

FluidCare: Automated Fluid Management in Healthcare Using IOT

Prof. Rakhi S. Punwatkar¹, Aniket Nitin Kumawat², Kartik Anil Nagare², Atharva Manish Shivari², Sahil Sanjay Barwal²

¹Department of Computer Engineering (Faculty)

²Department of Computer Engineering

Zeal College of Engineering and Research, Pune, India

{rakhi.punwatkar08, aniketkumawat8, kartiknagare3165, shivariatharva, sahilbarwal22}@gmail.com

Abstract—Manual monitoring of intravenous (IV) fluid bags in hospitals remains prone to delays, overflow events, and adverse patient outcomes, particularly in understaffed or rural facilities. This paper presents FLUIDCARE, a working prototype with a production-oriented architecture for real-time fluid-bag weight monitoring; components remaining for deployment are enumerated in Section VII. The architecture comprises ESP32-based sensor nodes equipped with HX711 24-bit analog-to-digital converters and single-point load cells, a repeater relay layer for range extension, a master gateway that bridges ESP-NOW to MQTT over TLS, and a Django/Channels backend with Redis-backed signal processing and a React dashboard. We detail the complete signal pipeline: on-device interquartile-range (IQR) median filtering, a calibrated linear weight conversion ($m = 1.18721 \times 10^{-3}$ g/count, $b = 683.22$ g), server-side monotonic-depletion validation, and exponentially weighted moving average (EWMA) smoothing ($\alpha = 0.2$). Calibration experiments conducted over 13 reference masses spanning 0–575 g and $N = 7,855$ raw observations yield $R^2 = 0.9995$, RMSE = 3.47 g, and MAE = 2.87 g. The device exhibits short-term repeatability of $\pm 1-2$ g (1σ) and an absolute calibration accuracy of RMSE = 3.47 g (MAE = 2.87 g) across the 0–575 g range. We further describe the threshold-based alert subsystem, the WebSocket real-time broadcast pipeline, and a multi-tenant hospital data model supporting floors, wards, beds, and patients. Four open prototype limitations are discussed, along with directions for future work.

Index Terms—Internet of Things, IV fluid monitoring, ESP32, ESP-NOW, MQTT, HX711, load cell, EWMA, hospital automation, patient safety, real-time alerting, Django Channels, WebSocket.

I. INTRODUCTION

Intravenous (IV) fluid therapy is one of the most ubiquitous interventions in modern hospital care, routinely administered to critically ill, surgical, and general medical patients across all acuity levels [1]. Despite its prevalence, the monitoring of individual fluid bags remains a manual task delegated almost entirely to nursing staff. A nurse managing a multi-bed ward must conduct periodic visual inspections of every bag—a workflow that breaks down under high patient loads, staff fatigue, or night-shift conditions. The consequences range from bag exhaustion and air embolism to blood backflow and delayed resuscitation, all of which are preventable with timely intervention [2].

Prior automated approaches have addressed fragments of this problem. Early systems relied on buzzer-based in-room

alerts or GSM-based SMS notifications [3], which introduced noise pollution and incurred cellular-network costs without enabling centralised monitoring. Camera-based computer-vision solutions demonstrated high accuracy but require GPU infrastructure, making them impractical for resource-constrained clinics [9]. Load-cell approaches using ESP32 microcontrollers and MQTT messaging have shown promise for lightweight sensing [7], but published implementations typically cover a single fluid type, omit relay infrastructure, lack server-side signal validation, and do not describe full backend architectures.

This paper addresses these gaps by presenting FLUIDCARE. Our contributions are:

- A three-tier wireless mesh—sensor nodes → repeater nodes → master gateway—enabling coverage across multi-floor hospital environments without infrastructure Wi-Fi dependency.
- A two-tier signal pipeline combining on-device IQR-median filtering with server-side monotonic-depletion validation and EWMA smoothing.
- A calibrated and characterised load-cell sensor model over 7,855 observations across 13 reference masses ($R^2 = 0.9995$, RMSE = 3.47 g).
- A multi-tenant backend with a WebSocket broadcast pipeline, threshold-based alerts with 30-minute debouncing, and a React dashboard.
- An honest enumeration of four open prototype limitations with a concrete roadmap toward clinical deployment.

The remainder of this paper is structured as follows. Section II reviews related work. Section III describes the system architecture. Section IV details the sensing and signal-processing pipeline. Section V presents the calibration methodology and results. Section VI covers the backend, alerting, and data model. Section VIII discusses prototype limitations. Section IX concludes.

II. RELATED WORK

The automated monitoring of IV fluids has attracted increasing research interest over the past decade. We organise the literature into four categories.

