Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.080490

ARTICLE

A Low-Code Orchestration Middleware for Secure and Transparent IoT–Blockchain Integration

Jesús Rosa-Bilbao*

UCASE Software Engineering Research Group, Department of Computer Science and Engineering, University of Cadiz, Cádiz, Spain

*Corresponding Author: Jesús Rosa-Bilbao. Email: jesus.rosa@uca.es

Received: 11 February 2026; Accepted: 03 April 2026; Published: 08 May 2026

ABSTRACT: The integration of Internet of Things (IoT) infrastructures with Distributed Ledger Technologies (DLT) remains challenging due to the reliance on complex, tightly coupled back-end systems or centralized oracle services that hinder scalability, maintainability, and trust. This paper introduces a lightweight middleware architecture based on a Low-Code Development Platform (LCDP) that enables flexible and secure IoT-to-blockchain orchestration. We develop a custom workflow extension for the n8n platform that supports direct interaction with smart contracts, thereby removing the need for third-party oracle intermediaries. The proposed system was evaluated in a real-world deployment involving a network of Netatmo environmental sensors and the Alastria consortium blockchain. Experimental results show that the middleware can process 12 concurrent sensor data streams with an average end-to-end latency of 37.4 s, a delay dominated by the blockchain consensus time rather than middleware overhead, while ensuring the generation of immutable and verifiable audit trails. These findings demonstrate that low-code orchestration can deliver an effective, scalable, and fault-tolerant alternative for integrating IoT infrastructures with blockchain in Industry 4.0 environments.

KEYWORDS: Low-code development platform; IoT–blockchain integration; distributed ledger technologies; workflow orchestration; smart-contract interaction; data integrity

1 Introduction

Industry 4.0 has transformed the monitoring and management of physical infrastructures through the large-scale deployment of Internet of Things (IoT) devices that continuously generate operational telemetry [1,2]. As organizations increasingly rely on these data streams for compliance and decision making, integrity, traceability, and non-repudiation become mandatory requirements. In domains such as environmental monitoring and regulated logistics, tampered or missing sensor records can directly affect legal accountability and operational safety [3].

Distributed Ledger Technologies (DLT), particularly blockchain systems, provide a strong foundation for tamper-evident record keeping in untrusted environments [4,5]. However, integrating heterogeneous and resource-constrained IoT infrastructures with deterministic consortium blockchains remains difficult in practice [6,7]. A central obstacle is that blockchains cannot directly consume off-chain sensor information, which shifts trust to middleware and oracle layers [8,9].

Current integration strategies typically rely on centralized middleware, custom monolithic back-end services, or external oracle operators. Although functional, these approaches often increase development effort, hinder maintainability, and concentrate trust in components that are hard to audit end to end. This

limitation is especially problematic in evolving IoT environments where devices, payload schemas, and compliance requirements change frequently.

Low-Code Development Platforms (LCDPs) have emerged as an alternative for rapid orchestration, but their role in secure IoT–DLT integration is still underexplored in academic literature [10]. In particular, there is limited evidence on whether a workflow-centric platform such as n8n can provide cryptographic transaction handling and deterministic IoT-to-blockchain settlement without resorting to a custom monolithic back end.

Recent studies on blockchain-enabled distributed coordination further motivate explicit treatment of determinism, collaboration, and deployment practicality in orchestration-layer design [11]. At the same time, IoT security literature emphasizes that network-layer hardening does not remove the need for robust application-layer trust boundaries in data-settlement pipelines [12].

To address this gap, this paper proposes and evaluates a low-code orchestration middleware that extends n8n with domain-specific modules for Netatmo data ingestion and smart-contract transaction signing. The architecture targets secure and transparent IoT–Blockchain integration by reducing coupling between sensing, orchestration, and on-chain settlement.

The study is guided by three research questions. **RQ1:** Can a low-code workflow engine be extended to execute secure and deterministic IoT-to-blockchain settlement without third-party oracle services? **RQ2:** What are the latency and computational-cost characteristics of the proposed architecture under continuous multi-sensor operation? **RQ3:** Which architectural trade-offs emerge when comparing this approach with conventional integration paradigms reported in the literature?

The contributions are threefold. First, we define a layered reference architecture that decouples IoT acquisition, orchestration, and blockchain settlement while preserving auditability. Second, we implement reusable n8n extensions for authenticated Netatmo ingestion and local raw Ethereum transaction signing. Third, we provide an empirical evaluation in a real deployment with Netatmo sensors and the Alastria consortium blockchain, reporting latency, gas consumption, reliability boundaries, and observed operational limitations.

The remainder of this paper is organized as follows. Section 2 reviews related work and critically contrasts architectural paradigms for IoT–DLT integration. Section 3 details the proposed middleware and smart-contract interaction flow. Section 4 presents the experimental design, reproducibility parameters, and evaluation metrics. Section 5 reports and discusses the empirical findings. Section 6 concludes the paper and outlines future work.

