Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

AquaGuard: An IoT-Enabled Real-Time Water Leak Detection and Automated Shutoff System Using Isolation Forest

Deekshitha MV
Dept. of CSE
Atria Institute of Technology
Bengaluru, India
1AT23CS040

Prof. Nayanika Saha
Dept. of CSE
Atria Institute of Technology
Bengaluru, India
Assistant Professor

Aishwarya V
Dept. of CSE
Atria Institute of Technology
Bengaluru, India
1AT23CS008

Ananya Nityanand Naik
Dept. of CSE
Atria Institute of Technology
Bengaluru, India
1AT23CS015

Chandana P
Dept. of CSE
Atria Institute of Technology
Bengaluru, India
1AT23CS031

Abstract - Water loss caused by undetected leakage remains a persistent inefficiency in residential and hostel-scale buildings, where consumption is typically verified only through monthly billing rather than continuous measurement. This delay lets leaks progress for weeks before discovery, compounding structural damage and financial loss. This paper presents AquaGuard, an IoT-enabled monitoring platform that detects abnormal per-floor water-flow behaviour within seconds of onset using an unsupervised Isolation Forest model. We give a formal statement of the detection task as a scoring function over a per-floor, per-time-slot feature space, and specify the flow-to-feature pipeline, the Isolation Forest anomaly score, and the alert-band rule that maps a score to a dashboard state. The system pairs ESP32 microcontrollers and YF-S201 flow sensors with an MQTT communication layer, a FastAPI/PostgreSQL backend, and a React dashboard that can trigger an automated solenoid-valve shutoff once an anomaly is confirmed. Bench testing under a simulated leak shows the system separating a controlled leak (anomaly score 0.97) from stable baseline flow. We describe the architecture, the stage-wise formulation, the requirements, and the planned evaluation protocol, and outline the open problems that remain: multi-floor validation, false-alarm-rate quantification, and long-term baseline drift.

Keywords - Internet of Things, water leak detection, Isolation Forest, anomaly detection, ESP32, MQTT, smart buildings, unsupervised learning.

I. INTRODUCTION

Treated water is costly to produce and distribute, yet a substantial share of it never reaches the end user. World Bank estimates place non-revenue water losses above one-third of treated volume in many developing countries, climbing to roughly half in the worst-affected regions. A meaningful part of this loss originates inside building-level plumbing rather than in the distribution network itself, where a leaking joint or fitting can run undetected for an extended period. Because most residential and commercial buildings still rely on manual, monthly meter readings rather than continuous monitoring, floor-wise consumption is effectively invisible between billing cycles.

A. Motivation

Three developments make continuous, floor-level leak monitoring practical today. Commodity microcontrollers such as the ESP32 combine adequate processing headroom, built-in Wi-Fi, and low unit cost, removing the need for a dedicated industrial data-acquisition system. Lightweight messaging protocols such as MQTT let many low-power sensor nodes report to a central backend without the overhead of a full HTTP stack. Finally, unsupervised anomaly-detection methods such as Isolation Forest remove the traditional requirement for a labelled dataset of past leaks, making per-building, per-floor model training realistic without a centralized leak archive.

B. Proposed System

AquaGuard ingests per-floor flow measurements and returns a real-time normal/watch/alert status per floor, together with an automated shutoff path when an anomaly is confirmed. Section IV gives the composition this pipeline implements formally; Section V and Section VI describe the architecture and methodology that realize it.

C. Paper Organization

Section II reviews related work. Section III states the problem and the research gap it leaves open. Section IV gives the formal, stage-wise formulation of the detection pipeline. Section V and VI describe the system architecture and methodology. Section VII specifies requirements, Section VIII sets out the principal use cases, Section IX reports bench-test results, Section X discusses the evaluation plan and threats to validity, and Section XI concludes.

