
Proceedings Paper:
Al-rubaye, M., Ahmed, H. and Aral, A. (2026) Demo: Resilient control for water infrastructure. In: MobiSys Companion '26: Proceedings of the 24th Annual International Conference on Mobile Systems, Applications and Services Companion. ACM MobiSys 2026 The 24th ACM International Conference on Mobile Systems, Applications, and Services, 21-25 Jun 2026, Cambridge, United Kingdom. ACM, pp. 173-174. ISBN: 9798400727115.

https://doi.org/10.1145/3812835.3814974

Demo: Resilient Control for Water Infrastructure

Maitham Al-rubaye
University of Vienna, Austria
maitham.al-rubaye@univie.ac.at

Hafiz Ahmed
University of Sheffield, UK
hafiz.ahmed@sheffield.ac.uk

Atakan Aral
University of Vienna, Austria
atakan.aral@univie.ac.at

CCS Concepts

• **Computer systems organization → Embedded and cyber-physical systems**; **Sensors and actuators**; *Dependable and fault-tolerant systems and networks.*

Keywords

Edge computing, Edge AI, cyber-physical systems, instrumentation and control systems, critical infrastructures

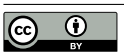
ACM Reference Format:
Maitham Al-rubaye, Hafiz Ahmed, and Atakan Aral. 2026. Demo: Resilient Control for Water Infrastructure. In *The 24th Annual International Conference on Mobile Systems, Applications and Services (MobiSys Companion '26), June 21–25, 2026, Cambridge, United Kingdom.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3812835.3814974

1 Introduction

Reliable operation of critical infrastructures, such as water distribution networks and large-scale industrial facilities, depends increasingly on networked sensor systems [6, 7]. However, this connectivity introduces security risks that can lead to incorrect operational decisions with serious consequences [4]. Protecting these systems requires both secure communication and decision logic that can block suspicious or inconsistent requests. Edge computing is particularly suited to this domain: processing sensor data locally on low-cost devices reduces reliance on cloud connectivity, enables real-time actuation even during network disruptions, and keeps sensitive operational data within the infrastructure boundary.

TROCI (Towards Resilient Operation of Critical Infrastructure) is an EU project that investigates resilient architectures for Instrumentation and Control (I&C) systems that remain correct under sensor compromise and network disruption [5]. Target application domains include **(i)** urban water distribution, in which citywide tanks and elevated reservoirs often require coordinated transfers to balance supply, meet fluctuating demand, or perform maintenance, and **(ii)** industrial cooling systems, such as those in nuclear power plants, which use multiple storage tanks and internal pump houses to circulate coolant.

Previous work within TROCI demonstrated AI-driven security on edge devices and across the cloud–edge continuum [1, 2]. These studies addressed the AI and computational aspects of securing monitoring systems. In this paper, we shift the focus to the physical control layer and present a hardware prototype that demonstrates how a distributed edge-based I&C system can make safe, validated transfer decisions across interconnected tanks using IoT hardware and MQTT-based communication. The contributions are: **(i)** a practical edge I&C architecture for distributed water monitoring; **(ii)** a decision protocol that checks water availability, priority, and message integrity; and **(iii)** a live demonstration of the complete request–decision–actuation cycle.



MobiSys Companion '26, Cambridge, United Kingdom

ACM ISBN 979-8-4007-2711-5/2026/08
https://doi.org/10.1145/3812835.3814974

2 System Architecture

2.1 Hardware Nodes

The system consists of two ESP8266 microcontroller nodes and a Raspberry Pi 5 that acts as an edge server and the central hub (Fig. 1). The prototype models a simplified water infrastructure, including Node 1, which represents a primary reservoir or storage tank that holds the main water supply and controls outgoing distribution, and Node 2, which represents a downstream storage point, such as a neighborhood-level distribution tank that draws water from the primary reservoir on demand.

Node 1 (main tank) is equipped with an analog water level sensor, a digital pressure sensor (GY-68), an MPU6050 inertial measurement unit, a digital temperature sensor, and a digital water flow sensor. It also controls a water pump through a relay module. This node hosts the decision logic and maintains a registry of connected nodes.

