Auditable Real-Time Cold-Chain Monitoring with IoT and Blockchain Anchoring

Mohamed DOUBIZ[1], Mouad BANANE[2], Abdelali ZAKRANI[3], Allae ERRAISSI[4]
Laboratory of Artificial Intelligence-Modeling and Computational Engineering, Hassan II University, Morocco[1,2,3]
Chouaib Doukkali University, El Jadida, Morocco[4]

***Abstract*—Safe vaccine storage hinges on continuous, trustworthy temperature supervision and evidence that records have not been altered. Yet many cold rooms still rely on fragmented logging tools that lack real-time alerts, end-to-end traceability, and audit-ready data. This paper presents a practical, low-cost architecture that integrates Internet of Things (IoT) sensing with blockchain anchoring to deliver real-time monitoring and visibility, reliable anomaly detection, and tamper-evident provenance for cold-chain storage. The design couples minute-level telemetry and dashboarding with alert debouncing/hysteresis to reduce false alarms, while anchoring hourly summaries and event alerts on-chain to create a verifiable trail without exposing raw data or incurring recurring fees during experimentation. A prototype in a vaccine cold-room scenario demonstrates that the approach is simple to deploy on commodity hardware, scales by adding rooms/sensors, and produces operator-friendly notifications alongside independently verifiable records. This combination of edge retention and cryptographic anchoring provides a pragmatic path for pharmacies, clinics, and warehouses to upgrade from basic loggers to transparent, audit-ready monitoring, bridging operational needs (alerts) and compliance needs (provenance) in one system.**

***Keywords*—Internet of Things; blockchain; cold chain; vaccine storage; real-time monitoring; tamper-evident; data provenance; traceability**

I. Introduction

In 2021, Morocco ordered over 60 million doses of COVID-19 vaccines from Sinopharm, Sinovac, and AstraZeneca [1]. Maintaining these vaccines between 2 °C and 8 °C remains a major logistical challenge, particularly in constrained environments. The COVID-19 pandemic made this clear at scale: unprecedented volumes moved in just-in-time flows among manufacturers, carriers, regional warehouses, vaccination centers, and mobile units, with each link responsible for continuous temperature monitoring and shared traceability. Some vaccines, such as Pfizer-BioNTech, required ultra-cold storage (−90 °C to −60 °C) upstream and, after thawing, allowed storage between 2 °C and 8 °C for up to 10 weeks [2], multiplying the thermal-regime transitions that had to be controlled and attested at every hand-off.

This complexity strengthened the demand for end-to-end traceability and tamper-evident records of thermal excursions; several studies therefore proposed IoT-plus-blockchain architectures to record, anchor, and share readings among stakeholders, with the promise of better combating counterfeiting and disputes [3] [4].

However, most existing proposals suffer from at least two limitations: high integration and operating costs (proprietary hardware, cloud services, transaction fees); and a lack of end-to-end evaluations showing how to reconcile continuous monitoring with tamper-evident anchoring at low cost.

This article addresses that gap by presenting and experimentally evaluating a low-cost IoT–blockchain integration architecture for a vaccine cold room. We ask: how can real-time supervision, immediate anomaly detection, and a verifiable proof of integrity be ensured while minimizing operational complexity and cost?

Our hypothesis is that the decoupling rates (fine-grained acquisition with parsimonious anchoring) combined with completeness control can achieve this objective.

Our contributions are as follows.

- An end-to-end prototype built from standard components and open-source building blocks: DS18B20 sensor → ESP8266 (NodeMCU) → MQTT (Mosquitto on Raspberry Pi 4).
- A Python collector (venv) logs measurements, triggers alerts if T < 2 °C or T > 8 °C (Telegram + on-chain event), and anchors an hourly statistical summary to the blockchain (number of samples, min/max/mean temperature, CSV hash).
- A frugal anchoring scheme: sampling is performed every minute (expected: 60 readings/hour), while on-chain anchoring is hourly via a smart contract (anchoring & alerts) on a local Ethereum node (geth, dev mode), eliminating transaction fees during trials. A completeness watchdog flags any hour with fewer than 60 readings (sensor, network, or logging loss).
- Alerting and Visualization Chain: Telegram immediately notifies threshold breaches and each hourly commit (count, min, max, mean, hash).
- A Node-RED dashboard provides live monitoring (gauge + time series) with set-point lines (2 °C/8 °C) and device online/offline status.

Organization Section II surveys related work on IoT–blockchain for the cold chain. Section III details the materials and methods (sensors, MQTT, Python collector, smart contract), including the alerting protocol, hourly anchoring, and the completeness watchdog. Section IV presents the results

and evaluation, including a comparison with representative approaches. Section V discusses limitations and extensions. Section VI concludes.

II. Literature Review

Digital vaccine supply chains (DVSCs) increasingly adopt sensing, connectivity, and data systems to monitor vaccines end-to-end and strengthen resilience [5]. Persistent quality and access gaps in resource-constrained contexts motivate pragmatic storage and distribution solutions that fit local constraints [6]. During COVID-19, large-scale social-media analytics further highlighted operational bottlenecks across logistics [7], [8]. Together, these streams justify lightweight, field-deployable *continuous* monitoring coupled with *tamper-evident* record-keeping as a practical research target.

Within healthcare, the Internet of Things (IoT) has progressed from pilots to domain applications; recent edited volumes synthesize IoT-blockchain convergence patterns, use cases, and architectural considerations [9], [10], [11]. For regulated cold-chain products such as vaccines, raw telemetry is insufficient without trustworthy context: integrity, provenance, and traceability are required to transform temperature streams into audit-ready evidence. Surveys of blockchain-based decentralized trust for IoT conclude that distributed ledgers offer tamper-resistant logs and programmable workflows (smart contracts), while emphasizing throughput, latency, and privacy trade-offs that guide design-to-cost choices [12].

From a supply-chain perspective, engineering-management work documents transparency and governance benefits of blockchain alongside adoption and interoperability barriers, issues that intensify when budgets and connectivity are limited [13]. Implementation studies show that permissioned frameworks (e.g., Hyperledger) can meet trust requirements at lower operational cost when value exchange is not the primary objective [14]. These insights motivate architectures that keep the blockchain surface minimal (hash anchoring / event notarization) while handling bulk telemetry off-chain—well aligned with a low-cost cold-room deployment.

Closer to medicine logistics, a blockchain-based traceability system for *cold-chain medicine logistics* demonstrates end-to-end traceability and automatic quality assessment with compact on-chain proofs and off-chain sensor data [15]. In the blood cold chain, analyses detail vulnerabilities (excursions, custody hand-offs, counterfeit risk) and argue for immutable, condition-aware traceability, directly analogous to vaccine storage and transport [16]. A vaccine-focused chapter surveys blockchain opportunities across the lifecycle (counterfeit mitigation, cold-chain compliance, smart-contract-based conformance) while noting scalability limits that favor minimalist, permissioned deployments [17].