A. Alert-Only and SMS-Based Systems

Early systems triggered an audible alarm when fluid reached a critical level. While conceptually simple, buzzer alerts produce noise pollution unsuitable for clinical environments and offer no remote monitoring capability. GSM-based SMS systems [3] improved reach but depend on cellular infrastructure, incur per-message costs, and cannot support sub-second alert latency. Moovendan et al. [3] remain a widely cited baseline; their Arduino-based architecture is limited to a single IV-bag type and provides no multi-bed dashboard.

B. Load-Cell and ESP32 Platforms

Ghosh et al. [7] proposed an ESP32 plus HX711 load-cell sensor transmitting via MQTT-S, demonstrating low-power Wi-Fi-compatible sensing suitable for continuous monitoring. Paranthaman and Annalakshmi [4] extended this by integrating solenoid valve control for active infusion management. Patil et al. [5] presented IVigilance, a multi-bed centralised dashboard, confirming the feasibility of ward-level monitoring. None of these implementations describe a relay layer for range extension, systematic calibration over many observations, or server-side plausibility filtering.

C. Computer-Vision Approaches

Giaquinto et al. [9] applied a YOLO-based detector to drip-rate monitoring from camera images, achieving high accuracy but requiring a GPU and continuous internet connectivity. Bai et al. [8] deployed an IoT infusion control system in a hospital setting at high per-bed hardware cost. These approaches are unsuitable for resource-constrained rural hospitals.

D. Holistic IoT Health Platforms

Nimi et al. [6] integrated IV fluid monitoring with vital-sign sensing into a unified cloud platform, providing the most comprehensive monitoring scope among the reviewed works. Their reliance on continuous internet limits applicability in settings with intermittent connectivity—a gap FLUIDCARE addresses through the ESP-NOW local mesh.

Table I summarises the comparative landscape.

III. SYSTEM ARCHITECTURE

FLUIDCARE is organised into four functional layers: (1) bedside sensor nodes, (2) repeater relay nodes, (3) a master gateway node, and (4) a cloud backend with a web dashboard. All intra-device communication uses the ESP-NOW protocol on IEEE 802.11 channel 6, eliminating dependence on hospital Wi-Fi infrastructure for the sensing path. Cloud connectivity is maintained only by the master node, which bridges local ESP-NOW traffic to a HiveMQ MQTT broker over TLS 1.2 on port 8883.

A. Sensor Node (Bedside Layer)

Each sensor node is built around an ESP32 SoC connected to a single-point strain-gauge load cell via an HX711 24-bit sigma-delta ADC operating at gain 128 on channel A. The node operates in Wi-Fi station mode with sleep disabled to

sustain ESP-NOW receive windows. On first boot, a UUID-based registration handshake (protocol codes 200–202) persists the assigned identifier to non-volatile storage (NVS), so subsequent reboots skip re-registration. The node samples the load cell at an 80 ms cadence, applies IQR-median filtering over a ten-sample rolling buffer (Section IV), and transmits a weight reading (code 203) when the delta from the last transmitted weight exceeds 1 g and at least 3,000 ms have elapsed. If direct delivery to the master fails, the node retries once via a designated repeater node.

B. Repeater Node (Relay Layer)

Intermediate repeater nodes extend ESP-NOW coverage in hospital environments with thick walls, multiple floors, or long corridors. The master maintains a route table of up to ten node-to-repeater MAC-address mappings (first-writes). Downlink messages (UUID assignment, NVS-erase commands) are routed through the appropriate repeater when an entry exists.

C. Master Gateway Node

The master aggregates all incoming ESP-NOW traffic into a FreeRTOS queue of depth 10, decoupling the receive interrupt from MQTT publish operations. A 30 ms main-loop cadence drains the queue and publishes to `be_project/node/data`. The master also drives a 16×2 LCD showing connected node counts, repeater registrations, and MQTT status.

Scalability note: The FreeRTOS queue depth of 10 and the ten-entry route table cap the number of concurrently active nodes per master at approximately ten. This is in tension with the multi-floor hospital scenario; scaling beyond one master requires either increasing these firmware constants or deploying multiple master gateways, each managing a subset of wards. This is identified as a known prototype limitation in Section VIII.

D. Cloud Backend and Dashboard

The backend is a Django 4.x application served by Daphne (ASGI), with Django Channels providing WebSocket support, Celery for asynchronous task processing, and Redis as both the Celery broker and a per-node signal cache. An MQTT client thread dispatches incoming messages to the appropriate handlers. Sensor readings are validated synchronously, EWMA-smoothed, and batch-persisted to SQLite (development) or PostgreSQL (production). A React dashboard renders per-device fluid-level cards, alert badges, and assignment management views in real time.

