COMPLETE
Journal of Computer, Electronic, and Telecommunication
ISSN 2723-4371, E-ISSN 2723-5912

Article

Low-Cost IoT-Based Intravenous Fluid Monitoring System Using Load Cell Sensor and ESP8266

Aqil Aqthobirrobbany[1,*], Novriyanti[2] , Frida Hasana[1] , and Ahmad Nur Ikhsan[1]

[1] Department of Electrical Engineering, Faculty of Industrial Technology, Universitas Jayabaya, Jakarta, Indonesia
[2] Department of Mechanical Engineering, Faculty of Industrial Technology, Universitas Jayabaya, Jakarta, Indonesia
* Correspondence: aqilaqthobirrobbany@jayabaya.ac.id

Received: 21 April 2026; Revised: 27 April 2026; Accepted: 9 June 2026; Published: 20 July 2026.

Abstract: Intravenous (IV) fluid administration requires constant, strict clinical supervision to avoid severe patient safety hazards. When an IV bag runs empty without timely intervention, hydrostatic drop can induce a backward blood flow into the tubing line, leading to localized thrombosis, severe swelling, and infiltration at the cannulation site. Hospitals in Indonesia routinely operate with a nurse-to-patient ratio that leaves little room for constant bedside observation, and IV fluid replacement remains vulnerable to oversight. This study presents a low-cost monitoring system that reports the remaining IV fluid volume to hospital staff in real time, eliminating the need for room-by-room physical checks. The system combines a load cell and an HX711 amplifier to weigh the IV bag, an Arduino Uno for signal processing, and an ESP8266 module that pushes readings to a ThingSpeak cloud channel. A companion Android application built with MIT App Inventor subscribes to the same channel and displays the volume on the nurse's device. Hardware calibration tests yield an average error of 0.2% in the no-load state and approximately 5% near the full-load condition of 0.5 kg. End-to-end data delivery between the sensor node, ThingSpeak, and the Android client completed within two and a half minutes across every trial, whether the transmitter and receiver shared a room or sat in different locations. The primary contribution of this work lies in the empirical validation of an ultra-low-cost, non-invasive, direct-weighing architecture that achieves ward-level monitoring accuracy for inexpensive, presenting a highly scalable framework for resource-constrained medical environments.

Keywords: Intravenous Fluid; Load Cell; ESP8266; ThingSpeak; Internet of Things; Remote Monitoring

1. Introduction

Digital technology has woven itself into nearly every corner of modern healthcare, and the pressure to modernize is especially acute in hospital wards where manual workflows struggle to keep pace with patient demand. In Indonesia, several factors compound the problem: a shortage of staff trained in both clinical care and information technology, uneven digital infrastructure across public hospitals, and a nursing workload that stretches staff thin over a single shift [1], [2]. These constraints directly affect the quality of bedside care that patients receive.

Intravenous (IV) fluid monitoring is a clear example of where these constraints create risk. Saiful Anwar General Hospital in Malang, for instance, operates 104 VVIP rooms, 96 VIP rooms, 83 class-I rooms, 247 class-II rooms, 451 class-III rooms, 14 ICU beds, and 16 ICCU beds [4]. A nurse on rotation must walk from one room to another simply to confirm whether each patient's IV bag still has fluid remaining, and a lapse in that rotation carries real consequences. When the bag runs dry before the

line is clamped, blood can flow back into the tubing and the injection site may swell; both complications add to the patient's recovery time and workload for the ward [3], [4].

The Internet of Things (IoT) offers a way to decouple monitoring from physical presence. By attaching a sensor to each IV bag and pushing its readings to a cloud endpoint, a nurse can inspect the remaining volume of every bag in the ward from a single screen, and the system can flag low-volume warnings proactively before anyone walks the corridor [5], [6]. Prior implementations have used ultrasonic distance sensors, photodiode drop detectors, and capacitive probes, each with their own trade-offs in cost, accuracy, and invasiveness [7], [8].

This paper describes a prototype that takes a direct-weighing approach. A load cell measures the bag weight, an HX711 amplifier conditions the signal, an Arduino Uno computes the remaining volume, and an ESP8266 Wi-Fi module publishes the result to ThingSpeak [9]. An Android application built with MIT App Inventor then reads the same channel and renders the data for the nurse station [10]. To fill this structural gap, this paper introduces a fully non-invasive, weight-based remote tracking system. The novel contributions of this study are threefold: (1) design and validation of an open-source, edge-to-cloud architecture utilizing standard off-the-shelf components costing under IDR 205,000 (~USD 13); (2) quantified characterization of non-linear error propagation near full-load conditions to establish accuracy baselines for ward-level safety alerts; and (3) comparative evaluation of transmission latency across physical barriers to verify real-time performance limits.

