IoT-Enabled Multi-Method Sensor Failure Diagnosis for Industrial Monitoring Using ESP32 with Cross-Validation of Soil, Temperature and Inductive Proximity Channels

Mr.Shailesh A. Shinde[1], Pranali V. Bhore[2], Arpita S. Khot[3], Samiksha S. Dinde[4], Revati S. Sankpal[5]
[1,2,3,4,5]*D.K.T.E. Society's Textile & Engineering Institute Yashwantaro Chavan Polytechnic, Ichalkaranji, Maharashtra, India.416115*

***Abstract*—Undetected sensor failures quietly corrupt process data and lead to wrong decisions, product loss and unplanned stoppages in industrial monitoring. This paper presents a low-cost, IoT-enabled framework that diagnoses sensor failures in real time, before faulty readings reach the control or reporting layer. The prototype is built around an ESP32 microcontroller and observes three heterogeneous channels: a capacitive soil-moisture sensor, a digital temperature sensor and an inductive proximity switch that detects metal. Six lightweight detection methods are combined: range and open/short-circuit screening, rate-of-change limiting, stuck-at (flatline) detection, noise-level monitoring, timeout and chatter supervision of the discrete proximity output, and cross-channel common-mode checking. A persistence rule turns individual flags into a three-level health state (normal, warning, fault) for every channel. The state is shown on an I2C LCD, a relay drives an alarm or isolates the affected line, and readings with fault codes are streamed to ThingSpeak for remote trend analysis. A fault-injection protocol and the evaluation metrics (detection rate, false-alarm rate and detection latency) are defined so that the methods can be compared on equal terms. The framework needs no process model, runs on commodity hardware and can be extended to further sensor types.**

***Index Terms*—Sensor fault detection, condition monitoring, Internet of Things, ESP32, ThingSpeak, inductive proximity switch, industrial diagnostics.**

I. INTRODUCTION

Modern plants rely on large numbers of low-cost sensors whose outputs feed displays, alarms and closed-loop controllers. When one of these sensors fails, the failure is rarely announced: a broken wire may read as a plausible constant, a degraded probe may drift slowly, and a worn proximity switch may chatter only under vibration. Because the downstream system still receives numbers, the error is treated as truth. Diagnosing the health of the sensor itself, before the data is used, is therefore a necessary layer of any dependable monitoring system [1], [3].

Classical fault-diagnosis theory offers model-based, signal-based and knowledge-based routes [1], [2], [5], [6]. Model-based methods need an accurate process model, which small and medium plants seldom have. Signal-based checks, in contrast, work directly on the measured waveform and are cheap enough to run on a microcontroller. Much of the available literature treats each check in isolation and on a single sensor type, and few studies place several checks side by side on a platform that is also connected to the cloud.

This work therefore contributes

(i) a sensor-health framework that applies six complementary signal-based checks to analog, one-wire digital and discrete-switch channels on one low-cost controller;

(ii) a persistence-based decision rule that limits false alarms;

(iii) an actuation and visualisation path consisting of a local I2C LCD, a relay and ThingSpeak; and

(iv) a reproducible fault-injection plan with metrics for comparing the methods.

The remainder of the paper is organised as follows. Section II reviews background, Section III describes the system, Section IV details the detection methods, Section V explains the IoT layer, Section VI gives the evaluation plan, Section VII presents the results format, and Section VIII concludes.

II. BACKGROUND AND RELATED WORK

Surveys of fault detection distinguish methods by the knowledge they need about the monitored process [2], [5]. Quantitative model-based approaches compare a model output with the measurement and examine the residual [1], [6]. Instrument-level studies focus on the sensor as the object of diagnosis and highlight redundancy, limit checking and statistical tests as practical tools [3]. Anomaly-detection research adds a general vocabulary of point, contextual and collective anomalies that maps naturally onto spikes, drift and flatlines in sensor streams [4].

For discrete devices, the standard for proximity switches defines their electrical behaviour and operating distances [9], but it does not prescribe how a user should supervise a switch that has silently stuck or begun to chatter. Digital temperature sensors also expose characteristic error codes that can be used as free diagnostic information [10]. The present work builds on these ideas and combines them in a single, inexpensive, internet-connected diagnostic unit.

