AquaShield: An Isolation Based IoT & Digital Twin Approach for Automatic Leak Detection and Shutdown

Akanksha Dubey[1], Bhoomika M[2], Deeksha R[3], Inchara B S[4], Mrs. Pallavi R[5]
[1,2,3,4]*Department of Computer Science and Engineering, Jyothy Institute of Technology, Bengaluru, India*
[5]*Project Guide, Department of Computer Science and Engineering, Jyothy Institute of Technology, Bengaluru, India*

***Abstract*—Leakage in water distribution pipelines without detection not only leads to the wastage of treated water but also raises the operational cost and damages the structure, but all cheap detection systems just send alarm and leave the isolation to humans. In this paper, an IoT system named AquaShield is proposed which detects the leak and also automatically isolates the faulty branch. The flow sensors using YF-S201 sensor are used in the main pipe and in each of the branches using ESP32. Leaks are detected from three criteria which are the difference between the flow rate in the main-line and the total flow rate in the branches, imbalance between the flow rate in the two branches and flow in the branch while its valve is closed. Using debounce counter, the leak will be detected if in N=3 continuous one-second reading, there are indications for leak detection. Also, a solenoid valve operated through relay will close the faulty branch and isolate it and keep the other branch open to supply the water. The valve control is implemented within the microcontroller so that even in case of the failure of the communication network, isolation process continues. Also, data is sent to React-based web application which includes 3D digital twin, notifications, and citizen complaints handling. All test cases were done successfully and Monte-Carlo simulation of the firmware's decision-making process shows mean latency of 2.8 second, minimum size of leak of 0.65 L/min and false isolation probability of 0.2 percent in 10-minute leak-free period.**

***Index Terms*—Automatic valve shut-off, Digital Twin, ESP32, Firebase, IoT, Smart Water Management, Water Leakage Detection.**

I. INTRODUCTION

There is also considerable loss of water from such water distribution networks due to leakage, which often goes unnoticed because it is a manual process and the response is often delayed [1] - [3]. Burst pipes cause pressure imbalances, among others, in addition to wastage of energy during the pumping process.

The idea of IoT makes continuous monitoring feasible in an economic way [23]. An economical microcontroller such as ESP32 may be used to acquire data from the sensors for flow, pressure, and moisture and send the data to the cloud. However, in most situations, monitoring takes place up to the level of alarm or even further [4], [5]. The valve is closed manually while there is no live representation of the system network by the operator.

In the paper, AquaShield is described as a prototype for bridging this gap in a small branched pipeline system.

The prototype identifies the leakage, confirms it, and isolates only the leaking branch without affecting the others. This process is performed through the use of a web dashboard that features the three-dimensional digital twin of the pipeline network. Citizenry can report any leaks within their localities, and such leaks are allocated to the engineers from the water department.

Contributions made by this paper include:

1) Detection method that employs the flow equilibrium between the trunk and branch pipes, flow between branches, and flow without the valve being opened.
2) Stage of confirmation with debouncing to reduce the false shutdowns caused by noise and variations in demand; there exist explicit solutions for the time taken by the detector and the minimum leak detected.
3) A self-latching level of branch isolation with a manual override and software level fail-safe function that works independent of network connection.

4) Real time dashboard made up of digital twin visualizations, alert center, and workflow for citizens complains and engineers' task allocation.
5) Testing for functionality and evaluation of quantitative performance in terms of latency in detection, false positives and negatives, shutting off valves offline and saving water.

The rest of this paper is organized as follows. In Section II, there will be provided a review of the literature as well as a gap in research. In Section III, the problem formulation will be done. The architecture of the system will be described in Section IV, while its implementation will be covered in Section V. In Section VI, the detection method will be described including calibration, time measurement and parameter adjustment.

II. RELATED WORKS AND RESEARCH GAP

The leak detection approaches can be divided into acoustic, learning, pressure, and modeling, and IoT threshold approach. Reviews of systematic literature show that machine learning and threshold-based algorithms have dominated the area, and flow, acoustic, and vibration sensors have been the most common ones due to their low cost and ease of installation [1]. Another review considers intelligent approaches which were applied to detect and locate leaks as well as open problems in leak detection of large-scale systems [2], [3]. Comparison of surveyed approaches and AquaShield is presented in Table I.

A. Acoustic and Learning-Based Detection

Even though the acoustic sensors can be precise, they cannot be installed easily. As part of their study, Bakhtawar et al. use fast Fourier transform features in conjunction with ensemble learning techniques like random forest and AdaBoost for classifying the hydrophones; but this technique needs specific permissions and is tough to install [6]. In order to classify the vibration, pressure, and acoustic sensors, Liu et al. have used frequency-aware Transformer; but the technique is computationally costly and complicated to calibrate [7].

Jackson uses convolutional leak classification on resource-limited low-power devices such as Arduino (TinyML); but the technique relies on the acoustic signal and is affected by the environmental noise [8]. The detection accuracy of Kang et al. using a graph-based localization technique with 1D CNN and SVM is 99.3 %, with a localization error of less than 3 m [9].

B. Pressure, Model-Based, and Sensor-Network Methods

Pressure sensing and sensor network technologies are relatively inexpensive. In their work, Leonzio et al. implement a convolutional feature extraction technique along with the one-class support vector machine using only non-leakage pressures [10]. In addition, Pérez et al. detect leakage using a model-based system that applies pressure sensors to an operational network in Barcelona [11].

On the other hand, Primawan applies fuzzy logic to flow, soil moisture, and pressure sensors that is relatively easy and limited by the rules [12]. Investigations on underground wireless sensor networks and hybrid time domain reflectometry and magnetic induction technique prove that there is accurate localization, but the primary issues include communication range, cost, and maintenance [13], [14]. PIPENET is one of the initial wireless sensor network systems applied to pipelines [15].

C. Low-Cost IoT Prototypes

IoT-based low-cost devices that detect the flow rate and pressure in the pipeline are being used. The prototypes generate an alert when the measurement crosses a certain threshold value. Andini et al. have compared the measurement with a static threshold value and have generated alerts via a cloud-based service [4]. In the same way, Sateesh et al. have used Raspberry Pi with soil moisture, temperature-humidity, and flow sensors.