II. RELATED WORK

Existing approaches to water-leak detection differ in sensing modality, learning strategy, and deployment scale; these differences matter when the target is floor-level monitoring inside a single multi-storey building rather than a distribution network.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

A. IoT-based monitoring and feature-driven detection

Ali et al. [1] proposed an IoT framework for water management that streams sensor readings to a cloud platform for monitoring and leak identification, demonstrating the feasibility of continuous, remote data acquisition without addressing per-floor anomaly modelling. Wu et al. [2] used high-frequency pressure data and multi-feature extraction to flag leakage-related changes in distribution networks, showing the value of engineered features over raw threshold checks — a principle this work adapts to flow-rate rather than pressure signals.

B. Supervised and deep-learning approaches

Coelho et al. [3] benchmarked Random Forest, Decision Tree, Neural Network, and Support Vector Machine classifiers on real sensor data and reported 75% leak-detection accuracy; their dependence on labelled leakage events for training illustrates the limitation this work is designed to avoid. Boujelben et al. [4] combined acoustic sensing with a lightweight one-dimensional CNN for detection and a regression model for localization, reporting 97% detection accuracy, at the cost of acoustic hardware that a flow-based approach does not require. Sousa et al. [5] applied supervised machine-learning classifiers to network-level leakage detection, reinforcing the broader pattern that most published systems assume a labelled training set is available.

C. Unsupervised and limited-label learning

Liu, Zayed, and Xiao [6] investigated contrastive learning for leak detection and showed it remains useful when only a small number of labelled signals exist, pointing toward the same underlying gap that motivates this work. This paper instead adopts the original Isolation Forest method of Liu, Ting, and Zhou [7], an unsupervised technique that isolates anomalous observations through randomized partitioning rather than comparison against labelled examples.

TABLE I. Comparison of Representative Water-Leakage Detection Approaches

System	Method	Limitation
Ali et al. [1]	IoT + cloud	No per-floor anomaly model
Wu et al. [2]	Pressure feature extraction	Network-scale, not building-level
Coelho et al. [3]	RF / DT / NN / SVM	Requires labelled leak data
Boujelben et al. [4]	Acoustic + lightweight CNN	Needs acoustic hardware
Sousa et al. [5]	Supervised ML	Requires labelled leak data
Liu et al. [6]	Contrastive learning	Still needs some labelled signals
Proposed	Isolation Forest	Single-floor prototype, bench-tested only

D. Research Gap

The reviewed studies confirm that IoT sensing and machine learning can support automated leak detection, but the strongest-performing systems either require labelled leakage data for training or rely on acoustic or pressure hardware not typically installed inside a single residential building. AquaGuard is scoped to a building-level, multi-floor setting that learns a normal-usage baseline per floor and per time slot without a labelled leak history.

III. PROBLEM STATEMENT AND RESEARCH GAP

A. Problem Statement

No low-cost, floor-level solution currently exists for real-time detection of abnormal water flow in residential and hostel-scale buildings. Fixed-threshold systems either raise false alarms during legitimate high-usage periods or fail to catch slow leaks that never cross the configured threshold. Systems that avoid this trade-off tend to be designed for industrial or municipal-network scale and are priced accordingly.

B. Problem Definition

The system must accept a stream of per-floor flow readings and produce, for each reading, a status in {Normal, Watch, Alert} without requiring a labelled history of past leaks, while additionally (i) attributing the status to a specific floor, (ii) accounting for expected variation across time of day, and (iii) triggering an automated shutoff when the status reaches Alert. Section IV states this formally.

C. Objectives

- Detect abnormal water-flow conditions within seconds of occurrence rather than weeks, and notify the building administrator through a real-time dashboard and alert.
- Scale to a multi-floor building using a single ESP32 microcontroller per building and one flow sensor per floor.
- Achieve reliable anomaly-detection accuracy without requiring a labelled history of past leaks.

D. Scope

The current implementation targets residential apartments and hostel buildings, using a single floor-level sensor as the initial deployment configuration, with the architecture designed to extend to buildings of five or more floors without a redesign of the underlying pipeline.