2 Background and Related Work

This section summarizes the technological foundations of IoT–DLT integration and positions the proposed middleware against existing architectural alternatives.

2.1 IoT Data Heterogeneity and Auditability Requirements

IoT ecosystems are heterogeneous by design, with devices exposing different communication protocols, data schemas, update frequencies, and reliability profiles [7,13]. In practical deployments, this heterogeneity introduces temporal misalignment and schema inconsistency before any on-chain operation can be executed [6,7]. Consequently, integration quality depends not only on data acquisition but also on deterministic normalization, validation, and transformation into blockchain-compatible payloads [2].

From an architectural perspective, this challenge is tightly coupled with edge-to-cloud partitioning decisions. Prior IoT studies show that preprocessing and filtering closer to data sources can reduce upstream

bottlenecks but also introduce consistency-management complexity across distributed layers [14,15]. In blockchain-backed auditing scenarios, this means that middleware design must jointly optimize data quality, temporal consistency, and settlement readiness, rather than treating these concerns as independent steps.

2.2 Consortium Blockchains for Industrial Telemetry

Public blockchains offer strong decentralization but often exhibit variable latency and volatile transaction fees, which can be unsuitable for continuous telemetry workloads [16]. Consortium blockchains provide a different trade-off by combining distributed trust with controlled governance and more predictable performance [17,18]. In this work, we use Alastria, a Quorum-based consortium network, because it supports enterprise governance and deterministic finality requirements typically demanded in compliance-oriented IoT scenarios [19].

This choice is also aligned with prior evidence that enterprise blockchain deployments are often selected when governance control, organizational accountability, and predictable operation are prioritized alongside distributed trust [17,19].

2.3 Smart Contracts as Audit Anchors

Smart contracts provide deterministic state transitions and immutable event logs that can support non-repudiable audit trails [20]. Since direct on-chain storage of full telemetry streams is inefficient, prior work recommends storing compact representations and application-relevant fields while preserving verifiability [21]. Within this context, the contract layer should be interpreted as a settlement and verification anchor, while pre-settlement data preparation remains a middleware responsibility.

Recent blockchain-in-information-systems reviews reinforce this separation by showing that most practical failures originate in integration and governance boundaries, not in contract execution itself [22]. This motivates our focus on the orchestration layer as a first-class research object.

2.4 Critical Comparison of Integration Paradigms

Prior literature on IoT–DLT integration can be grouped into three dominant paradigms. The first paradigm uses centralized oracle or gateway services that simplify implementation but introduce single points of trust and failure [8]. The second paradigm uses monolithic custom back ends (typically in Python, Java, or Go), which can provide full control but at high maintenance cost and limited reconfigurability when devices or schemas change. The third paradigm uses flow-based low-code tools that improve development speed and ease process reconfiguration in industrial settings [23,24]. For IoT–blockchain research, however, the available evidence on blockchain-specific cryptographic support and rigorous evaluation remains comparatively limited.

Our approach adopts the third paradigm but extends it with cryptographic and domain-specific modules to address its usual limitations. Compared with centralized oracle solutions, the proposed architecture reduces trust concentration by performing local transaction signing and direct settlement to the consortium chain. Compared with monolithic back-end implementations, it emphasizes modularity and maintainability through reusable workflow components. In addition, unlike traditional IoT flow tools mainly used for event routing and interoperability brokering [25], the proposed n8n extensions target deterministic blockchain settlement semantics and key-handling constraints that are usually handled outside low-code environments. This design choice also differs from generic low-code platforms whose primary focus is execution-strategy composition rather than blockchain-oriented settlement guarantees [26,27]. These trade-offs are directly connected to RQ3 and are revisited in Section 5 through a structured discussion of strengths and limitations.

2.5 *Structured Comparison with Node-RED-Centric Integrations*

To make the novelty claim auditable, Table 1 summarizes a feature-level comparison between representative Node-RED-centric IoT–Blockchain integrations and the proposed n8n-based middleware.

Table 1: Feature-level comparison between Node-RED-centric and proposed n8n-based middleware approaches.

Dimension	Node-RED-Centric Integrations	Proposed n8n Middleware
Cryptographic support	Usually delegated to external scripts/services; limited first-class support for transaction signing in workflow nodes.	Native workflow integration through custom smart-contract node with local raw-transaction signing.
Determinism of settlement path	Event-driven orchestration with potential ad-hoc script branches depending on deployment style.	Fixed linear pipeline for ingestion, normalization, signing, submission, and receipt logging.
Scalability model	Scales with additional flows and external microservices; operational consistency depends on custom glue code.	Scales through reusable typed nodes and explicit fan-out handling in one orchestrated workflow runtime.
Deployment overhead	Fast initial setup, but blockchain-specific extensions often require external wrappers and maintenance scripts.	Single Dockerized runtime plus domain-specific nodes; blockchain interaction logic embedded in reusable modules.
Trust concentration	Can rely on centralized external oracle handlers for signing and relay.	Self-hosted middleware signs locally and writes directly to chain; no third-party oracle operator in the execution path.