IV. SENSING AND SIGNAL PROCESSING

A. On-Device IQR-Median Filtering

The HX711 raw output in a clinical environment is subject to impulsive disturbances from line sway, patient contact, and mechanical vibration. Each node maintains a circular buffer of the ten most recent 24-bit raw counts. On every update,

TABLE I
COMPARISON OF RELATED SYSTEMS VS. FLUIDCARE

headerblue					
Moovendan et al. [3] 2023	Arduino + sensors	SMS (GSM)	No	None	No cloud; IV-only
rowgray Ghosh et al. [7] 2022	ESP32 + HX711	MQTT-S	No	None	Saline-only; no relay
Paranthaman & Annalakshmi [4] 2025	ESP32 + load cell	Wi-Fi / App	No	None	Limited scalability
rowgray Patil et al. [5] 2024	ESP32	MQTT	Yes	None	Short battery; no predictive alerts
Nimi et al. [6] 2022	IoT + vital sensors	Cloud	Yes	Basic	Requires constant internet
rowgray Giaquinto et al. [9] 2020	Camera + GPU	HTTP/cloud	No	YOLO (vision)	High cost; GPU required
Bai et al. [8] 2021	IoT sensors	HTTP	No	Basic	High cost; limited scalability
rowgreen FluidCare (This Work)	ESP32 + HX711	ESP-NOW + MQTT TLS	Yes	IQR-median + EWMA	See Section VIII

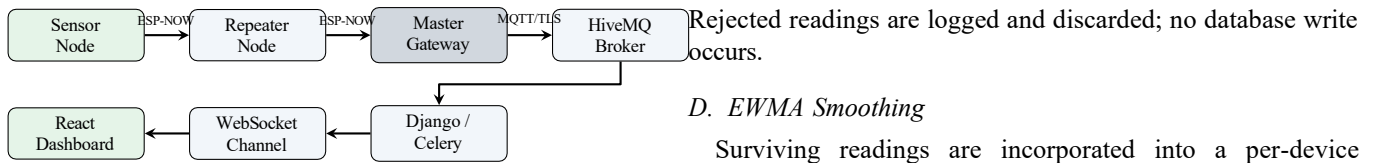


Fig. 1. End-to-end FLUIDCARE data flow across four system tiers.

the first quartile Q_1 and third quartile Q_3 of the buffer are computed, and any sample outside

$$Q_1 - 1.5 \text{ IQR}, Q_3 + 1.5 \text{ IQR}, \text{ IQR} = Q_3 - Q_1, \quad (1)$$

is discarded. The median of the surviving inliers is taken as the conditioned reading r_c . This stage executes entirely on the sensor node before wireless transmission, ensuring corrupted readings cannot trigger spurious alarms at the backend.

B. Calibration-Based Weight Conversion

The conditioned raw count r_c is converted to estimated mass $\hat{w}$ by

$$\hat{w} = m r_c + b, \quad (2)$$

with coefficients from the OLS regression described in Section V. The inverse sensitivity is approximately 843 ADC counts per gram.

C. Server-Side Monotonic-Depletion Validation

Each accepted reading w_t is compared against the previous reading w_{t-1} stored in Redis (TTL 3,000 s). Two rejection rules apply before EWMA computation:

- 1) *Monotonic-depletion enforcement*: if $\Delta = w_t - w_{t-1} > 0$, the reading is rejected (fluid mass is expected to be non-increasing during normal infusion).
- 2) *Implausible-drop rejection*: if $\Delta < -50$ g, the reading is rejected (50 g covers aggressive drip rates while blocking noise spikes).

D. EWMA Smoothing

Surviving readings are incorporated into a per-device EWMA maintained in Redis:

$$\hat{w}_t = \alpha w_t + (1 - \alpha) \hat{w}_{t-1}, \quad \alpha = 0.2. \quad (3)$$

Both the raw validated weight and the EWMA-smoothed weight are persisted to the `SensorReading` table on every accepted measurement.

V. CALIBRATION METHODOLOGY AND RESULTS

A. Experimental Setup

Calibration was performed using thirteen reference masses spanning the intended operating range of the device: 0, 11, 35, 50, 73, 94, 126, 147, 172, 202, 277, 386, and 575 g. Reference masses were commercial calibration weights; their traceability to national standards was not independently verified in this study, which is a limitation noted in Section VIII. For each reference mass, approximately 600 consecutive raw HX711 samples were recorded at the HX711 80 SPS (samples per second) output rate, corresponding to the node's nominal 80 ms polling cadence, yielding a total of $N = 7,855$ raw observations. The load cell was mounted in a fixed, horizontal orientation on a rigid aluminium bracket; the bag hook was attached to the rated loading point. A single tare/zero was applied at the outset under constant ambient conditions (laboratory room temperature; specific temperature not recorded, which limits reproducibility—see Section VIII). The rated capacity of the load cell is 1,000 g; operation above this limit was not tested.