IV. FORMAL PROBLEM FORMULATION

Let f_i denote the flow-rate reading at index i for a given floor, and let τ(i) ∈ {Morning, Afternoon, Evening, Night} be its time slot. The detection task is the composition

$$F = A \circ s \circ \varphi, \quad F(f\_i) = \text{alert-status} \in \{\text{Normal, Watch, Alert}\} \quad (1)$$

where φ builds the feature vector, s is the Isolation Forest anomaly score, and A is the alert-band rule. Fig. 2 shows the stages and the intermediate quantity each one produces.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

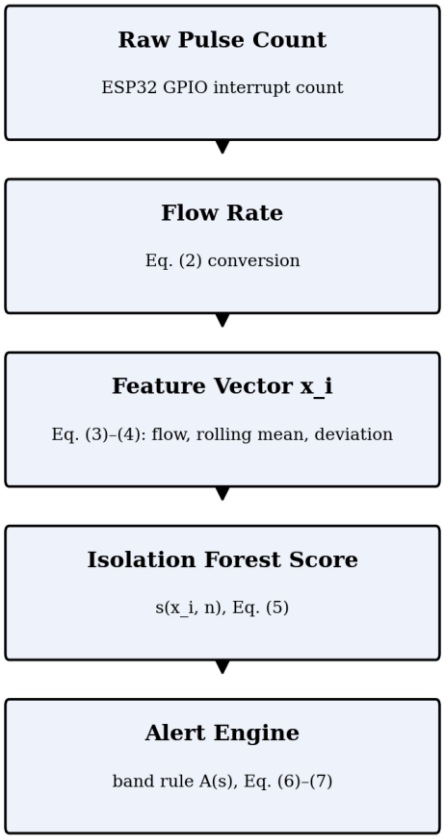

Fig. 2. Processing pipeline and the intermediate quantity produced by each stage, following the composition in (1).

A. Feature Construction

The raw pulse count is converted to a flow rate at the sensing device,

$$f_i\ (L/hr) = (PulseCount_i / 7.5) \times 60 \quad (2)$$

The five-sample rolling mean and the slot-relative deviation are then computed as

$$m_i = (1/5)\ \Sigma_{j=i-4}^{i}\ f_j\ , \quad d_i = (f_i - \mu_{\tau(i)}) / \sigma_{\tau(i)} \quad (3)$$

where $\mu_{\tau(i)}$ and $\sigma_{\tau(i)}$ are the mean and standard deviation of historical flow for time slot $\tau(i)$ on that floor. The resulting feature vector is

$$x_i = [f_i, m_i, d_i] \in \mathbb{R}^3 \quad (4)$$

B. Isolation Forest Score

For an ensemble of t randomized isolation trees, each isolating x_i after $h(x_i)$ partitioning steps, the anomaly score is

$$s(x_i, n) = 2^{ - E[h(x_i)] / c(n) }, \quad c(n) = 2H(n-1) - 2(n-1)/n \quad (5)$$

where $E[h(x_i)]$ is the average path length over the ensemble, n is the training sub-sample size, $H(k)$ is the k-th harmonic number, and $c(n)$ normalizes path length so that $s(x_i, n) \to 1$ for a strongly isolated (anomalous) point and $s(x_i, n) \to 0.5$ or below for a typical point. Fig. 3 sketches how the score responds to the deviation feature d_i for a few illustrative values of the rolling mean m_i; the curves are schematic, intended to show the shape of the response rather than a fitted relationship.

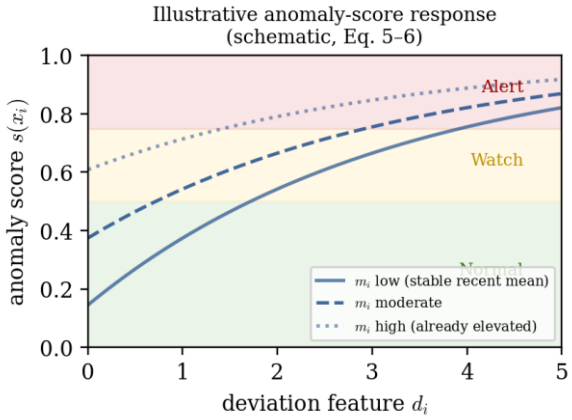

Fig. 3. Illustrative anomaly-score response to the deviation feature (schematic, Eq. 5–6).

