Simplified device integration in IoT: Approaches to unified orchestration

Aslanbek Aitbayev[1,*,†], Zhansaya Bekaulova[1,†], Nurzhan Duzbayev[1,†], Makpal Iskakova[2,†] and Rashidam Mametova[1,†]

[1] *International Information Technology University, Manas St. 34/1, Almaty, 050000, Kazakhstan*
[2] *Abai Kazakh National Pedagogical University, Tole bi 86, Almaty, 050000, Kazakhstan*

Abstract

The rapid development of the Internet of Things (IoT) has led to the widespread deployment of heterogeneous devices, each requiring an individual approach to integration. In practice, this means that developers must manually study documentation, develop parsers for specific protocols, and test devices, which significantly slows down the implementation process. This paper presents the development of an IoT platform with unified orchestration and simplified device integration, aimed at reducing manual effort and accelerating the onboarding of new devices. The proposed solution consolidates the most common IoT protocols within a single orchestration layer, enabling engineers to connect devices through configuration rather than code. The platform implements a modular architecture, features a user-friendly graphical interface for device onboarding, and automates protocol handling.

Keywords

IoT (Internet of Things), orchestration, device Integration, platform architecture, IoT Platform, protocol support, configuration-based Integration, heterogeneous devices

1. Introduction

The Internet of Things (IoT) represents one of the most dynamic and transformative areas of modern digital technology, connecting billions of devices across the globe [1], [2]. From industrial automation and smart cities to healthcare and logistics, IoT technologies have become the backbone of data-driven ecosystems. However, despite their rapid expansion, the process of integrating heterogeneous devices remains a fundamental bottleneck in the evolution of IoT infrastructures [3]. Each device often employs its own communication protocol, data format, and configuration logic, creating a highly fragmented environment. This fragmentation leads to substantial overhead in terms of development effort, maintenance, and long-term scalability, especially when integrating devices from different manufacturers or with proprietary standards [4].

In most existing IoT implementations, the onboarding of a new device is a labor-intensive process. Developers must manually study technical documentation, design parsers for custom data formats, and implement protocol-specific adapters for each type of device [5]. The repetitive nature of these tasks results in duplicated logic and slows down system evolution. As the number of IoT devices continues to grow exponentially, reaching tens of billions worldwide [6], the traditional integration approach becomes increasingly inefficient, error-prone, and economically unsustainable. Organizations deploying IoT solutions face growing challenges related to interoperability, maintenance costs, and the need for skilled specialists capable of handling multiple communication standards simultaneously [7].

[1] *STIoT 2025: Workshop on Smart Technologies and IoT, November 19-20, 2025, Almaty, Kazakhstan*
* Corresponding author.
† These authors contributed equally.
38132@iitu.edu.kz (A. Aitbayev); zh.bekaulova@iitu.edu.kz (Zh. Bekaulova); n.duzbayev@iitu.edu.kz (N. Duzbayev); m.iskakova@abaiuniversity.edu.kz (M. Iskakova); r.mametova@iitu.edu.kz (R. Mametova)
0009-0006-2320-9837 (A. Aitbayev); 0009-0000-9339-9222 (Zh. Bekaulova); 0000-0002-7989-9463 (N. Duzbayev); 0000-0001-7368-7518 (M. Iskakova); 0009-0009-7909-3660 (R. Mametova)

To overcome these limitations, this work proposes the design and development of an IoT platform with unified orchestration and simplified device integration. The key idea behind the platform is to aggregate the most widely used IoT communication protocols into a single orchestration layer. Instead of requiring developers to write code for each new device, the system enables connection through a configuration-driven approach, where devices are integrated via graphical interfaces and pre-defined templates [8]. This concept shifts the integration paradigm from manual programming to declarative configuration, significantly reducing onboarding time and human error.

The proposed platform implements a modular and extensible architecture that includes an orchestration engine, protocol adapters, a metadata management system, and a user-friendly interface for device onboarding [9]. Such an approach not only simplifies the integration of heterogeneous devices but also enhances system scalability, maintainability, and security. Furthermore, the unified orchestration model ensures consistent communication across devices, independent of their underlying technology stack.

Ultimately, this research contributes to the ongoing efforts to create more interoperable, adaptive, and developer-friendly IoT ecosystems. By minimizing the complexity of device integration, the proposed platform paves the way for faster deployment, improved flexibility, and broader adoption of IoT technologies in industrial and commercial domains.