Calibration range vs. clinical use: The maximum reference mass of 575 g does not cover a full 1 L IV bag, which weighs approximately 1,050 g when filled with saline

($\rho \approx 1.00$ g/mL). Operation above 575 g therefore constitutes extrapolation beyond the validated range. The system is intended for monitoring bags as they deplete below 575 g; initial bag loading should be confirmed against the load cell's rated capacity.

The per-mass median of the raw counts was taken as the representative value for regression, providing a robust estimator insensitive to the noise-induced outliers that occur even in controlled conditions.

B. Linear Regression, Goodness of Fit, and Uncertainty

OLS regression of the thirteen (median raw count, reference mass) pairs yields:

$$m = 1.187 \times 10^{-3} \text{ g/count}, \quad b = 683.22 \text{ g}, \quad R^2 = 0.9995. \quad (4)$$

The 95% confidence intervals derived from the OLS fit (Student's t -distribution, $\nu = 11$ d.f.) are:

$$m = (1.187 \pm 0.017) \times 10^{-3} \text{ g/count}, \quad (5)$$

$$b = 683.2 \pm 7.9 \text{ g}. \quad (6)$$

The mean absolute error is $\text{MAE} = 2.87$ g and the root-mean-square error is $\text{RMSE} = 3.47$ g. The largest single-point deviation is 8.69 g at the 11 g reference point, attributed to either a small reference-mass error or incomplete zero-offset settling at very light loads; excluding this point reduces RMSE to 2.37 g.

Important: These metrics are *in-sample*—the same 13 calibration points were used for both fitting and evaluation. A held-out or cross-validated assessment is left to future work.

A quadratic fit reduces RMSE only marginally to 3.08 g, insufficient to justify the added computation on a resource-constrained microcontroller; the linear model is retained.

C. Residual Analysis

Fig. 2 plots the signed residuals ($w_{\text{ref}} - \hat{w}$) against reference mass. A mild systematic pattern is visible: the model over-predicts slightly in the 0–35 g range (negative residuals), under-predicts from 50–172 g (positive residuals), then over-predicts again at 202–277 g and 575 g. This sign reversal is consistent with mild nonlinearity in the load-cell response at light and heavy loads—a known characteristic of single-point strain-gauge cells operating near the limits of their rated range. The maximum absolute residual (8.69 g at 11 g) is an outlier likely attributable to zero-offset error rather than model nonlinearity; all other residuals lie within ± 5 g. The pattern does not justify a quadratic model given the measurement noise floor (see Section V-D), but it should be noted as a source of systematic error at the extremes of the range.

D. Repeatability, Noise Floor, and Post-Filter Precision

Raw noise: The within-capture standard deviation of the raw HX711 counts—before IQR-median filtering—ranges from 224 to 1,164 counts (Table II), corresponding to approximately ± 0.3 – 1.4 g (1σ) in mass units. The worst case (± 1.4 g at 0 g) places the raw noise floor just above the 1 g change-detection threshold.

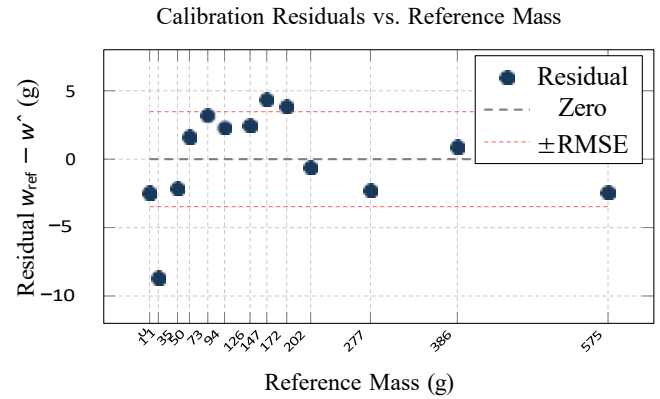


Fig. 2. Signed calibration residuals. Dashed red lines show $\pm \text{RMSE}$ (± 3.47 g). The outlier at 11 g (-8.69 g) is discussed in the text. Mild systematic nonlinearity is visible at the range boundaries.

Post-filter precision: After IQR-median filtering over a ten-sample buffer, the effective noise is reduced to the median absolute deviation (MAD) of the inlier set. For a Gaussian distribution the MAD relates to σ by $\text{MAD} \approx 0.674 \sigma$, yielding a post-filter standard deviation of approximately ± 0.2 – 0.9 g across the measurement range. This confirms that the 1 g change-detection threshold is supported by the filtered signal; the on-device filter successfully pushes the effective noise floor below the transmission threshold.