D. Digital Twins for Water Systems

The platform that was proposed by López et al. makes use of LSTM and gradient boosting for forecasting water demand with the help of artificial intelligence but needs a lot of sensors and a complex architecture [16].

In general, the domain of digital twins has been surveyed based on the technologies available to create digital twins, their limitations, architecture used in industry-level and Internet of Things level applications, along with the values and enablers from the modeling perspective [17]– [19], [20].

E. Sensor Reliability and Security

The question of the reliability of the sensors is quite another story. Sinha and Das have discovered that low-cost sensors are drifting, getting biased and wearing out, and they propose an approach of deep reinforcement learning which can detect such faults and recalibrate the sensor according to its own past good state [21]. Jan et al. employ a support vector machine to diagnose faults in the sensor using time-domain statistical measures [22]. Such work is devoted to sensor reliability rather than to leak response; however, it provides the foundation for the sensors extension discussed in Section X. Leak response is the actuation of a physical valve and thus it creates challenges for cybersecurity of the system.

F. Research Gap

However, in the case of the system being analyzed, detection is where everything ends. Isolation of the failing section or detecting the operation performed during the disconnection process is not considered. The problem is addressed by AquaShield through an amalgamation of straightforward detection and branch-level valves under the control of the microcontroller and providing a digital twin of the network to the user.

TABLE I Comparison between Surveyed Systems and AquaShield

Aspect	Surveyed systems [4]– [16]	AquaShield (this work)
Leak detection	Thresholds, acoustic or learned classifiers	Flow balance and branch comparison with debounce confirmation
Control response	Alert to operator; valve closed by hand	Automatic closure of the faulty branch valve
Isolation scope	Whole-line shut-off	Only the faulty branch is isolated
Network failure	Not reported	Valve logic runs on the ESP32 and works offline
Digital twin	Simulation platforms [16] with many sensors	Lightweight live 3-D twin with valves, flow, and leak marker
User interface	Basic dashboards	Role-based dashboard with alerts and history
Citizen role	Not considered	Leak reporting and engineer assignment
Cost	Hydrophones, many sensors	ESP32 with off-the-shelf sensors
Scalability	Hardware-limited	A new branch adds one flow sensor and one valve

TABLE I-A Methodology, Implementation, and Results Presented by Surveyed Literature

Ref	Methodology	Implementation	Reported result	Limitation
[4]	Static threshold on measured values	IoT prototype with cloud alerting	Alert raised when threshold is crossed; no valve action	Alert only
[5]	Historical-data learning on flow, soil-moisture, temperature-humidity readings	Raspberry Pi node	Leak alerts from learned data; no branch isolation	No isolation logic
[6]	FFT features, random forest, AdaBoost	Hydrophones	High classification accuracy (Section VIII-A)	Special permissions; hard to install
[7]	Frequency-aware Transformer	Vibration, pressure, acoustic sensors	High accuracy (Section VIII-A)	Heavy computation; hard to calibrate
[8]	Convolutional leak classification (TinyML)	Arduino-class device	Low-power on-device classification	Needs acoustic signals; noise-prone
[9]	1-D CNN + SVM, graph-based localization	Wireless sensor network	99.3 % detection; localization error < 3 m	Larger sensing deployment
[10]	CNN feature extractor + one-class SVM on no-leak data	Pressure sensors at network nodes	Outperforms recent methods on a simulated Modena network dataset	Needs clean no-leak training data

Ref	Methodology	Implementation	Reported result	Limitation
[11]	Model-based localization	Pressure sensors	Applied and tested on a real network in Barcelona	Needs a hydraulic model
[12]	Fuzzy logic rules	Flow, soil-moisture, pressure sensors	Rule-based leak decision	Simple, restrictive rules
[13], [14]	Underground WSN; hybrid TDR–magnetic induction	Buried sensor nodes	Accurate localization	Short range, cost, maintenance
[15]	Early pipeline-monitoring WSN	Wireless sensor network	Pipeline monitoring	Monitoring only
[16]	LSTM and gradient boosting	Digital-twin platform	Demand forecasting	Many sensors; complex architecture
AquaShield	Flow balance + branch comparison + debounce	ESP32, 3× YF-S201, relay-driven valves	2.8 s mean latency; Lmin ≈ 0.65 L/min; 0.2 % false isolation (simulation)	Bench scale; simulation-based figures

The detection techniques in the cited references belong to three categories: thresholding techniques [4], [5], [12], classifier-based approaches [6]-[10], and modeling-based localizations [11]. The methods vary from economical microcontroller-based nodes [4], [8] to acoustic sensor and hydrophone deployments [6], [9], [13]. The accuracy is the highest in classifier-based methods with a large number of nodes [9]; yet none of the referenced studies describes an automated valve activation or functionality during network disruption, leading to the development of AquaShield.

III. PROBLEM FORMULATION

The proposed architecture involves a system of pipes in which a main pipe transfers its flows to two branches, that is, Road 1 and Road 2, where each branch has a solenoid valve installed. The flows in the main pipe and branches are represented by Qm and Qi, respectively, where $i = \{1, 2\}$ in litre/min. The notations used in this paper are as presented in Table II. Leak refers to the flow which goes into waste and does not get transferred as demand.

Leak can be in any one of the following forms:

1) Leak occurring after branch flow sensor causes an increase in Qi than the other branch flow.
2) The leakage between the master and the branch sensors increases Qm more than the addition of the flow in the branches.
3) Water that flows through the branch with a closed valve results in Qi having a positive value.

The measurements are noisy due to turbulent flows, air bubbles, pulse quantization, and electrical noise, and legitimate changes occur whenever the valves are opened or closed. It is, thus, the task of finding out a leak with a small probability of false alarms and shutting off only the faulty branch in order to avoid either missing the leak or shutting off the wrong one. Since the controller measures in equal time intervals Ts and decides about a leak after observing *N* suspect measurement values in a row, the time delay is about *N* Ts.