C. Alert-Band Rule

The score is mapped to a discrete status by

$$A(s) = \{ Normal,\ 0.00 \le s < 0.50;\ Watch,\ 0.50 \le s < 0.75;\ Alert,\ 0.75 \le s \le 1.00 \} \quad (6)$$

A shutoff action is triggered only when the Alert status persists for k consecutive readings, which limits the effect of a single noisy sample on the solenoid valve:

$$Shutoff_i = 1[\ A(s_j) = Alert\ \forall j \in \{i-k+1, \ldots, i\}\ ] \quad (7)$$

D. Backend Ingestion Latency

The backend is stateless per MQTT message, so with c worker processes each ingesting and scoring at rate μ readings/s under offered load λ, the expected per-reading response time follows the standard M/M/1-per-worker approximation

$$W = (1/\mu) \cdot 1/(1-\rho), \quad \rho = \lambda / (c \cdot \mu) < 1 \quad (8)$$

which sets the worker count needed to keep ingestion latency within budget as the number of monitored floors grows. Fig. 6 (Section VI) plots this relation for illustrative μ; the curves are a design-time queueing model, not measured latencies.

V. SYSTEM ARCHITECTURE

AquaGuard is organized into four functional layers — Hardware, Communication, Backend and AI, and Frontend — that separate sensing, transport, analysis, and visualization into independently maintainable components, shown in Fig. 1.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

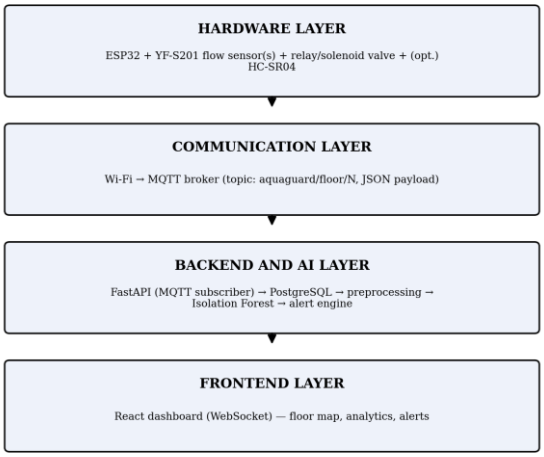

Fig. 1. Four-layer architecture of AquaGuard.

A. Hardware Layer

The Hardware Layer senses and controls water-flow conditions. Its principal components are the ESP32 microcontroller, one or more YF-S201 flow sensors, a relay module, and a solenoid valve; an optional HC-SR04 ultrasonic sensor monitors the level of a rooftop storage tank. Because one ESP32 can address multiple sensors on separate GPIO pins, the same board scales from a single sensor to several floors without any change to the underlying design.

B. Communication Layer

The Communication Layer moves sensor measurements from the ESP32 to the backend over Wi-Fi using MQTT. Each ESP32 publishes a JSON payload containing a floor identifier, flow rate, and timestamp to a topic of the form aquaguard/floor/N, and the broker forwards matching messages to the subscribed backend service.

C. Backend and AI Layer

FastAPI acts as both the MQTT subscriber and the REST service; on each new message it validates the payload, writes the reading to PostgreSQL, computes the feature vector of Eq. (4), and scores it under Eq. (5)–(6). A confirmed anomaly triggers the alert engine, which updates the record shown on the dashboard and, under Eq. (7), the solenoid valve.

D. Frontend and Monitoring Layer

The Frontend Layer is a React application that presents current flow readings, floor-wise status, historical usage, and alert records through a single dashboard, updated over WebSocket without a manual refresh. Fig. 5 places these four layers alongside the data store and external actuation/notification paths.

