal	([65,80,100,100])

Algorithm 1
Hourly rainfall-risk alert generation.

1 : Input : rainfall R_t, relative humidity H_t, current air pressure P_t, previous-hour air pressure P_{t-1}

2 : Output : defuzzified output y_t^* , alert label L_t

3 : Compute one-hour air-pressure change:

$$\Delta P_t = P_t - P_{t-1}$$

4 : Fuzzify the input variables:

rainfall R_t → {Low, Medium, High}
relative humidity H_t → {Low, Medium, High}
air-pressure change ΔP_t → {HighDrop, Normal, LowChange}

5 : Apply Mamdani rules $R1-R11$ using the minimum operator (**AND**)

6 : Aggregate the rule outputs using the maximum operator (**OR**)

7 : Perform centroid defuzzification:

$$y_t^* = \frac{\sum_j y_j \, \mu_{\text{agg}}(y_j)}{\sum_j \mu_{\text{agg}}(y_j)}$$

where $\mu_{\text{agg}}(y)$ is the aggregated output membership function.

8 : Assign alert label L_t based on y_t^*:

$$L_t = \begin{cases} \text{Safe}, & \text{if } y_t^* < 35, \\ \text{Warning}, & \text{if } 35 \le y_t^* < 65, \\ \text{Danger}, & \text{if } y_t^* \ge 65. \end{cases}$$

9 : **R**eturn y_t^* and L_t

For each rule k, the firing strength was calculated from the minimum membership degree of its antecedents as

$$\alpha_k = \min\left[\mu_{A_{k,1}}(R_t), \mu_{A_{k,2}}(H_t), \mu_{A_{k,3}}(\Delta P_t)\right]. \tag{4}$$

The consequent of each activated rule was clipped in accordance with a_k, and all rule outputs were combined using the maximum aggregation operator, as defined by

$$\mu_{agg}(y) = \max_k\left[\min\left(\alpha_k, \mu_{B_k}(y)\right)\right]. \tag{5}$$

The aggregated output was then converted into a crisp alert score through centroid defuzzification over 1001 discrete points in the output range from 0 to 100, as follows.

$$y^* = \frac{\sum_{j=1}^{M} y_j \mu_{agg}(y_j)}{\sum_{j-1}^{M} \mu_{agg}(y_j)}, \quad M = 1001 \tag{6}$$

Finally, the defuzzified score was converted into the Safe, Warning, or Danger class using the following boundaries:

$$C\left(y^*\right)=\begin{cases}\text{Safe}, & y^* < 35\\ \text{Warning}, & 35 \le y^* < 65\\ \text{Danger}, & y^* \ge 65\end{cases} \tag{7}$$

When no rule was activated, the denominator in Eq. (6) became zero; therefore, both MATLAB and the ESP32 assigned $y^* = 0$, corresponding to the Safe class.

Table 2 shows the 11 Mamdani if–then rules used to relate the three weather inputs to the alert level. The rules were designed to emphasize high rainfall and pronounced pressure drops as higher-risk conditions, while low rainfall and low humidity were generally associated with Safe conditions. The term "Any" indicates that the corresponding variable was not included in the antecedent of that rule.

The same rule base was implemented in MATLAB and the ESP32. Therefore, in the subsequent validation, two implementations of the same fuzzy model were compared rather than two different decision structures.

2.3 GAN-based data augmentation

The observational dataset was highly imbalanced, with 4226 Safe, 4487 Warning, and only 67 Danger samples. A conditional Wasserstein generative adversarial network with gradient penalty (WGAN-GP) was therefore used to generate rainfall, relative humidity, and atmospheric pressure change for rare-class augmentation.[14,15,24] The model used class-balanced minibatches, a latent dimension of 16, a batch size of 256, 1200 training steps, and a random seed of 20260712. After calibration and screening based on physical ranges, FIS-label consistency, and duplicate detection, 774 Safe, 513 Warning, and 4933 Danger samples were retained. The synthetic records were used only for implementation stress testing, not as substitutes for field observations or

Table 2
Mamdani fuzzy rule base.

Rule	Rainfall	Humidity	Pressure change	Alert
R1	High	Any	Any	Danger
R2	High	Any	HighDrop	Danger
R3	Medium	Low	LowChange	Safe
R4	Medium	Medium	LowChange	Warning
R5	Medium	High	LowChange	Warning
R6	Medium	High	HighDrop	Danger
R7	Low	Low	Any	Safe
R8	Low	High	LowChange	Warning
R9	Low	Medium	Normal	Warning
R10	Low	High	Normal	Danger
R11	Low	Any	HighDrop	Danger

time-series forecasting. Because a generated sample was retained only when it conformed to the predefined fuzzy model, this screening procedure is circular by construction and does not constitute the independent validation of the WGAN-GP outputs; it is used here solely to verify that both implementations behave consistently on stress-test inputs.

The WGAN-GP critic and generator losses were defined following the gradient-penalty formulation proposed by Gulrajani *et al.*[24]

$$\mathcal{L}_{\mathcal{D}} = \mathbb{E}\left[D\left(\tilde{x},c\right)\right] - \mathbb{E}\left[D\left(x,c\right)\right] + \lambda\mathbb{E}\left[\left(\left|\nabla_{\hat{x}}D\left(\hat{x},c\right)\right|_2 - 1\right)^2\right] \tag{8}$$

$$\mathcal{L}_{\mathcal{G}} = -\mathbb{E}\left[D\left(\tilde{x},c\right)\right] \tag{9}$$

Here, x and $\tilde{x}$ denote observational and generated samples, respectively, c is the class condition, and λ is the gradient-penalty coefficient.