TABLE II Notation

Symbol	Meaning
Qm, Qi	Flow on the main line and on branch *i* (L/min)
f, P	Pulse frequency (Hz) and calibration factor (pulses per litre)
ΔQ	Flow resolution of one pulse in a window (L/min)
δs, δb	Flow-balance mismatch and branch imbalance
τ	Relative threshold for a suspect sample
si, ci	Suspect indicator and consecutive-suspect counter of branch *i*
N, Li	Confirmation count and confirmed-leak flag of branch *i*
Ts, Td	Sampling period and detection delay (s)
q, Lmin	Branch demand and minimum detectable leak (L/min)

IV. PROPOSED SYSTEM ARCHITECTURE

A. System Overview and Workflow

AquaShield consists of three layers that are illustrated in Fig. 1. Field layer includes ESP32 microcontroller, three YF-S201 Hall-effect flow sensors (main line, Road 1, Road 2), pressure sensor, moisture sensor, two relay modules, and two solenoid valves.

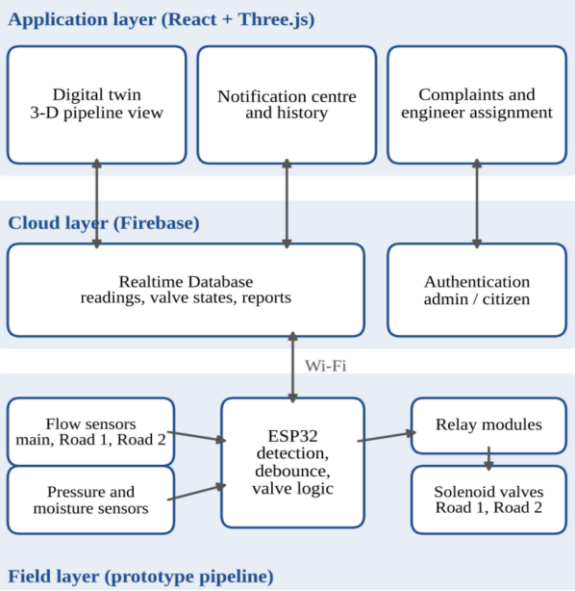


Fig. 1. Layered architecture of AquaShield: field layer, cloud layer, and application layer.

B. Hardware Design

Table III lists the hardware. The three YF-S201 outputs connect to interrupt-capable inputs of the ESP32, so that no pulse is lost while the firmware performs other tasks. Each relay module switches the supply of one solenoid valve, so that the 3.3 V logic of the ESP32 never drives the valve coil directly. The pressure and moisture sensors are fitted for corroboration and for the dashboard; the isolation decision of the present firmware is made from the flow indicators of Section VI. Fig. 2 shows the connections, and Fig. 3 shows the assembled prototype.

TABLE III Hardware Components of the Prototype

Component	Role
ESP32	Field controller, Wi-Fi link, local valve logic
YF-S201 flow sensors (3)	Flow on the main line, Road 1, and Road 2
Pressure sensor	Pressure trend and drop detection
Moisture sensor	Physical confirmation of water at joints
Relay modules (2)	Switch the solenoid valves
Solenoid valves (2)	Close a branch to isolate a leak

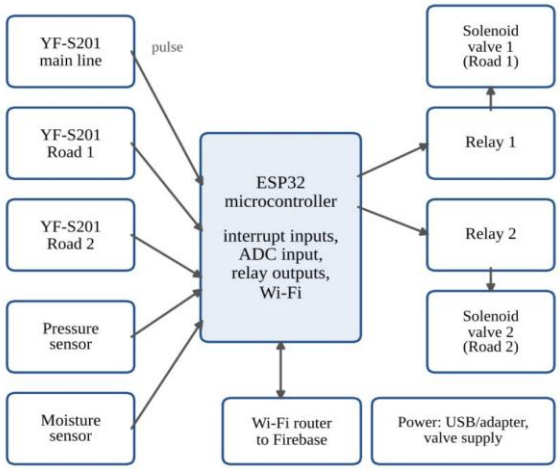


Fig. 2. Hardware connection diagram of the AquaShield prototype.



Fig. 3. Assembled AquaShield prototype: ESP32 controller board, power supply, pump, flow sensors, solenoid valves, and the two-branch test pipeline.

C. Pin Assignment

Table IV gives the pin assignment of the firmware. The three flow inputs are attached to hardware interrupts on the falling edge of the sensor output.

The relay boards are active-low, so a low level on the relay pin keeps the valve open and a high level closes it. Both valves are driven to the open state at start-up so that the system always begins from a known condition.

TABLE IV ESP32 Pin Assignment

Signal	ESP32 pin	Function
Main-line flow pulse	GPIO 34	Interrupt, falling edge
Road 1 flow pulse	GPIO 35	Interrupt, falling edge
Road 2 flow pulse	GPIO 32	Interrupt, falling edge
Relay, Road 1 valve	GPIO 26	Active-low: LOW = open, HIGH = closed
Relay, Road 2 valve	GPIO 27	Active-low: LOW = open, HIGH = closed

D. Data Model

The data model of Table V separates users, pipelines, readings, alerts, valve actions, and citizen reports. Readings and valve states are time-stamped so that the dashboard can draw live and historical graphs, and every automatic valve action is logged with its time and cause. Records are stored in the Firebase Realtime Database. The controller writes its readings under a single root node that holds one child per measurement point (MainFlow, Road1, Road2) and a System child with the pump state, Wi-Fi state, time stamp, and device status. Each branch node carries the flow rate, the cumulative litres, the valve status, and the leak status, and the dashboard subscribes to these paths.

TABLE V Data Entities and Main Fields

Entity	Main fields
User	Identifier, name, e-mail, phone, role (administrator or citizen)
Pipeline	Identifier, name, diameter, location, status
SensorData	Pipeline, timestamp, flow rate, pressure, moisture state
Alert	Pipeline, alert type, severity, timestamp, status
ValveControl	Valve, pipeline, status, action (automatic or manual), timestamp
ReportedLeak	Citizen, phone, area, landmark, description, photograph, time, engineer, status

E. Digital Twin Dashboard

The digital twin view, rendered with Three.js, displays the two-branch layout with live flow values, valve and pump controls, and a marker on the affected pipe section (Fig. 4). It is driven by the same database paths as the controller, so the twin reflects the physical valve states within the update delay of the cloud. When a leak is confirmed, the notification center names the branch, states that its valve was closed automatically, and marks the event as requiring repair (Fig. 5).