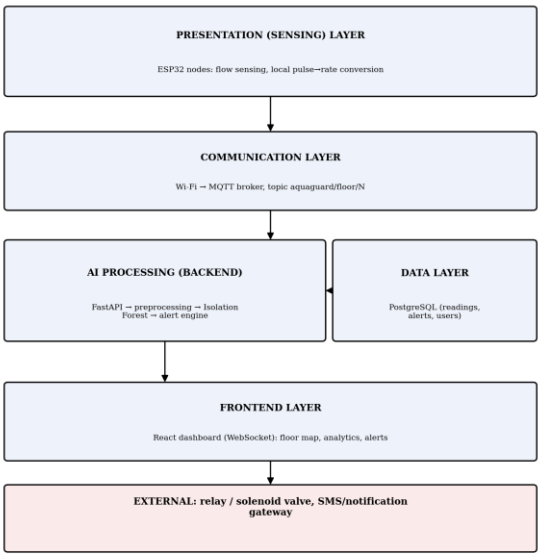

Fig. 5. Layered architecture with data and external paths. The AI Processing block implements Eq. (4)–(7) and holds no per-request state, which is what permits the horizontal scaling modelled by Eq. (8).

VI. METHODOLOGY

A. Time-Slot Classification and Missing Data

Because normal usage varies through the day, each reading is assigned to one of four time slots — Morning (06:00–10:00), Afternoon (10:00–18:00), Evening (18:00–23:00), and Night (23:00–06:00) — so $\mu_{\tau(i)}$ and $\sigma_{\tau(i)}$ in Eq. (3) are computed separately per slot. A missing reading is stored as NULL rather than zero, since zero flow is itself a valid measurement; missing observations are excluded from training, and three or more consecutive missing readings are treated as a possible sensor or link fault.

B. Normalization and Training-Data Outlier Removal

Features are normalized separately per floor and time slot using a StandardScaler. Training data are additionally screened with the interquartile-range method, excluding observations above Q3 + 3×IQR so that unusually high early-stage or setup-related readings do not distort the learned baseline before Eq. (5) is fitted.

C. Model Training and Inference

Observations are grouped by floor and time slot, and the feature vector of Eq. (4) for each valid reading is used to fit the Isolation Forest of Eq. (5). No individual reading needs to be labelled as leak or non-leak. At inference time, the backend repeats the preprocessing pipeline for each new measurement and evaluates Eq. (5)–(6) against the trained model, as summarized in Fig. 2.

D. Backend Scaling

Because the AI layer holds no per-request state, additional floors are handled by adding MQTT topics and, if ingestion approaches the latency budget of Eq. (8), additional FastAPI worker processes rather than by re-

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

architecting the service. Fig. 6 illustrates the relationship Eq. (8) predicts between offered load and response time for a small number of workers.

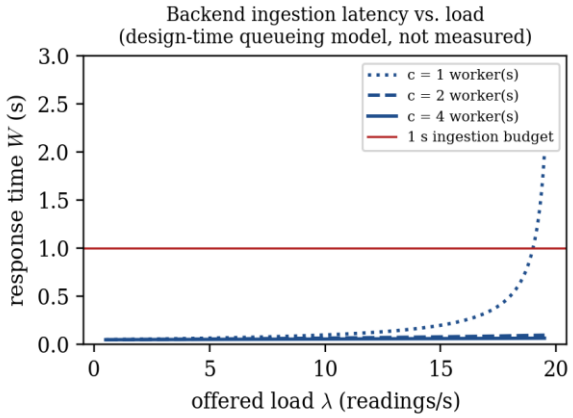

Fig. 6. Backend ingestion latency vs. offered load, Eq. (8). Design-time queueing curves, not measured values.