The target was set to 5000 samples per class. Thus, 774 Safe, 513 Warning, and 4933 Danger samples were generated, resulting in 6220 synthetic samples and a balanced combined dataset of 15000 samples, as shown in Table 3.

The generated samples were screened using physical-range checks, class-wise Kolmogorov–Smirnov (KS) statistics, Wasserstein distances, FIS-label consistency, and duplicate detection. The consistency rate was calculated as

$$CR = \frac{N_{consistent}}{N_{evaluated}} \times 100\% \,. \tag{10}$$

All retained samples satisfied the FIS-label consistency criterion, yielding a consistency rate of 100%, and no exact duplicates were detected. The synthetic samples were used for rare-class and boundary-condition stress testing rather than as substitutes for independent field observations. Consequently, the observational and GAN-generated subsets were evaluated separately during MATLAB–ESP32 validation.

2.4 Experimental implementation and validation

Figure 2 shows the experimental implementation and validation flow. The Mamdani fuzzy system was implemented in MATLAB and on the ESP32 using identical membership functions,

Table 3
Composition of the observational and GAN-augmented datasets.

Class	Observational	GAN-generated	Combined
Safe	4,226	774	5,000
Warning	4,487	513	5,000
Danger	67	4,933	5,000
Total	8,780	6,220	15,000

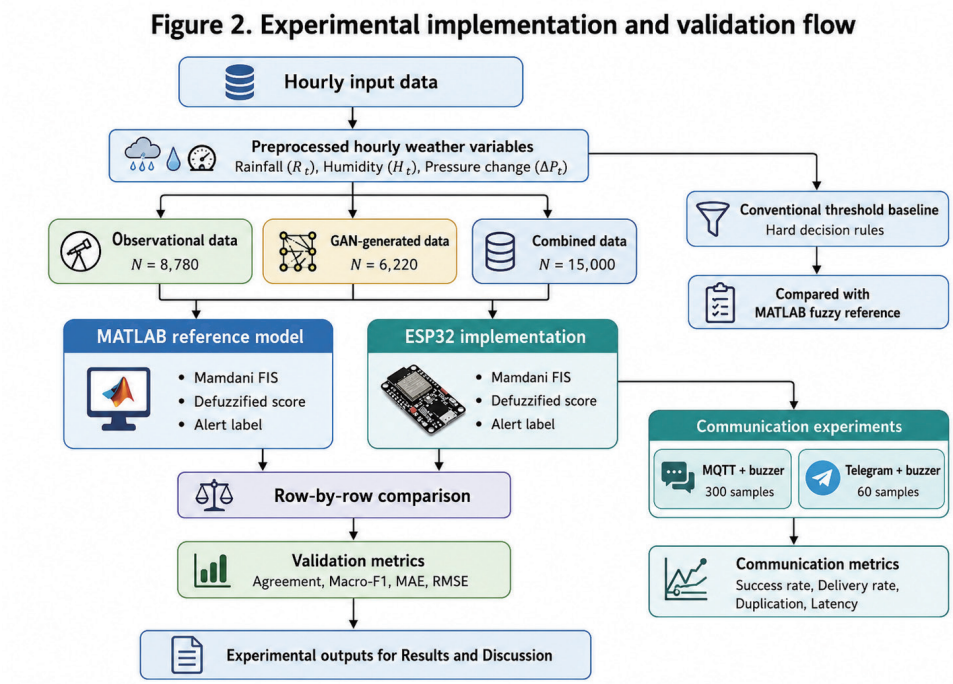


Fig. 2. (Color online) Experimental implementation and validation flow.

11 rules, inference operators, output discretization, fallback handling, and alert boundaries. A similar integration of fuzzy logic and IoT has been demonstrated in agricultural monitoring and control systems.[23]

MATLAB was used as the reference implementation because it provides a standardized environment for designing and validating Mamdani fuzzy inference systems, whereas the ESP32 executes the same inference on a low-cost embedded platform with limited RAM and Flash memory. Although identical membership functions and fuzzy rules were implemented, differences in floating-point precision, manual membership-function evaluation, and discrete centroid computation could introduce implementation-level numerical deviations. Therefore, the comparison was performed to verify that the embedded implementation faithfully reproduced the MATLAB reference rather than to assess the accuracy of the fuzzy model itself.

For each sample, MATLAB sent the input vector (R_t, H_t, ΔP_t) to the ESP32 through a serial connection at 115200 baud. The ESP32 returned the defuzzified score and alert label. These outputs were compared row by row using 8,780 observational samples, 6220 GAN-generated samples, and the combined dataset of 15000 samples.

A conventional threshold-based classifier was evaluated using the same inputs. Danger was assigned when rainfall exceeded 4 mm/h. Warning was assigned when rainfall was at least 2 mm/h, relative humidity was at least 65%, or pressure change was at most −1.5 hPa/h. All other

conditions were classified as Safe. The resulting labels were compared with the MATLAB fuzzy reference.