3 Proposed Architecture

To address interoperability, auditability, and maintainability challenges in industrial IoT environments, we propose a three-layer architecture composed of a perception layer, an orchestration layer, and a settlement layer. Fig. 1 presents the high-level design.

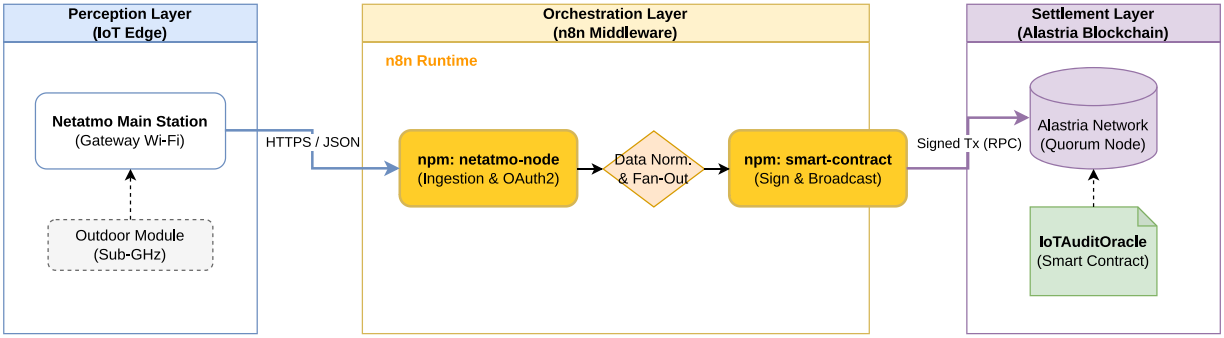

Figure 1: High-level architecture of the proposed solution.

The perception layer includes Netatmo stations that expose telemetry through cloud Application Programming Interfaces (APIs) with asynchronous update behavior. The orchestration layer is a self-hosted n8n runtime extended with two custom nodes. The first node handles authenticated Netatmo ingestion and canonical payload preparation. The second node handles smart-contract interaction, including Application Binary Interface (ABI)-driven parameter mapping, local transaction construction, and raw transaction signing. The settlement layer is the Alastria permissioned blockchain, where the IoTAuditOracle contract stores and indexes validated telemetry events.

This separation of concerns is central to the proposal. Sensing remains device-specific, orchestration remains workflow-centric and reconfigurable, and settlement remains immutable and verifiable. As a result, new IoT sources can be incorporated by extending ingestion mappings without redesigning blockchain logic, while contract-level audit guarantees remain stable.

3.1 Oracle-Service vs. Self-Hosted Middleware Clarification

The manuscript distinguishes between two often conflated concepts. A third-party oracle service is an external trust intermediary that signs or relays off-chain information on behalf of the application and therefore introduces an additional trust dependency. In contrast, the proposed orchestration middleware is self-hosted by the data owner, executes deterministic transformation rules, and signs transactions locally before direct RPC submission. The middleware still plays an oracle *function* (bridging off-chain to on-chain data), but not an external oracle *service* role in the trust model. This distinction is critical for interpreting the claim of eliminating third-party oracle intermediaries.

3.2 Custom n8n Extensions

The first extension, `n8n-nodes-netatmo-weather-station`, encapsulates OAuth2 credential handling, token refresh, and hierarchical JSON parsing [28]. Instead of exposing raw vendor payloads downstream, it outputs a normalized object model with explicit module identifiers and typed fields suitable for deterministic mapping. The node also distinguishes between upstream API failures and partial sensor unavailability to avoid silent data loss.

The second extension, `n8n-nodes-smart-contract`, performs contract interaction without requiring a general-purpose back-end service [29]. The node loads the contract ABI, builds function-call payloads, signs transactions locally using environment-injected private keys, and submits signed transactions to the authenticated RPC endpoint. This design keeps secrets outside visual workflow definitions and improves traceability by exposing execution status and transaction identifiers at each run.

3.3 IoTAuditOracle Contract Interface and Storage Logic

For clarity and reproducibility, this section provides explicit contract-level detail. The core contract operation receives a device identifier, a normalized metric value, and a timestamp generated by the orchestration layer. Conceptually, the write path is represented by the following interface and event model:

```
function recordMetric(
       bytes32 deviceId,
       int256 metricValue,
       uint256 ts
) external;
event  MetricRecorded(
```

```
        bytes32  indexed  deviceId,
        int256  metricValue,
        uint256  ts,
        address  indexed  sender
    );
```

At execution time, `recordMetric` updates the latest value associated with each `deviceId` and emits `MetricRecorded` for chronological indexing. The storage update guarantees deterministic overwrite semantics for the latest state, while event logs preserve an append-only historical trace for external auditing. This dual pattern enables efficient state retrieval and complete historical verification.