Node 2 (secondary tank) is equipped with an analog water level sensor, a digital pressure sensor (GY-68), an MPU6050 inertial measurement unit, and a digital temperature sensor. It publishes its sensor readings periodically and sends a fill request when the water level drops below a threshold.

On both nodes, analog water level and pressure sensors provide two independent readings. This redundancy serves a dual purpose: if either sensor fails or is compromised, the other can still confirm the actual level, and a significant discrepancy between the two readings signals a potential fault or manipulation attempt. The MPU6050 complements this by detecting physical tampering through abnormal acceleration or vibration on the node enclosure. The edge server runs the Mosquitto MQTT broker and hosts the Node-RED dashboard. It acts as the central point for all message routing and provides the operator with a real-time interface.

2.2 Communication and Dashboard

All communication between the two nodes passes through the Mosquitto MQTT broker on the edge server over Wi-Fi. Each node publishes and subscribes under its own topic namespace (`node/{id}/...`), which keeps messages separate and allows the system to scale to additional nodes without changing the protocol. The main topics are: `node/{id}/sensors` for sensor readings, `node/{id}/water/request` for requests, and `node/{id}/water/decision` for responses (GRANT, DENY, or DEFER). The Node-RED dashboard, hosted on the edge server, subscribes to all topics and provides a real-time interface with gauges, charts, pump status indicators, and a decision log.

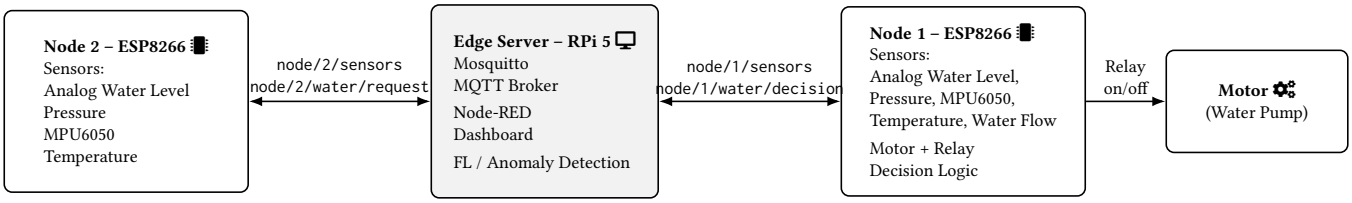


Figure 1: System architecture showing two ESP nodes communicating through an MQTT on a Raspberry Pi 5.

3 Request Handling and Decision Logic

Each node must register before it can request water transfer. On boot, Node 2 publishes a registration message; Node 1 validates it against a whitelist and responds with ACCEPT or REJECT. Every fill request from a registered node is evaluated through a sequence of checks on Node 1. First, the node verifies that the requester is registered. It then checks that the main tank level exceeds a reserve threshold and that the requesting tank actually needs water (i.e. its reported level is below the refill threshold). If the reported level falls outside a physically realistic range, the request is deferred as a suspected sensor fault or manipulation. A rate limiter rejects requests that arrive faster than a configured maximum within a 60 s window, guarding against replay or flooding attacks. The system also verifies that the previous transfer completed successfully by confirming flow sensor readings and that the main tank pressure is adequate for transfer. Additionally, if the controller node reports abnormal vibration or tilt beyond a configured threshold, the request is deferred as a suspected physical tampering attempt. Each failed check returns and logs a typed DENY or DEFER code.

When all checks pass, the main node issues GRANT and activates the pump. Node 2 then switches from 300 s to 1 s reporting for fine-grained fill monitoring. The pump stops when the level reaches 80% or after a 120 s safety timeout.

More advanced anomaly detection is handled by a Federated Learning model. Each edge server trains a local anomaly detection model on the aggregated sensor streams from its connected nodes, detecting patterns such as gradual sensor drift, abnormal pressure drops, or coordinated manipulation attempts. When the model flags a suspicious pattern, the edge server publishes an advisory alert on a dedicated MQTT topic that Node 1 incorporates as an additional check in its decision logic, allowing it to defer or deny requests associated with the flagged anomaly. A central server would periodically aggregate and redistribute model updates, enabling cross-segment learning without centralizing raw data [1, 2].