The communication experiments used 300 samples for MQTT and buzzer testing and 60 samples for Telegram and buzzer testing. MQTT messages were transmitted through the public HiveMQ broker and recorded by a Python subscriber. Telegram notifications were sent through the Telegram Bot API. Message reception, duplication, latency, buzzer activation, and Wi-Fi signal strength were recorded. Similar communication metrics, including latency and data loss, have been used to evaluate MQTT-based agricultural IoT systems.[17,25]

2.5 Evaluation metrics

The MATLAB output was used as the reference for evaluating the computational consistency of the ESP32 implementation and the conventional threshold-based baseline. Label agreement was calculated as

$$Agreement = \frac{N_{match}}{N} \times 100\%, \tag{11}$$

where N_{match} is the number of samples with identical labels and N is the total number of evaluated samples.

The precision and recall for alert class c were calculated as

$$P_c = \frac{TP_c}{TP_c + FP_c}, \tag{12}$$

$$R_c = \frac{TP_c}{TP_c + FN_c}, \tag{13}$$

where TP_c, FP_c, and FN_c denote the true-positive, false-positive, and false-negative counts for class c, respectively. The class-wise F1 score was calculated as

$$F1_c = \frac{2P_cR_c}{P_c + R_c}. \tag{14}$$

The *Macro-F*1 score was obtained by averaging the F1 scores of the Safe, Warning, and Danger classes.

$$Macro\text{-}F1 = \frac{1}{3}\sum_{c=1}^{3} F1_c \tag{15}$$

Differences between the defuzzified MATLAB and ESP32 outputs were evaluated using *MAE* and *RMSE*.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_i^{ESP32} - y_i^{MATLAB}\right| \tag{16}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i^{ESP32} - y_i^{MATLAB}\right)^2} \tag{17}$$

Communication performance was evaluated using the success rate:

$$Success\ Rate = \frac{N_{successful}}{N_{tested}} \times 100\%, \tag{18}$$

where $N_{successful}$ is the number of successful buzzer activations, MQTT transmissions, or Telegram API responses. The MQTT delivery rate was calculated as

$$Delivery\ Rate = \frac{N_{received}}{N_{published}} \times 100\%. \tag{19}$$

The latency of test i was calculated as

$$L_i = t_i^{end} - t_i^{start}. \tag{20}$$

For MQTT, latency represented the approximate one-way interval between message transmission by the ESP32 and reception by the Python subscriber. For Telegram, it represented the interval between sending the API request and receiving the HTTP response. Buzzer latency represented the local interval between the alert command and the buzzer activation. The mean, median, 95th percentile, and maximum latency were reported. MQTT duplicates and the average Wi-Fi received signal strength indicator were also recorded. Agreement, *Macro-F*1, *MAE*, and *RMSE* represent consistency with the MATLAB reference and should not be interpreted as real-world rainfall-prediction accuracy.

3. Results and Discussion

3.1 Dataset characteristics and GAN augmentation results

The observational dataset was heavily imbalanced with 4226 Safe samples, 4487 Warning samples, and only 67 Danger samples. Thus, the Danger class represented approximately 0.76%

of the 8780 observational samples. After conditional WGAN-GP augmentation and screening, 774 Safe, 513 Warning, and 4933 Danger samples were retained. The resulting combined dataset contained 5000 samples per class and 15000 samples in total.

All retained synthetic samples remained within the accepted variable ranges. The generated rainfall values ranged from 0 to 26.945 mm/h, relative humidity from 26.813 to 100%, and 1 h atmospheric pressure change from −2.591 to 2.170 hPa/h. The corresponding observational ranges were 0–27 mm/h, 21.985–100%, and −2.620–3.330 hPa/h, respectively. Table 4 presents the class-wise comparison between the observational and GAN-generated samples.

The KS statistics ranged from 0.003 to 0.225, with the largest deviation observed for Warning-class rainfall. Wasserstein distances were generally small relative to the measurement range of each variable, although Danger-class humidity showed a larger difference, likely owing to the limited number of observational Danger samples. Following previous agricultural studies on GAN-based augmentation,[14,15] the generated weather samples were screened using physical-range checks, distributional comparison, FIS-label consistency, and duplicate detection before retention.

All retained synthetic samples matched the corresponding conditioning classes, yielding a 100% consistency rate, with no exact duplicates detected. However, this result does not establish meteorological accuracy because label consistency was part of the screening procedure, and the Danger class was derived from only 67 observational samples. The balanced dataset was therefore suitable for implementation stress testing.

Future research should include collecting additional field observations, particularly for the Danger class, and evaluating the augmentation procedure across different seasons, locations, and weather patterns. Overall, the WGAN-GP augmentation produced a balanced and screened dataset suitable for use in implementation stress testing, with the separate reporting of observational and synthetic subsets to avoid treating the generated samples as equivalent to independent field measurements.

Table 4
Class-wise observational and GAN-generated samples.