Table 2 summarizes the core storage and event structure used by the contract.

Table 2: Core IoTAuditOracle storage and event layout.

Element	Type	Purpose
`latestMetric [deviceId]`	Mapping(bytes32 → int256)	Stores latest normalized metric per device for efficient current-state lookup.
`latestTimestamp [deviceId]`	Mapping(bytes32 → uint256)	Stores latest ingestion timestamp per device for temporal consistency checks.
`MetricRecorded`	Event	Emits append-only audit record: indexed device, value, timestamp, and sender address.
`recordMetric(...)`	Function	Validates input and updates mappings, then emits event for immutable chronological indexing.

3.4 Deterministic Numeric Normalization

To ensure compatibility with integer arithmetic in the Ethereum Virtual Machine, the orchestration layer applies fixed-point conversion before settlement. For a sensor value V_{sensor} and precision factor k, the transformed on-chain value V_{chain} is calculated as in Eq. (1).

$$V_{chain} = \lfloor V_{sensor} \times 10^{k} \rfloor \tag{1}$$

In this study, $k = 2$ for temperature and humidity fields. This strategy avoids floating-point emulation in Solidity, keeps contract logic deterministic, and preserves reversible scaling for audit reconstruction.

4 Experimental Setup and Methodology

To evaluate the proposed middleware, we executed a real-world deployment focused on reliability, end-to-end latency, and computational overhead under periodic operation.

4.1 IoT Testbed and Deployment Context

The sensing layer consisted of one Netatmo main station and three satellite modules, producing a total of 12 telemetry streams after fan-out processing in n8n. The main station was connected through Wi-Fi and satellite modules communicated with the gateway using the vendor radio link. Sensor updates were polled every 10 min, aligned with the provider refresh cycle.

For reproducibility, the deployment context is made explicit. The test was executed at the Escuela Superior de Ingenieria of the University of Cadiz, located in Puerto Real (Cadiz, Spain). The observation window covered the period from 25 January 2026 to 8 February 2026. Network behavior was not controlled through bandwidth throttling and the workflow operated over the institution's standard Internet access profile.

4.2 Execution Infrastructure

The orchestration service was deployed as a self-hosted n8n instance in Docker over a Raspberry Pi 3 host with 128 GB microSD storage. At container level, no explicit CPU or memory caps were enforced; all middleware components were deployed in Docker with default resource scheduling. The blockchain endpoint was an authenticated Alastria Red-T RPC service using a dedicated externally owned account for transaction submission.

4.3 Security Model and Operational Controls

Security analysis is structured using STRIDE categories at architecture level. Spoofing and tampering risks are mitigated by authenticated API access, local private-key signing, and immutable on-chain event anchoring. Repudiation risk is reduced through transaction hashes and indexed events. Information-disclosure risk is mitigated by keeping signing keys outside workflow JSON and injecting secrets through runtime environment variables. Denial-of-service and elevation-of-privilege risks remain partially open because this study does not include adversarial load testing or container-breakout campaigns.

The main container-security concern is potential Docker escape affecting key material or workflow integrity. The deployment therefore assumes host hardening, least-privilege container configuration, restricted network exposure, and periodic image patching. A production-grade key-rotation mechanism should be implemented through external secret managers and scheduled key replacement procedures; in this study, key lifecycle management followed controlled manual rotation outside the workflow payload.

4.4 Workflow Execution Pipeline

Fig. 2 shows the implemented n8n workflow used during the evaluation campaign.

Each scheduled run executed the following deterministic sequence: (i) trigger activation, (ii) high-precision start timestamp capture, (iii) Netatmo data ingestion, (iv) payload normalization and fixed-point conversion, (v) smart-contract transaction signing and submission, (vi) receipt-based metric extraction, and (vii) persistence of operational logs. This sequence was intentionally linear to simplify traceability and avoid hidden branching effects in latency measurements.

At implementation level, the normalization stage performs module fan-out, null filtering for temporarily unreachable devices, and canonical field mapping before transaction construction. This design avoids mixing transport-specific parsing logic with contract-specific encoding logic and simplifies failure diagnosis in execution logs.

Figure 2: Implemented orchestration workflow in n8n.

4.5 Evaluation Metrics and Symbol Definitions

End-to-end latency is defined in Eq. (2):

$$L_{E2E} = T_{conf} - T_{start} \tag{2}$$

where T_{start} is the timestamp captured at workflow initialization and T_{conf} is the timestamp when transaction confirmation is received.

Reliability rate is defined in Eq. (3):

$$R = \frac{Tx_{success}}{Tx_{total}} \times 100 \tag{3}$$

where $Tx_{success}$ denotes the number of confirmed transactions and Tx_{total} denotes the total number of attempted submissions in the observation window.