Precision vs. accuracy: The device exhibits *short-term repeatability* (precision) of ± 1 – 2 g (1σ , raw) and an *absolute calibration accuracy* of $\text{RMSE} = 3.47$ g, $\text{MAE} = 2.87$ g, across the 0–575 g range. These are distinct quantities: the repeatability characterises the spread of repeated readings at a fixed load, while the RMSE characterises the systematic + random error of the calibration model relative to the reference masses.

E. Per-Mass Calibration Summary

Table II reports the per-mass statistical summary. The coefficient of variation $\text{CV} = \sigma/|\mu| \times 100\%$ remains below 0.2% across most of the range, confirming excellent short-term repeatability. The largest relative dispersion ($\text{CV} = 0.713\%$) occurs at 575 g, where the raw count is smallest in magnitude.

VI. BACKEND, DATA MODEL, AND ALERTING

A. Multi-Tenant Data Model

FLUIDCARE employs a relational schema structured around the hospital physical hierarchy:

Floor $\rightarrow$ Ward $\rightarrow$ Bed $\rightarrow$ PatientDeviceBedAssignment.

Uniqueness constraints enforce that at most one device is active per bed, one bed is assigned per device, and one patient occupies each bed simultaneously. Ward and floor identifiers are denormalised from the Bed foreign key at assignment-save time, enabling efficient queries without repeated joins. The attribution chain for each reading is:

SensorReading $\rightarrow$ FluidBag $\rightarrow$ Device $\rightarrow$ PatientDeviceBedAssignment.

TABLE II

PER-MASS CALIBRATION SUMMARY ($N \approx 600$ PER LEVEL)

headerblue							
headerblue							
0	599	-573,481	1164	0.203	2.48	-2.48	
rowgray612		-558,985	677	0.121	19.69	-8.69*	
11							
35	626	-544,275	727	0.134	37.15	-2.15	
rowgray602		-534,825	408	0.076	48.37	+1.63	
50							
73	595	-516,775	653	0.126	69.80	+3.20	
rowgray597		-498,325	363	0.073	91.70	+2.30	
94							
126	602	-471,481	336	0.071	123.56	+2.44	
rowgray588		-455,392	224	0.049	142.66	+4.34	
147							
172	612	-433,903	774	0.178	168.17	+3.83	
rowgray608		-404,889	500	0.123	202.61	-0.61	
202							
277	605	-340,297	467	0.137	279.28	-2.28	
rowgray606		-251,139	436	0.174	385.11	+0.89	
386							
575	603	-89,117	636	0.713	577.43	-2.43	

$R^2 = 0.9995$; RMSE = 3.47 g; MAE = 2.87 g; $N = 7,855$
 *Outlier—see text. Metrics are *in-sample*.

The SensorReading model stores the raw validated weight (integer grams) and the EWMA-smoothed weight (float), auto-timestamped at server receipt. The Notification model captures alert scope (global, all-users, role-scoped, per-user), severity (low, medium, high), acknowledgement state, and a high-severity retry counter.

B. Ingestion and Batch Processing

All sensor traffic arrives via MQTT QoS 1 over TLS 1.2. The MQTT client thread validates each reading synchronously, updates the EWMA accumulator in Redis, and enqueues the validated payload. A Celery beat task drains this queue every 3 seconds using `bulk_create` with `ignore_conflicts` inside `transaction.atomic()`, protected by a 20-second distributed Redis lock. The queue accommodates up to 5,000 items per batch cycle; overflow items are re-queued. A Web-Socket `group_send` to the `sensor_monitoring` channel group is performed synchronously within the MQTT callback, decoupling dashboard latency from the slower database write path.

C. Threshold-Based Alerting

Alert generation occurs inside the Celery batch task. The preferred alert value (EWMA-smoothed if available, otherwise raw, in **grams**) is compared to `fluid_bag.threshold_low`, which is stored in **grams** in the database. If the condition is met and no warning-type notification for the same device exists within

TABLE III

DASHBOARD ALERT CLASSIFICATION (NON-OVERLAPPING BANDS)

headerblue		
Device offline or completed	<i>offline</i>	Grey
rowgray $p \leq \theta_{low}$	<i>critical</i>	Red
$\theta_{low} < p \leq 1.2 \theta_{low}$	<i>warning</i>	Yellow
rowgray $p \geq \theta_{high}$	<i>overflow</i>	Purple
$1.2 \theta_{low} < p < \theta_{high}$	<i>normal</i>	Green

the preceding 30 minutes, a Notification record is written and broadcast via the WebSocket channel layer.