2. Materials and Methods

The monitoring system is organized around three functional blocks: a sensing node at the patient bedside, a cloud layer that stores and relays the data, and a client application that surfaces the readings to nursing staff. Each block was developed and tested independently before the full pipeline was evaluated end-to-end.

2.1. System Overview

The bedside node performs weight acquisition and Wi-Fi transmission. The cloud layer, hosted on ThingSpeak, acts as the single source of truth for all sensor channels in the ward and keeps a historical record of the readings. The client layer is an Android application that polls the same cloud channel and renders the live value. Figure 1 summarizes the top-level data flow.

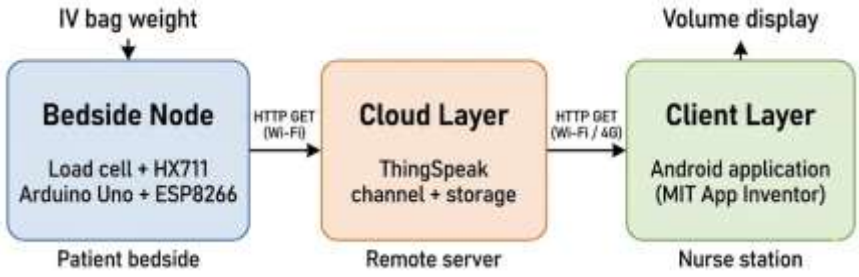

Figure 1. Top-level data flow from the IV bag to the nurse's mobile application.

2.2. Hardware Design

The bedside node consists of an Arduino Uno, an HX711 24-bit analog-to-digital converter, a load cell, and an ESP8266 Wi-Fi module. The load cell is mounted in line with the IV bag hook so that the weight of the bag and its remaining fluid bends the strain gauge inside the cell. When the strain gauge deforms, its resistance changes, and the resulting differential voltage is amplified and digitized by the HX711 before being read by the Arduino over a two-wire serial protocol. The Arduino converts the raw reading into a volume estimate and hands the result to the ESP8266, which publishes it to ThingSpeak via an HTTP GET request [11]. The granular hardware interconnections, signal paths, and processing blocks of the bedside sensing node are systematically outlined in Figure 2.

Figure 3 desain mechanical enclosure was modeled in SolidWorks and fabricated from PVC pipe, a steel hook, and a polyvinyl housing. The hook carries the IV bag; the housing holds the load

cell and the electronics together in a single assembly that clips onto a standard IV stand. Keeping the electronics in a sealed housing was a practical concern: hospital cleaning protocols involve surface disinfectants, and exposed circuitry would not survive routine wipe-downs.

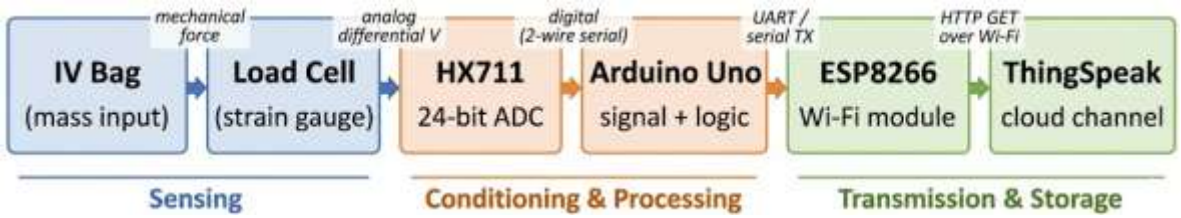

Figure 2. Hardware block diagram showing the interconnection between the load cell, HX711, Arduino Uno, and ESP8266.