Class	Feature	Observational mean ± SD	GAN mean ± SD	KS statistic	Wasserstein distance
Safe	Rainfall (mm/h)	0.008 ± 0.084	0.006 ± 0.058	0.003	0.002
Safe	Humidity (%)	56.091 ± 13.038	56.992 ± 12.440	0.048	1.108
Safe	Pressure change (hPa/h)	−0.029 ± 0.430	−0.100 ± 0.465	0.067	0.070
Warning	Rainfall (mm/h)	0.084 ± 0.370	0.134 ± 0.414	0.225	0.051
Warning	Humidity (%)	87.742 ± 8.201	88.523 ± 7.556	0.074	1.039
Warning	Pressure change (hPa/h)	0.033 ± 0.469	−0.021 ± 0.477	0.045	0.056
Danger	Rainfall (mm/h)	3.142 ± 4.746	3.186 ± 4.469	0.039	0.230
Danger	Humidity (%)	85.341 ± 11.634	86.930 ± 8.314	0.120	1.876
Danger	Pressure change (hPa/h)	−0.774 ± 0.921	−0.796 ± 0.864	0.059	0.075

3.2 ESP32–MATLAB validation and baseline comparison

The ESP32 implementation consistently reproduced the MATLAB fuzzy reference across all evaluated datasets. As summarized in Table 5, no implementation-induced label differences were observed between the desktop and embedded implementations, resulting in identical classifications and a *Macro-F*1 score of 100%. The negligible *MAE* and *RMSE* values further indicate that implementation-level numerical effects did not alter the resulting alert classifications.

For the combined dataset, the ESP32–MATLAB comparison produced an *MAE* of 7.2458×10^{-5} and an *RMSE* of 1.3630×10^{-4}. Although the GAN-generated subset showed slightly larger errors than the observational subset, the deviations remained too small to alter any alert label. These minor differences were attributed to floating-point precision, membership-function evaluation, rule aggregation, and discrete centroid calculation, confirming the consistent reproduction of the MATLAB fuzzy decisions on the ESP32.

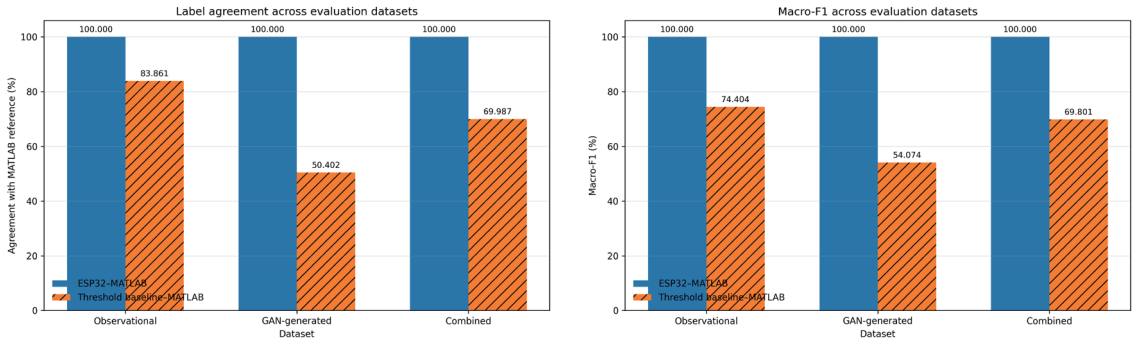
No implementation-induced label differences were observed between the ESP32 and the MATLAB reference across the evaluated datasets. The identical classifications indicate that the embedded implementation preserved the intended Mamdani fuzzy decision process despite the computational constraints of the low-cost ESP32, including its limited RAM and Flash memory. The corresponding *MAE* and *RMSE* values remained negligible, confirming implementation consistency between the desktop and embedded implementations.

Figure 3 shows the ESP32 implementation and the conventional threshold-based baseline using label agreement and *Macro-F*1. The ESP32 reproduced the MATLAB reference consistently across all datasets, whereas the threshold baseline achieved agreement rates of 83.861, 50.402, and 69.987% for the observational, GAN-generated, and combined datasets, respectively, with corresponding *Macro-F*1 scores of 74.404, 54.074, and 69.801%. The larger difference on the GAN-generated subset shows that the fixed thresholds led to different classifications for the rare and boundary-condition samples, while overlapping fuzzy membership functions allowed for more gradual transitions between alert levels.[10,12,23]

A total of 1633 samples activated no fuzzy rule and were therefore assigned $y^* = 0$ and classified as Safe by both MATLAB and the ESP32. Although this preserved implementation agreement, it indicates incomplete rule coverage. Moreover, the evaluated datasets contained no pressure changes below −3 hPa/h, so the HighDrop-related rules were not activated. In future

Table 5
ESP32–MATLAB validation and threshold baselines.

Dataset	Samples	ESP32–MATLAB agreement (%)	*Macro-F*1 (%)	*MAE*	*RMSE*	Threshold–MATLAB agreement (%)	Threshold *Macro-F*1 (%)
Observational	8780	100	100	2.8445×10^{-5}	6.4515×10^{-5}	83.861	74.404
GAN-generated	6220	100	100	1.3459×10^{-4}	1.9729×10^{-4}	50.402	54.074
Combined	15000	100	100	7.2458×10^{-5}	1.3630×10^{-4}	69.987	69.801



(a) (b)

Fig. 3. (Color online) Results of ESP32 implementation and conventional threshold-based baseline with the MATLAB fuzzy reference: (a) label agreement and (b) *Macro-F*1 score for the observational, GAN-generated, and combined datasets.

work, the rule base should be expanded and stronger pressure-drop conditions should be evaluated using field observations.

