

FloodGuard AI: An AIoT Platform for Urban Flood-Risk Sensing, Edge-Assisted Telemetry, Predictive Risk Scoring and Municipal Response Support

1) Uppu Chandrasekhar
Founder and CEO
Sujju Software Solutions Pvt Ltd

2) Dr. K. Siva Rama Krishna
Professor, Department of AI&DS
Vasireddy Venkatadri International
Technological University, Namburu,
Guntur District, Andhra Pradesh,
India

3) Dr. M. Chinarao
Head of Department, Information
Technology
Andhra Loyola Institute of Engineering
and Technology, Vijayawada, Andhra
Pradesh, India

4) G. Jyothi
Department of Information
Technology Andhra Loyola Institute
of Engineering and Technology,
Vijayawada, Andhra Pradesh, India

5) M. Vishnu Vardhan
Department of CSE (AIML)
Andhra Loyola Institute of Engineering
and Technology, Vijayawada, Andhra
Pradesh, India

6) A. Tejaswini
Department of Information Technology
Andhra Loyola Institute of Engineering
and Technology, Vijayawada, Andhra
Pradesh, India

7) G. Nageswari
Department of CSE (DS)
Andhra Loyola Institute of Engineering and Technology, Vijayawada, Andhra Pradesh, India

Abstract— Flash floods in towns and river basins build up when heavy rain, rising channel levels and choked drainage act together, yet many monitoring set-ups still present these signals separately. FloodGuard AI is an Artificial-Intelligence-of-Things (AIoT) platform that joins low-cost ESP32 sensing nodes, a Raspberry Pi edge gateway, a Node.js/MongoDB cloud service and a GIS command centre in one closed loop from sensing to logged response. A transparent risk engine converts level, rainfall intensity, flow velocity and smoothed rate of rise into a bounded 0–100 score and a four-state label (SAFE, WATCH, HIGH RISK, CRITICAL), with a crest estimate, anomaly flags and an advisory directive; every alert action is written to an append-only log. Beyond the architecture, the paper contributes (i) a provenance-class discipline that separates bench-measured, scenario-injected, illustrative and planned-field data; (ii) a fault-versus-flood routing rule that sends technicians, not response teams, when redundant sensors disagree; and (iii) a pre-specified evaluation protocol with acceptance criteria. To give the design a first quantitative test before field trials, we report a synthetic Monte-Carlo study (300 runs per scenario family). Adding rate-of-rise escalation to a fixed 3.80 m alarm raised the median warning lead time before the 4.60 m critical crest from 60 to 130 min, and the weighted score raised it to 195 min, with no missed critical events; the price was more alerts on events that peaked below the high-risk stage. These figures describe the logic on simulated hydrographs, not the Krishna basin. No field accuracy figure is claimed.

Keywords— flood early warning, AIoT, ESP32, Raspberry Pi gateway, LoRa, rate of rise, risk scoring, GIS decision support, Krishna basin.

I. Introduction

Urban and peri-urban flooding is rarely caused by one factor. A storm cell rains on a catchment, channels fill, culverts and underpasses choke on debris, a barrage or regulator is operated, and within hours a road or embankment that looked fine is under water. The Krishna basin around Vijayawada combines a large upstream

catchment, a barrage that passes very high discharges and a dense network of urban canals and drains that respond within hours to local rain.

In practice the data needed to see such a situation early are scattered: gauge readings in one register, rainfall in another, drain conditions learnt by telephone. Alerts are usually raised when one reading crosses a fixed mark, which is blind to how fast the reading moves. A level of 3 m that has been steady for a day is a very different situation from a level of 3 m that gained half a metre since morning. Manual channels also leave little trace of who knew what, when, and what was done.

FloodGuard AI addresses this at prototype scale. Water level, flow, rainfall and device health from a handful of stations are brought into one stream; a few hydrologically meaningful indicators are computed on it; and the result is presented so an authorised operator can decide where to look and what to dispatch. Emergency decisions stay with people; the software ranks, explains and records. The loop is Sense → Stream → Detect → Forecast → Explain → Alert → Respond → Verify → Learn.

Research questions. **RQ1:** does adding rate of rise and rainfall to a level threshold lengthen warning lead time, and at what cost in extra alerts? **RQ2:** can a station-specific, traceable linear score be audited by operators better than an opaque model at the cold-start stage? **RQ3:** can one ingestion path serve field nodes, test tools and scenario injection, so laboratory behaviour is what the field will see?

Contributions.

1. An end-to-end AIoT architecture (ESP32 nodes, Raspberry Pi gateway with in-order buffering, cloud ingestion, React GIS console).
2. A transparent risk formulation with two explicit safeguards (float-switch veto of SAFE; rate-of-rise escalation by one state) and an anomaly layer that distinguishes *sensor fault* from *flood signal* and routes each to the right responder.
3. An alert-to-response workflow with named attribution and an append-only log, whose records are designed to become labelled history for the Verify and Learn stages.
4. A provenance-class framework (Table III) applied to every value in the paper, with a same-path validation toolkit (hardware diagnostic page, scenario suite, packet terminal).
5. A reproducible synthetic study (Section VII-B) and a pre-specified field protocol with acceptance criteria (Section VIII).

II. Related Work

A. Sensor-based flood monitoring. The IoT made it economical to deploy many small measuring points where only a few expensive gauges were affordable [1]. Basha et al. combined in-situ measurements with a physical channel model to detect river rise early [2]; Mitra et al. demonstrated neural-network forecasts fed by low-cost IoT readings [3]; Mousa et al. studied ultrasonic and infrared sensing for urban flash-flood detection [18]. The UrbanFlood project described a full early-warning architecture with sensor networks and computational modules [17]. These works establish sensing and prediction, but few describe the operator's next step: how an alert is acknowledged, who is dispatched, and how the response is recorded for review.

B. Machine learning for flood prediction. Surveys report a steady move from physically based models to data-driven learners [4]. Random forests [5] and gradient-boosted trees [6] handle mixed tabular features with modest data; LSTM [7] and GRU [8] networks capture temporal dependence in rainfall–runoff modelling [9]. Kratzert et al. showed that LSTMs trained across many catchments can generalise better than models fitted to one [20], and Nevo et al. described an operational large-scale ML forecasting framework [19]. Such results depend on long labelled records or regional training sets, which a new municipal deployment does not have.

FloodGuard AI therefore begins with an interpretable score and a physical rate-of-rise baseline, and treats learned models as an upgrade that must beat the baseline on data the platform itself collects.

C. Edge processing and communication. Edge computation reduces reaction time and dependence on the uplink [10]. MQTT [11] suits constrained devices, and LoRa reaches gateways over kilometres in open terrain [12], though duty-cycle and capacity limits matter at scale [22]. Flood sites often have patchy cellular coverage and limited power, which motivates the gateway-with-buffer pattern adopted here.

D. Anomaly detection, decision support and warning practice. Drift, floating-debris spikes and packet corruption are routine in outdoor sensing; statistical outlier tests and temporal filters are standard remedies [13]. Operational warning guidance stresses that a warning is useful only if timely, understood and acted upon [21], and forecast-verification literature separates consistency, quality and value [23]. Few IoT flood papers address that chain end to end.

E. Positioning and research gap. Table I positions this work on the dimensions that matter for operations rather than sensing alone. The comparison is qualitative, reflects published design descriptions and the present prototype, and is not a measured ranking.

TABLE I. Positioning of the proposed platform. [VERIFY each cell against the cited paper before submission.]

Dimension	Sensor-only IoT [1],[3]	Model-based sensing [2]	Early-warning architecture [17]	Large-scale ML [19]	This work
Low-cost node with edge buffering	Partly	Partly	Not the focus	No	Yes
Rate of rise and rainfall in one score	Rare	Via model	Via models	Learned	Yes, explicit
Station-specific thresholds	Rare	Via model	Yes	Learned	Yes
Sensor-fault vs flood routing	Rare	Not the focus	Not the focus	Not the focus	Yes
Logged acknowledge / dispatch	Rare	No	Partly	Alert delivery	Yes, append-only
Works without site history	Yes	Needs model	Needs models	No	Yes
Field-validated performance	Varies	Yes	Varies	Yes	Not yet

Gap. Existing IoT, ML and GIS work is often built as separate components; many rely on one fixed threshold; and few combine level, rate of rise, rainfall, flow, location-specific thresholds, fault routing and a response log in one inspectable loop that can be tested before a storm arrives. The novelty claimed here is this integration and its evaluation discipline, not a new forecasting algorithm.

III. Problem Statement

Urban areas face water on roads and underpasses, culverts blocked by debris, drains overloaded by intense rain, and river reaches rising faster than operators can track. The principal challenges are: (1) manual, periodic monitoring that cannot follow changes within minutes; (2) fragmented level, rainfall and flow data; (3) fixed thresholds that ignore location and rainfall; (4) limited use of rate of rise and trend; (5) location-specific risk that needs per-station profiles; and (6) delayed and untraceable response. The need is for a system that collects field data continuously, detects abnormal patterns, shows location-specific risk understandably, and records the response while leaving emergency decisions to authorised humans.

IV. Proposed System

A. Architecture. Sensor nodes are placed at flood-prone roads, drains, culverts, underpasses and river reaches. An ESP32 on each node samples the sensors and sends compact packets to a Raspberry Pi gateway; the

gateway forwards validated records over MQTT or HTTPS to a cloud Express service; the service stores them in MongoDB Atlas, runs the risk engine and broadcasts updates to browsers; operator actions flow back to the database, closing the loop (Figs. 1 and 2).