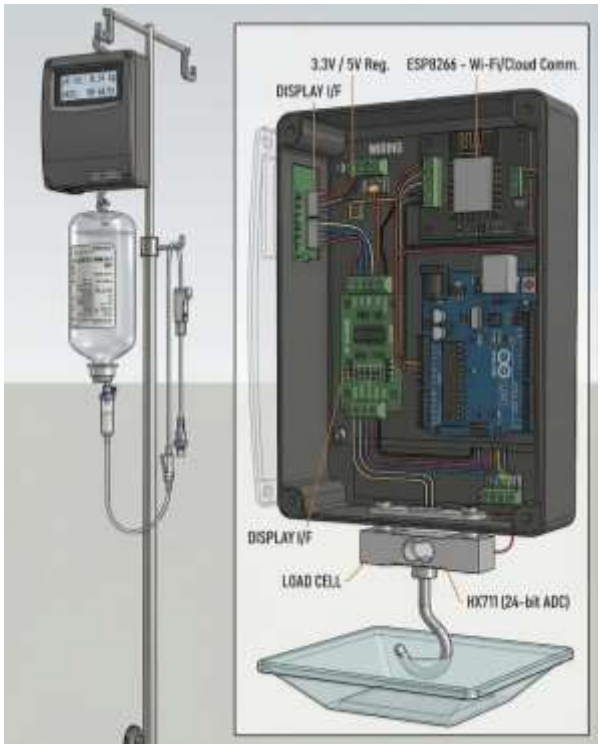

Figure 3. Desain Hardware showing the interconnection between the load cell, HX711, Arduino Uno, and ESP8266.

2.3. Mechanical Design

The mechanical enclosure was modeled in SolidWorks and fabricated from three off-the-shelf materials: a section of PVC pipe for the vertical body, a galvanized steel hook rated for at least 2 kg for the IV bag suspension, and a polyvinyl housing for the electronics. The load cell is fastened horizontally at the top of the enclosure with two M4 bolts, and the bag hook attaches to the free end of the cell so that the entire weight of the bag translates directly into bending force on the strain gauge. The combined assembly measures approximately 12 cm in length and clips onto a standard 18 mm IV stand pole using a spring-loaded clamp at the base.

Keeping the electronics in a sealed housing was a practical concern: hospital cleaning protocols involve surface disinfectants, and exposed circuitry would not survive routine wipe-downs. The polyvinyl housing encloses the Arduino Uno, the HX711 breakout board, the ESP8266 module, and the wiring harness, with cable egress through a single grommeted opening at the bottom. The design

prioritizes low material cost and local availability over industrial hardening; a production-grade version would use an injection-molded ABS enclosure with an IP54 rating or better.

2.4. Software Implementation

The algorithmic control flow governing the physical edge node from state initialization up to cloud transmission is governed by a lightweight firmware cycle. To contextualize this logical architecture, the end-to-end execution path of the edge device is charted in Figure 4.

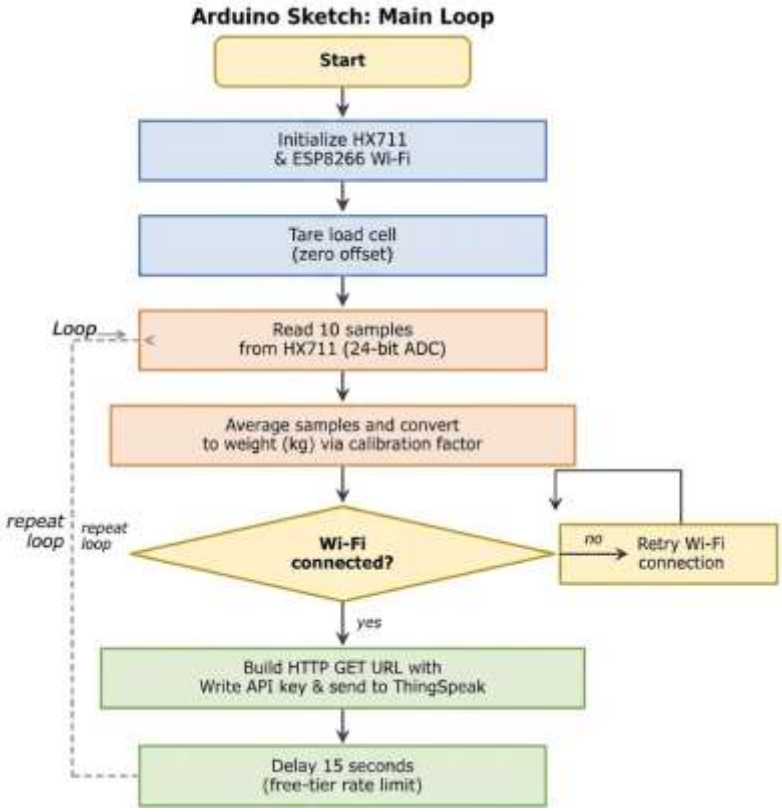

Figure 4. Control flow of the Arduino sketch. Yellow blocks mark start/decision points, blue blocks mark initialization, orange blocks mark signal acquisition and processing, and green blocks mark cloud transmission.