The baseline results should not be interpreted as evidence that the fuzzy model was meteorologically more accurate. Both the agreement and *Macro-F*1 values were calculated using the MATLAB fuzzy output as the reference rather than independently labeled rainfall-risk events. Moreover, the samples generated by the GAN were filtered and conditioned on the fuzzy classes. Consequently, the comparison primarily demonstrates differences between fuzzy reasoning and conventional hard-threshold classification.

In future work, both decision methods should be validated against independently labeled field events, and their ability to identify actual hazardous rainfall conditions should be assessed. The results show that, in general, the ESP32 can reproduce the MATLAB Mamdani fuzzy model with small numerical error, while the traditional threshold method provides very different decisions, especially for rare and boundary-condition samples.

3.3 Alert communication performance

The local and remote alert channels were evaluated using 300 MQTT trials and 60 Telegram trials. All MQTT messages were received successfully by the subscriber, all Telegram requests returned successful HTTP responses, and the buzzer was activated in every test. The communication and local-alert performance are summarized in Table 6.

In the MQTT experiment, all 300 published messages were received by the Python subscriber, resulting in a delivery rate of 100%. Each message was assigned a unique test identifier and there were no duplicate messages. The mean approximate one-way latency was 4701.837 ms, with a median of 4556 ms, a 95th-percentile value of 5902.1 ms, and a maximum of 6453 ms.

Table 6
Performance of the alert communication channels.

Alert channel	Trials	Success or delivery rate (%)	Mean latency	Median latency	P95 latency	Maximum latency	Mean RSSI (dBm)
MQTT message delivery	300	100	4701.837 ms	4556 ms	5902.1 ms	6453 ms	−78.85
Telegram API response	60	100	2014.8 ms	2019.5 ms	2229.5 ms	2745 ms	−75.92
MQTT-associated buzzer	300	100	6.82 µs	7 µs	7 µs	14 µs	—
Telegram-associated buzzer	60	100	7.90 µs	8 µs	8 µs	8 µs	—

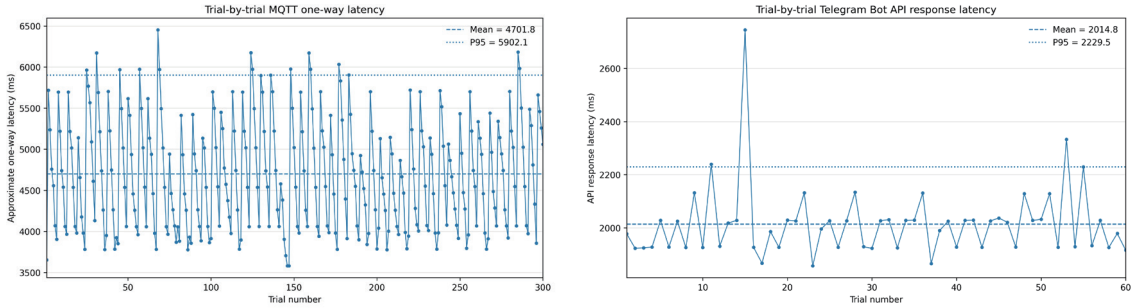
Figure 4 presents the trial-by-trial latency measurements for the two remote communication channels. The MQTT results showed greater fluctuation than the Telegram API response times. However, the MQTT latency was calculated using transmission and reception timestamps recorded on separate devices. Consequently, the values may include clock offset in addition to public-broker routing and Internet delay. The MQTT results should therefore be interpreted as approximate one-way end-to-end latency rather than broker-processing latency. In the Telegram experiment, all 60 requests returned HTTP status code 200, yielding a success rate of 100%. The mean API response latency was 2014.8 ms, while the median, 95th-percentile, and maximum latencies were 2019.5, 2229.5, and 2745 ms, respectively. These values represent the interval between sending a request to the Telegram Bot API and receiving the HTTP response. They do not represent the time required for a notification to appear on the user's device or for the user to read it.

Figure 5 presents examples of the Telegram notifications generated for Warning and Danger conditions. Each message includes the input measurements, defuzzified output, alert status, and corresponding recommended action. The displayed messages confirm that the fuzzy classification results were converted into readable remote notifications containing both the measured conditions and the recommended response.

The active buzzer also responded successfully in every trial. The mean buzzer command latency was determined to be 6.82 µs during the MQTT experiment and 7.90 µs during the Telegram experiment. The corresponding maximum values were 14 and 8 µs, respectively. Compared with the remote communication delays, the local buzzer latency was negligible. Because the buzzer measurements were nearly constant and several orders of magnitude smaller than the network latencies, they were summarized in Table 6 rather than included in Fig. 4.

The mean Wi-Fi received signal strength was determined to be −78.85 dBm during the MQTT experiment and −75.92 dBm during the Telegram experiment. Despite these signal conditions, no MQTT message loss, message duplication, Telegram request failure, or buzzer activation failure was observed during the controlled tests.

MQTT-based communication has also been evaluated in previous studies using delivery rate, latency, and message-loss indicators.[17,25] However, direct numerical comparison with previous results is limited because broker location, broker type, MQTT quality-of-service level, payload size, network conditions, synchronization method, and latency definition may differ among experiments. Therefore, the present communication results are primarily used to demonstrate the feasibility and consistency of the prototype rather than to claim superior network performance.