4 Live Demonstration

The demonstration runs on a physical setup with two water tanks, two ESP8266 nodes, and a Raspberry Pi displaying the Node-RED dashboard. The audience can observe four phases and access the dashboard on their mobile devices. A preview video is available at https://www.youtube.com/watch?v=CtezynZY35Q.

(1) Startup and Registration. Both nodes connect to the MQTT broker. Node 2 registers, and Node 1 validates it against the whitelist. The dashboard shows both nodes online and logs the ACCEPT response.

(2) Steady-State Monitoring. The dashboard displays live readings of water level, pressure, and temperature from each tank, updated every 300 s. No new entries are recorded in the decision log.

(3) Fill Request Cycle. Water is physically removed from the secondary tank until the level drops below 25%. Node 2 detects the low level and publishes a fill request. Node 1 evaluates the request, responds with GRANT, and activates the pump. Node 2 switches to 1 s reporting, and its level gauge rises in real time on the dashboard. The pump stops when the level reaches 80% or after a 120 s safety timeout.

(4) Fault and Attack Scenarios. The following are triggered on demand: unregistered-node rejection (UNREGISTER-ED), rate limiting (RATE_EXCEEDED), low denial (INSUFFICIENT _RESERVE), and tamper detection (TAMPER_DETECTED). Each response appears in the log.

While the prototype models a simplified water-transfer scenario, the underlying design generalizes to a broader class of distributed water-control systems in safety-critical environments. The key novelty lies in combining lightweight edge-based decision logic with multi-sensor validation and secure message handling to ensure that decisions remain safe even under faulty or adversarial conditions. Unlike cloud-only approaches, the system enforces local consistency checks and continues to operate under network disruptions. Future versions would reduce communication and energy overhead through event-driven sensing mechanisms [3].

Acknowledgments

This research was funded in part by CHIST-ERA-22-SPiDDS-07, the Austrian Science Fund (FWF): 10.55776/I6647, and the Engineering and Physical Sciences Research Council (EPSRC): EP/Y036344/1.

References

[1] M. Al-rubaye and A. Aral. 2024. Towards Enhanced AI-Driven Security in Monitoring Systems with Low-Cost IoT Devices. In *Proc. IoT 2024.* ACM, New York, NY, USA, 255–260. doi:10.1145/3703790.3703819

[2] M. Al-rubaye and A. Aral. 2025. Performance Analysis of AI-Driven Security Models in the Cloud-Edge Continuum for Monitoring Critical Infrastructures. In *Proc. AINA 2025.* Springer, Cham, 273–283. doi:10.1007/978-3-031-87778-0_27

[3] A. Aral. 2025. EdgeSynapse: Towards Leaky-Spike Transmission for Sustainable Edge Sensing. In *Proceedings of the Tenth ACM/IEEE Symposium on Edge Computing (SEC '25).* ACM, New York, NY, USA, 5. doi:10.1145/3769102.3774704

[4] A. A. Cárdenas, S. Amin, and S. Sastry. 2008. Research Challenges for the Security of Control Systems. In *Proc. HotSec 2008.* USENIX, Berkeley, CA, USA, 6:1–6:6.

[5] TROCI Consortium. 2024. Towards Resilient Operation of Critical Infrastructures. https://troci.holisun.com/. CHIST-ERA-22-SPiDDS-07.

[6] M. Mendoza and P. V. Tsvetkov. 2023. An intelligent fault detection and diagnosis monitoring system for reactor operational resilience. *Progress in Nuclear Energy* 156 (2023), 104529. doi:10.1016/j.pnucene.2022.104529

[7] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline. 2007. PIPENET: A wireless sensor network for pipeline monitoring. In *Proc. IPSN 2007.* ACM, New York, NY, USA, 264–273. doi:10.1145/1236360.1236396