Operations-research work on vaccine logistics consolidates objectives (wastage, service level, cost) and decision layers (facility, inventory, routing) and underlines the value of reliable, high-granularity data for calibration and evaluation [18]. In this sense, an IoT-blockchain stack is both a monitoring/compliance tool and a *data infrastructure*: continuous telemetry (IoT) anchored with immutable proofs (blockchain) yields decision-grade streams for control, alarms, and post-hoc audits. At the interface with public-facing technologies, scoping reviews of social media and digital tools in vaccination stress trustworthy records and transparent workflows for programs and communities, with blockchain among DVSC enablers [19].

Synthesis for the present work. Three design patterns consistently align with a cold-room experiment.

- Separation of concerns: retain high-rate temperature streams off-chain and anchor compact hashes/events on a permissioned ledger [12], [15], [16], [17];
- Programmable compliance: smart contracts for threshold alerts, excursion attestation, and custody milestones [12], [16], [17];
- Design-to-cost: lightweight nodes, batching, and cadence tuned to regulatory and network constraints typical of resource-limited settings [20], [6], [13], [14], [18], [19].

III. Materials and Methods

We built and operated a verifiable cold chain monitor for a pharmacy-grade chamber maintained between 2 °C and 8 °C. A waterproof DS18B20 probe, sampled by a NodeMCU ESP8266 every 60 s, publishes JSON over MQTT to a Raspberry Pi gateway connected via Wi-Fi (wlan0). On the Pi, a Python process ingests the stream, writes hour-aligned CSV files with milli-degree precision, and applies excursion detection with debounce and hysteresis. Immediate alerts and per-hour file-integrity digests (CSV SHA-256) are anchored on a local, feeless Geth developer chain; operators receive human-readable Telegram messages, and a Node-RED dashboard presents a live integer gauge plus a Temperature Evolution chart with fixed 2 °C and 8 °C reference lines. All parameters, file paths, and commands reported below mirror the deployed system, emphasizing reproducibility, auditability, and operational simplicity in resource-constrained settings (Fig.1).

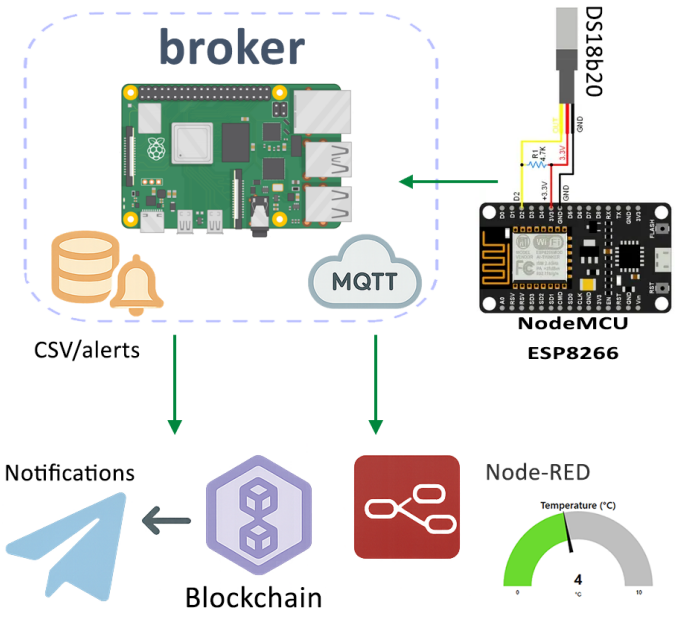

Fig. 1. Architecture: ESP8266 → MQTT → Pi monitor → CSV/alerts → on-chain → Telegram → Node-RED.

A. Hardware and Wiring (DS18B20 / ESP8266 / Raspberry Pi)

1) Cold room (process under control): The monitored environment whose air temperature must remain within 2 °C to 8 °C. It provides the real thermal dynamics (door openings, defrost cycles) against which the system is validated.

2) DS18B20 temperature probe (measurement): Digital 1-Wire thermometer (waterproof lead) that senses air temperature in the chamber. We placed the tip in the airflow, away from coils/doors to limit local bias. The resolution is fine enough for vaccine ranges; readings are consumed as is in this iteration (no correction applied).

4.7 KΩ pull-up (bus integrity). Keeps the 1-Wire data line idle-high and ensures reliable edges on long, thin leads.

3) NodeMCU ESP8266 V3 (sensor node & publisher):

a) Roles: powers the DS18B20 (3.3 V), polls it every 60 s, formats readings as JSON, and publishes via MQTT over Wi-Fi. It also maintains a retained liveness topic and a Last Will ("`offline`") for fault detection

b) What we store in each message: `temp_c` (float), `seq` (monotonic counter), and `ms` (`millis()`), which help the gateway spot gaps or reorders.

c) Why ESP8266: integrated 2.4 GHz Wi-Fi and very low BOM cost.

4) Raspberry Pi 4 (edge gateway & verifier):

- Roles: single-box aggregator on wlan0. It runs:
  - Mosquitto (MQTT broker) to receive ESP8266 messages,
  - our Python collector to append hour-aligned CSV, detect excursions (debounce + hysteresis), and compute SHA-256 digests,
  - Geth (dev mode) to anchor immediate alerts and hourly CSV integrity,
  - Telegram notifications, and
  - Node-RED dashboard (gauge + Temperature Evolution chart with 2 °C/ 8 °C reference lines).
- Networking: the Pi's wlan0 serves both the ESP8266 (same Wi-Fi) and Internet egress for Telegram.
- Wi-Fi access point/router (connectivity) Provides the WLAN where both ESP8266 and Pi's wlan0 are associated (DHCP + IP reachability). Internet access from this WLAN allows Telegram delivery.
- Power ESP8266 via stable 5 V USB (on-board 3.3 V regulator); Raspberry Pi via its PSU. A common ground exists between DS18B20 and ESP8266.
- Wiring we used
  - DS18B20 DATA → D2 (GPIO4) on ESP8266
  - DS18B20 VCC → 3.3 V, GND → GND
  - 4.7 KΩ resistor between DATA and 3.3 V.

B. Sensor Calibration

For this experiment we did not apply a numeric correction to the DS18B20. We performed a sanity check (ice-water bath $\approx 0\,^{\circ}\mathrm{C}$ and ambient $\approx 24\,^{\circ}\mathrm{C}$) and observed readings within the expected DS18B20 accuracy. Because the main objective was demonstrating the end-to-end chain (ingestion → detection → anchoring → notification), we accepted nominal accuracy and deferred formal calibration.