(a) (b)

Fig. 4. (Color online) Trial-by-trial latency measurements during the alert communication experiments: (a) approximate MQTT one-way latency for 300 messages and (b) Telegram Bot API response latency for 60 requests. The dashed and dotted horizontal lines indicate the mean and 95th-percentile values, respectively.

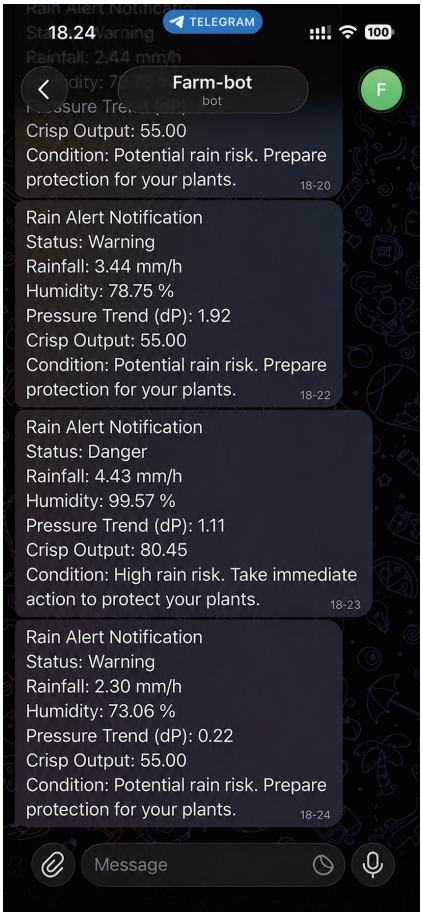


Fig. 5. (Color online) Examples of Telegram notifications generated for Warning and Danger conditions.

The 100% success rates obtained in these experiments should not be generalized to long-term agricultural deployment. Future testing should include longer observation periods, synchronized device clocks, different Wi-Fi signal levels, varying Internet conditions, multiple MQTT quality-of-service settings, and comparisons between public, private, and local brokers. Overall, the local buzzer provided immediate on-site warnings, whereas MQTT and Telegram successfully delivered remote notifications in all controlled trials.

4. Practical Implications and Limitations

The proposed system demonstrates the feasibility of implementing a Mamdani fuzzy rainfall-risk model on a low-cost ESP32 platform. Local fuzzy inference and buzzer activation provide immediate on-site warnings, while MQTT and Telegram enable remote notification. No implementation-induced label differences were observed between the ESP32 and the MATLAB reference for the observational, GAN-generated, and combined datasets. The negligible *MAE* and *RMSE* values further indicate that implementation-level numerical effects did not alter the resulting Safe, Warning, or Danger classifications, confirming implementation consistency between the desktop and embedded implementations. The results presented herein support the use of fuzzy inference on resource-constrained hardware for agricultural monitoring applications.[10,12,23] The conventional threshold-based baseline achieved agreement rates of 83.861, 50.402, and 69.987%, with corresponding *Macro-F*1 scores of 74.404, 54.074, and 69.801% for the observational, GAN-generated, and combined datasets, respectively. The lower scores, in particular, for the rare and boundary-condition samples, show that the fixed thresholds produce different decisions than the gradual Mamdani inference process. The comparison, however, was referenced to the MATLAB fuzzy output and not to independently verified field events, and therefore should not be interpreted as evidence of superior meteorological accuracy.

The study has several limitations that should be considered when interpreting the results. Table 7 summarizes the practical implications, limitations, and potential future improvements of the proposed system. These limitations are discussed in the following points. First, the observational data were obtained from a public weather dataset rather than from an agricultural field deployment. Therefore, sensor calibration, environmental interference, hardware durability, energy consumption, and long-term operation were not evaluated. The model was also limited to rainfall, relative humidity, and 1 h atmospheric pressure change, whereas actual agricultural risk may also depend on soil moisture, antecedent rainfall, crop conditions, drainage, wind, and local topography.

Second, only 67 of the 8780 observational samples belonged to the Danger class. Although WGAN-GP generated 4933 additional Danger samples, these data were derived from a limited observational basis. Because FIS-label consistency was used during screening, the resulting 100% consistency indicates conformity with the predefined fuzzy model rather than the independent evidence of real-world accuracy.

Third, 1633 samples did not activate any of the 11 fuzzy rules and were classified as Safe using the predefined fallback procedure. Although MATLAB and the ESP32 handled these cases consistently, assigning all uncovered conditions to Safe may be unsuitable for practical

Table 7
Practical implications, limitations, and future improvements.

Aspect	Practical implication	Current limitation	Future improvement
Embedded fuzzy inference	Enables local and interpretable risk classification	Validated using a public weather dataset	Conduct field deployment over multiple seasons
ESP32 implementation	Reproduces the MATLAB fuzzy logic consistently	MATLAB was used as the reference model	Validate against expert-labeled or official warning events
Fuzzy versus threshold decision	Provides gradual transitions near decision boundaries	No independent field ground truth was available	Compare both methods using real hazardous events
GAN augmentation	Increases rare and boundary-condition samples	Only 67 observational Danger samples were available	Collect additional real Danger-class observations
Fuzzy rule base	Supports simple rule-based decision-making	1633 samples did not activate any rule	Expand and validate the rule coverage
Local buzzer	Provides an immediate on-site warning	Tested only under controlled conditions	Evaluate outdoor durability and energy consumption
MQTT and Telegram	Enables remote warning delivery	Limited test duration and public-broker dependence	Test longer periods, different QoS levels, and private brokers

deployment. The communication tests were also conducted under controlled conditions using a public MQTT broker and a limited number of trials. In addition, the MQTT latency measurements may include clock offset between devices.