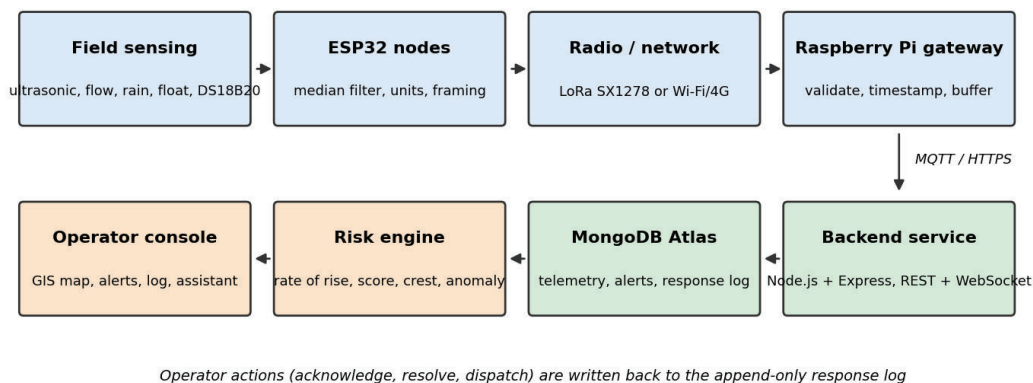


Fig. 1. FloodGuard AI data path from field sensing to operator response and back to the audit log.

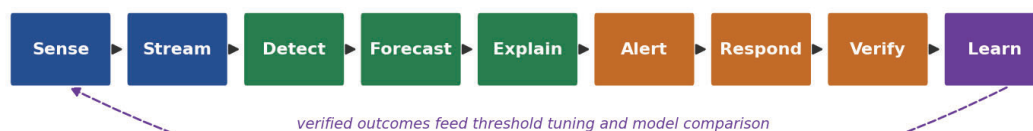


Fig. 2. The nine-stage closed loop.

B. Requirements and design goals. Requirements were drawn from how a municipal control room works during a rain event (Table II). Two design choices follow: every derived number is traceable to raw inputs and a stated formula, so a weighted score was preferred over an opaque model; and one ingestion interface serves real nodes, the test terminal and the scenario suite, so laboratory behaviour passes through the path field data will take.

TABLE II. Principal requirements.

Class	Requirement
Sensing	Measure level, flow, rainfall and float-switch state at configurable intervals
Edge	Validate ranges, attach a trusted timestamp, buffer readings while the uplink is down
Transport	Deliver over MQTT or HTTPS and tolerate duplicates after reconnect
Analytics	Rate of rise, bounded score, four-state label and crest estimate per station
Anomaly	Flag implausible jumps and sensor faults instead of silently scoring them
Alerting / Traceability	Raise, acknowledge, resolve, dispatch with named attribution; append-only record
Usability / Safety	Map and cards with plain-language explanation; human in the loop, no automatic public warning
Scalability / Reliability	Add stations by configuration; degrade gracefully; replaceable components

C. Technology stack. Sensing: JSN-SR04T / AJ-SR04M ultrasonic level, YF-S201 flow, rain module (tipping bucket in field), float switch, DS18B20, BME280, NEO-6M GPS. Node: ESP32 DevKit with median filter, unit conversion and packet framing. Gateway: Raspberry Pi 4/5 with schema and range validation, timestamping, local queue and MQTT/HTTPS forwarder. Link: LoRa SX1278 (RA-02) or Wi-Fi/4G. Backend: Node.js, Express, REST and WebSocket. Database: MongoDB Atlas. Frontend: React, Vite, Tailwind CSS, Leaflet/OpenStreetMap. Intelligence: rule-and-statistics engine, with Python, Pandas, NumPy and scikit-learn [14] reserved for the learned-model stage.

V. Methodology

A. Development approach and provenance. The prototype was built in exercisable stages: bench integration with a transparent tank rig; a validating gateway; a cloud ingestion service; the risk engine and scenario tests; the GIS console and alert workflow; and finally diagnostic pages exposing the health of the whole chain. Every value in this paper belongs to exactly one provenance class (Table III), and classes are never mixed when making claims. Hardware nodes are connected only at ST-001 on the bench; ST-002 to ST-006 receive readings through the same interface in scenario mode.

TABLE III. Data provenance classes.

Class	Description	Used for
Bench-measured	ESP32 readings on the bench tank rig (ST-001 hardware path); serial and live diagnostics	Commissioning, fault-state demonstration
Scenario / injected	Packets from the terminal, test-ingest button or scenario suite; stations ST-002 to ST-006	Logic and interface demonstration
Synthetic simulation	Monte-Carlo hydrographs generated by code (Section VII-B)	Comparing alert rules under stated assumptions
Illustrative	Archive cards, sample directives, example alert narratives	Interface layout only
Planned field	Monsoon-season trials with controlled reference measurements	Future accuracy claims

B. Pre-processing. Each record is range-checked against physical limits for its station. Missing values are not imputed with zeros; the engine carries forward the last valid value for a short horizon and flags the station degraded if the gap persists. Gateway timestamps are aligned to a regular grid before differences are taken. At the node, a median filter over a burst of ultrasonic echoes suppresses single-echo spikes before level is computed as mounting height minus measured distance.

C. Rate of rise. With water level $h(t)$ in metres, the instantaneous rate of rise over interval Δt is

$$r(t) = [h(t) - h(t - \Delta t)] / \Delta t \quad (1)$$

reported in cm/h and smoothed by an exponential moving average

$$\hat{r}(t) = \alpha \cdot r(t) + (1 - \alpha) \cdot \hat{r}(t - 1), \quad 0 < \alpha \leq 1 \quad (2)$$

The dashboard shows the instantaneous value, a short moving window and a longer slope so that an operator can tell a spike from a sustained climb. A negative value is labelled a falling trend.

D. Station threshold profiles and channel utilisation. Each station has four stages: safe freeboard, watch advisory, high-risk stage and critical crest. For ferry-ghat station ST-001 the profile is 1.80, 2.80, 3.80 and 4.60 m (Fig. 3). Channel utilisation is

$$U = h / H_c \times 100 \% \quad (3)$$

where H_c is the critical crest. At the ST-001 snapshot level of 2.65 m, $U \approx 58 \%$. The dashboard's capacity percentage (51 % for the same reading) is computed against configured channel capacity, a separate quantity from the critical crest; the two must not be quoted interchangeably.

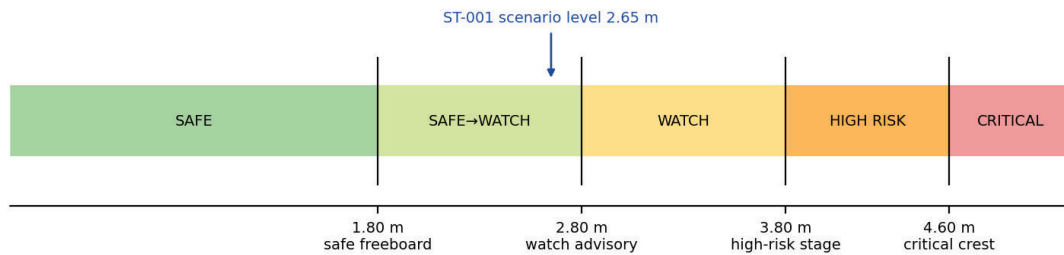


Fig. 3. Four-stage threshold profile of station ST-001 with the scenario-mode level of 2.65 m.

E. Weighted risk score. Four inputs are scaled to the unit interval,

$$x_i = \min\{1, \max\{0, (v_i - v_{i,\min}) / (v_{i,\max} - v_{i,\min})\}\} \quad (4)$$

and combined as a convex sum scaled to 0–100,

$$R = 100 \times \sum w_i x_i, \sum w_i = 1, w_i \geq 0 \quad (5)$$

with level and rate of rise weighted most and rainfall and flow acting as corroborating evidence. A bounded linear score was preferred for the first release because an operator can reason about it, any input moves the output predictably, and it is a transparent baseline for later learned models. The radar view (Fig. 19b) shows six vectors, adding basin saturation and a debris anomaly; the published score uses the four above, and the role of the two extra vectors should be stated [VERIFY].

Worked example. Take $H_c = 4.60$ m, level 3.45 m, rain 35 mm/h, flow 2.4 m/s, rise +6.5 cm/h, and normalisation ranges level 0– H_c , rain 0–80 mm/h, flow 0–6 m/s, rise 0–20 cm/h. Then $x = (0.75, 0.44, 0.40, 0.33)$. With illustrative weights $w = (0.35, 0.20, 0.15, 0.30)$, $R = 100 \times (0.2625 + 0.088 + 0.060 + 0.099) \approx 51$, in the WATCH band (Fig. 4); the production sandbox returned 50 for the same inputs (Fig. 16) [VERIFY: state the production weights and ranges so the one-point difference is explained].

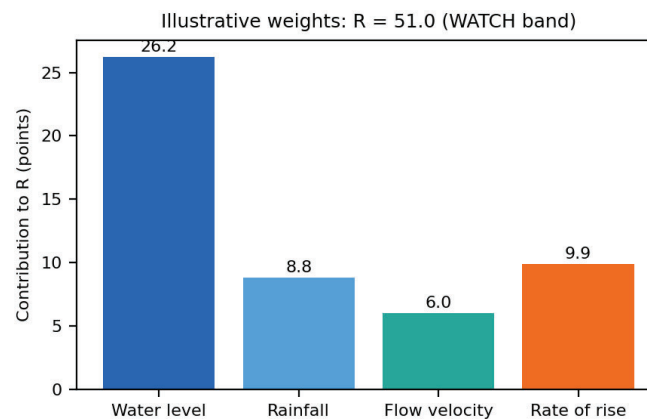


Fig. 4. Contribution of each normalised input to the illustrative score of the worked example.

Sensitivity. Because R is linear, the marginal effect of each input is constant and can be stated in advance (Table IV). With the illustrative weights and ranges, one centimetre of level moves R by 0.076 points, whereas 1 cm/h of rise moves it by 1.5 points.

TABLE IV. Analytic sensitivity of the score (illustrative weights).