C. Firmware (Sampling, JSON, Topics, LWT)

File: `esp_ds18b20_mqtt.ino` (Arduino IDE).
Libraries: `ESP8266WiFi`, `PubSubClient`, `OneWire`, `DallasTemperature`.

- Sampling: every 60 s; the loop is non-blocking and calls `mqtt.loop()` continuously.
- JSON payload example:
  `{"temp_c": 5.500, "seq": 21, "ms": 1271041}`
  where `seq` increments per publish and `ms = millis()` assists drop/out-of-order detection
- Topics:
  - `coldchain/cold_room_1/status` (retained) → "`online`" at connect; LWT retained "`offline`".
  - `coldchain/cold_room_1/reading` (live) → readings (QoS 0)

D. Transport (MQTT): Broker, Auth, QoS, Tests

We ran Mosquitto on the Pi (listening on 0.0.0.0:1883 so the ESP8266 on Wi-Fi can connect).

- Auth: username/password (user `iotuser`, password stored in `/etc/mosquitto/passwd`).
- Config: `/etc / mosquitto / conf . d / coldchain.conf`
  with `allow_anonymous false`, persistence on, listener on `1883`.
- QoS: 0 to minimize latency; liveness comes from the retained `status`.
- Sanity tests we used repeatedly:

```
mosquitto_sub -h 127.0.0.1 -u iotuser -P
'********' -v -t 'coldchain/#'
mosquitto_pub -h 127.0.0.1 -u iotuser -P
'********' \  -t 'coldchain/cold_room_1/reading
' -m '{"temp_c":5.8,"seq":1,"ms":1000}'
```

E. Collector (CSV Windowing, Quality Policy, Timekeeping)

File: `/home/nuevomar/coldchain/monitor_from_mqtt_onchain.py` (Python 3.11 in a `.venv`). The process subscribes to `coldchain/+/reading` and `coldchain/+/status`, writes slot-based CSV archives, detects excursions, anchors integrity on-chain, and notifies Telegram. Runtime secrets (MQTT credentials, Telegram token, RPC URL) are loaded from `.env`.

Windowing → CSV: Readings are grouped into fixed slots of `SLOT_SECONDS`.

- Production profile: `SLOT_SECONDS = 3600` (one commit/hour).
- Lab profile: `SLOT_SECONDS = 60` (faster iteration).

CSV format: `iso_utc,epoch,temp_C,temp_mC`. We store milli-°C as integers to enable deterministic hashing.

Timekeeping: UTC timestamps from the Pi (NTP-synced). Slot boundaries aligned to the top of the hour in production.

Data quality: Outliers rejected if $|\Delta T/\Delta t| > 5\,^{\circ}\mathrm{C/min}$; missing/late samples recorded as `NaN` and excluded from averages, with `count` reflecting valid points. No smoothing (debounce/hysteresis covers noise).

Artifacts: hourly CSVs in `/var/log/coldchain/hourly/` and per-day alert logs in `/var/log/coldchain/alerts/`.

F. Alarm Logic (Band, Debounce, Hysteresis, Latency)

We enforced a $[2, 8]\,^{\circ}\mathrm{C}$ band with hysteresis and debounce:

- Thresholds: `LOW = 2.0 °C`, `HIGH = 8.0 °C`.
- Hysteresis: `HYST_MARGIN = 0.2 °C` ⇒ recovery band $[2.2, 7.8]\,^{\circ}\mathrm{C}$.
- Debounce: `DEBOUNCE_S = 180 s` (production); 30 s in lab.
- Events: `ALERT_START` after a sustained breach; `ALERT_END` after sustained recovery.
- Observed alert latency: $\approx 1.5-2$ min (lab profile, 60 s sampling / 30 s debounce) and $\approx 4-5$ min (production profile).

G. Blockchain Anchoring (Contract, Events, Schedule)

We used a local Geth developer chain (feeless, ephemeral) to anchor alerts and hourly CSV integrity.

Node startup used:

```
/usr/local/bin/geth --dev \
  --datadir /home/nuevomar/geth-dev \
  --http --http.addr 127.0.0.1 --http.port 8545 \
  --http.api eth,net,web3 --ipcdisable --dev.period 2
```

Contract: `ColdChainVax.sol` (deployed once). Artifacts persisted as `ColdChainVax.abi.json` and `ColdChainVax.address`.

Immediate anchoring (alerts): On `ALERT_START/END`, the collector called `submitReading(sensorId, epoch, temp_mC, kind)` with `sensorId = keccak256("cold_room_1")`, emitting an `Alert` event.

Periodic anchoring (per slot): The collector computed `(count, min, max, avg)` and the SHA-256 digest of the slot CSV, then emitted `HourCommitted(sensorId, slot, count, min_mC, max_mC, avg_mC, csvHash)`.

This yields tamper-evident on-chain proofs for alerts and per-hour CSV integrity while keeping full data off-chain.

H. Notifications (Telegram)

File: `/home/nuevomar/coldchain/notify.py` with `send_telegram(text)`; bot token and chat ID in `.env`.

Messages included:

```
[ALERT START] cold_room_1 2025-09-13T...Z T=...
 C (Above upper)
[CHAIN] submitReading tx: 0x...
[COMMIT] cold_room_1 slot=... count=... min=... max=...
avg=... hash=... tx=0x...
```

Using `wlan0` only, ensured outbound Telegram API access once Wi-Fi was up.

I. Visualization (Node-RED) + Reference-Line Configuration

We installed **Node-RED** with node-red-dashboard and built a single-tab dashboard:

- **Gauge** (°C): receives rounded integer values; Format `{{value | number:0}}`.
- **Temperature Evolution chart:** three series per measurement:
  - live temperature (`topic = cold_room_1`),
  - constant LOW line (`topic = LOW`, payload `2`),
  - constant HIGH line (`topic = HIGH`, payload `8`).
- The function node (two outputs) emits three messages to the chart and a rounded value to the gauge. Chart settings: Legend=ON, Use-one-color=OFF.

The flow is stored at `~/.node-red/flows_<hostname>.json`.

J. Security and Deployment (Secrets, Network, Persistence)

Secrets (`.env`) were set to file mode 0600 and not versioned; Telegram tokens were rotated as needed. MQTT required username/password; the broker listened on `0.0.0.0:1883` so the ESP8266 on the same Wi-Fi could reach it (TLS was not required on this isolated WLAN). Write access to `/var/log/coldchain/*` was restricted. We ran the collector as a single `systemd` unit (`coldchain-monitor-mqtt.service`) when needed to prevent multiple instances. CSV and alert logs persisted under

/var/log/coldchain; retention/backup can be handled with logrotate/rsync. (we preserved full logs during the experiment).

K. Software Versions and Reproducibility Commands

Environment we used As summarized in Table I, our deployment used commodity components and mainstream toolchains, which favors reproducibility.