Future work should include multiseason field deployment, additional real Danger-class observations, expanded and expert-validated fuzzy rules, synchronized communication testing, and evaluation against independently labeled agricultural-risk events. Additional environmental variables, energy consumption, sensor drift, false alarms, missed warnings, and long-term communication reliability should also be examined. Overall, the results demonstrate prototype feasibility and MATLAB–ESP32 consistency, but not field-level predictive validity.

5. Conclusions

An IoT-based rainfall-risk alert system using a Mamdani fuzzy inference model was implemented on an ESP32 and validated against a MATLAB reference. The system classified rainfall conditions into Safe, Warning, and Danger levels using hourly rainfall, relative humidity, and 1 h atmospheric pressure change.

No implementation-induced label differences were observed between the ESP32 and MATLAB implementations across the evaluated datasets. The defuzzified outputs also showed only negligible numerical deviations. The defuzzified outputs produced an *MAE* of 7.2458×10^{-5} and an *RMSE* of 1.3630×10^{-4}, indicating negligible implementation-level numerical differences. In comparison, the conventional threshold-based baseline achieved an agreement of 69.987% and a *Macro-F*1 score of 69.801% on the combined dataset, demonstrating substantial differences between hard-threshold decisions and gradual fuzzy inference.

The WGAN-GP augmentation increased the representation of rare and boundary-condition samples, producing a balanced dataset of 15000 samples. However, the synthetic results should be interpreted as implementation stress-testing data because only 67 observational Danger samples were available and FIS-label consistency was included in the screening procedure.

The local buzzer, MQTT, and Telegram channels achieved 100% success during the controlled experiments. The buzzer provided nearly immediate local activation, while MQTT and Telegram enabled remote warning delivery. Nevertheless, the communication results were obtained under limited and controlled test conditions.

Overall, the study demonstrates the feasibility of transferring a MATLAB-designed Mamdani fuzzy model to an ESP32 while preserving its decision behavior. The current results establish prototype feasibility and computational consistency rather than field-level predictive accuracy. Future work should include multiseason field deployment, additional real Danger-class observations, expanded fuzzy-rule coverage, synchronized communication testing, and validation against independently labeled agricultural-risk events. Such an evaluation should report a full confusion matrix, class-wise precision, recall, and F1 score, balanced accuracy, false-alarm rate, missed-warning rate, and confidence intervals, computed against independently verified rainfall-risk labels rather than the MATLAB fuzzy output used as the reference in the present study.

Acknowledgments

The authors would like to express their sincere gratitude to the Graduate School of Natural Science and Technology, Kanazawa University, for providing the academic environment and research facilities that supported this study. The authors also appreciate the valuable guidance and support received during this research.

References

1 M. J. O'Grady and G. M. P. O'Hare: Inf. Process. Agric. **4** (2017) 179. https://doi.org/10.1016/j.inpa.2017.05.001
2 D. Pivoto, P. D. Waquil, E. Talamini, C. P. S. Finocchio, V. F. Dalla Corte, and G. de Vargas Mores: Inf. Process. Agric. **5** (2018) 21. https://doi.org/10.1016/j.inpa.2017.12.002
3 A. Goap, D. Sharma, A. K. Shukla, and C. Rama Krishna: Comput. Electron. Agric. **155** (2018) 41. https://doi.org/10.1016/j.compag.2018.09.040
4 D. P. Rubanga, K. Hatanaka, and S. Shimada: Sens. Mater. **31** (2019) 831. https://doi.org/10.18494/SAM.2019.2154
5 S. R. J. Ramson, W. D. León-Salas, Z. Brecheisen, E. J. Foster, C. T. Johnston, D. G. Schulze, T. Filley, R. Rahimi, M. J. C. Villalta Soto, J. A. Lopa Bolivar, and M. Postigo Málaga: IEEE Internet Things J. **8** (2021) 9278. https://doi.org/10.1109/JIOT.2021.3056586
6 P.-H. Kuo, R.-J. Liou, P. Nilaphruek, K. Kanchanasatian, T.-H. Chen, and R.-M. Lee: Sens. Mater. **33** (2021) 3547. https://doi.org/10.18494/SAM.2021.3580
7 A. Liopa-Tsakalidi, V. Thomopoulos, P. Barouchas, A. D. Boursianis, and S. K. Goudos: Smart Agric. Technol. **9** (2024) 100673. https://doi.org/10.1016/j.atech.2024.100673
8 E. A. Abioye, M. S. Z. Abidin, M. S. A. Mahmud, S. Buyamin, M. H. I. Ishak, M. K. I. A. Rahman, A. O. Otuoze, P. Onotu, and M. S. A. Ramli: Comput. Electron. Agric. **173** (2020) 105441. https://doi.org/10.1016/j.compag.2020.105441
9 J. Xie, Y. Chen, P. Gao, D. Sun, X. Xue, D. Yin, Y. Han, and W. Wang: Comput. Electron. Agric. **201** (2022) 107287. https://doi.org/10.1016/j.compag.2022.107287