Fig. 4. Digital twin dashboard showing the two-branch pipeline, live flow, valve and pump controls, and a leak marker.

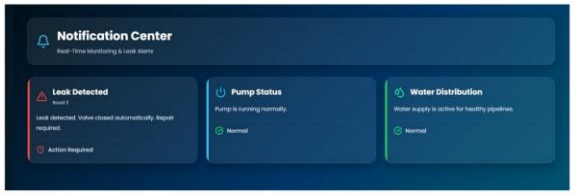


Fig. 5. Notification center reporting a detected leak on Road 2 and the automatic valve closure.

F. User Roles, Complaint Workflow, and Security

Two roles are supported. The administrator, representing the water department, has a dashboard, notifications, history, and complaint management. The citizen has a dashboard with report counts by status, a leak reporting form, a list of personal reports, and a profile. A citizen reports a leak with an area, landmark, description, and photograph (Fig. 7a). The report enters the administrator queue with status pending. The administrator assigns an engineer, which changes the status to in progress, and a repair-completed action sets it to resolved (Figs. 6 and 7b). The citizen follows the status on the personal dashboard.

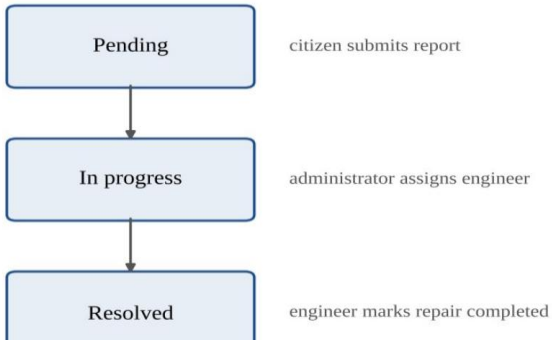


Fig. 6. Status workflow of a citizen leak report.

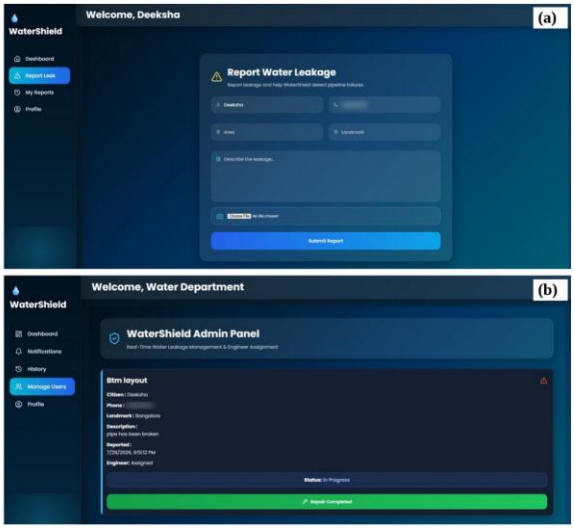


Fig. 7. (a) Citizen leak-reporting form and (b) administrator complaint-management panel (personal details blurred).

Access is controlled by Firebase Authentication with role checks. A citizen can read and write only personal reports, and valve and pump commands are accepted only from the administrator role. Data is exchanged over encrypted connections provided by the platform. Unauthorized valve commands were tested and rejected (Section VII-B). Restricting actuation to authenticated roles follows the general guidance for cyber-physical systems, where a compromised command channel can cause physical harm [24].

V. IMPLEMENTATION TECHNOLOGIES

Table VI lists the technologies used in each layer of the prototype. The controller firmware runs on the ESP32 and handles pulse counting by interrupts, flow computation, the detection logic of Section VI, and relay control, so that isolation does not depend on the cloud. The ESP32 sends readings and valve states to the Firebase Realtime Database over Wi-Fi, and it receives operator commands from the same database.

TABLE VI Implementation Technologies

Layer	Technologies
Controller	ESP32 microcontroller with on-board Wi-Fi; C/C++ firmware; hardware interrupts for pulse counting
Sensors	YF-S201 Hall-effect flow sensors, pressure sensor, moisture sensor
Actuators	Relay modules and solenoid valves (one per branch); water pump
Cloud	Firebase Realtime Database for readings, valve states, alerts, and reports; Firebase Authentication for users and roles
Dashboard	React with Vite; Three.js for the 3-D digital twin; responsive layouts for administrator and citizen roles
Notifications	In-app notification center and leak alert messages to the registered user

A. Firmware Organization

Three interrupt service routines increment one volatile counter each on the falling edge of the corresponding sensor output. The main loop runs a fixed one-second cycle with five steps: (i) the three counters are copied and cleared inside a short critical section; (ii) the pulse counts are converted to flow rates and to cumulative litres; (iii) the leak check and valve control run; (iv) the operator command is read from the database; and (v) the readings and states are printed on the serial port and uploaded. Step (iii) runs before any network call and does not use the network, which is what allows the valves to act during an outage. A Wi-Fi supervisor checks the link every 5 s and re-associates automatically when it is lost.

All three counters are copied at the same instant, so any stretch of the cycle caused by slow network calls scales the three flow readings equally and leaves the ratios δ_s and δ_b of Section VI unchanged. Only the absolute limits and the litre totals depend on the cycle length. Table VII lists the parameters of the firmware.

TABLE VII Firmware Parameters

Parameter	Value	Role
Cycle period Ts	1 s	Pulse window, detection, and upload
Pulse-to-flow factor	7.5	Q = pulses per second / 7.5 (P = 450 pulses/L)
Noise floor	0.05 L/min	Smaller readings are set to zero
Threshold τ	0.15	Balance and branch-comparison limit
Confirmation count N	3 cycles	Debounce before shut-off

Parameter	Value	Role
Balance check active	Q_m > 1.0 L/min	Avoids ratios on idle flow
Branch comparison active	Mean Q_i > 0.5 L/min	Both valves open
Closed-valve limit	0.3 L/min	Valve-fault flag
Wi-Fi check period	5 s	Automatic re-association

B. Manual Override and Reopening

The dashboard writes the requested valve state to the database path of the branch. A request to close is executed at once and marks the branch as manually overridden, and the firmware never overrules it. A request to open is executed only when no leak flag is active for that branch, so that an operator cannot reopen a branch that still leaks. After repair, the operator reopens the latched branch from the dashboard.