TABLE I. SOFTWARE ENVIRONMENT USED IN THIS DEPLOYMENT

Layer	Component	Version / Notes
Hardware	Raspberry Pi	Pi 4, 64-bit OS
Firmware	ESP8266 Arduino core	3.x
Sensor libs	OneWire / DallasTemperature	2.x / 3.x
MQTT client	PubSubClient (ESP)	2.x
Messaging	Mosquitto	2.x (system pkg)
OS runtime	Python	3.11
Python pkgs	paho-mqtt / web3 / dotenv	1.6.x / 6.x / 1.x
Blockchain	Geth	v1.16.3 (-dev)
Smart contract	Solidity	0.8.20 (ABI/address persisted)
Node runtime	Node.js	v22.19.0
Edge app	Node-RED (+ dashboard)	3.x (+ node-red-dashboard 3.x)

Commands actually used:

```
# Python venv (+ deps)
cd /home/nuevomar/coldchain
python3 -m venv .venv && source .venv/bin/activate
pip install -U pip paho-mqtt web3 python-dotenv

# Start Geth dev (feeless, ephemeral)
mkdir -p /home/nuevomar/geth-dev
/usr/local/bin/geth --dev \
  --datadir /home/nuevomar/geth-dev \
  --http --http.addr 127.0.0.1 --http.port 8545 \
  --http.api eth,net,web3 --ipcdisable --dev.period 2

# Export runtime env (edit .env beforehand)
set -a; . ./.env; set +a
export RPC_URL=http://127.0.0.1:8545

# Run the monitor
python monitor_from_mqtt_onchain.py

# Check MQTT intake (during tests)
mosquitto_sub -h 127.0.0.1 -u iotuser -P '********' \
  -v -t 'coldchain/#'

# Inspect latest CSV & digest
LAST=$(ls -1t /var/log/coldchain/hourly/\
cold_room_1_*.csv | head -n1)
tail -n5 "$LAST"
sha256sum "$LAST" | cut -d' ' -f1
```

IV. RESULTS AND EVALUATION

A. Setup and Metrics

We sample once per minute (ESP8266→MQTT→Raspberry Pi), aggregate readings into fixed hour-aligned slots, and anchor both immediate alerts and per-slot integrity on chain. Two operating profiles were exercised: a *test profile* (debounce = 30 s) and a *production profile* (debounce = 180 s). We report alert latency (telemetry → alert emission), excursion statistics (count, total minutes out of band, median/P95 excursion duration), and integrity checks (on-chain digest vs. local CSV hash).

B. Network Observations and Mitigation (Wi-Fi → LAN)

During early trials on an isolated WLAN (MQTT QoS 0), we observed intermittent disconnections of the gateway interface (wlan0), visible as Last Will & Testament (LWT) toggles and short gaps in ingestion. Minor hiccups were absorbed by the collector's buffering, but multi-second drops inflated end-to-end alert latency and occasionally deferred Telegram delivery. To remove this confounder for the evaluation, we migrated the gateway to wired Ethernet (eth0); subsequent runs showed a stable 60 s cadence with no disconnects, while the anchoring logic and results remained unchanged.

C. Telemetry Ingestion Over Wi-Fi/MQTT

The ESP8266 (NodeMCU) published one DS18B20 measurement every 60 s over Wi-Fi to the Raspberry Pi Mosquitto broker (wlan0). The Python collector subscribed to coldchain/+/reading, persisted time-windowed CSV files, and computed per-slot statistics.

Minute-level telemetry and slot summary. In a representative one-hour window (slot 488279), we observe 60 samples at a stable 60 s cadence (median/P95 60 s/60 s). Temperatures range from 2.250 °C to 8.563 °C with a mean of 5.025 °C. Band compliance is 54 / 60 (90%) inside [2 °C, 8 °C]; 6 above 8 °C and 0 below 2 °C. The first out-of-range sample above 8 °C occurs at 2025-09-13T23:21:51Z; the last at 23:26:51Z (Fig. 2).

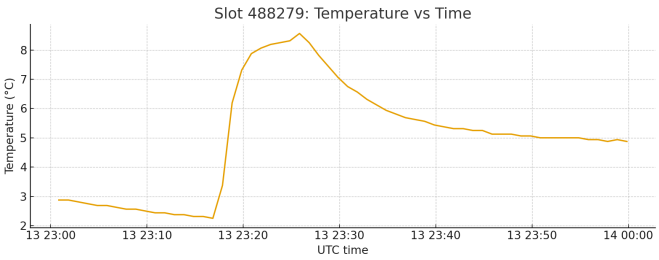

Fig. 2. Slot 488279 time series. The trace shows a brief excursion above 8 °C followed by recovery toward 5 °C.

For the representative slot 488279 (60 min window), we observe Table II:

TABLE II. SLOT 488279 SUMMARY

Rows	60
Sampling interval (median/P95)	60 s / 60 s
Min / Max / Avg	2.250 °C / 8.563 °C / 5.025 °C
Inside band [2 °C, 8 °C]	54 / 60 (90%)
Above 8 °C	6 / 60 (10%)
Below 2 °C	0 / 60 (0%)
CSV SHA-256	105c9c8...b0d49

D. Alarm Detection (2–8 °C Band; Debounce & Hysteresis)

Alerting behavior and latency. On 2025-09-13, the system detected 7 complete excursions (ALERT_START/END)

totaling 13 620 s out of range; median excursion 600 s, P95 6336 s. These timings align with the configured sampling and debounce: under the test profile (60 s sampling, 30 s debounce), end-to-end alert latency was ∼1.5–2 min; under production parameters, ∼4–5 min.

The collector enforced a 2 °C to 8 °C band with 0.2 °C hysteresis and a debounce window (test profile: 30 s; production: 180 s). A visual timeline of the `START` → `END` intervals appears in Fig. 3.

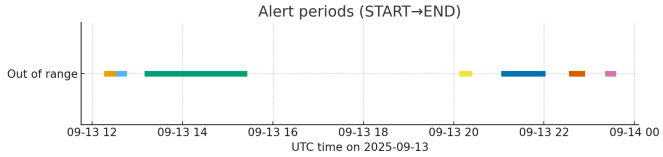

Fig. 3. Alert timeline on 2025-09-13 (START→END intervals).

E. On-Chain Anchoring (Devnet, Feeless)

We anchored both immediate alerts and slot integrity on a local Geth developer chain.

- **Alerts.** Each `ALERT_START/END` triggers `submitReading(sensorId, epoch, temp_mC, kind)`, emitting an `Alert` event; Telegram messages include the transaction hash (e.g., `[CHAIN] submitReading tx: 0xcbc8619c...`).
- **Hourly (slot) commits.** At the end of each slot we compute (count, min, max, avg) and the SHA-256 of the slot CSV and emit `HourCommitted(sensorId, slot, count, min_mC, max_mC, avg_mC, csvHash)`.