Gas usage was extracted from each transaction receipt as an implementation-level proxy of computational cost at contract execution time. Although Alastria does not impose public-network fees, gas remains informative for relative cost analysis across initialization and steady-state phases.

4.6 Scope of Reliability Assessment

The reliability analysis is explicitly scoped to observed operational conditions. It does not claim adversarial robustness because no dedicated fault-injection campaign (e.g., forced RPC failures or byzantine message perturbation) was executed. To avoid overstatement, reliability findings are interpreted as empirical operational stability under the tested conditions, and adversarial validation is reported as future work.

4.7 Open Reproducibility Artifacts

To support open-science reproducibility, the full artifact package (smart contract, n8n workflow export, workflow figure, deployment files, and execution dataset) has been released in a public Zenodo record [30]. In addition, the custom n8n nodes used in this study are publicly available as open-source repositories for direct reuse and inspection [28,29].

5 Results and Discussion

The evaluation dataset contains consecutive workflow executions under full sensor fan-out, enabling the analysis of latency behavior, gas profile, and operational reliability.

5.1 Latency Behavior under Continuous Operation

Fig. 3 shows the end-to-end latency for each execution cycle.

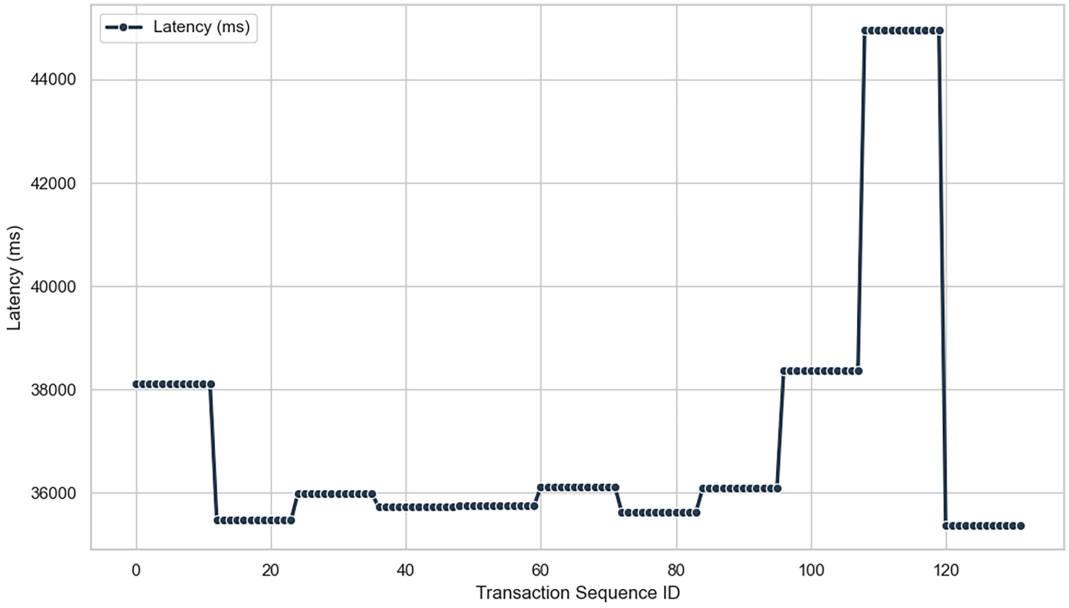

Figure 3: End-to-end latency over execution cycles.

The observed mean latency was 37,422 ms (approximately 37.4 s), with values concentrated between 35 and 44 s. The time series does not show progressive drift across cycles, which indicates stable orchestration behavior under the tested load. The dominant delay component is blockchain confirmation time rather than local workflow processing, as already suggested by the near-constant processing profile in n8n execution logs.

To improve statistical clarity, this study distinguishes descriptive claims from inferential claims. The current manuscript reports observed central tendency from the available run logs. Standard deviation and confidence intervals should be included once the complete raw log export is consolidated as a tabular annex (fields: cycle ID, module ID, L_{E2E}, gas, and transaction outcome).

The released open dataset enables calculation of additional robustness indicators (standard deviation, percentile metrics such as P95/P99, and boxplot-based outlier diagnostics) directly from per-transaction records [30]. This manuscript therefore reports validated descriptive findings while providing the underlying evidence for expanded statistical inspection.

5.2 Computational Cost Profile

Fig. 4 reports average gas consumption per sensor across cycles.

The first cycle shows an average near 126,846 gas due to storage-slot initialization. From the second cycle onward, values stabilize around 52,846 gas, which is consistent with update operations over already-initialized storage. This shift implies a 58.3% reduction from cold-start to steady-state execution and supports the practical viability of periodic auditing workloads in permissioned networks.