VI. METHODOLOGY

A. Flow Measurement and Calibration

The YF-S201 emits pulses whose frequency is proportional to flow. The firmware counts the pulses with hardware interrupts over a fixed one-second window and converts the pulse frequency f to the flow rate Q in litres per minute using the calibration factor P in pulses per litre:

$$Q = \frac{60f}{P} \quad (1)$$

The datasheet value of about 450 pulses per litre is only a starting point, because the factor depends on the pipe fitting and the flow range. It corresponds to the factor 7.5 of Table VII, since $Q = 60f / 450 = f / 7.5$. P is therefore calibrated for each sensor by passing a known volume V_{ref} of water through it, counting the pulses N_{ref}, and computing

$$P = \frac{N_{ref}}{V_{ref}} \quad (2)$$

The calibration is repeated until the measured and collected volumes agree. After calibration, the total volume delivered is obtained from the pulse count N_p as

$$V = \frac{N_p}{P} \quad (3)$$

Because the pulses are counted in a window of length T_s, the flow reading is quantized. One pulse corresponds to

$$\Delta Q = \frac{60}{P T_s} \quad (4)$$

which is 0.133 L/min for P = 450 and T_s = 1 s. At a branch flow of 4 L/min the window holds about 30 pulses, so a single missed or extra pulse changes the reading by about 3.3 %. This quantization sets the noise floor against which the threshold τ must be chosen.

B. Signal Conditioning

Raw pulse trains contain electrical jitter, air-bubble spikes, and turbulence. AquaShield conditions the signal in three light-weight stages that fit the ESP32 loop. First, the one-second counting window acts as a boxcar average of the pulse train. Second, a noise floor of 0.05 L/min sets readings below it to zero, which removes isolated stray pulses on an idle line. Third, the debounce counter of Section VI-D suppresses any disturbance that does not persist for N cycles. A median filter or a moving average over a window of W samples would smooth the readings further, but it would add a delay of about $W\,T_s$ to every decision; it is therefore left to future work.

C. Leak Indicators

Three indicators are evaluated in every cycle. First, the flow balance is evaluated when Q_m exceeds 1.0 L/min. The relative mismatch between the main line and the branches is

$$\delta_s = \frac{|Q_m - (Q_1 + Q_2)|}{Q_m} \quad (5)$$

and the system is flagged when δ_s exceeds the threshold τ. Because such a mismatch cannot be attributed to a single branch, it raises an alert but does not close a valve.

Second, when both branch valves are open and the mean branch flow exceeds 0.5 L/min, the branches are compared with each other:

$$\delta_b = \frac{Q_1 - Q_2}{(Q_1 + Q_2)/2} \quad (6)$$

Road 1 is suspect when $\delta_b > \tau$, and Road 2 is suspect when $-\delta_b > \tau$. The comparison is disabled once a branch is isolated, since the closed branch would otherwise make the open branch appear to be leaking. Third, if the valve of a branch is closed but its flow

exceeds 0.3 L/min, the branch is flagged as a valve fault and an alert is raised.

Pressure and moisture readings are logged and shown on the dashboard as supporting evidence. A sudden pressure drops or a wet moisture sensor near a branch adds weight to the suspicion for the operator, while the shut-off decision itself rests on the three flow indicators.

D. Confirmation by Debounce

A single noisy sample must not shut off a supply. For each branch i, let $s_i(t) \in \{0, 1\}$ indicate whether the branch is suspect in cycle t. A counter c_i grows by one for every consecutive suspect cycle and resets to zero otherwise:

$$c_i(t) = (c_i(t-1) + 1) \cdot s_i(t) \tag{7}$$

A leak is confirmed only when the counter reaches the confirmation count N:

$$L_i(t) = [\, c_i(t) \geq N \,] \tag{8}$$

where $[\cdot]$ equals 1 when the condition holds and 0 otherwise. The threshold τ and the count N are configurable and were tuned during testing to separate leaks from normal tap-like demand changes.

Algorithm 1 gives the complete procedure that runs in every sampling cycle, and Fig. 8 shows the decision flow for one branch.

Algorithm 1 Leak Confirmation and Branch Isolation
1: Input: pulse counts, valve states v1, v2, threshold τ, count N
2: for every sampling cycle do
3: Compute Qm, Q1, Q2 using (1); apply the noise floor
4: if Qm > 1.0 and δs > τ then raise a system alert
5: if v1 and v2 are open and (Q1 + Q2)/2 > 0.5 then
6: s1 ← [δb > τ]; s2 ← [−δb > τ]
7: else s1 ← 0; s2 ← 0
8: for i ∈ {1, 2} do
9: if vi is closed and Qi > 0.3 then raise a valve-fault alert
10: ci ← (ci + 1) · si
11: if ci ≥ N and vi is open then
12: close valve i, latch it closed, raise an alert, and log the event
13: publish readings and valve states to the cloud when connected

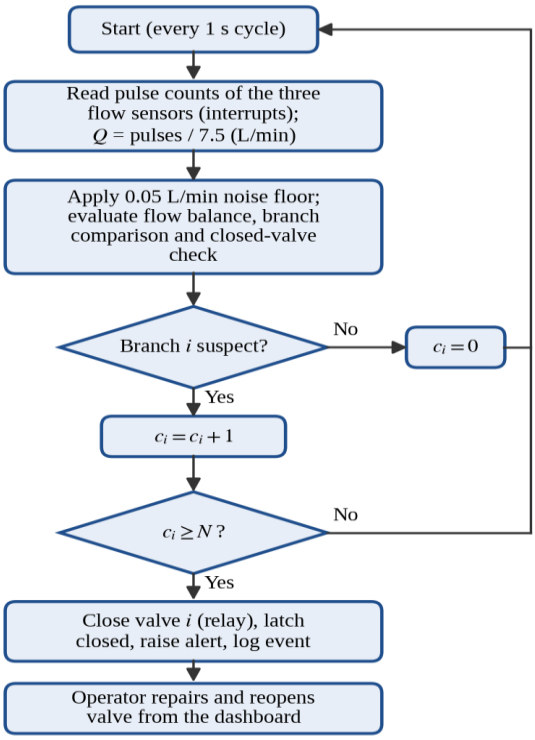


Fig. 8. Flowchart of leak confirmation and automatic branch isolation for branch i.