For slot 488279, the CSV file's digest exactly matches the on-chain `csvHash` reported in Telegram (`105c9c8...b0d49`), demonstrating reproducible integrity anchoring.

F. Human-in-the-Loop Notifications (Telegram)

The system delivered human-readable messages for `ALERT_START/END`, on-chain submissions, and per-slot commits. Once the Pi default route used `wlan0`, Telegram delivery was reliable. A representative excerpt is shown in Fig. 4, including an alert burst and two consecutive slot commits with their hashes and transaction IDs.

G. Visualization (Node-RED)

The dashboard (Fig. 5) presents an integer gauge and a Temperature Evolution Chart with continuous reference lines at 2 °C and 8 °C. Transient excursions and recovery dynamics are clearly visible; retained MQTT `status` reflects sensor liveness.

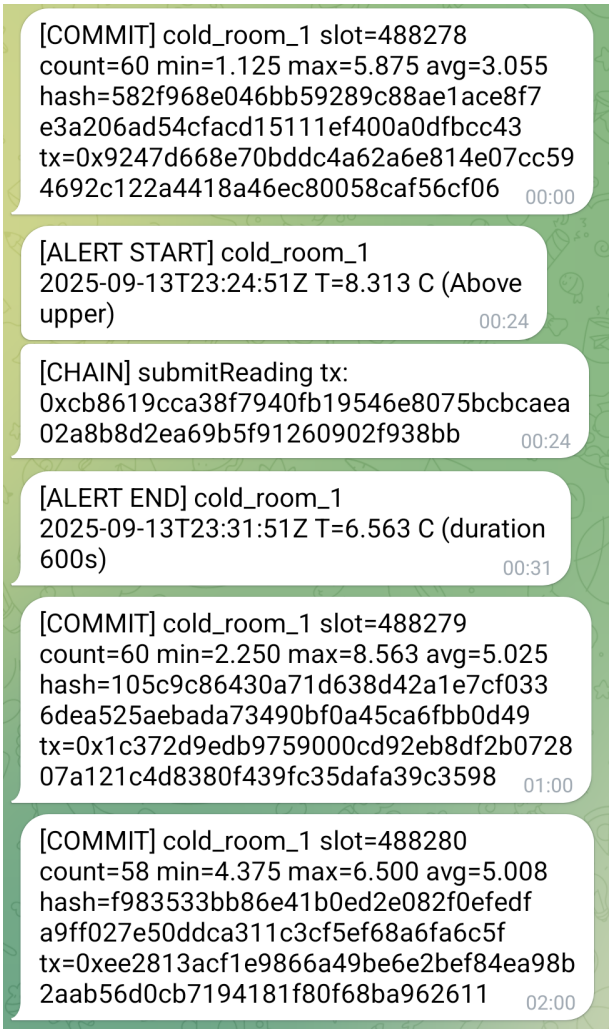

Fig. 4. Telegram excerpt showing an alert, the corresponding on-chain transaction, and per-slot commits with SHA-256 hashes.

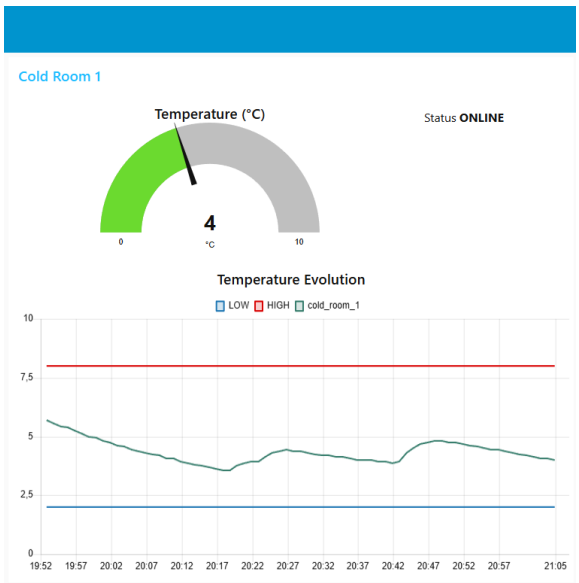

Fig. 5. Node-RED dashboard.

H. System Stability and Resource Use

On a Raspberry Pi 4, the stack (Mosquitto + Python collector + Node-RED + Geth dev) ran comfortably. Mosquitto and the collector exhibited low, bursty CPU at 1 msg/min; Geth's `-dev.period 2 s` caused modest periodic CPU spikes. After configuration stabilized, no restarts were required; a single systemd unit prevented concurrent collectors.

I. Integrity Verification Spot-Checks

We verified that:

- `sha256sum` of the slot CSV equals the on-chain `csvHash`;
- `eth_getTransactionReceipt` + ABI decoding reproduces `Alert` parameters;
- slot statistics `(count/min/max/avg)` recompute from the CSV and match the values posted on chain.

J. Slot Statistics

We partition the stream into fixed, hour-aligned slots ($\Delta_{\text{slot}} = 60\,\text{min}$). For slot 488279, we observe $N_s = 60$ records, median/P95 sampling interval 60 s/60 s, and temperatures in [2.250 °C, 8.563 °C] with a mean of 5.025 °C. Band compliance is 90% inside [2 °C, 8 °C], with 10% above 8 °C. The CSV's SHA-256 digest matches the on-chain `csvHash`, enabling independent verification of the stored file's integrity. Beyond per-slot completeness, we summarize anchoring regularity via inter-anchor intervals and report $p50/p90/p99$ when available.

K. Comparison with Representative Approaches

We contrast our decoupled design with two baselines used in the literature: continuous on-chain logging and batch-only notarization. The Table III is qualitative and highlights the trade-offs reviewers asked to see.

TABLE III. QUALITATIVE COMPARISON WITH REPRESENTATIVE BASELINES

Approach	Characteristics (from our deployment and standard baselines)
Continuous on-chain logging	Per-sample transactions; strong auditability but high transaction volume and on-chain data exposure; alerting often gated by confirmation time; cost scales with telemetry rate.
Batch-only notarization	Daily (or coarse) digests reduce cost but delay provenance; alerts typically derived post hoc; weaker near-real-time guarantees; low on-chain exposure.
This work (decoupled sparse anchoring)	**Real-time alerts from the live stream ;** hourly per-slot digests + event-level alerts provide verifiable provenance; raw data stay off chain (privacy); **cost bounded by slot cadence;** works on feeless devnet and is migration-ready for PoA/L2.