Input	Range used	Weight	ΔR per unit
Level	0–4.60 m	0.35	7.6 points per m
Rainfall	0–80 mm/h	0.20	0.25 points per mm/h

Input	Range used	Weight	ΔR per unit
Flow velocity	0–6 m/s	0.15	2.5 points per m/s
Rate of rise	0–20 cm/h	0.30	1.5 points per cm/h

F. State classification. The score and station profile jointly decide the state (Table V). Two safeguards prevent the label being driven by the score alone: a station is never reported SAFE while its float switch is triggered, and a rate of rise above the station alarm value raises the state by one level even if the absolute level is below the next stage. In both cases the explanation names the reason.

TABLE V. Operational meaning of the four states.

State	Meaning	Expected operator action
SAFE	Normal monitored conditions	Routine monitoring
WATCH	Conditions are changing	Watch trend; check upstream rain and drain screens
HIGH RISK	Significant abnormal conditions	Acknowledge; prepare teams; consider dispatch
CRITICAL	Severe conditions	Immediate review; dispatch and public-advisory decision

G. Crest estimate and hydrograph projection. The baseline extrapolates the smoothed rate,

$$\hat{h}(t + \tau) = h(t) + \hat{r}(t) \cdot \tau, \quad t_c = (H_c - h(t)) / \hat{r}(t) \quad (6)$$

where τ is the look-ahead horizon and t_c the time to reach the crest if the present rate persisted. For the worked example, $t_c = (4.60 - 3.45) / 0.065 \approx 17.7$ h. This is a floor on sophistication; the dashboard also projects a six-hour hydrograph with a mass-balance style update (Fig. 11). It is a projection under current inflow, not a prediction of rain that has not yet fallen.

H. Anomaly and fault detection. A Z-score filter on the rate of rise,

$$z = (r - \mu_r) / \sigma_r, \text{ flagged when } |z| > \tau_z \quad (7)$$

uses a trailing window for the same station. A jump that is statistically implausible and unsupported by rainfall or neighbouring stations is labelled a sonar-noise or drift suspect and held out of the score until corroborated. A second check compares redundant measurements (two ultrasonic probes, or ultrasonic level against the float switch); disagreement raises a *sensor-fault flag* so that a technician is sent to the node, not a response team to the river. A debris-restriction signature (level rising while flow velocity falls, the pattern of a blocked culvert) is surfaced as a high-risk anomaly.

I. Confidence and path to learned models. The displayed model-confidence percentage describes internal consistency of the inputs and is **not** a calibrated probability. Calibration against observed outcomes (reliability diagrams, Brier score [24]) is future work. As labelled history accumulates, the score is to be compared with statistical anomaly detection, then random forest [5], gradient-boosted trees [6] and LSTM/GRU [7],[8]; any learned model must beat the baseline of Eq. (6) and the score on the same event-wise split before replacing it. Directives are advisory.

VI. Implementation

A. Sensing layer. Level is measured by downward-facing ultrasonic ranging (JSN-SR04T and AJ-SR04M probes; HC-SR04 on the bench). Flow is estimated from a Hall-effect turbine meter (YF-S201 class); rainfall from a rain module on the bench, to be replaced by a tipping-bucket gauge in the field. A mechanical float switch gives an independent high-water indication, a DS18B20 records temperature, and a BME280 and NEO-6M GPS provide environment and time reference. Two ultrasonic probes are kept deliberately so that disagreement is raised as a data-quality flag instead of being averaged away.

B. ESP32 node. Firmware samples all sensors on a fixed schedule, applies the median filter, converts to engineering units and sends a compact packet (station ID, level, rain intensity, flow velocity, temperature, float

and relay state). OLED/LCD, LEDs and a buzzer assist technicians; a relay output for a gate or pump is operated only from the operator console in this prototype. Principal hardware (Table VI): 3 ESP32 nodes, 1 Raspberry Pi gateway, 2 waterproof and 1 bench ultrasonic probe, 2 flow meters, 1 rain module, 1 GPS, 2 LoRa modules, 2 float switches, INA219 monitors and batteries, and a bench hydraulic rig.

TABLE VI. Principal prototype hardware.

Function	Component	Qty
Node controller	ESP32 DevKit	3
Edge gateway	Raspberry Pi 4/5	1
Water level	JSN-SR04T (waterproof) / HC-SR04 (bench)	2 / 1
Flow	YF-S201 / YF-B5 turbine meter	2
Rain; environment	Rain module; BME280, DHT-series, DS18B20	1; 1 each
Position and time	NEO-6M GPS	1
Radio and fieldbus	LoRa SX1278 / RA-02 with antenna; MAX485 RS-485	2; 2
Local display and alert	0.96-in OLED, 16×2 I ² C LCD; buzzers; R/Y/G and RGB LEDs	2 / 1; 2; 17
Safety inputs and actuation	Float and tamper switches; 5 V relay modules	2 / 2; 2
Power	LM2596 buck converter, INA219 current monitor, batteries	2 each
Test rig	Pump, transparent tank, PVC pipe, elbows and valves	1 set
Enclosure	Waterproof boxes, cable glands, connectors, heat-shrink	2 / 10

C. Edge gateway. The gateway validates each packet against range limits and schema, attaches a gateway timestamp so that rate of rise does not depend on a drifting node clock, and stores accepted records in a local queue forwarded in order when the uplink returns. The dashboard exposes heartbeat age, link protocol, signal strength and packet loss per station, plus a diagnostic echo with round-trip time (Figs. 14, 15).

D. Communication profiles. Stations with line of sight to a gateway use LoRa from node to gateway and MQTT beyond; stations near cellular coverage use HTTPS over 4G or Wi-Fi. The first costs less power and tolerates poor coverage; the second is simpler to deploy. Each station is labelled with its profile so loss statistics can be read in context.

E. Power and protection. Outdoor nodes use a battery and small solar panel with a buck converter and INA219 current monitor, waterproof enclosures, surge protection and tamper switches. The bench tank is the intended arrangement for the calibration of Section VIII.

F. Backend and data model. A single POST endpoint under the telemetry path accepts station ID, level, rainfall (mm/h), flow (m/s), temperature, float and relay state. On receipt the server validates, stores, recomputes rate of rise, runs the anomaly check, updates the score and, if a state boundary is crossed, creates or updates an alert and emits a WebSocket event. Four collections hold state: stations (type, coordinates, elevation, threshold profile, gateway), telemetry, alerts (severity, station, trigger time, acknowledgement and resolution attribution) and a response log (category, summary, instructions, assigned unit, dispatching officer). Keeping the log separate from alerts lets one action address several alerts and one alert receive several actions.

G. Operator console. A single-page application with live badges offers the modules in Table VII.

TABLE VII. Dashboard modules.

Module	Purpose
Command Center	Basin risk index, flashpoints, peak rate of rise and rain, map and alert queue
Live Hardware Test	End-to-end check of backend, database, gateway and node values
ML Test Suite	Scenario cases and interactive input sandbox for the risk model
Advanced Visualizations	Multi-station hydrograph, six-axis risk radar, rain–flow scatter
Alarm Notifications	Browser push, acoustic siren engine, broadcast simulator

Module	Purpose
GIS Flood Map	Gauge markers, flood plains, shelters, barrage status panel
Sensor Telemetry	Per-station level, rate of rise, rain, flow and link state
IoT Packet Terminal	Live packet feed and field-node telemetry injector
AI Risk Engine	Risk index, crest forecast, anomalies, directive
Alerts and Incidents	Acknowledge, resolve and dispatch workflow
Response Action Log; Historical Trends; Flood Archive	Audit entries, past-event cards, hydrograph trend page
IoT Gateway Health	Battery, solar, signal, loss and heartbeat per node

VII. Results and Discussion

Part A reports what the prototype does when exercised (scenario and bench provenance; these show behaviour, not accuracy). Part B reports a synthetic simulation study that tests the alert logic quantitatively under stated assumptions. All screenshots were captured on 17 October 2026 from the local development build.

A. Prototype observations

1) Configuration and telemetry. Six stations are configured: ST-001 ferry-ghat upstream basin, ST-002 Prakasam Barrage and Krishnalanka, ST-003 Budameru diversion, ST-004 Eluru canal corridor, ST-005 Benz Circle underpass and sump, ST-006 Ryves and Bandar delta canal sluices, served by three gateways. Table VIII and Fig. 8 show the telemetry page, a scenario-mode snapshot and not a measured dataset. Station cards show level against critical stage, capacity bar, rate of rise, precipitation, flow discharge and node and link state.

TABLE VIII. Telemetry snapshot (scenario mode). States depend on each station's own profile, so equal levels need not give equal states.

Station	Location	Level (m)	State
ST-001	Ferry Ghat	2.65	SAFE
ST-002	Prakasam Barrage	4.67	HIGH RISK
ST-003	Budameru Diversion	2.98	CRITICAL
ST-004	Eluru Canal Corridor	1.79	SAFE
ST-005	Benz Circle Underpass	0.95	WATCH
ST-006	Ryves & Bandar Canal	3.18	SAFE

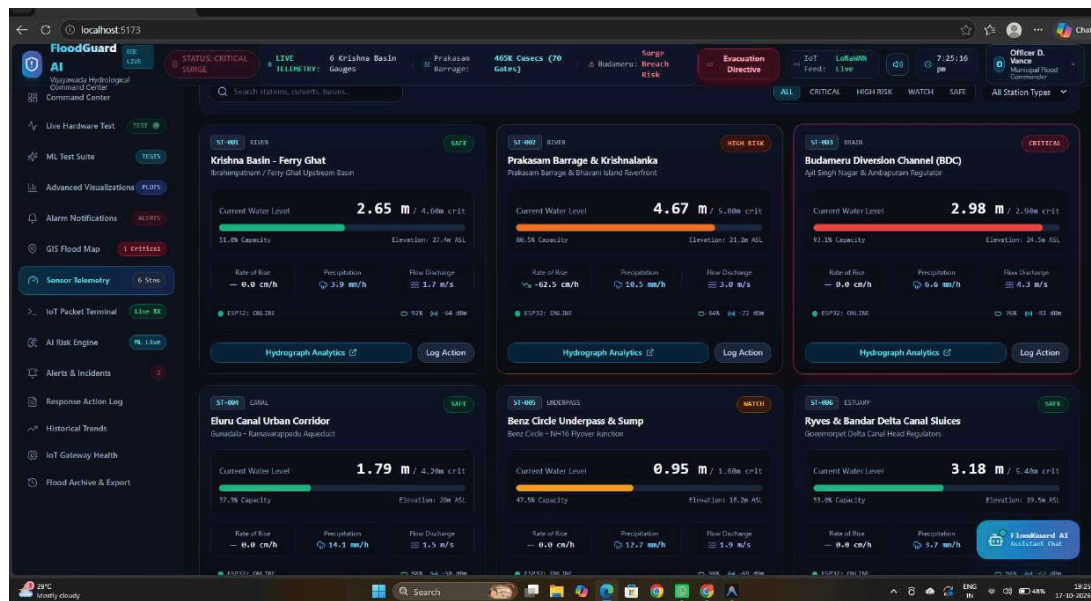


Fig. 8. Sensor-telemetry page: one card per station with level, capacity, rate of rise, precipitation, flow and link state (scenario-mode data).