III. PROPOSED SYSTEM

A. Architecture

Fig. 1 shows the architecture. The ESP32 [7] samples the three sensing channels, runs the diagnosis engine, and acts on the result through three output paths: the I2C LCD for the operator at the machine, the relay for an immediate physical response, and the Wi-Fi link to ThingSpeak [8] for remote supervision. The proximity switch output is conditioned by a level shifter because its supply and output swing exceed the 3.3 V tolerance of the ESP32 pins.

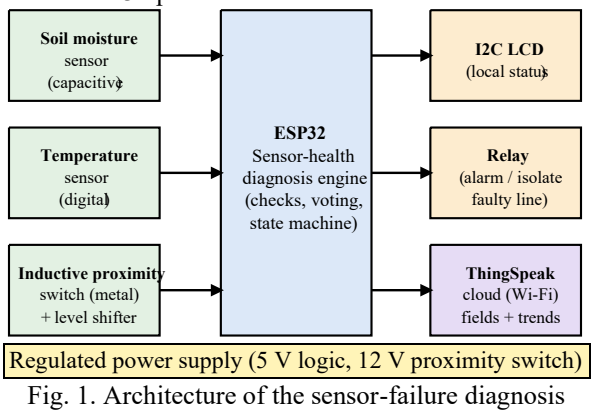

Fig. 1. Architecture of the sensor-failure diagnosis system.

B. Hardware

The soil-moisture channel uses a capacitive probe with an analog output read by the 12-bit ADC. Capacitive probes are preferred to resistive ones because their exposed electrodes do not corrode, which removes a major cause of slow drift. The temperature channel uses a one-wire digital sensor. The metal-detection channel uses an inductive proximity switch with a discrete output. Table I lists the components and their roles.

TABLE I COMPONENTS AND ROLES

Block	Component	Role
Controller	ESP32 module	Sampling, diagnosis, Wi-Fi
Soil channel	Capacitive soil-moisture sensor	Analog moisture level
Temperature channel	One-wire digital temperature sensor	Temperature, error codes
Metal channel	Inductive proximity switch + level shifter	Presence of metal part
Local display	16x2 LCD with I2C backpack	Health state per channel
Actuation	Relay module	Alarm / line isolation
Cloud	ThingSpeak channel	Logging, plots, alerts

IV. FAULT DETECTION METHODOLOGY

Each sampling cycle produces a measurement x for every k

channel and a set of binary flags, one per applicable check. The checks are deliberately simple so that they execute within milliseconds on the microcontroller.

A. Range and Open/Short-Circuit Check

A reading outside the physically possible band of the sensor, or equal to a known error signature, is flagged. Open or shorted signal lines typically push an analog input to a rail or to an implausible constant, and the digital temperature sensor reports fixed codes on bus loss or reset [10].

$$x \le x \le x \quad (1) \text{ min } k \text{ max}$$

B. Rate-of-Change Check

Thermal and moisture processes are slow. A step larger than the process can physically produce between samples indicates a spike, loose contact or interference rather than a real change.

$$| x - x | / T > R \qquad (2)$$ k k−1 s max

C. Stuck-at (Flatline) Check

A healthy analog channel always carries a small amount of natural variation. If the variance over a window of N samples stays below a small threshold for a hold time T , the channel hold
is considered frozen.

$$\mathrm{Var}(x \ldots x) < \varepsilon \text{ for } t > T \qquad (3)$$ k−N+1 k hold

D. Noise-Level Check

The opposite symptom, excessive scatter, signals electromagnetic interference, a poor connector or a failing element. The windowed standard deviation is compared with a reference value learned during commissioning.

$$S > K \cdot S \qquad (4)$$ w ref E. Proximity-Switch Supervision

A discrete output cannot be tested with range or variance rules, so two timing rules are used. First, a timeout rule flags a switch that has not changed state for longer than the expected process cycle T while the process is known to be running; max
this catches stuck-on and stuck-off outputs. Second, a chatter rule counts state toggles inside a short window and flags the switch when the count exceeds a limit C , which reveals max
marginal mounting distance or vibration damage. A cable break on a normally-open output looks identical to the absence of metal; the framework therefore recommends a normally-closed or supervised wiring arrangement for critical positions.