L. Cost Analysis and Total Cost of Ownership (TCO)

We analyse the costs for the implemented pipeline (DS18B20 → ESP8266 → MQTT/Mosquitto on Raspberry Pi → Python collector/CSV/alerts → hourly anchoring on local Geth (feeless) → Telegram/Node-RED), running at one reading per minute per room with hourly anchoring and immediate alert events. For concreteness, set the average powers to $P_{\text{Pi}} = 4\,\text{W}$ (light-load gateway) and $P_{\text{ESP}} = 0.15\,\text{W}$ (low-duty ESP8266 at 1 msg/min); with 24/7 operation the monthly gateway energy is $(0.004 + 0.00015\,N) \times 720 = 2.88 + 0.108N$ kWh for N rooms per gateway, i.e., per room $E_{\text{room}} = (2.88/N + 0.108)$ kWh/month. From the experiment logs (2025-09-13), one hourly CSV contains 60 samples in 2 671 B (min 2.250 °C, max 8.563 °C, mean 5.025 °C; 54/60 within $[2, 8]$ °C), extrapolating to ≈ 62.6 KB/day/room and ≈ 1.92 MB/month/room; the representative JSON payload `{"temp_c":5.500,"seq":21,"ms":1271041}` is $\approx$ 43 B, giving $\approx$ 1.86 MB/month/room of payload-only traffic, or $\approx$ 4.3–8.6 MB/month/room assuming a 100–200 B/message envelope (the broker is local; WAN usage is negligible). Anchoring is feeless in trials; on a PoA/L2 network the monthly on-chain volume per room is $720 + 2 \cdot \#\text{exc}$ transactions (hourly commits plus START/END), on 2025-09-13 there were 7 complete excursions (14 alert events), so the daily total would have been 24 hourly commits + 14 alerts = 38 transactions. This "alert-now, attest-later" scheme preserves near-real-time responsiveness while bounding OPEX by the hourly cadence and shared gateway resources (see Sections III–IV).

V. DISCUSSION

The experimental system delivered near–real-time visibility, reliable and latency-bounded alerts, and cryptographically verifiable provenance of stored data—using inexpensive hardware and a feeless local chain during testing. By combining edge CSV retention with per-slot hashing and event-level anchoring, the approach offers a pragmatic path to transparent, auditable cold-chain monitoring for vaccine storage while keeping operational complexity modest.

A. Contributions and Advantages

1) Real-time alerting without blockchain gating: In our deployment, alerts are produced directly from the live stream, achieving minute-scale responsiveness ($\approx 1.5\,\text{min}$ to $2\,\text{min}$) in the test profile and $\approx 4\,\text{min}$ to $5\,\text{min}$ in the production profile) rather than waiting for block confirmation. This preserves real-time operation while reserving the anchoring path for integrity and audit.

2) Tamper-evident provenance without exposing raw data: Event-level alert emissions and per-slot (hourly) CSV integrity digests (SHA-256) allow operators to recompute hashes and match on-chain values, yielding an auditable trail without placing raw telemetry on chain.

3) Operational simplicity on low-cost hardware: A single Raspberry Pi aggregates MQTT readings, applies debounce/hysteresis, writes hour-aligned CSV, and drives a Node-RED dashboard and Telegram notifications. The setup uses commodity components and documented versions/commands, emphasizing reproducibility.

4) Scalability and predictable cost: Anchoring at a slot/hour cadence bounds the number of commits—and therefore cost—while the topic-based design scales to multiple rooms. The same pattern can migrate from a feeless local

devnet to a private PoA or low-cost L2 without changing application logic.

5) Actionable operator visibility: The dashboard presents the averaged room series with fixed 2 °C and 8 °C reference lines, liveness indicators, and notifications, turning telemetry into decisions while retaining verifiable integrity of the underlying records.

B. Alignment with Objectives

1) Real-time visibility: With a 60 s sampling interval (ESP8266→MQTT→Raspberry Pi over Wi-Fi), the system provided continuous telemetry on a Node-RED dashboard (gauge + Temperature Evolution chart) and retained a minute-level historical trace. In a representative one-hour window (slot 488279) we observed 60 samples with 2.250 °C to 8.563 °C (mean 5.025 °C). Minute-level resolution is operationally adequate for pharmacy/clinic cold rooms, and the data volume remains tractable for edge storage.

2) Reliable, Interpretable alerting: The collector enforced a 2 °C to 8 °C band with 0.2 °C hysteresis and debounce. The behavior matches the design intent: short spikes did not trigger alerts, while sustained deviations did. End-to-end alert latency was ($\approx$ 1.5 min to 2 min) (test profile) and $\approx$ 4 min to 5 min (production), satisfying the operational need to react to sustained thermal risk rather than instantaneous noise.

3) Transparency and tamper-evidence: Two on-chain commitments were used: an immediate alert event (`Alert`) carrying the offending sample and timestamp, and a per-slot integrity event (`HourCommitted`) posting (*count, min, max, avg*) and the SHA-256 digest of the slot CSV. For slot 488279, the local `sha256sum` exactly matched the on-chain `csvHash`, enabling third-party verification that the plotted/analyzed file is bit-for-bit identical to the original at commit time.

4) Secure(er) retention: All raw measurements were kept off chain as CSV on the Pi, while only digests and minimal statistics were published on chain. MQTT authentication (username/password) and file permission hardening for secrets (`.env`) were applied. This separation reduces privacy/volume risk while preserving public verifiability of data integrity.

C. Reliability, Latency, and Human-in-the-Loop Operation

Although MQTT ran at QoS 0 on an isolated WLAN, reliability was high because the sampling rate is modest (1 msg/min) and the retained Last Will & Testament (LWT) topic exposed liveness ("online/offline"). Debounce and hysteresis suppressed flapping near thresholds. Telegram messaging closed the loop for operators: each `[ALERT START]` was paired with an on-chain submission (`[CHAIN] submitReading tx:...`), and each slot closure yielded a `[COMMIT] ...hash=...` line. From an operational standpoint, the latency budget is dominated by the sampling period and debounce; measured timings indicate the stack introduces minimal additional delay, which is appropriate for cold-chain supervision where cumulative exposure governs disposition decisions.

D. Auditability via Local, Feeless Anchoring

A low-friction audit path is available: an operator (or auditor) recomputes a slot CSV's SHA-256, retrieves the corresponding `HourCommitted` log, and verifies a match without accessing raw on-chain data beyond the digest and summary statistics. Using a local Geth developer network (`-dev`) removed transaction fees during experimentation and provided deterministic inclusion (typically $\sim$2–3 s with `-dev.period 2`). The anchoring pattern itself is chain-agnostic and can migrate to a private Proof-of-Authority network or a low-cost L2, preserving integrity guarantees while bounding cost through hourly (or daily) batching.