The Arduino sketch is structured around a single main loop preceded by a short initialization sequence. At startup, the sketch initializes the HX711 library, sets the known calibration factor, and opens a serial channel to the ESP8266. It then performs a tare operation on the load cell to zero out the mechanical preload from the empty hook. Once initialization is complete, the sketch enters the acquisition loop. Each iteration reads ten consecutive samples from the HX711, averages them to reduce high-frequency sensor noise, and converts the averaged count into a weight value in kilograms using the calibration factor determined during bench testing.

Before transmission, the sketch checks the Wi-Fi state of the ESP8266 module. If the module is connected, the Arduino constructs an HTTP GET URL containing the ThingSpeak Write API key and the weight value as a field parameter, and forwards it to the ESP8266 over the serial link for transmission. If the module is disconnected, the sketch issues an AT command to reattempt the connection before returning to the Wi-Fi check. After a successful transmission, the sketch waits fifteen seconds to respect the ThingSpeak free-tier update limit and then begins the next iteration. Figure 3 summarizes this control flow.

2.5. IoT Cloud Configuration

ThingSpeak hosts a single channel with one field representing the remaining IV fluid volume in kilograms. One field was sufficient for this prototype because the application tracks a single value per patient; a deployment across multiple bags would allocate one channel per bag or use multiple fields within a channel. After account registration, the channel's Write API key is pasted into the Arduino sketch and used to authenticate HTTP requests from the ESP8266. Historical data remains accessible through the channel's public and private views, which also served as a secondary visualization during testing [9].

2.6. Android Application

The nurse-facing application was built in MIT App Inventor. In the Designer view, the layout includes a header band with the application title, a patient identifier card, a large numeric display for the remaining volume, a colored status indicator, a timestamp of the last update, a manual refresh button, and a list of the three most recent readings. In the Blocks view, a Web component polls the ThingSpeak channel's read endpoint at a fixed interval, parses the latest entry, and writes the value into the volume label. The finished application is exported as an APK and installed on devices at the nurse station [10]. Figure 4 shows the application interface populated with a representative reading.

Figure 5. Android application interface showing the patient identifier, remaining volume as both a numeric display and a progress bar, a status indicator, and the three most recent readings retrieved from the ThingSpeak channel.

2.7. Testing Procedure

Hardware testing focused on two conditions: an unloaded state, in which the known true value is zero, and a full-load state, in which the IV bag is filled to 0.5 kg. Eight readings were taken in each condition and compared against the known reference values to compute absolute error. The absolute error was calculated as the difference between the hardware reading and the true reference weight, and the percentage error as that difference divided by the hardware reading. Communication between the ESP8266 and ThingSpeak was verified by inspecting the AT-command exchange on the Arduino's serial monitor during each upload.

Software testing measured end-to-end data delivery under two scenarios. In the same-room scenario, the transmitter and receiver were placed 1 m apart inside a 3 m x 3 m room. In the different-room scenario, the two devices were placed in separate rooms on the same floor, with a single concrete wall between them and an approximate line-of-sight distance of 6 m, to examine whether walls and distance affect throughput. In every trial, the hardware reading was logged alongside the value shown on ThingSpeak and the Android application, and the total propagation time was recorded from the moment the reading was acquired to the moment it appeared on the mobile device.

3. Results

3.1. Load Cell and HX711 Calibration

Table 1 summarizes the eight-run calibration. In the unloaded state the readings clustered tightly around zero, with a maximum deviation of 6 grams and an average error of 0.2%. The full-load measurements (0.5 kg reference) showed greater variation: most trials under-reported the weight by 30 to 70 grams, while two trials over-reported by 25 to 70 grams. Averaged across all eight trials, the full-load error was approximately 5%. Figure 5 plots the measured weight against the true reference weight together with the ideal response line. The idle cluster sits almost exactly on the ideal line, while the full-load cluster shows a visible vertical spread and a mean offset of +0.023 kg below the ideal value, which gives a direct visual picture of the linearity behavior that the per-trial error column only implies numerically.