2) Command centre and GIS map. The command centre (Fig. 5) opens on four tiles: basin risk index (41/100 in the snapshot), critical flashpoints (two of six: one critical, one high-risk), peak rate of rise with its location, and peak precipitation with its location. The map and alert queue share the screen so that selecting an alert centres the map on the station. The GIS page (Fig. 6) adds a barrage status panel (discharge, gates open, reservoir level against full level, tailwater velocity), flood-plain and shelter layers, and a browser notification when an alert is dispatched.

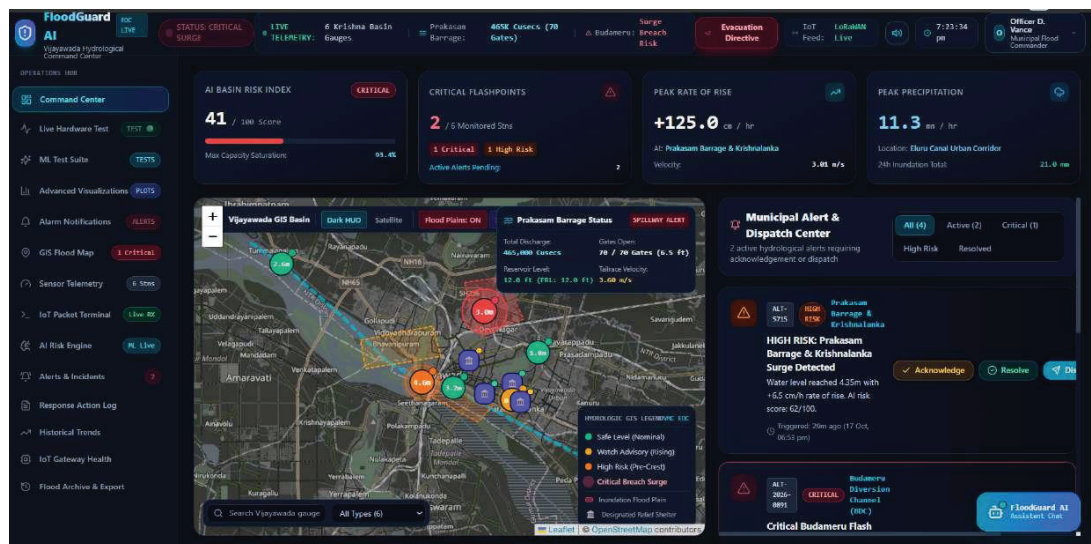


Fig. 5. Command centre: basin risk index, flashpoints, peak rate of rise and rainfall, GIS map with barrage panel and alert queue (scenario-mode data).

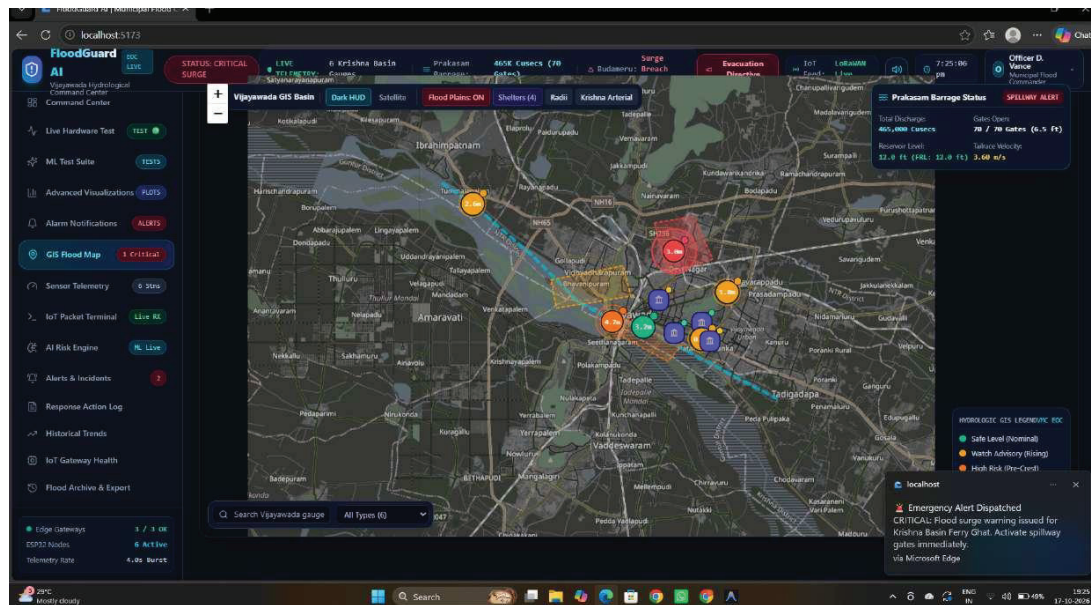


Fig. 6. GIS flood map with flood-plain and shelter layers, barrage status panel and a browser alert notification (scenario-mode data).

3) Station inspector, risk-engine view and trends. Opening a station shows the inspector (Fig. 9): four-stage profile, utilisation bar, rate of rise and trend, 24-hour rainfall, predicted crest window, mitigation directive and a hardware telemetry strip, with a direct button into the dispatch form (Fig. 10). The risk-engine page (Fig. 11) shows the risk gauge, crest time and height, drivers, anomaly, rate-of-rise derivatives (instantaneous, short moving average, slope), a six-hour hydrograph against the embankment-critical line and the directive. In the captured injected scenario ST-001 reads 47/100 (WATCH) with a surge anomaly at +62.5 cm/h; the displayed confidence is not a calibrated probability (Section V-I). The advanced-visualisation page (Fig. 19) offers a 24-hour multi-station hydrograph, a six-vector radar and a rainfall-versus-flow scatter plot.

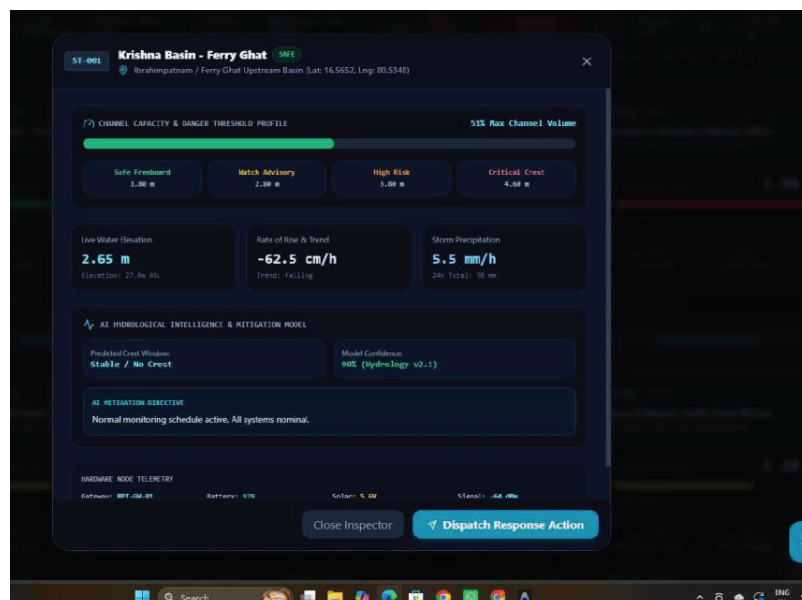


Fig. 9. Station inspector for ST-001 (scenario-mode data).

Fig. 10. Response-action form: station, action-category presets, summary, instructions, assigned unit and dispatching officer.



Fig. 11. AI risk-engine view for ST-001: gauge, six-hour hydrograph projection, drivers, surge anomaly and directive (injected scenario).



Fig. 19a. Advanced visualisations: 24-hour multi-station hydrograph comparison (scenario-mode data).

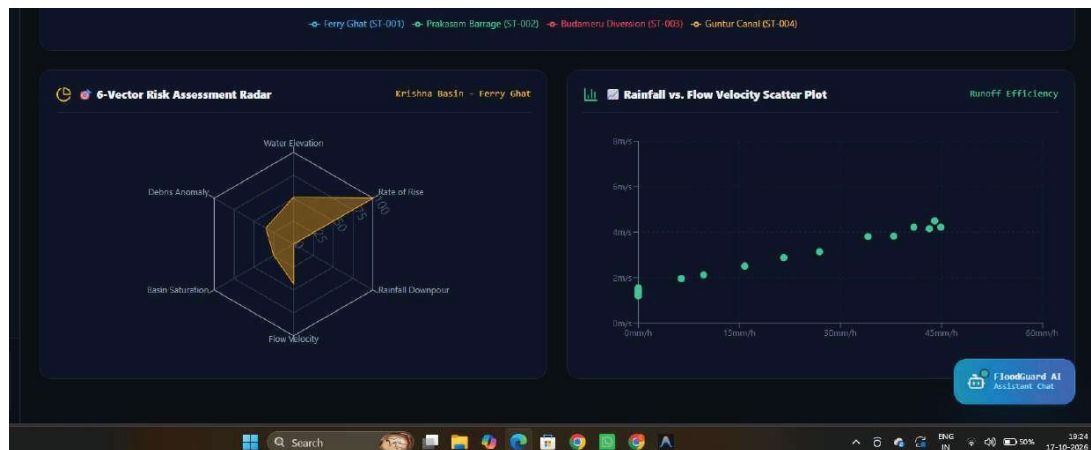


Fig. 19b. Six-vector risk radar and rainfall-versus-flow scatter plot for ST-001.