VII. Requirements Specification

A. Functional Requirements

ID	Description
FR-01	Acquire per-floor flow-rate readings from YF-S201 sensors via ESP32 and publish them over MQTT.
FR-02	Tag each reading with a time slot per Section VI-A.
FR-03	Compute the feature vector of Eq. (4) and score it under Eq. (5).
FR-04	Present current flow, tank level, and floor-wise status on a real-time dashboard.
FR-05	Generate and log an alert per the band rule of Eq. (6).
FR-06	Trigger an automated solenoid-valve shutoff per the persistence rule of Eq. (7).

B. Non-Functional Requirements

The dashboard is a React single-page application updated over WebSocket. The backend, built on FastAPI with a PostgreSQL data store, is designed to add floors by subscribing to additional MQTT topics rather than by re-architecting the service, subject to the worker-scaling relation of Eq. (8).

C. Software and Hardware Stack

Category	Technology	Role
Microcontroller	ESP32	Sensing, local flow-rate conversion, Wi-Fi
Flow sensor	YF-S201	Pulse-based flow measurement
Tank sensor	HC-SR04 (optional)	Ultrasonic tank-level monitoring
Messaging	MQTT	Sensor-to-backend transport
Backend API	FastAPI	MQTT subscriber, REST service
Database	PostgreSQL	Readings, floors, alerts, users
ML model	Isolation Forest (scikit-learn)	Unsupervised anomaly scoring
Frontend	React + WebSocket	Real-time monitoring dashboard

VIII. SYSTEM MODEL

The principal use cases, each with its main actor and pre/postcondition pair:

- Registration and authentication. Actor: administrator. Pre: valid account credentials. Post: an authenticated session.
- Sensor onboarding. Actor: administrator. Pre: an ESP32 flashed with floor identifier. Post: the floor appears on the dashboard and begins contributing to μ_{τ}, σ_{τ} in Eq. (3).
- Real-time monitoring. Actor: system. Pre: a valid MQTT reading. Post: a status in {Normal, Watch, Alert} per Eq. (6).
- Alert review. Actor: administrator. Pre: a logged alert. Post: the alert is inspected and marked resolved.
- Automated shutoff. Actor: system. Pre: the persistence condition of Eq. (7) holds. Post: the solenoid valve for the affected floor closes and the action is logged.

IX. RESULTS AND DISCUSSION

The prototype was exercised under normal operation and under a controlled leakage condition on a single-floor bench rig. During normal operation, flow stayed within the expected range for each time slot and the dashboard reported a Normal status throughout. A controlled leak was then introduced by opening a bypass valve; the Isolation Forest model flagged the resulting change as anomalous, and the dashboard raised a leakage alert that was recorded in the alert history.

For the recorded event, the dashboard reported a flow rate of 318 L/hr against an average baseline of 48 L/hr for that floor and time slot, yielding an anomaly score of 0.97 under Eq. (5) — inside the Alert band of Eq. (6). Fig. 4 gives an illustrative reconstruction of this episode; the exact per-second sensor log is not reproduced here.

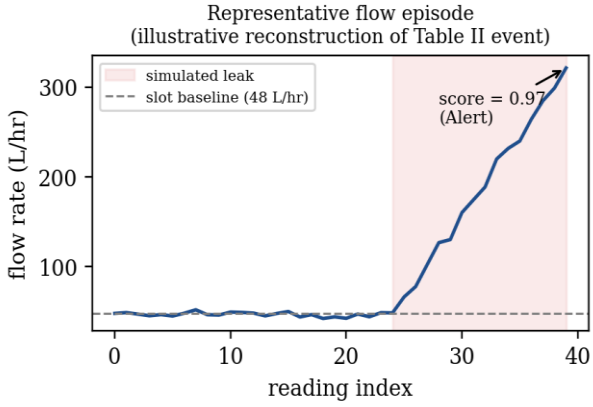

Fig. 4. Illustrative reconstruction of the recorded bench-test episode (Table II).