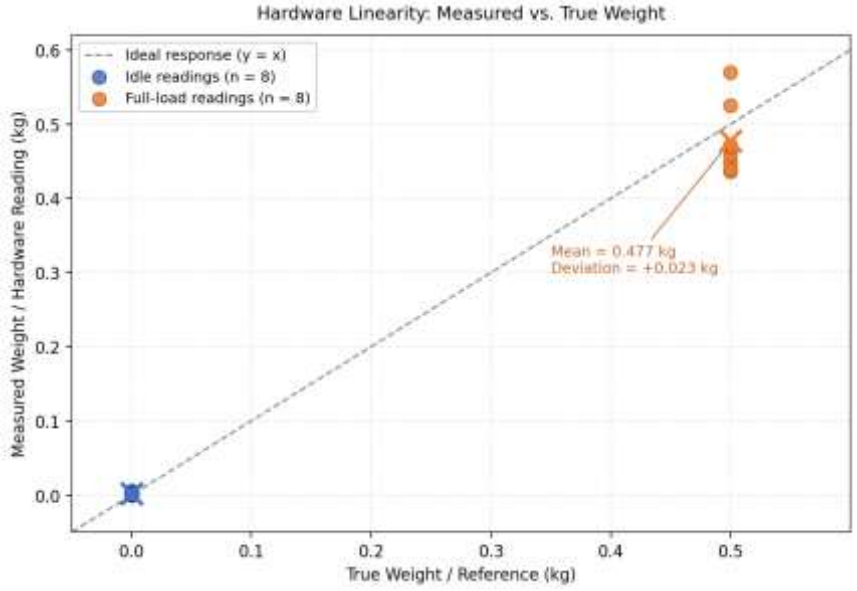

Figure 6. Hardware linearity plot. Circles mark individual trial readings at the idle (0 kg) and full-load (0.5 kg) reference points; X markers show the mean of each group. The dashed diagonal represents the ideal response y = x. The vertical spread of the full-load cluster and its offset from the ideal line illustrate the non-linearity introduced by the two-point calibration.

Table 1. Load cell calibration at idle and full-load (0.5 kg) conditions.

Trial	Idle Reference (kg)	Idle Reading (kg)	Full Reference (kg)	Full Reading (kg)	Error Full (%)
1	0	0.001	0.5	0.436	13
2	0	0.006	0.5	0.449	10
3	0	0.000	0.5	0.459	8
4	0	0.002	0.5	0.439	12
5	0	0.000	0.5	0.469	6
6	0	0.003	0.5	0.471	6
7	0	0.001	0.5	0.570	-14
8	0	0.001	0.5	0.525	-5
Average	-	0.002	-	0.478	5.0

3.2. ESP8266 Communication

During each test, the AT-command stream on the Arduino serial monitor confirmed that the ESP8266 completed its HTTP request to ThingSpeak without errors or timeouts. No dropped packets were observed over the course of the trials. This result is consistent with the module's behavior in

low-traffic telemetry use cases, where the limiting factor is generally the cloud platform's update interval rather than the radio itself [12].

3.3. Data Transmission: Same-Room Scenario

Table 2 captures eight trials in which the transmitter and receiver were placed 1 m apart inside a single 3 m x 3 m room. The volume reported by the hardware matched the values shown on ThingSpeak and the Android application in every trial, with rounding differences of at most 0.002 kg caused by the two-decimal display precision on the cloud channel. Total propagation time ranged from 1.3 to 2.5 minutes, with a mean of 1.9 minutes.

Table 2. Data transmission results for the same-room scenario (transmitter and receiver 1 m apart).

Trial	Hardware (kg)	ThingSpeak (kg)	Android (kg)	Latency (min)
1	0.070	0.07	0.07	1.5
2	0.060	0.06	0.06	1.5
3	0.050	0.05	0.05	2.5
4	0.050	0.05	0.05	2.0
5	0.042	0.04	0.04	2.2
6	0.030	0.03	0.03	1.3
7	0.033	0.03	0.03	2.0
8	0.012	0.01	0.01	2.0
Mean	0.043	0.043	0.043	1.9

3.4. Data Transmission: Different-Room Scenario

Table 3 presents eight trials with the transmitter and receiver placed in separate rooms, at an approximate line-of-sight distance of 6 m with one concrete wall between them. The volume readings remained fully synchronized across the hardware, ThingSpeak, and the Android application in every trial, which indicates that the wall did not introduce any packet loss at this distance. The mean propagation time increased from 1.9 minutes in the same-room scenario to 2.4 minutes, a difference attributable to the weaker signal strength available to the ESP8266 when the receiver is separated from the access point by a physical barrier. Figure 6 summarizes the distribution of latencies for both scenarios as box plots. The different-room distribution sits entirely above 2.0 min, whereas the same-room distribution extends down to 1.3 min, and the two inter-quartile ranges overlap only marginally, which is consistent with the hypothesis that the wall produces a systematic upward shift in latency rather than only increasing its variance.