A separate hourly Celery task marks devices as *offline* if their `last_seen` timestamp is older than 45 seconds and emits a high-severity notification. High-severity unread notifications are eligible for up to two re-delivery attempts at **5-minute** (300 s) intervals.

D. Dashboard Alert Classification

The React dashboard converts the EWMA-smoothed weight (in grams) to a percentage of nominal capacity client-side:

$$p = \text{clamp} \left(\frac{\hat{w}_{\text{smooth}}}{C_g} \times 100, 0, 100 \right), \quad (7)$$

where C_g is the bag nominal capacity converted to grams. **Density assumption:** this conversion assumes a fluid density of 1 g/mL (valid for normal saline and most aqueous IV solutions). Blood ($\rho \approx 1.06$ g/mL) and high-dextrose solutions ($\rho \approx 1.04$ g/mL) deviate from this assumption; a per-fluid density factor should be applied for accurate percentage reporting in those cases, even though FLUIDCARE claims multi-fluid support. The threshold θ_{low} used for server-side alert generation remains in grams (Section VI-C), so server alerts are unaffected by the density assumption.

The dashboard classification uses strictly non-overlapping bands (Table III):

VII. IMPLEMENTATION

The proposed FluidCare system was implemented using a multi-tier IoT architecture consisting of bedside sensor nodes, repeater nodes, a master gateway node, cloud communication infrastructure, and a centralized monitoring dashboard. The implementation focuses on providing accurate IV fluid level monitoring, reliable wireless communication, and real-time alert generation for healthcare professionals.

A. Sensor Node Hardware

The sensor node is the primary sensing unit of the Fluid-Care system. It is installed near the patient's IV stand and continuously measures the weight of the fluid bottle. Each sensor node consists of an ESP32 microcontroller, HX711 load-cell amplifier, single-point load cell sensor, and battery power supply.

The load cell converts the weight of the IV fluid bottle into electrical signals, which are amplified and digitized by the HX711 module. The ESP32 processes these readings and

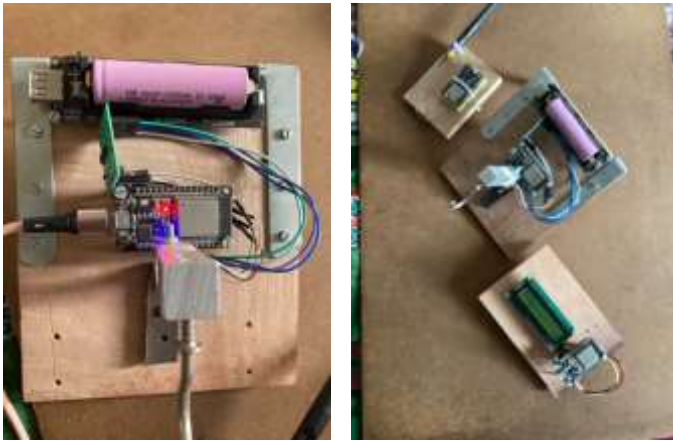


Fig. 3. Left: Sensor node consisting of ESP32, HX711 amplifier module, load cell sensor, and battery unit. Right: Complete FluidCare prototype demonstrating IV fluid monitoring using load-cell technology.

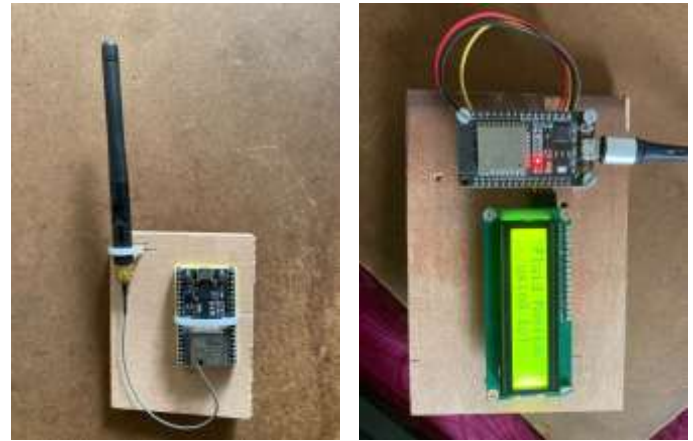


Fig. 4. Left: ESP32-based repeater node equipped with an external antenna for communication range extension. Right: Master gateway node with LCD display used for centralized monitoring and MQTT communication.

determines the remaining fluid quantity. Based on predefined threshold values, the node categorizes the fluid level into Normal, Warning, and Critical states before transmitting the information through the ESP-NOW communication network.