4) Alerts, dispatch and notification. The alert queue (Fig. 7) shows severity, station, one-line justification, time since trigger and attribution; its filters (all, active, critical, high risk, resolved) follow the lifecycle. Acknowledgement stamps the officer name; resolution closes the alert once conditions have recovered; dispatch opens the form of Fig. 10, whose presets cover hydraulic barrier deployment, auxiliary pump start, mobile sandbag reinforcement, traffic and underpass detour signage, public siren and cell-broadcast advisory, trash-grate and culvert clearing, and sensor calibration and hardware inspection. Each saved action enters the response log with its time. Three channels reach operators (Fig. 18): browser push, a Web Audio dual-tone siren (880/440 Hz) with a mute control, and a broadcast simulator with audience, channel and priority-override selection. The broadcast is a simulation and is deliberately not connected to an SMS or cell-broadcast gateway.

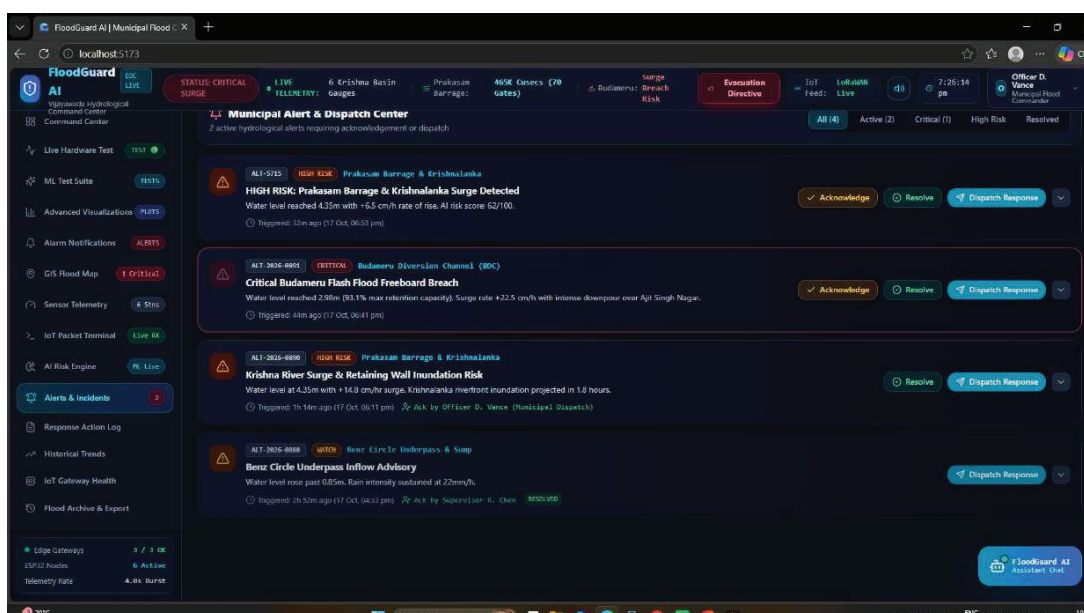


Fig. 7. Municipal alert and dispatch centre with acknowledge, resolve and dispatch actions and attribution (scenario-mode data).

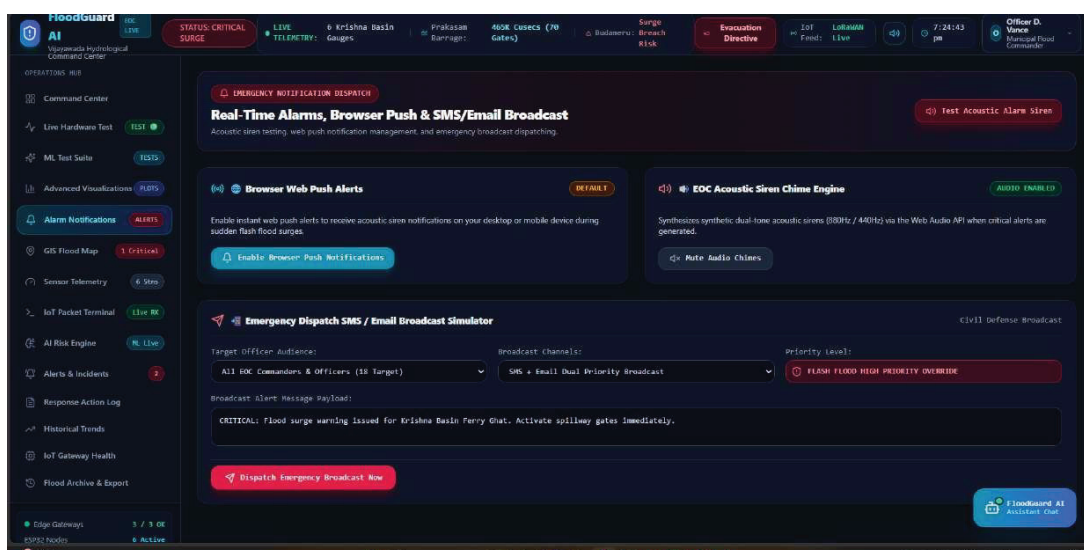


Fig. 18. Alarm-notification page: browser push, acoustic siren engine and broadcast simulator.

5) Hardware diagnostics. The live-hardware page (Fig. 12) checks backend, database, gateway and last-telemetry time with automatic refresh, and lists one card per ESP32 node. A test-ingest button sends a packet through the full chain and returns the resulting analysis with the curl command used (Fig. 13); the captured ST-001 run returned SAFE, 17/100, no rise. A bench serial capture during commissioning reported water status danger, float switch triggered, AJ-SR04M passing at 172.8 cm, DS18B20 passing at 27.8 °C, GPS searching, RGB red and the DHT module failing [KEEP original serial screenshot as Fig. 12b]. These captures show what the diagnostics are for: each sensor passes or reports a distinct fault state. They are not calibration and are not used to claim accuracy.

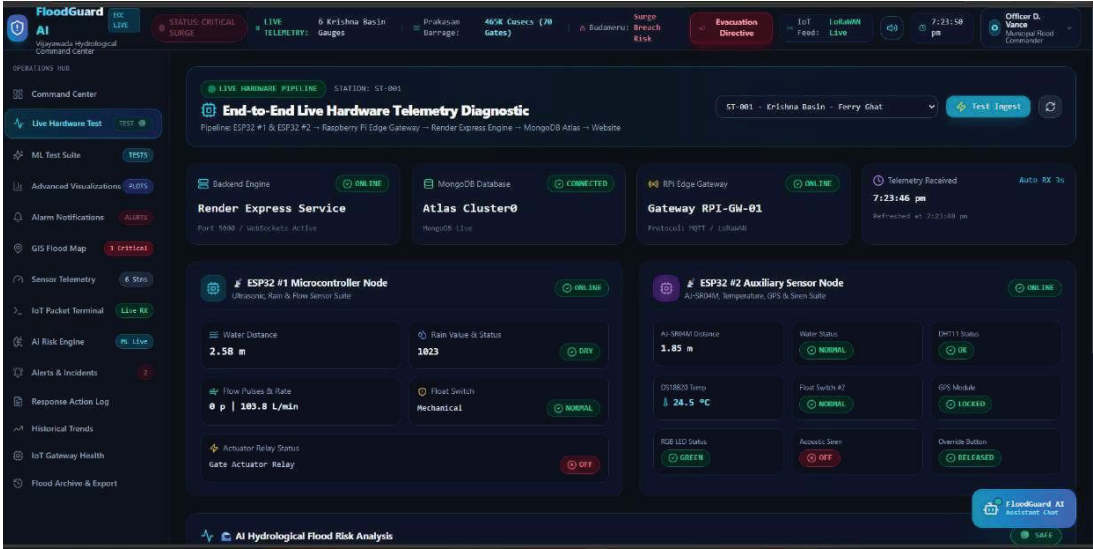


Fig. 12. End-to-end live hardware diagnostic: backend, database, gateway and both ESP32 node cards.



Fig. 13. Test-ingest result and ingestion-API tester.

6) **Gateway health.** The gateway-health page (Fig. 14) shows one card per station with link protocol, battery percentage and voltage, solar state, signal strength, packet loss, firmware build (v2.4.2-esp32-flood) and heartbeat age, plus a ping button. In the ST-001 capture (Fig. 15) the echo returned in 33 ms with battery 92 %, solar charging, −64 dBm and 5 % loss; this single echo illustrates the page and is not a latency benchmark. Table IX lists the readings; stations that discharge their battery under weak solar input and show high loss (ST-003) are those the page is designed to surface early.

TABLE IX. Gateway-health snapshot (scenario mode).