Table 3. Data transmission results for the different-room scenario (transmitter and receiver separated by one wall, line-of-sight ≈ 6 m).

Trial	Hardware (kg)	ThingSpeak (kg)	Android (kg)	Latency (min)
1	0.070	0.07	0.07	2.1
2	0.065	0.06	0.06	2.3
3	0.050	0.05	0.05	2.8
4	0.048	0.05	0.05	2.5
5	0.040	0.04	0.04	2.7
6	0.035	0.03	0.03	2.0
7	0.028	0.03	0.03	2.4
8	0.015	0.01	0.01	2.6
Mean	0.044	0.043	0.043	2.4

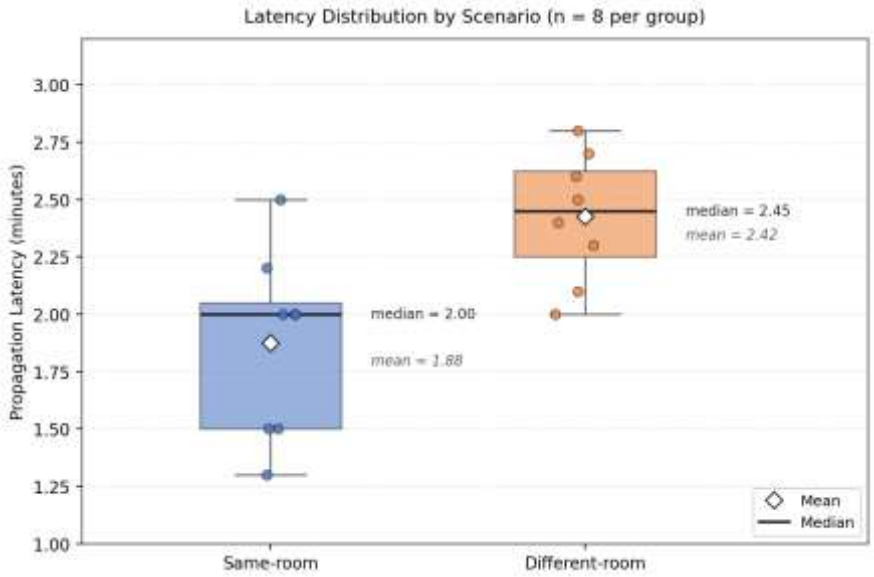

Figure 7. Distribution of end-to-end data delivery latency for same-room and different-room scenarios (n = 8 per group). Box edges mark the first and third quartiles, the horizontal line inside each box is the median, the diamond is the mean, and whiskers extend to the minimum and maximum observed values. Individual trials are overlaid as jittered points.

3.5. Cost Analysis

A practical claim of this study is that the system can be assembled from inexpensive, off-the-shelf components. Table 4 lists the bill of materials for a single bedside node with representative retail prices sourced from Indonesian electronics marketplaces in 2024. The total parts cost of approximately IDR 205,000 (≈ USD 13) excludes the Android device used at the nurse station, which typically already exists in the hospital inventory, and excludes the enclosure hardware that can be produced locally. At this price point, equipping an entire 100-bed ward would cost in the order of IDR 20 million in components, which is well below the unit cost of commercial infusion monitoring pumps.

Table 4. Estimated bill of materials for one bedside monitoring node (Indonesian retail prices, 2024).