E. Autonomous Isolation and Fail-Safe

When $L_i = 1$ and the valve of branch i is open, the controller closes that valve through its relay and latches the closed state. The other branch is untouched, so supply continues to unaffected areas. The latched valve is reopened only by the operator from the dashboard after the repair is complete, and an operator command that closes a valve is never overruled by the firmware.

Because this logic runs on the ESP32, the valves still close during a Wi-Fi outage, and the firmware re-associates with the access point when the link returns.

F. Timing and Sensitivity Analysis

Two closed-form results follow from the indicators and the debounce rule and guide the choice of τ and N. First, a leak that is well above the threshold makes every cycle suspect from the first full window, so the confirmed flag is raised after

$$T_d \approx N T_s \tag{9}$$

seconds, which is 3 s for the firmware values; the exact delay lies between $(N - 1)T_s$ and about $N\,T_s$ plus one window, depending on where in a cycle the leak starts.

Second, let q be the demand of each branch and L the leak flow added to Road 1. Then $\delta_b = L / (q + L / 2)$, and the leak is suspect when $\delta_b > \tau$, which gives the minimum detectable leak

$$L_{min} = \frac{\tau q}{1 - \tau/2} \quad (10)$$

For $\tau = 0.15$ and $q = 4$ L/min this is 0.65 L/min. A smaller leak stays below the threshold, and a larger τ raises L_{min} proportionally. A short demand burst on one branch is rejected only if it produces fewer than N consecutive suspect cycles. A burst of duration D that is not aligned with the cycle touches about $D + 1$ windows, so bursts up to $(N - 2)$ Ts are rejected reliably and bursts of N T_s or longer are always confirmed. Section VII-D confirms both statements by simulation.

G. Parameter Calibration and Tuning

AquaShield uses a rule-based detector, so there is no learned model and no training data set. The equivalent of training is a calibration and tuning phase performed once for each installation. First, P of every flow sensor is calibrated by (2) with a known collected volume. Second, the pipeline runs for a period with no leak under normal and varied demand, and the largest branch imbalance δ_b and balance mismatch δ_s are recorded. The threshold τ is then set above these leak-free maxima with a safety margin. Third, a leak is injected through a bypass valve at several flow rates, and the confirmation count N is chosen as the smallest value that rejects the recorded bursts, so that the detection delay N T_s stays within the tolerated loss of water. Learned detection that needs a training set is left to future work.

VII. RESULTS

A. Validation Procedure

Because the system controls a live water supply, testing prioritized safety: a missed leak causes wastage and damage, while a false positive cut off water unnecessarily. Black-box tests ran water through the pipe, simulated leaks with a bypass or a wetted moisture sensor, disconnected sensors, and sent corrupted packets. White-box tests logged raw pulses against computed flow values in the firmware and backend. The levels covered unit tests of flow computation, leak scoring, moisture detection, and valve control; integration tests from sensor to alert and from reading to dashboard; and end-to-end tests from registration to valve reopening after repair.

B. Functional Results

Table VIII lists the ten test cases. The calibration factor, the idle-flow threshold, the confirmation count, the burst-duration filter, the sensor sampling rate, the relay trigger, the data pagination and caching, the access-control checks, and the notification route were tuned during iterative development. All ten cases produced the expected outcome and passed in the final system.

TABLE VIII Functional Test Cases and Results

No.	Test case	Expected outcome	Observed outcome	Result
1	Continuous low flow at idle	Leak alert triggered	Alert triggered	Pass
2	Flow rate against measured volume	Reading matches measured volume	Matched after calibration of P	Pass
3	Normal tap bursts	No alert	No alert; debounce rejects the burst	Pass
4	Moisture sensor wetted	Wet state detected; alert sent	Detected and alert sent	Pass
5	Wi-Fi disconnect mid-session	Auto-reconnect; valve logic unaffected	Reconnected; valve logic unaffected	Pass
6	Confirmed-leak shut-off	Valve closes automatically	Valve closed automatically	Pass
7	Dashboard with large history	Smooth display	Smooth; pagination and caching	Pass
8	Unauthorized valve command	Request denied	Request denied	Pass
9	High-rate data streaming	No dropped packets	No packets dropped; queue buffering	Pass
10	Leak alert SMS delivery	User receives alert instantly	Alert delivered promptly	Pass

The tests show that the detection chain works end to end: a simulated leak is detected, classified, announced to the operator, and isolated by the valve, and normal tap-like usage does not raise an alert. The local fail-safe logic kept the valves functional with Wi-Fi disabled, and unauthorized valve commands were rejected.

C. Illustration of Detection and Isolation

Fig. 9 illustrates the decision logic of Section VI on a simulated scenario. The traces were generated by implementing the indicators and the debounce rule with τ = 0.15, N = 3, a one-second cycle, pulse quantization, and Gaussian flow noise of 0.15 L/min on each sensor; they are not measured data. A two-second flow burst on Road 1 at t = 15 s crosses the threshold, but the counter reaches only 2 and resets, so no valve is closed. A leak on Road 1 starts at t = 50.4 s and raises Q_1 by about 3 L/min. The counter reaches N = 3 at t = 53 s, the Road 1 valve closes, and the main-line flow falls to the flow of Road 2, which is never interrupted.

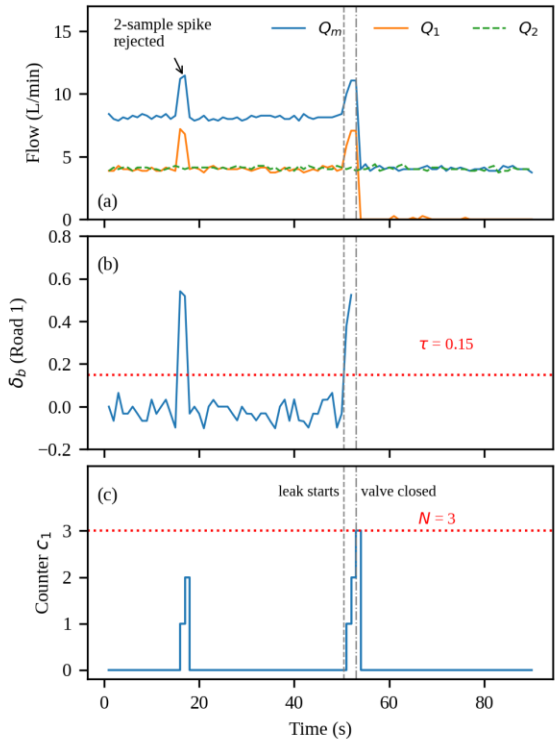


Fig. 9. Simulated detection and isolation of a leak on Road 1: (a) flows, (b) branch imbalance, and (c) confirmation counter. Parameters τ = 0.15 and N = 3 follow the firmware.