Stn	Link	Battery	Signal	Loss
001	MQTT/LoRa	92 %, charging	−64 dBm	5.0 %
002	HTTPS/4G	84 %, discharging	−72 dBm	12.0 %
003	MQTT/LoRa	76 %, discharging	−81 dBm	45.0 %
004	MQTT/LoRa	98 %, charging	−58 dBm	1.0 %
005	HTTPS/4G	90 %, charging	−60 dBm	8.0 %
006	MQTT/LoRa	94 %, charging	−62 dBm	2.0 %

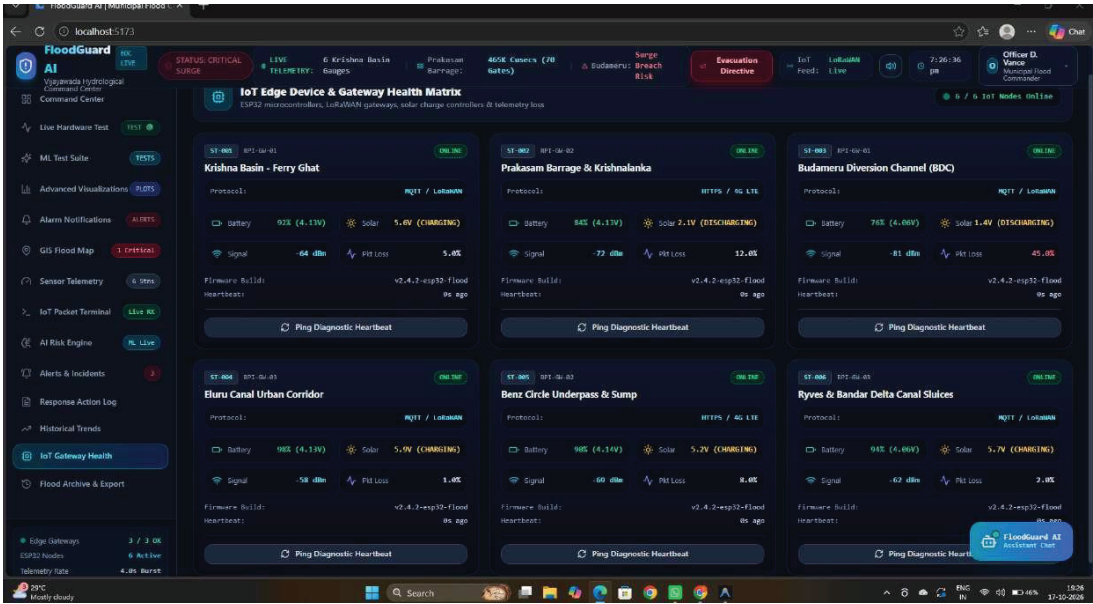


Fig. 14. Gateway and edge-device health matrix for the six stations (scenario-mode data).



Fig. 15. Gateway card for ST-001 with ping echo.

7) **Scenario suite, packet terminal and assistant.** The test suite (Fig. 16) holds four stored cases (Table X) and a sandbox with four sliders; inputs 3.45 m, 35 mm/h, 2.4 m/s and +6.5 cm/h returned 50/100, WATCH, confidence 88 %. The packet terminal (Fig. 17) injects a packet for a chosen station through the production path, triggering rate-of-rise recalculation, the anomaly check and a WebSocket broadcast. An assistant embedded in every page (Fig. 21) carries the selected station as context and answers status and active-alert queries from the same backend state: in the capture it reports ST-001 at 2.65 m, 51 % of channel capacity, SAFE, 20/100, and two active alerts, the latest a high-risk surge at the barrage and Krishnalanka.

TABLE X. Stored scenario cases. Only four hand-written cases exist; Section VIII extends this to a labelled set.

ID	Case	Inputs (level, rain, rise)	Intended label
TC-001	Cloudburst flash surge	4.65 m, 85 mm/h, +12 cm/h	CRITICAL
TC-002	Debris restriction	3.8 m, 25 mm/h, +7.5 cm/h	HIGH RISK
TC-003	Sonar noise and drift	2.1 m, 5 mm/h, +0.5 cm/h	WATCH

ID	Case	Inputs (level, rain, rise)	Intended label
TC-004	Spillway recovery	falling after discharge	SAFE

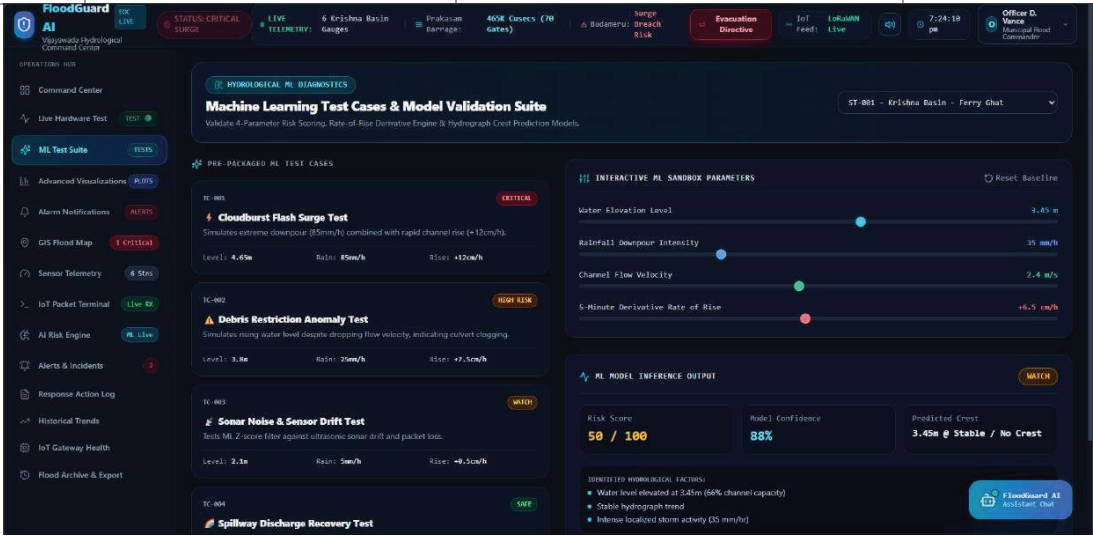


Fig. 16. ML test suite with pre-packaged cases and the interactive sandbox (3.45 m, 35 mm/h, 2.4 m/s, +6.5 cm/h → 50/100, WATCH).

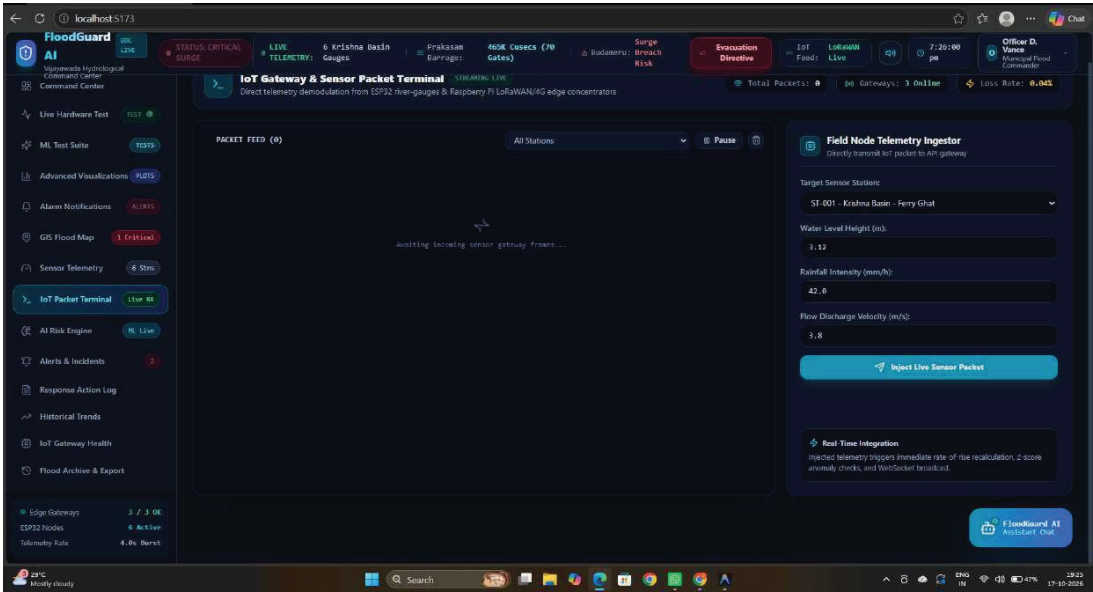


Fig. 17. IoT packet terminal and field-node telemetry injector.

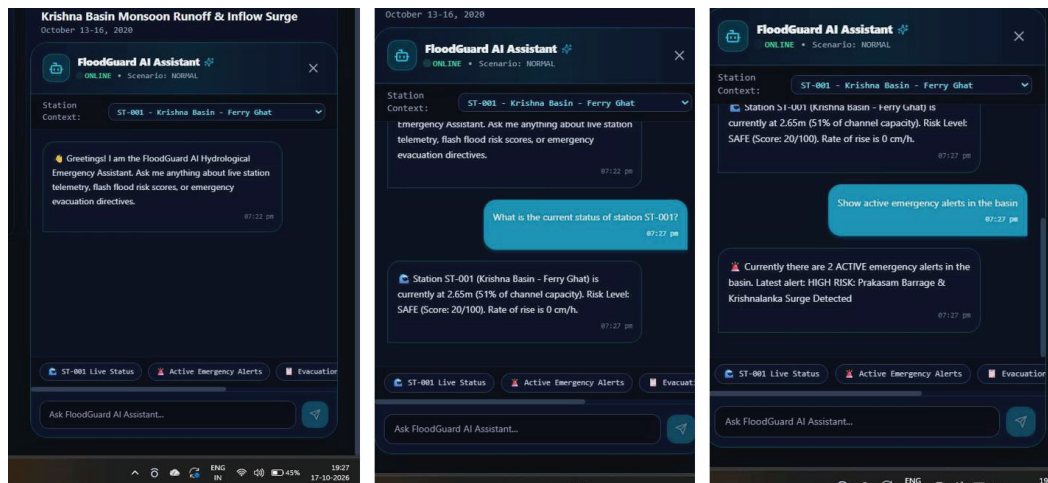


Fig. 21. Assistant exchanges on the ST-001 context: greeting, station status and active emergency alerts.