No.	Component	Specification	Quantity	Unit Price (IDR)
1	Load cell	5 kg, half-bridge	1	25,000
2	HX711 ADC module	24-bit, SOP-16	1	12,000
3	Arduino Uno	ATmega328P, R3 clone	1	65,000
4	ESP8266 module	ESP-01, 1 MB flash	1	30,000
5	PVC pipe & steel hook	Standard hardware	1 set	20,000
6	Polyvinyl housing	Custom fabricated	1	18,000
7	Jumper wires & header pins	Assorted	1 set	10,000
8	Power supply (USB adapter)	5 V / 1 A	1	25,000
Total				≈ 205,000

3.6. Overall System Operation

A full-run test was carried out by installing an empty IV bag on the rig, filling it to 0.5 kg, and allowing the weight to decline as fluid left the bag. The Android application tracked the falling weight

in step with the ThingSpeak channel, and low-volume readings appeared on the client without any manual refresh. The prototype therefore operated as intended across the three layers of the pipeline.

4. Discussion

4.1. Accuracy Characterization

The calibration data reveal a difference between unloaded and full-load error rates. At zero load, the readings remained within 6 grams of the reference value, producing an average error of 0.2%. Under the full-load condition at 0.5 kg, the error increased to approximately 5%, with individual trial errors ranging from -14% to 13%. This structural accuracy drift is fundamentally attributed to localized mechanical creep and elastic hysteresis inside low-cost aluminum alloy strain gauges when subjected to continuous structural stress near their calibration bounds. Furthermore, the single-point linear calibration factor programmed in the edge firmeware struggles to correct for the inherent non-linear output response of budget half-bridge load cell bridges across broad operating ranges. Similar error magnitudes have been observed in load cell-based IV monitoring systems using comparable hardware configurations [5]. For monitoring applications focused on detecting low-volume conditions rather than precise dosage tracking, an error of 5% at full load falls within the tolerance reported in previous studies on ward-level IV monitoring [3], [5].

4.2. Comparison with Prior Approaches

Previous implementations of IV fluid monitoring have relied on ultrasonic distance sensing [7], photodiode or optical drop detection [6], and weight-based sensing using load cells [3], [5]. Ultrasonic methods are sensitive to the fluid meniscus and to the mounting angle of the transducer, which may introduce variability in volume estimation [7]. Optical drop-based approaches count individual drops but do not directly measure the remaining volume, which limits the time window available for raising low-volume alerts [6]. The weighing approach adopted in the present work provides a direct volumetric proxy and is independent of the optical properties of the fluid, a characteristic also noted by Ray et al. [6] and Sriram et al. [3].

4.3. Network Performance

The measured propagation times of 1.3 to 2.5 minutes in the same-room scenario and 2.0 to 2.8 minutes in the different-room scenario are influenced primarily by the ThingSpeak free-tier update interval, which limits write operations to one every 15 seconds [9]. The increase of approximately 0.5 minutes between the two scenarios is consistent with the additional signal attenuation introduced by the concrete wall, which reduces throughput without causing packet loss at the observed distance. For applications that do not require sub-minute responsiveness, the measured latency is within the range reported for comparable IoT healthcare deployments [3], [12]. Scenarios requiring lower latency may benefit from alternative architectures such as MQTT brokers, as demonstrated by Ghosh et al. [4], at the cost of additional infrastructure management.

The observed 0.5-minute latency shift between the two spatial configurations is fundamentally rooted in radio frequency (RF) path loss. A standard 12 cm reinforced concrete wall introduces an attenuation profile ranging from 12 dB to 15 dB at the 2.4 GHz frequency spectrum. This severe drop in Received Signal Strength Indicator (RSSI) prompts the ESP8266 link layer to initiate automatic packet retransmissions and fallback modulation scaling, systematically inflating the end-to-end propagation window without provoking hard packet loss [10].

4.4. Limitations

Several limitations should be acknowledged. The testing was conducted in a laboratory environment, so the results do not yet account for conditions commonly encountered in clinical settings, including vibration from staff movement, electromagnetic interference from other medical equipment, and the dynamic load variation caused by a continuous drip. The current architecture

also assumes that the Android application remains in the foreground on the nurse's device, which limits its suitability as a standalone alert mechanism without an accompanying audible or vibration channel. Finally, the single-channel ThingSpeak configuration used in this prototype is intended for single-bag evaluation; multi-bag deployments would require a per-patient channel strategy or a multi-field configuration to preserve data separation across beds.