10 M. Benzaouia, B. Hajji, A. Mellit, and A. Rabhi: Comput. Electron. Agric. **215** (2023) 108407. https://doi.org/10.1016/j.compag.2023.108407
11 M. Bukhari, S. O. Athar, M. Ullah, and M. N. Aman: IEEE Internet Things J. **11** (2024) 37257. https://doi.org/10.1109/JIOT.2024.3440200
12 A. Morchid, Z. Said, A. Y. Abdelaziz, P. Siano, and H. Qjidaa: Smart Agric. Technol. **11** (2025) 100979. https://doi.org/10.1016/j.atech.2025.100979
13 A. Younes, Z. Elamrani Abou Elassad, O. El Meslouhi, D. Elamrani Abou Elassad, and E.-D. Abdel Majid: Smart Agric. Technol. **7** (2024) 100425. https://doi.org/10.1016/j.atech.2024.100425
14 Y. Lu, D. Chen, E. Olaniyi, and Y. Huang: Comput. Electron. Agric. **200** (2022) 107208. https://doi.org/10.1016/j.compag.2022.107208
15 W. Zhai, G. Mo, Y. Xiao, X. Xiong, C. Wu, X. Zhang, Z. Xu, and J. Pan: Comput. Electron. Agric. **216** (2024) 108457. https://doi.org/10.1016/j.compag.2023.108457
16 K. A. Jani and N. K. Chaubey: IEEE Internet Things J. **9** (2022) 11275. https://doi.org/10.1109/JIOT.2021.3128161
17 R. Rahmatullah, H. Gao, R. P. Utama, P. D. P. Adi, J. Mubashir, R. Muwardi, W. D. Gustian, H. Dwiyanti, and Yuliza: Int. J. Adv. Comput. Sci. Appl. **16** (2025) 296. https://doi.org/10.14569/IJACSA.2025.0160129
18 A. King: Weather Long-term Time Series Forecasting, Kaggle: https://www.kaggle.com/datasets/alistairking/weather-long-term-time-series-forecasting (accessed July 2026).
19 World Meteorological Organization: Guide to Instruments and Methods of Observation, Vol. I: Measurement of Meteorological Variables, WMO-No. 8 (World Meteorological Organization, Geneva, 2021) https://library.wmo.int/viewer/68695/ (accessed July 2026).
20 S. A. Asklany, K. Elhelow, I. K. Youssef, and M. Abd El-Wahab: Atmos. Res. **101** (2011) 228. https://doi.org/10.1016/j.atmosres.2011.02.015
21 A. Y. Ardiansyah, R. Sarno, and O. Giandi: Proc. 2018 Int. Conf. Information and Communications Technology (ICOIACT) (IEEE, 2018) 848–854. https://doi.org/10.1109/ICOIACT.2018.8350711
22 B. A. Kindhi, M. I. Triana, U. L. Yuhana, S. Damarnegara, F. Istiqomah, and M. H. Imaaduddiin: Proc. 2022 IEEE Int. Conf. Communication, Networks and Satellite (COMNETSAT) (IEEE, 2022) 67–72. https://doi.org/10.1109/COMNETSAT56033.2022.9994512
23 V. Thomopoulos, F. Tolis, T.-F. Blounas, D. Tsipianitis, and A. Kavga: Smart Agric. Technol. **8** (2024) 100446. https://doi.org/10.1016/j.atech.2024.100446
24 I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville: Adv. Neural Inf. Process. Syst. **30** (2017) 5767–5777. https://arxiv.org/abs/1704.00028
25 J. Simla A., R. Chakravarthy, and M. Leo L.: Meas. Sens. **24** (2022) 100470. https://doi.org/10.1016/j.measen.2022.100470

About the Authors



Rizky Rahmatullah received his B.Eng. (S.T.) degree in electrical engineering from Mercu Buana University, Jakarta, Indonesia, in 2022, and M.Sc. degree from Beijing Institute of Technology, Beijing, China, in 2025. He is currently pursuing a Ph.D. degree with the Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa, Japan. His research interests include the Artificial Intelligence of Things (AIoT), machine-learning-based forecasting, and computer vision. His awards include the Japanese Government Scholarship, awarded through the Ministry of Education, Culture, Sports, Science, and Technology (MEXT), for his Ph.D. study, and the Chinese Government Scholarship (CSC) for his M.Sc. study. (rizk032@stu.kanazawa-u.ac.jp)



Hidetaka Nambo (Member, IEEE) received his B.E., M.E., and D.E. degrees in electrical and information engineering from Kanazawa University, Kanazawa, Japan, in 1994, 1996, and 1999, respectively. He is currently a professor at the School of Entrepreneurial and Innovation Studies, College of Transdisciplinary Sciences for Innovation, Kanazawa University. His research focuses on monitoring systems using living plants, data mining, image classification with machine learning and deep learning, and related areas. His work uniquely bridges the gap between technology and emotional well-being, offering innovative perspectives on how AI can enhance human lives. He is an active member of the IEE of Japan, the IEICE of Japan, and the IPSJ of Japan