E. Security and Privacy Posture

The design keeps raw data off chain, publishes only hashes and aggregates, and authenticates MQTT clients. Secrets (MQTT password, Telegram token, RPC URL) are stored locally with restrictive permissions. While this posture is sufficient for a laboratory deployment, production environments should additionally enable TLS for MQTT and per-topic ACLs, rotate credentials, and consider a hardened key-management process for blockchain identities.

F. Generalizability and Scalability

The architecture scales by topic: additional rooms or probes publish to `coldchain/<room>/reading`, and the collector subscribes accordingly (wildcards). Edge compute load grows roughly linearly with sensor count and remains modest at minute-level sampling. On-chain costs scale with commit frequency, not raw telemetry size; therefore, batch anchoring (hourly/daily) keeps costs predictable. The same dashboard pattern (gauge + multi-series chart with reference lines) extends directly to multiple rooms.

G. Why the Design Works

The core advantage stems from *decoupling* sensing/alerting from anchoring. Alerts are produced directly from the live stream, so their latency is governed primarily by sampling and debounce rather than block confirmation. Hour-aligned slots provide a fixed cadence for integrity commitments, ensuring predictable anchoring overhead while keeping raw telemetry off chain. Room-level averaging (Eq. 1) concentrates operator attention on a single, representative series per room, reducing noise from probe-level idiosyncrasies yet preserving anomaly sensitivity through within-room spread and quorum checks.

H. Where it can Break (and How it Degrades)

1) Connectivity and buffering: With minute-level sampling, short Wi-Fi outages delay—but do not invalidate—alerts if the collector buffers; longer outages can push alert latency beyond the operational target. Hourly anchoring tolerates skipped anchors (later slots still authenticate prior CSVs), but prolonged loss creates gaps that must be explained operationally.

2) Sensing fidelity: Averaging can mask localized hot/cold spots if probes are poorly placed or if quorum degrades. The policy mitigates this via n_{valid} and the within-room spread (`std_C`); however, if $n_{\text{valid}} < 2$ the sample is marked missing, which reduces continuity for audit unless operators act on "quorum degraded" signals.

3) Chain and timing assumptions: Using a local developer chain removes fees and gives deterministic inclusion; migration to a private PoA or low-cost L2 preserves the pattern but changes absolute inclusion timings and operational procedures (RPC endpoints, keys, monitoring). The design remains chain-agnostic because it commits only digests and slot metadata.

I. Limitations and Threats to Validity

Ephemeral chain. The Geth developer chain is transient by design; for long-term evidence a persistent private chain or L2 must be used. The integrity mechanism remains unchanged.

- Single host and Wi-Fi only. The Raspberry Pi is a single point of failure; a UPS and hot-standby would increase availability. Wi-Fi loss would delay alerts; a wired or cellular fallback mitigates this risk.
- QoS 0 and no TLS (lab). Adequate for an isolated WLAN; regulated deployments should enable TLS and evaluate QoS based on loss/latency trade-offs.
- Temperature-only sensing. No door/humidity/compressor state was collected; adding these channels would improve root-cause attribution.
- Calibration not formalized. DS18B20 readings were sanity-checked but no traceable calibration curve was applied. A two-point correction (offset/slope) can be applied in firmware or at the collector without affecting anchoring.

J. Practical Implications

1) Parameterization: A production profile of 60 s sampling, 180 s debounce, and 0.2 °C hysteresis provided interpretable alerts while avoiding noise. Thresholds should match the product label (e.g., $+2$ to $+8$ °C for common vaccines).

2) Multi-sensor deployment with room-level averaging (recording policy): To improve accuracy and spatial representativeness while keeping the dataset compact, each cold room is equipped with ≥ 2 DS18B20 probes, but the *recorded and visualized data are the room-level average* rather than per-probe series. Let $T_i^{\mathrm{cal}}(t)$ denote the *calibrated* reading of probe i at time t (two-point offset/slope applied in firmware or at the collector). The stored value is

$$T_{\mathrm{room}}(t) \;=\; \frac{1}{N_t} \sum_{i \in \mathcal{V}(t)} T_i^{\mathrm{cal}}(t), \tag{1}$$

where $\mathcal{V}(t)$ is the set of *valid* probes at t (self-test passed, plausible range, not stale), and $N_t = |\mathcal{V}(t)|$. If fewer than a quorum (e.g., $N_t < 2$) are valid, the sample is marked missing.

CSV schema (recommended): store the averaged sample only, plus quality fields:

```
iso_utc, epoch, temp_C_mean,
temp_mC_mean, n_valid, std_C
```

As defined in Eq. (1), we average only valid probes at each timestamp. Here, `n_valid` is the number of valid probes contributing to Eq. (1) (i.e., `n_valid` $= N_t$), and `std_C` is the within-room standard deviation (a compact indicator of spatial spread without retaining per-probe traces).

3) Anchoring cadence (room-level only): Keep one on-chain commit per slot per room (e.g., hourly in production) with (count, min, max, avg) computed from the averaged series and the SHA-256 of the *room-level* CSV. Immediate alert events (`Alert`) are emitted only for room-level breaches. This preserves a tamper-evident trail while avoiding per-probe on-chain artifacts.

4) Multi-channel notifications: For operational resilience, deliver alerts in parallel:

- Telegram (already implemented) for rapid chat notifications,
- SMS (via provider API or GSM HAT) for reachability when data services are limited,
- E-mail (SMTP) for audit-friendly, searchable records.

Use a common message template (room ID, UTC time, averaged value, breach type, on-chain tx/hash), with delivery retries, rate-limiting (minimum interval between identical alerts), and escalation (e.g., notify a second contact after 10 min unresolved). Maintain daily liveness pings on all channels. In regulated environments, configure SMTP with SPF/DKIM/DMARC and manage provider API keys as encrypted secrets.

5) Operator workflow: Dashboards present the *single averaged room series* with continuous 2 °C/8 °C reference lines. Expose `n_valid` and `std_C` alongside the gauge to surface sensor health and spatial dispersion without storing individual probe traces. Provide cumulative minutes out of range and mean kinetic temperature (MKT) as secondary KPIs.

6) Security and migration: The averaging policy does not change security posture: secrets remain local; MQTT keeps authentication; raw samples stay off chain. Migrating anchoring from the local developer chain to a private PoA or low-cost L2 network, and enabling MQTT TLS + ACLs, are incremental steps that leave application logic and the averaging workflow unchanged.

VI. Conclusion

We presented a practical IoT–blockchain architecture for vaccine cold rooms that *decouples* real-time alerting from ledger anchoring. By producing alerts directly from the live stream and periodically committing hour-aligned SHA-256 digests of room-level records (plus event-level alerts), the system delivers minute-scale responsiveness, tamper-evident provenance, and predictable anchoring overhead—all on commodity hardware and without exposing raw telemetry on chain.