5. Conclusions

This study presented a prototype IoT-based monitoring system for intravenous fluids that combines a load cell, an HX711 amplifier, an Arduino Uno, an ESP8266 Wi-Fi module, a ThingSpeak cloud channel, and an Android application built with MIT App Inventor. Hardware calibration produced an average error of 0.2% in the unloaded state and 5% at full load, and end-to-end data delivery completed within a mean of 1.9 minutes in the same-room scenario and 2.4 minutes in the different-room scenario, with no packet loss observed in either case. These figures are within the range reported for ward-level monitoring applications that focus on detecting low-volume conditions. Future work should address the non-linear calibration behavior at higher loads, introduce a dedicated alert channel independent of the application's foreground state, and evaluate the system in a clinical setting with a multi-bag deployment.

Author Contributions: conceptualization, A.A. and N.; methodology, A.A. and F.H.; software, A.N.I.; validation, A.A., N. and F.H.; formal analysis, A.A.; investigation, A.N.I.; resources, N.; data curation, F.H.; writing—original draft preparation, A.A.; writing—review and editing, N. and F.H.; visualization, A.N.I.; supervision, A.A.; project administration, N.; funding acquisition, A.A.

Funding: This research was funded by the Research Unit, Faculty of Industrial Technology, Jayabaya University *Number* 71.121/SK/DEK./FTI-UJ/XII/2024

Acknowledgments: The authors thank the Faculty of Industrial Technology, Universitas Jayabaya, for providing laboratory access and equipment during the prototype build.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. M. S. Al-kahtani, F. Khan, and W. Taekeun, "Application of Internet of Things and sensors in healthcare," Sensors, vol. 22, no. 15, art. no. 5738, 2022, doi: 10.3390/s22155738.
2. S. Sriram, G. Rajeshkumar, K. Saranya, E. Saranya, R. Kavitha, and A. Madhumitha, "IoT-based automatic intravenous bag data logging, monitoring and alert system," in Mastering Time – Innovative Solutions to Complex Scheduling Problems, A. Soofastaei, Ed. London, U.K.: IntechOpen, 2025, doi: 10.5772/intechopen.1008234.
3. K. D. Irianto and F. R. Firdaus, "IoT-based intravenous (IV) therapy monitoring system for elderly care," Eng. Archive (engrXiv), preprint, 2024, doi: 10.31224/4035. [Online]. Available: https://engrxiv.org/preprint/view/4035
4. A. Paranthaman and K. Annalakshmi, "IV fluid bag monitoring, alert, and control system using ESP32 and IoT," Int. J. Sci. Technol. (IJSAT), vol. 16, no. 2, 2025. [Online]. Available: https://www.ijsat.org/research-paper.php?id=5881
5. S. Mitropoulos, D. Rimpas, S. Katsoulis, G. Hloupis & I. Christakis, "Integrated low cost, LoRa-based, real time fluid infusion flask monitoring system," Electronics, vol. 14, no. 5, pp. 869, 2025. https://doi.org/10.3390/electronics14050869
6. U. S. A. Redha, R. M. A. Salman, and T. E. A. Atiya, "An IoT-based intravenous fluid measuring device using ESP32 and ultrasonic sensor," Central Asian J. Med. Natural Sci., vol. 6, no. 4, pp. 2319–2324, 2025. [Online]. Available: https://cajmns.centralasianstudies.org/index.php/CAJMNS
7. K. V. Sriram, A. Udoji, S. Malagi, G. Pawar, and P. Desai, "Smart intravenous fluid monitoring, alert and control system using ESP32 and IoT," Int. Res. J. Modernization Eng. Technol. Sci. (IRJMETS), vol. 7, no. 12, 2025. [Online]. Available: https://www.irjmets.com/upload_newfiles/irjmets71200098558/paper_file/irjmets71200098558.pdf

8. MathWorks, Inc., "ThingSpeak: IoT analytics platform documentation," Natick, MA, USA, 2023. [Online]. Available: https://www.mathworks.com/help/thingspeak/
9. MIT App Inventor Team, "MIT App Inventor 2 documentation," Massachusetts Inst. Technol., Cambridge, MA, USA. [Online]. Available: https://appinventor.mit.edu/explore/documentation
10. Avia Semiconductor, "HX711: 24-bit analog-to-digital converter for weigh scales (datasheet)," Xiamen, China, Rev. 2.0, 2020. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf
11. Espressif Systems, "ESP8266EX datasheet," Version 6.8, Shanghai, China, 2021. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
12. K. D. Irianto and F. R. Firdaus, "IoT-based intravenous (IV) therapy monitoring system for elderly care," Eng. Archive (engrXiv), preprint, 2024, doi: 10.31224/4035. [Online]. Available: https://engrxiv.org/preprint/view/4035