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

TABLE II. Observed System Response

Condition	Flow Behaviour	Model Output	Dashboard Response
Normal	Stable	Normal pattern	Normal status
Simulated leak	Increased	Abnormal pattern	Leakage alert
After detection	Reading stored	Event identified	Alert history updated

These bench results indicate that the pipeline of Fig. 2 — sensing, MQTT transport, Isolation Forest scoring, and dashboard visualization — functions as intended for a single simulated leak on a single floor. The current evaluation does not yet establish a false-positive or false-negative rate across a realistic range of everyday usage patterns; that quantification is identified as necessary follow-up work in Section X.

X. EVALUATION PLAN AND THREATS TO VALIDITY

Because AquaGuard has so far been tested on a single-floor bench prototype, several open items remain. Score calibration: the thresholds in Eq. (6) have been validated against one controlled leak scenario; a systematic evaluation across varied leak sizes, floors, and usage patterns is needed to fix reliable operating points. Missing-data heuristic: the three-consecutive-reading rule of Section VI-A has not yet been validated against real network-interruption patterns. Scale: the architecture assumes one ESP32 per building; buildings with a larger physical footprint may require additional boards or a revised GPIO-to-floor mapping, and the worker-scaling relation of Eq. (8) has not yet been measured against a real MQTT load. Baseline drift: μ_{τ} and σ_{τ} in Eq. (3) are learned once from initial data; longer-term field data is needed to confirm the baseline remains representative as occupancy and usage habits change over a full year.

XI. CONCLUSION

AquaGuard composes floor-level flow sensing, MQTT-based communication, an Isolation Forest anomaly score (Eq. 5), and an alert-band rule (Eq. 6) into a single low-cost pipeline for building-level water-leak detection, with an automated shutoff path (Eq. 7) and a queueing-based account of backend scaling (Eq. 8). Bench testing under a simulated leak showed the system correctly distinguishing an abnormal flow condition from stable baseline usage. Because Isolation Forest requires no labelled leak examples, the approach is well suited to individual buildings with no historical leak record to draw on. Future work will extend testing to multiple floors, quantify detection accuracy and false-alarm rate at scale, and evaluate long-term baseline drift.

Acknowledgement

The authors thank Atria Institute of Technology, Bengaluru, for the facilities and resources that supported this work, and their project guide for guidance and feedback throughout its development.

REFERENCES

[1] A. S. Ali, M. N. Abdelmoez, M. Heshmat, and K. Ibrahim, “A solution for water management and leakage detection problems using IoTs based approach,” Internet of Things, vol. 18, Art. no. 100504, 2022, doi: 10.1016/j.iot.2022.100504.

[2] X. Wu, S. Peng, G. Zheng, X. Fang, and Y. Tian, “Leakage detection in water distribution networks based on multi-feature extraction from high-frequency pressure data,” Water, vol. 15, no. 6, Art. no. 1187, 2023, doi: 10.3390/w15061187.

[3] J. A. Coelho, A. Glória, and P. J. A. Sebastião, “Precise water leak detection using machine learning and real-time sensor data,” IoT, vol. 1, no. 2, pp. 474–493, 2020, doi: 10.3390/iot1020026.

[4] M. Boujelben, Z. Benmessaoud, M. Abid, and M. Elleuchi, “An efficient system for water leak detection and localization based on IoT and lightweight deep learning,” Internet of Things, vol. 24, Art. no. 100995, 2023, doi: 10.1016/j.iot.2023.100995.

[5] D. P. Sousa, R. Du, J. M. Barros da Silva Jr., C. C. Cavalcante, and C. Fischione, “Leakage detection in water distribution networks using machine-learning strategies,” Water Supply, vol. 23, no. 3, pp. 1115–1126, 2023, doi: 10.2166/ws.2023.054.

[6] R. Liu, T. Zayed, and R. Xiao, “Contrastive learning method for leak detection in water distribution networks,” npj Clean Water, vol. 7, Art. no. 118, 2024, doi: 10.1038/s41545-024-00406-6.

[7] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation Forest,” in Proc. 8th IEEE Int. Conf. Data Mining (ICDM), Pisa, Italy, 2008, pp. 413–422, doi: 10.1109/ICDM.2008.17.