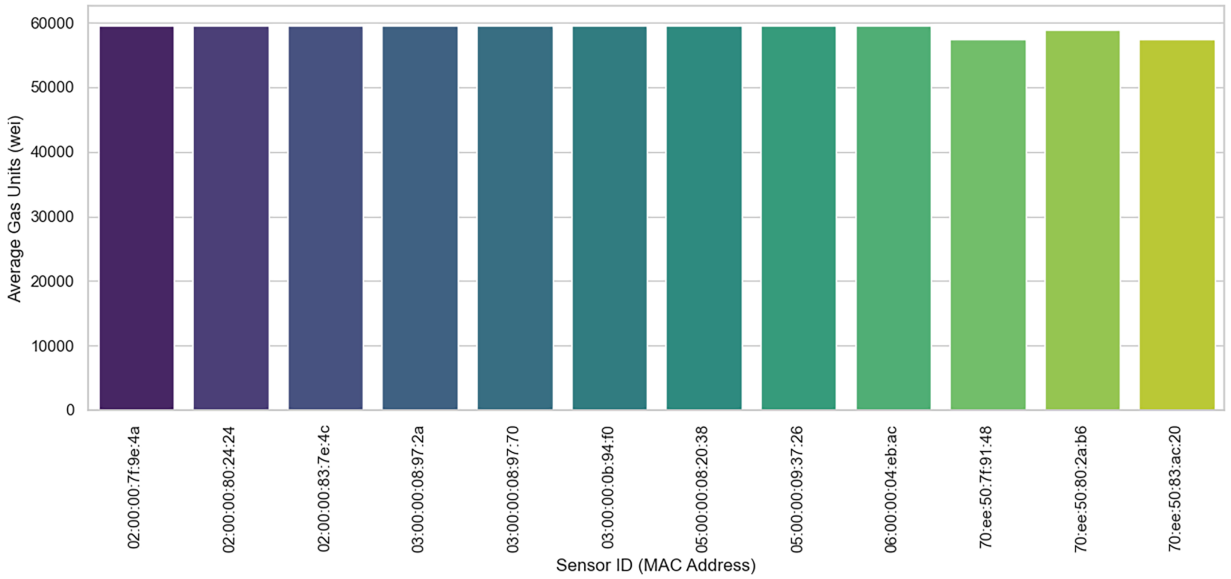

Figure 4: Average computational overhead (Gas Used) per sensor.

Per-sensor gas differences are explained by Ethereum Virtual Machine (EVM) storage-state transitions rather than by large payload-shape changes, because the function signature and field widths remain fixed across devices. In particular, first-write updates and non-zero state transitions can cause localized variability by device identifier. For this reason, deterministic cost should be interpreted as bounded operational behavior with predictable phase structure (initialization vs. steady update), not as identical gas for every transaction hash. Given the fixed payload schema of `recordMetric` (constant number and type of arguments), payload-size vs. gas correlation is not expected to be a dominant explanatory factor in this experiment; storage-state transitions are the primary driver.

5.3 Reliability Interpretation and Validity Boundaries

All recorded submissions in the observation window returned confirmed receipts, yielding an observed reliability of 100% according to Eq. (3). The claim is intentionally bounded to non-adversarial operational conditions. No targeted fault injection, RPC outage emulation, or malicious input campaign was executed; therefore, adversarial robustness is outside the empirical scope of this study.

This clarification addresses an important validity concern: operational success under stable infrastructure does not imply guaranteed behavior under hostile or degraded network states. The manuscript now separates these two interpretations explicitly.

5.4 Comparison with Existing Approaches

A direct numerical benchmark against prior studies is difficult because published works use heterogeneous datasets, blockchain stacks, and execution conditions. To still support comparative interpretation, we provide a qualitative architecture-level contrast aligned with RQ3.

Compared with centralized-oracle designs, the proposed middleware reduces trust concentration by signing transactions locally and writing directly to the consortium chain. Compared with monolithic custom middleware, it improves reconfigurability through visual workflow composition and reusable nodes. Compared with generic low-code orchestration without blockchain-aware extensions, it adds deterministic payload preparation and contract-native execution semantics.

The trade-off is that quantitative superiority over all state-of-the-art systems cannot be claimed from the current experimental scope. This manuscript therefore positions its contribution as an implementation and evaluation study that demonstrates feasibility, operational stability, and clear architectural advantages under documented conditions.

5.5 Latency Decomposition and Baseline Scope

The end-to-end path can be decomposed as:

$$L_{E2E} = L_{ingest} + L_{normalize} + L_{sign} + L_{rpc} + L_{consensus} \tag{4}$$

where middleware-local components correspond to ingestion, normalization, and signing, while network and chain components correspond to RPC transit and consensus finality. The current campaign did not include a dedicated mock-RPC or local-dev-chain control baseline; therefore, precise numeric isolation of middleware-only overhead is left for future controlled experiments. Even under this limitation, execution logs indicate that consensus confirmation dominates the observed latency envelope.