D. Quantitative Performance

The functional tests establish correct behavior. Table IX reports quantitative performance. The values are derived from the firmware parameters and from a Monte-Carlo simulation of the firmware decision logic, not from timed bench trials; the last column states the basis of each value. In the simulation each branch carries a demand of 4 L/min, each reading is the pulse count of a one-second window with a random phase, the flow noise is Gaussian with a standard deviation of 0.15 L/min, each sensor has a gain tolerance of ±3 %, and leaks are added downstream of the Road 1 sensor with the onset uniformly distributed within a cycle. The false-alarm figures come from 2000 leak-free runs of 600 s, and the detection figures from 600 trials per leak size.

TABLE IX Performance of the Prototype Logic

Metric	Value	Basis
Detection latency (2 L/min leak)	Mean 2.8 s, maximum 3.6 s; about N Ts for leaks ≥ 1.5 L/min	Simulation
Shut-off time (confirmed flag to relay command)	Less than 1 s; executed in the same cycle as the confirmation	Firmware
False-alarm rate in leak-free runs	0.2 % (4 of 2000 runs of 600 s); 0.75 % with a 1 s demand burst	Simulation
Missed-leak rate	0 % for leaks ≥ 0.8 L/min; 7.5 % at 0.6 L/min; Lmin ≈ 0.65 L/min	Simulation
Counting resolution	0.133 L/min per pulse (±3.3 % at 4 L/min); datasheet accuracy about ±10 % before calibration	Firmware, datasheet
Offline shut-off	Same as online (about 2.8 s); no dependence on Wi-Fi	Firmware
Water saved (2 L/min leak, 10 min manual response)	20 L lost without shut-off against 0.09 L with it (99.5 % less)	Simulation

Fig. 10(a) shows the probability that a leak of a given size is isolated. Leaks of 0.8 L/min or more were always isolated, the probability falls to 92.5 % at 0.6 L/min and to 42 % at 0.4 L/min, and leaks of 0.2 L/min were almost never isolated, in agreement with the $L_{\min}$ of (10). Fig. 10(b) shows the false-isolation probability for a demand burst on Road 1. With N = 3, a one-

second burst is rejected in 98 % of the runs, whereas bursts of 2 s or longer are mostly confirmed; with N = 5 bursts up to 3 s are rejected. This agrees with the (N − 2) Ts rule of Section VI-F and shows the cost of a larger N: the detection delay grows from about 3 s to about 5 s.

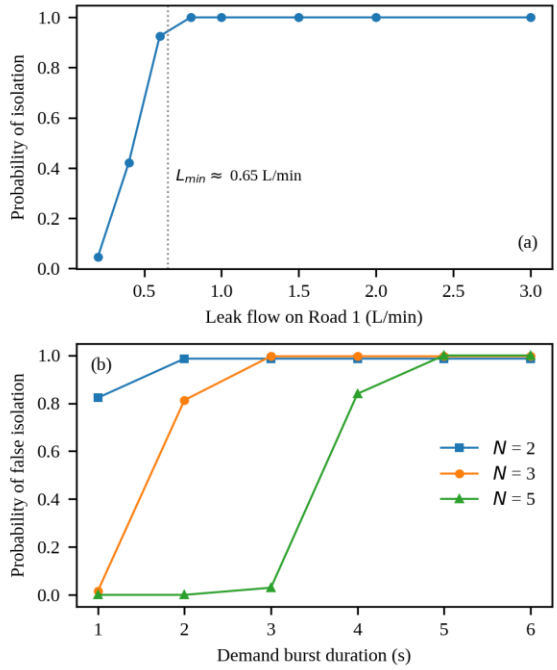


Fig. 10. Sensitivity of the debounce rule: (a) probability of isolating a leak against leak size for N = 3, and (b) probability of false isolation against the duration of a demand burst for N = 2, 3, and 5.

A further leak-free simulation applied a sustained demand difference between the two branches. A difference of 10 % (4.4 against 4.0 L/min) led to isolation of the busier branch in 73 % of the 600 s runs, and a difference of 20 % led to isolation in every run. The branch comparison therefore assumes similar demand on both branches, and τ must be set above the largest imbalance expected in service, as stated in Section IX.

VIII. DISCUSSION

The results show that a simple, interpretable rule set is enough to close the loop from detection to isolation on a branched pipeline. Four observations follow. First, the debounce counter is the element that makes automatic shut-off safe. Without it, a short burst or a sensor glitch would cut off a healthy branch, whereas with it the delay is a fixed, known multiple of the sampling period and bursts shorter than about (N − 2) Ts are ignored. Second, the three indicators cover different cases: the flow balance detects loss anywhere between the main sensor and the branch sensors, the branch comparison tells which branch is at fault, and the closed-valve check catches a valve that does not seal. Third, executing the decision on the microcontroller removes the cloud from the critical path, so a network outage delays the dashboard but not the protection of the pipeline. Fourth, the thresholds trade sensitivity against false shut-offs: a larger τ or N lowers the false-alarm rate but raises the minimum detectable leak or the detection delay, so both are installation parameters rather than constants.

The dashboard adds value beyond the controller. The digital twin gives the operator the location and state of the fault at a glance, and the complaint workflow connects citizen reports to engineer assignment, which shortens the time between a leak and its repair. The remaining threshold tuning is an installation task, described in Section VI-G.