B. Wiring Schematic

The wiring configuration establishes communication between the ESP32 microcontroller, HX711 amplifier module, and load cell sensor. The HX711 is powered by the ESP32 and converts the low-level analog output of the load cell into digital readings suitable for processing. Proper wiring ensures accurate weight measurement and stable operation of the monitoring system.

C. Master Gateway and Repeater Hardware

The master gateway acts as the central communication hub of the FluidCare system. It receives sensor readings from bedside nodes either directly or through repeater nodes using the ESP-NOW protocol. The gateway processes incoming data and forwards it to the MQTT broker through Wi-Fi connectivity.

To improve communication coverage in larger hospital environments, a repeater node equipped with an external antenna is deployed. The repeater receives data packets from sensor nodes and forwards them to the master gateway. This architecture improves signal strength, reduces packet loss, and supports deployment across multiple wards and hospital floors.

A 16×2 LCD display is integrated with the gateway to provide local visualization of system status, connected nodes, and communication activity.

D. Full Deployment Layout

The complete deployment consists of multiple sensor nodes installed on patient IV stands, repeater nodes positioned at strategic locations, and a master gateway located near the nursing station. Sensor nodes continuously monitor IV fluid levels and transmit measurements through the wireless communication network. The master gateway aggregates data from

all connected nodes and forwards it to the cloud infrastructure for storage and visualization.

E. Software Stack and Backend Services

The backend infrastructure of FluidCare is developed using Django and PostgreSQL. MQTT is used for communication between IoT devices and the cloud platform. Redis is used for temporary data caching and asynchronous task execution. Real-time updates are delivered through WebSocket communication, enabling instant visualization of sensor data on the monitoring dashboard.

The backend validates incoming measurements, stores patient records, maintains alert history, and manages device assignments. This architecture ensures reliable data processing and supports real-time healthcare monitoring.

F. Dashboard Implementation

A web-based dashboard was developed to provide healthcare professionals with a centralized monitoring interface. The dashboard enables patient registration, ward management, device assignment, fluid level monitoring, and alert management through a single platform.

Real-time sensor readings are displayed through dynamic dashboards and monitoring cards. When fluid levels reach predefined warning or critical thresholds, notifications are generated immediately and presented to healthcare staff. The dashboard also maintains historical records for future analysis and auditing purposes.

G. Prototype Validation

The complete FluidCare prototype was tested under simulated hospital conditions using IV fluid bottles of different capacities. The system successfully measured fluid levels, transmitted data through the ESP-NOW network, forwarded information to the MQTT broker, and displayed real-time updates on the monitoring dashboard.

Experimental testing confirmed that the system can detect low fluid conditions and generate alerts before complete

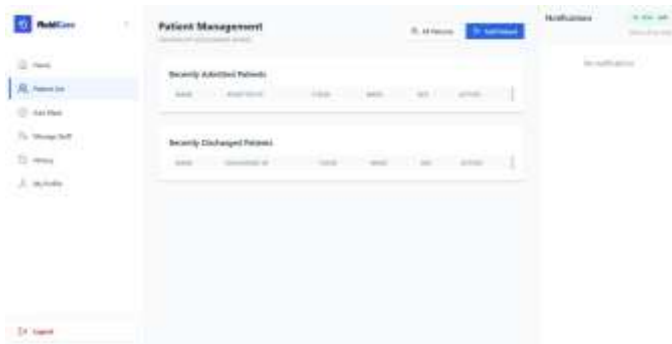


Fig. 5. FluidCare dashboard displaying patient management, ward assignment, notification handling, and real-time IV fluid monitoring features.

depletion of the IV bottle. The prototype demonstrates the feasibility of an IoT-based fluid monitoring solution capable of reducing manual workload and improving patient safety through continuous automated monitoring.

VIII. PROTOTYPE LIMITATIONS

The following are *known defects* of the current prototype. They do not affect the validity of the reported measurements, but must be resolved before clinical deployment.

A. Calibration Limitations at Light Load and Extrapolation above 575 g

The calibration was conducted under static, controlled conditions with a single tare, unverified reference-mass traceability, and unrecorded ambient temperature. In clinical use, thermal drift, creep under sustained load, zero-offset drift between power cycles, and mechanical vibration will introduce additional error. The 8.69 g residual at 11 g indicates that accuracy near the low end of the range may be insufficient for near-empty small-volume bags. Furthermore, a full 1 L saline bag ($\approx 1,050$ g) exceeds the 575 g calibrated range; readings above this level are extrapolations. Periodic re-tare, temperature-compensated recalibration, and independently traceable reference masses are recommended before deployment.