5.6 Answers to Research Questions

RQ1 (secure and deterministic low-code settlement): The implementation and execution results support an affirmative answer under the evaluated conditions. The custom n8n nodes enabled authenticated ingestion, deterministic payload normalization, local transaction signing, and direct smart-contract settlement without relying on an external third-party oracle service.

RQ2 (latency and computational-cost behavior): The observed average end-to-end latency was approximately 37.4 s, with bounded variation across cycles, and gas consumption showed the expected cold-start to steady-state transition (from approximately 126,846 to 52,846 gas). These findings indicate stable operational behavior for periodic telemetry anchoring in the tested consortium setting.

RQ3 (architectural trade-offs vs. conventional paradigms): The evidence suggests that the proposed architecture improves modularity and reconfigurability compared with monolithic middleware while reducing trust concentration compared with centralized oracle gateways. The main limitation is that broad quantitative superiority over heterogeneous state-of-the-art implementations cannot be established from the current experimental scope and should be addressed in future controlled benchmarking studies.

6 Conclusions and Future Work

This paper presented a low-code orchestration middleware for secure and transparent IoT–Blockchain integration and evaluated it in a real deployment using Netatmo sensors and the Alastria consortium network. The results support three main conclusions: first, a workflow-centric platform can be extended with custom modules to perform authenticated ingestion, deterministic normalization, and local smart-contract transaction signing without relying on external oracle operators; second, under the tested operational conditions, the architecture maintained stable end-to-end latency while handling multi-sensor fan-out and preserved complete transaction confirmation in the observed run window; and third, the gas profile shows the expected transition from higher initialization cost to lower steady-state update cost, which is relevant for continuous auditing scenarios. Accordingly, the manuscript provides explicit answers to RQ1–RQ3 in Section 5, linking design decisions and empirical observations with the stated research objectives. At the same time, the study explicitly recognizes scope boundaries: the current evidence is based on a limited number of devices and non-adversarial execution conditions, so broad claims about statistical generalization or attack-resilient reliability are not made. Future work will therefore prioritize larger-scale

experiments, formal statistical reporting from expanded logs, controlled fault-injection campaigns, and protocol extensions toward additional industrial interfaces and privacy-preserving verification mechanisms.

Acknowledgement: Not applicable.

Funding Statement: This publication is part of the I+D+i grant PID2021-122215NB-C33 funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU.

Availability of Data and Materials: The reproducibility package, including the Solidity smart contract, n8n workflow export, workflow image, and Docker Compose deployment files, is publicly available in Zenodo [30]. The custom n8n nodes used in this work are also openly available in GitHub repositories [28,29].

Ethics Approval: Not applicable.

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

References

1. Reyna A, Martín C, Chen J, Soler E, Díaz M. On blockchain and its integration with IoT. Challenges and opportunities. Future Generat Comput Syst. 2018;88(3):173–90. doi:10.1016/j.future.2018.05.046.
2. Panarello A, Tapas N, Merlino G, Longo F, Puliafito A. Blockchain and IoT integration: a systematic survey. Sensors. 2018;18(8):2572. doi:10.3390/s18082572.
3. Lu Y. Implementing blockchain in information systems: a review. Enterp Inform Syst. 2022;16(12):2008513. doi:10.1080/17517575.2021.2008513.
4. Al Breiki H, Al Qassem L, Salah K, Habib Ur Rehman M, Sevtinovic D. Decentralized access control for IoT data using blockchain and trusted oracles. In: Proceedings of the 2019 IEEE International Conference on Industrial Internet (ICII); 2019 Nov 11–12; Orlando, FL, USA. p. 248–57. doi:10.1109/ICII.2019.00051.
5. Lo SK, Xu X, Staples M, Yao L. Reliability analysis for blockchain oracles. Comput Elect Eng. 2020;83(4):106582. doi:10.1016/j.compeleceng.2020.106582.
6. Khan MA, Salah K. IoT security: review, blockchain solutions, and open challenges. Future Generat Comput Syst. 2018;82:395–411. doi:10.1016/j.future.2017.11.022.
7. Dai HN, Zheng Z, Zhang Y. Blockchain for internet of things: a survey. IEEE Internet Things J. 2019;6(5):8076–94. doi:10.1109/JIOT.2019.2920987.
8. Al-Breiki H, Rehman MHU, Salah K, Svetinovic D. Trustworthy blockchain oracles: review, comparison, and challenges. IEEE Access. 2020;8:85675–85. doi:10.1109/ACCESS.2020.2992698.
9. Caldarelli G. Overview of blockchain oracle research. Fut Inter. 2022;14(6):175. doi:10.3390/fi14060175.
10. Salagrama S, Bibhu V, Rana A. Blockchain based data integrity security management. Procedia Comput Sci. 2022;215(1):331–9. doi:10.1016/j.procs.2022.12.035.
11. Shen Y, Xu C, An B, Li A, Lin X. Blockchain-based distributed multi-agent reinforcement learning for collaborative multi-object tracking framework. IEEE Trans Comput. 2023;73(3):778–88.
12. Zhang Y, Cao W, Zhang S, Wang H, Duong TQ, Zhao Z, et al. Enhancing physical-layer security for IoT with nonorthogonal multiple access assisted semi-grant-free transmission. IEEE Internet Things J. 2023;10(4):2850–63. doi:10.1109/JIOT.2022.3193189.
13. Christidis K, Devetsikiotis M. Blockchains and smart contracts for the internet of things. IEEE Access. 2016;4:2292–303. doi:10.1109/ACCESS.2016.2566339.
14. Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: vision and challenges. IEEE Internet Things J. 2016;3(5):637–46. doi:10.1109/JIOT.2016.2579198.
15. Xu LD, He W, Li S. Internet of Things in industry: a survey. IEEE Trans Indust Inform. 2014;10(4):2233–43. doi:10.1109/TII.2014.2300753.
16. Vukolić M. The quest for scalable blockchain fabric: proof-of-work vs. BFT replication. Open Prob Netw Secur. 2015:112–25. doi:10.1007/978-3-319-39028-4_9.