8) **Archive and review.** The archive page (Fig. 20) holds event cards with peak 24-hour rain, peak water level, sandbags deployed, average response time and a mitigation outcome, supporting the Verify-and-Learn step: after an event, the response log and stored telemetry can be replayed to ask how long acknowledgement took, what was dispatched and what the river did. The cards in the prototype are **illustrative** entries and will be replaced by official records once the platform is deployed alongside the municipal corporation.

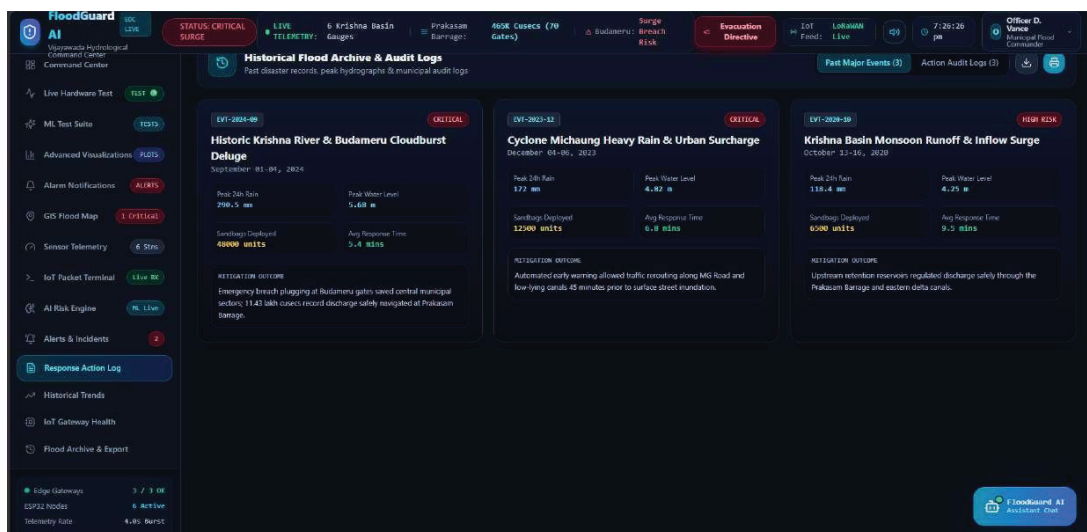


Fig. 20. Historical flood archive and audit-log page (illustrative cards).

B. Synthetic simulation study

Purpose. To answer RQ1 before field data exist, three alert rules were compared on simulated 24-hour, 5-minute-sampled hydrographs for a station with the ST-001 profile (1.80/2.80/3.80/4.60 m).

Rules. (R1) *Fixed threshold*: alert when level ≥ 3.80 m. (R2) *Level plus rate escalation*: alert when level ≥ 3.80 m, or level ≥ 2.80 m with smoothed rise ≥ 10 cm/h (the one-state escalation of Section V-F). (R3) *Weighted score*: alert when $R \geq 60$, using the weights and ranges of the worked example (the 60-point boundary is a simulation setting, not a stated production value [VERIFY]). Rate of rise used a 15-minute difference with $\alpha = 0.3$. Each rule was run with and without the Z-score filter ($\tau_z = 4$, applied when rainfall < 5 mm/h and $|r| > 60$ cm/h).

Generator. Four scenario families, 300 runs each per filter setting: *critical* (peak 4.7–5.6 m, rain pulse 40–100 mm/h leading the rise); *sub-critical* (peak 3.0–3.7 m); *minor* (peak 2.0–2.8 m); and *no flood* (drift, rainfall noise and rare single-echo spikes of 0.3–0.8 m). Rise duration was 5–9 h; sensor noise 0.7 cm. **Lead time** is the time from alert to the level first reaching 4.60 m. A **nuisance alert** is any alert in a run whose true level never reached 3.80 m.

TABLE XI. Synthetic simulation results (not field data), Z-filter on.

Rule	Median lead (min)	IQR (min)	Missed critical	Nuisance: sub-critical	Nuisance: minor	Nuisance: no flood
R1 Fixed 3.80 m	60	55–75	0/300	0/300	0/300	0/300
R2 Level + rate	130	110–150	0/300	300/300	1/300	0/300
R3 Weighted score	195	110–255	0/300	86/300	0/300	0/300

Without the Z-filter, medians were 65, 137.5 and 182.5 min; nuisance counts in the sub-critical family were 2, 300 and 91, and 7 for R2 in the minor family.

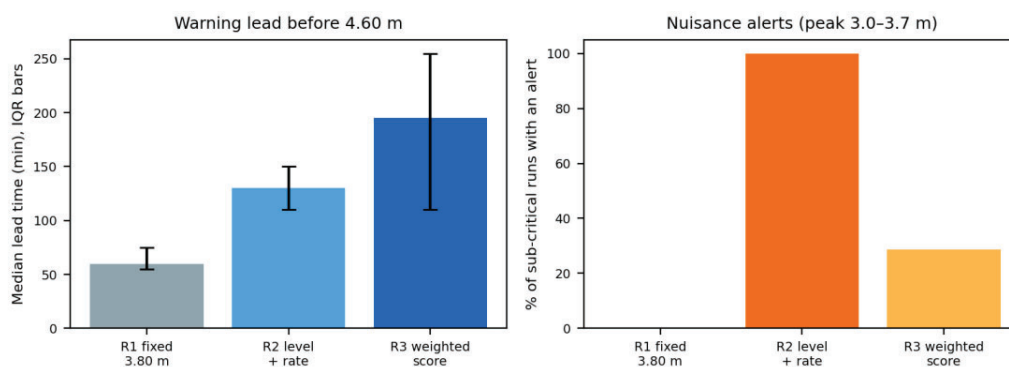


Fig. 22. Synthetic simulation: median warning lead time (IQR bars) and nuisance-alert share in sub-critical runs.

Findings. (i) Both multi-input rules gave longer lead time than the fixed threshold with no missed critical events: +70 min for R2 and +135 min for R3 at the median. (ii) The gain has a price. R2 alerted in every sub-critical run because any run passing 2.80 m at ≥ 10 cm/h triggers it; the 10 cm/h alarm value is therefore too sensitive for this generator, which shows why station alarm values must be tuned on events. R3 alerted in 29 % of sub-critical runs. (iii) The Z-filter had a small effect, removing the two fixed-threshold spike alerts and cutting R2 minor-event alerts from 7 to 1, at the cost of a slightly shorter R2 lead time (137.5 $\rightarrow$ 130 min). (iv) The no-flood family produced no alerts under any rule.

Limits of this evidence. Hydrograph shapes, the rain-leads-level assumption, the noise model and the nuisance definition were chosen by the authors, so absolute minutes should not be read as expected performance on the Krishna basin. The rain-leads-level assumption in particular favours R3. The study supports the *direction* of the effect and the existence of a lead-time/nuisance trade-off; the field protocol must supply the magnitudes.

C. What the prototype does and does not show. Together, the observations show that an inexpensive node, a small gateway and a standard web stack can carry readings from sensor to console and back to a logged response, with risk, alerts and directives computed on one stream. They do not show field accuracy, latency distributions, delivery ratios or classification scores.

VIII. Evaluation Methodology for Field Trials

All trials will record date, duration, ambient conditions, firmware build and sample size; raw logs are retained. Table XII lists measures, references and **proposed acceptance targets**, which are design goals to be agreed with the drainage and irrigation authorities, not results.

TABLE XII. Evaluation measures and proposed targets.

Aspect	Measure	Reference	Proposed target [VERIFY]
Level accuracy	MAE, RMSE, repeatability over working range	Graduated scale	MAE ≤ 1 cm on bench; set per site
Rate of rise	Error over fixed windows	Scale and timer	Within ± 2 cm/h at 15-min window
Latency	Node $\rightarrow$ gateway $\rightarrow$ storage $\rightarrow$ browser	Synchronised clocks	95th percentile ≤ 10 s (LoRa)
Delivery	PDR per link; ordered recovery of buffered records	Packet counters	PDR ≥ 95 %; 100 % ordered recovery
Classification	Precision, recall, F1, confusion matrix, false-alarm rate	Labelled events	Recall ≥ 0.9 for HIGH RISK and CRITICAL
Forecast	MAE, RMSE, Nash–Sutcliffe efficiency [25]	Held-out events	Beat Eq. (6) baseline
Alerts	Lead time, misses, false alerts, time to acknowledge	Event record	Median lead ≥ 60 min
Calibration	Brier score, reliability diagram [24]	Observed outcomes	Before confidence is shown as probability
Power	Current draw, uptime, recovery	INA219, logs	Survives ≥ 3 days without sun

A. Sensor accuracy. The bench tank is raised and lowered across the working range against a graduated scale. For N paired readings,

$$MAE = (1/N) \sum |h_i - h_{ref,i}|, RMSE = \sqrt{(1/N) \sum (h_i - h_{ref,i})^2} \quad (8)$$

with repeatability as the standard deviation at a fixed level [15]. Temperature, foam and floating-object cases are repeated deliberately.

B. Communication. Latency is the difference between node transmission, gateway receipt, backend storage and browser update times. Reliability is

$$PDR = N_{rx} / N_{tx} \times 100 \% \quad (9)$$

per link profile over a stated packet count, including trials with the uplink disconnected.

C. Classification and forecast skill. A labelled scenario set from bench runs and, when available, recorded events scores the four-class output [16]:

$$P = TP/(TP + FP), Rc = TP/(TP + FN), FI = 2 \cdot P \cdot Rc / (P + Rc) \quad (10)$$

Because flood classes are rare, precision–recall summaries are preferred over ROC curves alone [26]. To avoid leakage, tuning and test sets are split *by event*, never by sample; weights and alarm values are tuned on one monsoon period and tested on another. Confidence intervals come from bootstrap resampling of events.