B. Blood and Urine Bag Alerts Unimplemented

The `process_alert` Celery task implements threshold checks for blood and urine bags but is never invoked—no `.delay()` call exists in the codebase. Additionally, `send_alert_notification` is a stub function (`body: pass`), so SMS and email delivery channels are absent. These omissions limit the prototype to IV fluid types at present.

C. No Refill-Event Mechanism

The monotonic-depletion rule rejects any weight increase unconditionally. In practice, replacing an empty bag with a full one produces a large step increase that would be rejected. The device remains at its last recorded low weight until the Redis EWMA state expires (~ 50 minutes), risking continued low-fluid alerts after a successful refill. A refill-detection mechanism—detecting a weight increase exceeding a

configurable threshold following a low-alert event—is required to reset state correctly.

D. Per-Master Node Scalability Ceiling

The FreeRTOS queue depth of 10 and the ten-entry route table impose a hard cap of approximately ten simultaneously active sensor nodes per master gateway. Deployments exceeding this limit require either increasing these firmware constants (rebuilding and reflashing the master) or deploying multiple master gateways. The current firmware provides no automatic load balancing or gateway handover mechanism.

IX. CONCLUSION AND FUTURE WORK

This paper presented FLUIDCARE, a working prototype of a multi-tier IoT system for real-time hospital fluid-bag monitoring, with a production-oriented architecture; components remaining for deployment are enumerated in Section VIII. The four-tier architecture connects sensor nodes, a repeater mesh, and a master gateway through ESP-NOW, and bridges to the cloud via MQTT over TLS. A two-tier signal pipeline—on-device IQR-median filtering followed by server-side monotonic-depletion validation and EWMA smoothing—provides robust noise rejection for clinical environments. Calibration over 7,855 observations across 13 reference masses confirms a linear sensor model ($R^2 = 0.9995$, $RMSE = 3.47$ g, $MAE = 2.87$ g, both *in-sample*) with short-term repeatability of ± 1 – 2 g (1σ) and absolute calibration accuracy of $RMSE = 3.47$ g. The 1 g change-detection threshold implemented in firmware is supported by the post-filter noise floor. Four prototype defects have been identified and described in Section VIII: light-load and above-range calibration limitations, unimplemented blood/urine alert dispatch, the absence of a refill-detection mechanism, and the per-master scalability ceiling.

Future work will address three directions: (1) Resolving the identified prototype defects, with priority on the refill detection logic and blood/urine alert dispatch; (2) Extending the calibration to cover the full 0–1,050 g clinical range with temperature compensation, cross-validated error estimates, and independently traceable reference masses; (3) Developing machine-learning models for predictive bag-depletion forecasting from the historical `SensorReading` time series accumulated by the backend.

ACKNOWLEDGMENT

The authors thank the staff at Zeal College of Engineering and Research, Pune, for support in hardware procurement, laboratory access, and review of the clinical monitoring requirements that motivated this work.

REFERENCES

- [1] M. McLellan, R. Hayes, and F. Carter, "Clinical Utility of the Fluid Intravenous Alert Monitor," *Journal of Clinical Anesthesia*, Elsevier, 2016.
- [2] J. Park, S. B. You, and H. Kim, "Experience of Nurses with Intravenous Fluid Monitoring for Patient Safety: A Qualitative Descriptive Study," *PLOS ONE*, 2022.

- [3] M. Moovendan, R. Jeyaprakash, and K. Jayanthi, "Design of Automated Control and Monitoring System for Intravenous Infusion Based on IoT," in *Proc. IEEE ICCPCT*, 2023.
- [4] A. Paranthaman and K. Annalakshmi, "IV Fluid Bag Monitoring, Alert, and Control System Using ESP32 and IoT," *Int. J. Scientific Advances in Technology (IJSAT)*, 2025.
- [5] H. Patil, S. Deshmukh, and P. Kulkarni, "IVigilance – IoT Based Intravenous Drip Monitoring and Controlling System," *Int. J. Creative Research Thoughts (IJCRT)*, 2024.
- [6] W. S. Nimi, M. Joseph, and T. Kurian, "Reliable and Smart E-Healthcare System for IV Fluid and Vital Parameter Monitoring," *IEEE J. Translational Engineering in Health and Medicine*, 2022.
- [7] D. Ghosh, S. Dutta, and R. Nath, "Smart Saline Level Monitoring System Using ESP-32 and MQTT-S," in *Proc. IEEE Healthcom*, 2022.
- [8] X. Bai, J. Liu, and Y. Zhang, "Application of Infusion Control System Based on Internet of Things Technology," *J. Healthcare Engineering*, 2021.
- [9] N. Giaquinto, L. Romano, and M. Rossi, "Real-Time Drip Infusion Monitoring Through a Computer Vision System," *IEEE Trans. Biomedical Engineering*, 2020.