17. Baliga A, Subhod I, Kamat P, Chatterjee S. Performance evaluation of the quorum blockchain platform. arXiv:1809.03421. 2018.
18. Mazzoni M, Nicola V. Performance evaluation of permissioned blockchains for financial applications: the ConsenSys Quorum case study. Blockch Res Appl. 2021;3(1):100026. doi:10.1016/j.bcra.2021.100026.
19. Gómez A, Pérez-Solà C, Herrera-Joancomartí J. Alastria identity: self-sovereign identity on a permissioned blockchain. In: Proceedings of the 2021 IEEE International Conference on Blockchain (Blockchain); 2021 Dec 6–8; Melbourne, Australia. p. 482–9. doi:10.1109/Blockchain53845.2021.00073.
20. Casino F, Dasaklis TK, Patsakis C. A systematic literature review of blockchain-based applications: current status, classification and open issues. Telem Inform. 2019;36:55–81. doi:10.1016/j.tele.2018.11.006.
21. Eberhardt J, Tai S. On or off the blockchain? Insights on off-chaining computation and data. In: European Conference on Service-Oriented and Cloud Computing. Cham, Switzerland: Springer; 2017. p. 3–15. doi:10.1007/978-3-319-67262-5_1.
22. Fernández-Caramés TM, Fraga-Lamas P. A review on the use of blockchain for the Internet of Things. IEEE Access. 2018;6:32979–3001. doi:10.1109/ACCESS.2018.2842685.
23. Sanchis R, García-Perales O, Fraile F, Poler R. Low-code as an enabler of digital transformation in manufacturing industry. Appl Sci. 2020;10(1):12. doi:10.3390/app10010012.
24. Waszkowski R. Low-code platform for automating business processes in manufacturing. IFAC-PapersOnLine. 2019;52(10):376–81. doi:10.1016/j.ifacol.2019.10.060.
25. Blackstock M, Lea R. IoT interoperability: a hub-based approach. In: Proceedings of the 2014 International Conference on the Internet of Things (IOT); 2014 Oct 6–8; Cambridge, MA, USA. p. 79–84. doi:10.1109/IOT.2014.7030119.
26. Philippe J, Coullon H, Tisi M, Sunyé G. Towards transparent combination of model management execution strategies for low-code development platforms. In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings; 2020 Oct 16–23; Virtual. New York, NY, USA: Association for Computing Machinery; 2020. p. 1–10. doi:10.1145/3417990.3420206.
27. Ciccozzi F, Spalazzese R. MDE4IoT: supporting the internet of things with model-driven engineering. In: Badica C, El Fallah Seghrouchni A, Beynier A, Camacho D, Herpson C, Hindriks K, et al., editors. Intelligent distributed computing X. Cham, Switzerland: Springer International Publishing; 2017. p. 67–76. doi:10.1007/978-3-319-48829-5_7.
28. Rosa-Bilbao J. n8n-nodes-netatmo-weather-station; 2026. Custom n8n node for Netatmo telemetry ingestion. GitHub repository. [cited 2026 Apr 2]. Available from: https://github.com/JesusRosaB/n8n-nodes-netatmo-weather-station.
29. Rosa-Bilbao J. n8n-nodes-smart-contract; 2026. Custom n8n node for smart-contract interaction and raw transaction signing. GitHub repository. [cited 2026 Apr 2]. Available from: https://github.com/jesusrosab/n8n-nodes-smart-contract.
30. Rosa-Bilbao J. Dataset for a low-code orchestration middleware for secure and transparent IoT–blockchain integration. Zenodo. 2026. doi:10.5281/zenodo.18888366.