A. Comparison with the Existing Literature

AquaShield differs from the surveyed work in closing the loop (Table I). Acoustic and learned detectors [6]–[10] achieve high accuracy but end with a classification, and threshold prototypes [4], [5] end with an alert. AquaShield uses a simpler, interpretable detector, but it acts on the result by isolating the faulty branch on the microcontroller, and it keeps this ability when the network is unavailable. Its detection accuracy is not comparable with that of the learned detectors on large networks [9], [11], which localize leaks along the pipe; its purpose is fast, low-cost isolation at the branch level. A digital twin view and a citizen complaint workflow complete the operator-facing side, which the surveyed prototypes do not provide.

IX. LIMITATIONS

The detector is rule-based, so τ and N must be tuned for each installation, and a leak beyond the last flow sensor of a branch looks like consumption. Leaks smaller than $L_{\min}$ of (10) are not isolated. Comparing the two branches assumes similar demand: a sustained demand difference close to τ produces false isolation, and a demand burst that lasts N T_s or longer is

indistinguishable from a leak. The performance values of Table IX come from firmware parameters and simulation, so timed bench trials with water are still needed to confirm them, and the validation used a bench-scale, two-branch pipeline. These limits define the scope of the next version rather than the validity of the present results.

X. CONCLUSION

This paper presented AquaShield, an ESP32-based water leakage detection system that isolates a leaking branch automatically. By combining flow-balance, branch-comparison, and closed-valve checks with debounce confirmation, it closes the faulty branch while keeping the rest of the pipeline in service, and it does so on the microcontroller without relying on the network.

A React and Firebase dashboard with a digital twin view and a citizen complaint workflow gives operators and residents a shared picture of network health. Functional testing showed correct detection, shut-off, offline operation, and access control, and an analysis of the firmware logic gave a detection delay of about 3 s, a minimum detectable leak of about 0.65 L/min, and a false-isolation rate of 0.2 % per ten-minute leak-free run.

Future work will extend the framework in six directions: timed field trials on a larger pipeline with per-branch baseline normalization of demand; a median filter and moving average that are tuned against the added delay; sensor-health monitoring and adaptive fusion following [21], [22] to prevent false shut-offs; a self-calibrating digital twin built with EPANET or hydraulic equations that tunes pipe parameters from the error between simulated and measured values; learned detection and localization with LSTM, graph neural network, or Transformer models [7], [9]; and demand forecasting with pump speed control, together with pH and turbidity sensing and water-loss and cost estimation on the dashboard.

ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering, Jyothy Institute of Technology, Bengaluru, for the laboratory facilities and support.

REFERENCES

[1] M. R. Islam, S. Azam, B. Shanmugam, and D. Mathur, “A review on current technologies and future direction of water leakage detection in water distribution network,” *IEEE Access*, vol. 10, pp. 107177–107201, 2022.

[2] T. K. Chan, C. S. Chin, and X. Zhong, “Review of current technologies and proposed intelligent methodologies for water distributed network leakage detection,” IEEE Access, vol. 6, pp. 78846–78867, 2018.

[3] K. B. Adedeji, Y. Hamam, B. T. Abe, and A. M. Abu-Mahfouz, “Towards achieving a reliable leakage detection and localization algorithm for application in water piping networks: An overview,” IEEE Access, vol. 5, pp. 20272–20285, 2017.

[4] S. M. Andini, A. Harjanto, N. R. Alham, and H. Nugroho, “Prototype design and implementation of water distribution pipeline leak detection based on IoT,” in Proc. Int. Conf. Smart Infrastructure Technol., 2021.

[5] G. Sateesh, B. Jaswanth, P. Tharun Kumar, G. Divyamsi, and C. V. Subbaiah, “Smart water leakage detection and prevention system using IoT technology,” Int. J. Adv. Res. Comput. Sci. Eng., vol. 11, no. 5, pp. 45–58, 2022.

[6] B. Bakhtawar, A. Fares, and T. Zayed, “An IoT-driven leak detection in real water networks using hydrophones,” in Proc. IEEE Int. Conf. Smart Water Syst., 2021.

[7] F. Liu, D. Wang, J. Tang, and L. Wang, “Frequency-informed transformer for real-time water pipeline leak detection,” J. Intell. Water Syst., 2022.

[8] A. Jackson, “Smart buildings: Water leakage detection using TinyML,” J. Smart Building Technol., vol. 12, no. 3, pp. 78–92, 2022.

[9] J. Kang, Y.-J. Park, J. Lee, S.-H. Wang, and D.-S. Eom, “Novel leakage detection by ensemble CNN-SVM and graph-based localization in water distribution systems,” IEEE Trans. Ind. Electron., vol. 65, no. 5, pp. 4279–4289, May 2018.

[10] D. U. Leonzio, P. Bestagini, M. Marcon, and S. Tubaro, “Enhanced water leak detection with convolutional neural networks and one-class support vector machine,” IEEE Access, vol. 9, pp. 104856–104868, 2021.

[11] R. Pérez, V. Puig, J. Pascual, J. Quevedo, E. Landeros, and A. Peralta, "Leak localization in water networks: A model-based methodology using pressure sensors applied to a real network in Barcelona," IEEE Control Syst. Mag., vol. 34, no. 4, pp. 24–36, Aug. 2014.

[12] A. B. Primawan, "Water pipe leakage detection system based on fuzzy logic and sensor network," Int. J. Embedded Sensor Syst., 2020.

[13] T. Al-Kadi, Z. Al-Tuwaijri, and A. Al-Omran, "Wireless sensor networks for leakage detection in underground pipelines," IEEE Sensors J., vol. 20, no. 8, pp. 4125–4135, 2020.

[14] R. Vankdothu, H. Bhukya, and R. R. Bhukya, "Hybrid TDR-MI based wireless sensor network for underground water pipeline leakage detection," Int. J. Wireless Commun. Syst., 2022.

[15] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline, "PIPENET: A wireless sensor network for pipeline monitoring," in Proc. 6th Int. Symp. Information Processing in Sensor Networks (IPSN), 2007, pp. 264–273.

[16] M. López, J. Hernández, and A. Rodríguez, "AI-enhanced digital twin platform for smart water distribution," J. Smart Cities Water Manage., vol. 8, no. 4, pp. 115–130, 2023.

[17] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2405–2415, Apr. 2019.

[18] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108952–108971, 2020.