E. Cross-Channel Common-Mode Check

When two or more independent channels show an abrupt change in the same sampling cycle, the cause is more likely a shared fault, such as a supply dip, ground disturbance or reference error, than several simultaneous sensor failures. The system raises a common-mode flag in this case and suppresses individual fault declarations for that cycle, which reduces misleading alarms.

F. Decision Logic

To avoid reacting to a single glitch, a channel moves to WARNING when any flag is raised and to FAULT when flags occur in at least M of the last N cycles. It returns to NORMAL d
only after R consecutive clean cycles. All thresholds are set during a short commissioning run on the healthy installation. Table II summarises how the methods map to the channels.

TABLE II MAPPING OF DETECTION METHODS TO CHANNELS

Channel	Failure mode	Method	Indicator
Soil	Open / short, drift, stuck, noise	A, B, C, D	ADC at rail, flat or noisy
Temperature	Bus loss, reset, spike, stuck	A, B, C	Error code, step, flat
Proximity	Stuck on/off, chatter	E	Timeout, toggle count
All analog	Supply or ground fault	F	Simultaneous jump

V. IOT AND ACTUATION LAYER

Local display: the I2C LCD uses only two signal lines and shows the three channel states and the most recent fault code. Relay: on a confirmed fault the relay is energised to sound an alarm or to isolate the affected line, and it is released only after the channel has recovered. Cloud: each cycle the ESP32 uploads the sensor values and a numeric health code to ThingSpeak fields over HTTP. The free service accepts updates at intervals of roughly fifteen seconds [8], so fast fault logic runs locally on the controller while the cloud is used for supervision, trend plots and optional alert rules. Table III lists a possible field assignment.

TABLE III THINGSPEAK FIELD ASSIGNMENT

Field	Content
1	Soil moisture (calibrated %)
2	Temperature (°C)
3	Proximity state / pulse count
4	Health code, soil
5	Health code, temperature
6	Health code, proximity
7	Relay state

VI. EVALUATION PLAN

A. Fault-Injection Protocol

Faults are injected deliberately on the working prototype, one at a time, so that the ground truth is known. Table IV lists the scenarios. Each scenario is repeated n times, and a set of healthy runs is recorded to measure false alarms.

TABLE IV FAULT-INJECTION SCENARIOS

ID	Injected fault	Channel
F1	Signal wire disconnected	All
F2	Signal shorted to ground / supply	Soil, temp.
F3	Output frozen (stuck-at)	All
F4	Electrical noise near relay / motor	Analog
F5	Gradual drift (probe ageing)	Soil
F6	Marginal mounting, vibration	Proximity
F7	Supply dip	All

B. Metrics

With TP, FN, FP and TN denoting true positives, false negatives, false positives and true negatives, the following are computed for every method and scenario:

$$DR = TP / (TP + FN), \quad FAR = FP / (FP + TN) \qquad (5)$$

$$L = t - t \qquad (6) \text{ detect inject}$$

where DR is the detection rate, FAR the false-alarm rate and L the detection latency measured from the instant of injection to the instant the fault is declared. Each method is first evaluated alone and then in the combined framework so that its individual contribution is visible.

VII. RESULTS

The measured outcomes of the fault-injection campaign are reported in Table V, with the three metrics for each scenario and each method, together with the cloud-update delay observed on ThingSpeak. Discussion of the results should state which methods were effective for which failure modes, for example whether the stuck-at check alone is sufficient for the digital temperature channel, and how much the persistence rule reduced false alarms relative to single-flag alarms.

[INSERT MEASURED VALUES BEFORE SUBMISSION — do not publish unmeasured numbers]

VIII. DISCUSSION AND LIMITATIONS

The signal-based approach is attractive because it needs no process model and fits within the memory and speed of a low-cost controller. Its limits are also clear. Single-sensor checks cannot separate a true process excursion from a fault that happens to look like one; redundancy or an analytical model is needed for that. Thresholds depend on the installation and must be recommissioned when the process changes. The cloud path adds delay and depends on network availability, so safety-relevant actions stay local. Finally, the prototype uses one example of each sensor type and does not replace certified safety instrumentation.