Our experiments confirmed this design: with 60 s sampling and simple debounce/hysteresis, the pipeline achieved minute-scale end-to-end alert latency while suppressing transient spikes, and on-chain `csvHash` values exactly matched local CSV digests, enabling independent verification of stored data. The resulting operator loop (dashboard + notifications) turns telemetry into timely, auditable actions, and the topic-based design scales to multiple rooms while bounding cost by slot cadence.

A. Main Contribution

We contribute a low-cost, reproducible pipeline that separates *detection* from *attestation*—"alert now, attest later"—achieving real-time supervision with verifiable integrity and controllable cost.

B. Future Work: Cost and Deployment Studies

Perform a total cost of ownership (TCO) analysis comparing a private PoA chain and a low-cost L2 under different anchoring cadences (per-alert only, hourly+alerts, daily+alerts). Instrument the gateway to measure energy (W/kWh) for the full stack (`Mosquitto` + collector + `Node-RED` + chain client) and compare against alternatives (e.g., Raspberry Pi 4 vs. Pi Zero 2 W vs. x86 NUC/mini-PC). Report KPIs such as cost/room/month, cost/commit, kWh/day, and CPU/RAM headroom; derive the break-even cadence (hourly vs. daily) that minimizes cost/audit proof while meeting the SLA for evidence availability.

REFERENCES

[1] Y. Ait Hamdan, F. El Amerany, J. Desbrieres, A. Aghrinane, H. Oudadesse, and M. Rhazi, "The evolution of the global covid-19 epidemic in morocco and understanding the different therapeutic approaches of chitosan in the control of the pandemic," *Polymer Bulletin*, vol. 80, no. 10, pp. 10 633–10 659, 2023.

[2] M. Wallace, J. P. Collins, H. Moline, I. D. Plumb, M. Godfrey, R. L. Morgan, D. Campos-Outcalt, S. E. Oliver, K. Dooling, and J. W. Gargano, "Effectiveness of pfizer-biontech covid-19 vaccine as evidence for policy action: A rapid systematic review and meta-analysis of non-randomized studies," *PLoS One*, vol. 17, no. 12, p. e0278624, 2022.

[3] U. J. Munasinghe and M. N. Halgamuge, "Supply chain traceability and counterfeit detection of covid-19 vaccines using novel blockchain-based vacledger system," *Expert Systems with Applications*, vol. 228, p. 120293, 2023.

[4] A. Musamih, R. Jayaraman, K. Salah, H. R. Hasan, I. Yaqoob, and Y. Al-Hammadi, "Blockchain-based solution for distribution and delivery of covid-19 vaccines," *Ieee Access*, vol. 9, pp. 71 372–71 387, 2021.

[5] W. Ran, Z. Li, Y. Xue, and D. He, "Literature review: Current trends and future prospects of digital vaccine supply chain support technology," *Human Vaccines & Immunotherapeutics*, vol. 21, no. 1, p. 2553454, 2025.

[6] G. Y. Scott and et al., "Enhancing vaccine quality and accessibility: Strategies for efficient storage and distribution in resource-constrained environment," *Outbreak Management and Response*, vol. 1, no. 1, 2025.

[7] A. Erraissi and M. Banane, "Machine learning model to predict the number of cases contaminated by covid-19," *IJCDS Journal*, vol. 10, pp. 991–1001, 10 2021.

[8] E. Cano-Marin, D. Ribeiro-Soriano, A. Mardani, and C. Blanco Gonzalez-Tejero, "Exploring the challenges of the covid-19 vaccine supply chain using social media analytics: A global perspective," *Sustainable Technology and Entrepreneurship*, vol. 2, no. 3, p. 100047, 2023.

[9] D. Mohamed, M. Banane, A. Zakrani, and A. Erraissi, *Integrating IoT for Enhanced Traceability, Safety, and Real-Time Control in Smart Farming Systems*, 12 2024, pp. 199–206.

[10] B. el Khalyly, A. Belangour, A. Erraissi, and M. Banane, "Devops and microservices based internet of things meta-model," *International Journal of Emerging Trends in Engineering Research*, vol. 8, 09 2020.

[11] S. K. Gupta, J. Rosak-Szyrocka, A. Mittal, S. K. Singh, and O. Hrybiuk, Eds., *Blockchain-Enabled Internet of Things Applications in Healthcare: Current Practices and Future Directions*. Singapore: Bentham Science, 2025, eISBN: 978-981-5305-21-0.

[12] Q.-u.-A. Arshad, W. Z. Khan, F. Azam, M. K. Khan, H. Yu, and Y. B. Zikria, "Blockchain-based decentralized trust management in iot: systems, requirements and challenges," *Complex & Intelligent Systems*, vol. 9, no. 6, pp. 6155–6176, 2023.

[13] H. A. Fattahi Bafghi, "Leveraging blockchain technology for sustainable, transparent and efficient supply chain management: An integrative exploration from an engineering management perspective," 2024.

[14] R. C. Aguilera, M. P. Ortiz, J. P. Ortiz, and E. V. Lozada, "Funding hyperledger blockchain dapp for covid-19 pandemic," *Fractals*, vol. 30, no. 6, p. 22501286, 2022.

[15] W. Zeng, Y. Wang, K. Liang, J. Li, and X. Niu, "Advancing emergency supplies management: A blockchain-based traceability system for cold-chain medicine logistics," *Advanced Theory and Simulations*, vol. 7, no. 4, p. 2300704, 2024.

[16] D. Asokan, J. Sunny, V. M. Pillai, and H. V. Nath, "Blockchain technology: a troubleshooter for blood cold chains," *Journal of Global Operations and Strategic Sourcing*, vol. 15, no. 3, pp. 316–344, 2022.

[17] T. Mawela, H. Smuts, F. Adebesin, M. Hattingh, and G. Maramba, "Opportunities for blockchain technologies in vaccine supply chain management," in *Bio-Inspired Computing: Proceedings of IBICA 2023, Volume 5: Real World Applications*, ser. Lecture Notes in Networks and Systems. Cham: Springer, 2025.

[18] J. Locke, M. Taghavi, B. Mansouri, and A. Saif, "Optimizing vaccine logistics: A taxonomy and narrative review," *INFOR: Information Systems and Operational Research*, 2025.

[19] S. Zang, X. Zhang, Y. Xing, J. Chen, L. Lin, and Z. Hou, "Applications of social media and digital technologies in vaccination: Scoping review," JMIR Preprints, 2023, preprint; final published version available in *Journal of Medical Internet Research*.

[20] A. Erraissi and A. Belangour, "A big data security layer meta-model proposition," *Advances in Science Technology and Engineering Systems Journal*, vol. 4, pp. 409–418, 10 2019.