D. Alert performance. The warning lead time is

$$T_{lead} = t_{crit} - t_{alert} \quad (11)$$

summarised with misses, false alerts and alert-to-acknowledgement time, and compared against the fixed-threshold rule R1 on the same events, as in the simulation, so that RQ1 is answered with field data. An independent evaluator, not the authors, should choose or confirm the test scenarios.

IX. Comparison with Existing Practice

Table XIII compares design capability with manual practice and fixed-threshold telemetry; it describes capability, not measured superiority, except for the lead-time and nuisance-alert trade-off simulated in Section VII-B.

TABLE XIII. Comparison with existing approaches.

Feature	Manual / periodic	IoT telemetry + threshold	FloodGuard AI (proposed)
Data capture	Visits, registers	Continuous, single parameter	Level, flow, rain, float, temperature, device health
Edge handling	None	Often none	Validation, gateway timestamp, in-order buffer
Risk logic	Judgement	Fixed threshold	Weighted score + rate of rise + station profile
Fault awareness	None	Rarely	Z-score filter, redundant-probe and float-switch checks
Explanation	Verbal	Alarm only	Plain-language drivers and directive
Spatial view	Paper maps	Basic list	GIS map, flood plains, shelters, barrage panel
Alert workflow	Phone calls	Notification	Acknowledge / resolve / dispatch with attribution
Traceability	Minimal	Limited	Append-only log, replayable telemetry
Scalability	Very low	Moderate	Stations added by configuration

X. Limitations and Threats to Validity

What is not shown. No field accuracy, calibration curve, latency distribution, delivery ratio or classification score is reported. Stations ST-002 to ST-006 run in scenario mode. Model confidence is not a calibrated probability. Archive cards are illustrative. Section VII-B is synthetic.

Consistency items to resolve before any measured claim [VERIFY]. Review of the captured screens found the following, each of which affects how numbers may be quoted:

TABLE XIV. Observations in the captured screens that need reconciliation.

Item	Observation	Action
Float switch vs SAFE	The test-ingest curl shows floatStatus “TRIGGERED” and relay “ON”, yet the result is SAFE; Section V-F says a triggered float never yields SAFE	Confirm whether the curl text is a static template, or fix the veto
Redundant probes	Live page: ESP32 #1 distance 2.58 m, AJ-SR04M 1.85 m, both “normal”	State mounting heights; show how disagreement is flagged
ST-001 rate of rise	0 cm/h (telemetry card, assistant), -62.5 cm/h (inspector), +59.4/+62.5/+68.8 cm/h (risk engine)	Caption each as a different injected state with time
ST-001 score and confidence	47/100 (risk engine), 20/100 (assistant), 17/100 (test ingest); confidence 96 % (Krishna v2.4), 90 % (Hydrology v2.1), 88 % (sandbox)	State one model version per figure
ST-001 rainfall	3.9 mm/h on the card, 5.5 mm/h in the inspector	Use one source and refresh time
Basin index	41/100 labelled CRITICAL, while station scores of 47 and 62 are WATCH and HIGH RISK	State the basin-index bands
Station name	ST-004 is Eluru Canal Corridor in the cards but Guntur Canal in the hydrograph legend	Use one name
DHT device	Serial capture reports a failing DHT11; inventory lists a DHT22; live page shows DHT healthy	State the exact device per capture
Ingest level	Raw level in the curl command differs from the analysed level	Document the distance-to-level conversion
Flow card	Zero pulse count beside 103.8 L/min	Define pulse-to-rate conversion and refresh timing
Worked example	51 (illustrative weights) vs 50 (production)	Publish production weights and ranges

Threats to validity. Ultrasonic ranging is affected by temperature, turbulence, foam and debris, all worse on a real channel than on a bench. LoRa range depends on terrain and antenna placement, and floods may degrade

links. Thresholds must come from channel surveys and local engineering experience; score weights must be cross-checked on events outside the tuning set, and a score tuned on one catchment need not transfer. Scenarios chosen by the authors, and a generator written by the authors, may flatter the logic; independent validation with the municipal authorities is planned. The platform supports official procedures and does not replace them.

XI. Security, Scalability and Deployment

Security. Operators authenticate individually so each acknowledgement and dispatch is attributable; roles separate viewing, acknowledging and dispatching from administration. Devices carry individual credentials so a stolen node can be revoked alone; telemetry uses TLS on both profiles; the ingestion API rejects unknown station IDs and applies rate limits so a faulty or hostile client cannot flood the database or raise false alerts. The packet terminal and broadcast simulator must be disabled or admin-only in production, and role-based access, authenticated devices and encrypted transport are prerequisites for connecting the broadcast channel to a real gateway.

Scalability. Stations are configuration records (coordinates, type, elevation, four-stage profile). The ingestion service is stateless apart from its database, so instances can sit behind a load balancer and WebSocket channels can be sharded by basin. At scale the limit is likely operator attention, which the basin risk index, flashpoint count and sorted alert queue are meant to manage.

Roll-out and maintenance. Commodity parts let a municipality start at priority locations such as underpasses, canal head regulators and known choke points. A staged plan is proposed: bench calibration; a supervised pilot of one monsoon season in parallel with manual procedures; then formal linkage to the control room, with Section VIII measurements deciding each step. Probe cleaning, battery and solar checks guided by the gateway-health page, scheduled recalibration and firmware tracking turn maintenance from calendar-based to condition-based.

XII. Conclusion

FloodGuard AI connects ESP32 nodes, a Raspberry Pi gateway, a cloud service and a web console into one loop from sensing to logged response. Its transparent score, safeguards and fault routing keep operators in control, and its provenance discipline keeps demonstration values apart from evidence. A synthetic study indicates that rate-of-rise escalation and the weighted score can lengthen warning lead time over a fixed threshold, at the price of more alerts on sub-critical events, which is a tuning problem that only field data can settle. The central claim remains architectural and procedural; claims about accuracy, latency and real warning lead time await the protocol of Section VIII.

XIII. Future Scope

Calibrate the sensors on the bench tank and report error figures; run latency and delivery trials on both link profiles; collect a monsoon season of data and label events with local engineers; tune alarm values and weights on one period and test on another; compare against random forest, gradient boosting and recurrent models; calibrate the confidence value; replace the rain module with a tipping-bucket gauge and add camera verification; connect rainfall forecasts, shelter and road-closure data; connect the broadcast module to an official gateway under an agreed protocol; add Telugu, Hindi and English alerts, SMS and a field-team mobile app. The aim is a deployment in which every number on screen has a measured error bar behind it.

Acknowledgment

The authors thank the faculty and management of Andhra Loyola Institute of Engineering and Technology, Vijayawada, and the engineering team of Sujju Software Solutions for laboratory access, hardware support and review.

References

- [1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, 2010.
- [2] E. Basha, S. Ravela, and D. Rus, "Model-based monitoring for early warning flood detection," in *Proc. 6th ACM SenSys*, 2008, pp. 295–308.
- [3] P. Mitra et al., "Flood forecasting using Internet of Things and artificial neural networks," in *Proc. IEEE IEMCON*, 2016, pp. 1–5.
- [4] A. Mosavi, P. Ozturk, and K. Chau, "Flood prediction using machine learning models: Literature review," *Water*, vol. 10, no. 11, Art. 1536, 2018.
- [5] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
- [6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD*, 2016, pp. 785–794.
- [7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [8] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.
- [9] F. Kratzert et al., "Rainfall–runoff modelling using long short-term memory (LSTM) networks," *Hydrol. Earth Syst. Sci.*, vol. 22, no. 11, pp. 6005–6022, 2018.
- [10] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, 2016.
- [11] A. Banks, E. Briggs, K. Borgendale, and R. Gupta, Eds., *MQTT Version 5.0*, OASIS Standard, Mar. 2019.
- [12] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: The rising stars in the IoT and smart city scenarios," *IEEE Wireless Commun.*, vol. 23, no. 5, pp. 60–67, 2016.
- [13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Art. 15, 2009.
- [14] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
- [15] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *Int. J. Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.
- [16] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manage.*, vol. 45, no. 4, pp. 427–437, 2009.
- [17] V. V. Krzhizhanovskaya et al., "Flood early warning system: Design, implementation and computational modules," *Procedia Comput. Sci.*, vol. 4, pp. 106–115, 2011. [VERIFY]
- [18] M. Mousa, X. Zhang, and C. Claudel, "Flash flood detection in urban cities using ultrasonic and infrared sensors," *IEEE Sensors J.*, vol. 16, no. 19, pp. 7204–7216, 2016. [VERIFY]
- [19] G. Nevo et al., "Flood forecasting with machine learning models in an operational framework," *Hydrol. Earth Syst. Sci.*, vol. 26, no. 15, pp. 4013–4032, 2022. [VERIFY]
- [20] F. Kratzert et al., "Towards learning universal, regional, and local hydrological behaviors via machine learning applied to large-sample datasets," *Hydrol. Earth Syst. Sci.*, vol. 23, no. 12, pp. 5089–5110, 2019. [VERIFY]
- [21] World Meteorological Organization, *Manual on Flood Forecasting and Warning*, WMO-No. 1072, 2011. [VERIFY]
- [22] F. Adelantado et al., "Understanding the limits of LoRaWAN," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 34–40, 2017. [VERIFY]
- [23] A. H. Murphy, "What is a good forecast? An essay on the nature of goodness in weather forecasting," *Weather Forecast.*, vol. 8, no. 2, pp. 281–293, 1993. [VERIFY]
- [24] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. ICML*, 2017, pp. 1321–1330. [VERIFY]
- [25] J. E. Nash and J. V. Sutcliffe, "River flow forecasting through conceptual models part I," *J. Hydrol.*, vol. 10, no. 3, pp. 282–290, 1970. [VERIFY]
- [26] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS ONE*, vol. 10, no. 3, e0118432, 2015. [VERIFY]