IX. CONCLUSION AND FUTURE WORK

A multi-method sensor failure diagnosis framework was presented for an ESP32 platform with soil-moisture, temperature and inductive proximity channels, an I2C LCD, a relay and ThingSpeak. By combining six lightweight checks with a persistence-based decision rule, the system turns raw sensor data into an explicit health state that is available locally and remotely. Future work includes adding redundant sensors for voting, learning thresholds automatically from healthy data, using lightweight machine-learning classifiers for drift, and extending the framework to further industrial sensor types.

REFERENCES

[1] R. Isermann, Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance. Berlin, Germany: Springer, 2006.

[2] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri,“A review of process fault detection and diagnosis: Part I: Quantitative model-based methods,” Comput. Chem. Eng., vol. 27, no. 3, pp. 293–311, 2003.

[3] G. Betta and A. Pietrosanto, “Instrument fault detection and isolation: State of the art and new research trends,” IEEE Trans. Instrum. Meas., vol. 49, no. 1, pp. 100–107, Feb. 2000.

[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: Asurvey," ACM Comput. Surv., vol. 41, no. 3, Art. no. 15, 2009.
[5] R. Isermann, "Process fault detection based on modeling andestimation methods—A survey," Automatica, vol. 20, no. 4, pp. 387–404, 1984.
[6] S. X. Ding, Model-Based Fault Diagnosis Techniques. Berlin, Germany: Springer, 2008.
[7] Espressif Systems, "ESP32 series datasheet." [Online]. Available: https://www.espressif.com
[8] MathWorks, "ThingSpeak documentation." [Online]. Available: https://www.mathworks.com/help/thingspeak/
[9] Low-Voltage Switchgear and Controlgear—Part 5-2: Control Circuit Devices and Switching Elements—Proximity Switches, IEC Standard 60947-5-2.
[10] Maxim Integrated, "DS18B20 programmable resolution 1-wiredigital thermometer," datasheet.
[11] A. Gaddam, T. Wilkin, M. Angelova, and J. Gaddam, "Detecting Sensor Faults, Anomalies and Outliers in the Internet of Things: A Survey on the Challenges and Solutions," Electronics, vol. 9, no. 3, Art. no. 511, 2020.
[12] Edge-to-Cloud IIoT for Condition Monitoring in Manufacturing Systems with Ubiquitous Smart Sensors, Sensors, vol. 22, no. 15, Art. no. 5901, 2022.
[13] M. Uppal et al., "Cloud-Based Fault Prediction for Real-Time Monitoring of Sensor Data in Hospital Environment Using Machine Learning," Sustainability, vol. 14, no. 18, Art. no. 11667, 2022.
[14] H. Darvishi, D. Ciuonzo, and P. Salvo Rossi, "A Machine-Learning Architecture for Sensor Fault Detection, Isolation, and Accommodation in Digital Twins," IEEE Sensors Journal, 2023.
[15] D. Kong et al., "Sensor Anomaly Detection in the Industrial Internet of Things Based on Edge Computing," Turkish Journal of Electrical Engineering and Computer Sciences, vol. 28, no. 1, pp. 331–346, 2020.
[16] T. Yang et al., "Cloud-Edge Collaborative Data Anomaly Detection in Industrial Sensor Networks," PLOS ONE, vol. 20, no. 6, 2025.
[17] Espressif Systems, "ESP32 Series Datasheet," technical datasheet.
[18] Maxim Integrated, "DS18B20 Programmable Resolution 1-Wire Digital Thermometer," datasheet.
[19] MathWorks, "ThingSpeak: IoT Analytics Platform," technical documentation.
[20] Low-Voltage Switchgear and Controlgear—Part 5-2: Control Circuit Devices and Switching Elements—Proximity Switches, IEC Standard 60947-5-2. angeltechnohubsolutions.in