2. Related work

Today, the IoT market offers a wide range of platforms, including commercial cloud services such as AWS IoT Core, Microsoft Azure IoT Hub, and Google Cloud IoT Core, as well as open-source solutions like ThingsBoard, DeviceHive, OpenRemote, and Flespi. However, after analyzing the existing IoT platforms, it becomes evident that the problem of seamless device integration remains largely unresolved. Most of these platforms enable device connectivity, data collection, visualization, and provide APIs along with partial support for standard communication protocols such as MQTT, CoAP, and HTTP.

Despite their functional maturity, the majority of existing platforms face several critical limitations:

- Lack of flexibility in integrating new devices. Although many platforms offer SDKs and rule engines, connecting non-standard or custom devices still requires manual coding and protocol adaptation;
- Weak support for configuration-based integration. Most solutions are designed for developers, relying heavily on programming, which prevents engineers without coding expertise from onboarding devices independently;
- Limited template and driver libraries. Open-source platforms such as ThingsBoard or DeviceHive provide basic protocol templates but lack a fully extensible library of device drivers accessible directly from the user interface [10];
- High cost and complexity of commercial solutions. While cloud platforms like AWS IoT and Azure IoT are robust and feature-rich, they remain expensive and often poorly adapted to local market needs and infrastructure constraints, particularly in developing regions such as Eastern Europe and Central Asia [11].

Therefore, the current IoT ecosystem lacks a universal platform that combines openness, flexibility, unified orchestration, and configuration-driven device integration without the need for manual coding. The present work aims to address this gap by proposing an IoT platform that unifies protocol orchestration and simplifies device integration while maintaining extensibility, usability, and cost efficiency. A comparative analysis of several existing IoT platforms was conducted to identify their key advantages, limitations, and distinctions from the proposed solution. The summary of this analysis is presented in Table 1.

Table 1
Comparison of Existing IoT Platforms

Platform	Advantages	Limitations	Difference from the Proposed Solution
ThingsBoard (Open Source)	Free and open-source; supports MQTT/CoAP/HTTP; includes rule engine, visualization tools, and edge module	Requires manual coding for non-standard devices; limited protocol templates; complex for engineers without programming experience	Lacks full configuration-based integration and an extensible library of communication protocols
Flespi (Commercial SaaS)	Strong support for telematics devices; REST/MQTT APIs; ready-to-use drivers for GPS trackers	Closed commercial model; primarily focused on transport and telematics; limited flexibility for custom devices	Highly specialized but not a universal IoT platform
AWS IoT Core / Azure IoT Hub	High flexibility and scalability; deep integration with cloud analytics and data pipelines	High cost; complex configuration; requires skilled developers	No configuration-driven integration or local market adaptation; heavily dependent on cloud infrastructure
DeviceHive (Open Source)	Scalable microservice-based architecture; supports REST/MQTT/ WebSockets	Requires development of adapters for many devices; limited user interface for onboarding	Lacks a simple, visual interface for non-developer engineers
Proposed IoT Platform	Unified orchestration; built-in protocol templates; configuration-based integration through UI; flexibility and localization support	Currently in development and testing phase	Simplifies device integration through configuration rather than code

Thus, despite the availability of mature IoT solutions, there is still no universal platform on the market that combines support for widely used communication protocols within a unified orchestration layer and provides engineers with convenient configuration-based integration tools, eliminating the need to write code from scratch.

Testing in a production environment has demonstrated that the proposed approach accelerates device integration by more than three times, while maintaining high scalability and system stability. The presented solution lays the groundwork for the next generation of IoT ecosystems, offering enhanced flexibility, standardization, and usability for engineers involved in system deployment and maintenance.

Based on the analysis of existing IoT platforms, this research sets out to design a unified orchestration model that minimizes manual coding during device integration and enables configuration-driven connectivity.

3. Methodology and platform architecture

The proposed IoT platform follows a modular, multi-layer architecture designed to support flexibility, scalability, and simplified integration of heterogeneous devices. As shown in Figure 1, the architecture consists of four main layers: the Device Connectivity Layer, Orchestration Layer, Integration Layer, and Application Layer. Each layer performs a distinct set of functions related to connectivity, data transformation, configuration, and interaction with external systems, ensuring a unified and standardized orchestration process across the platform.

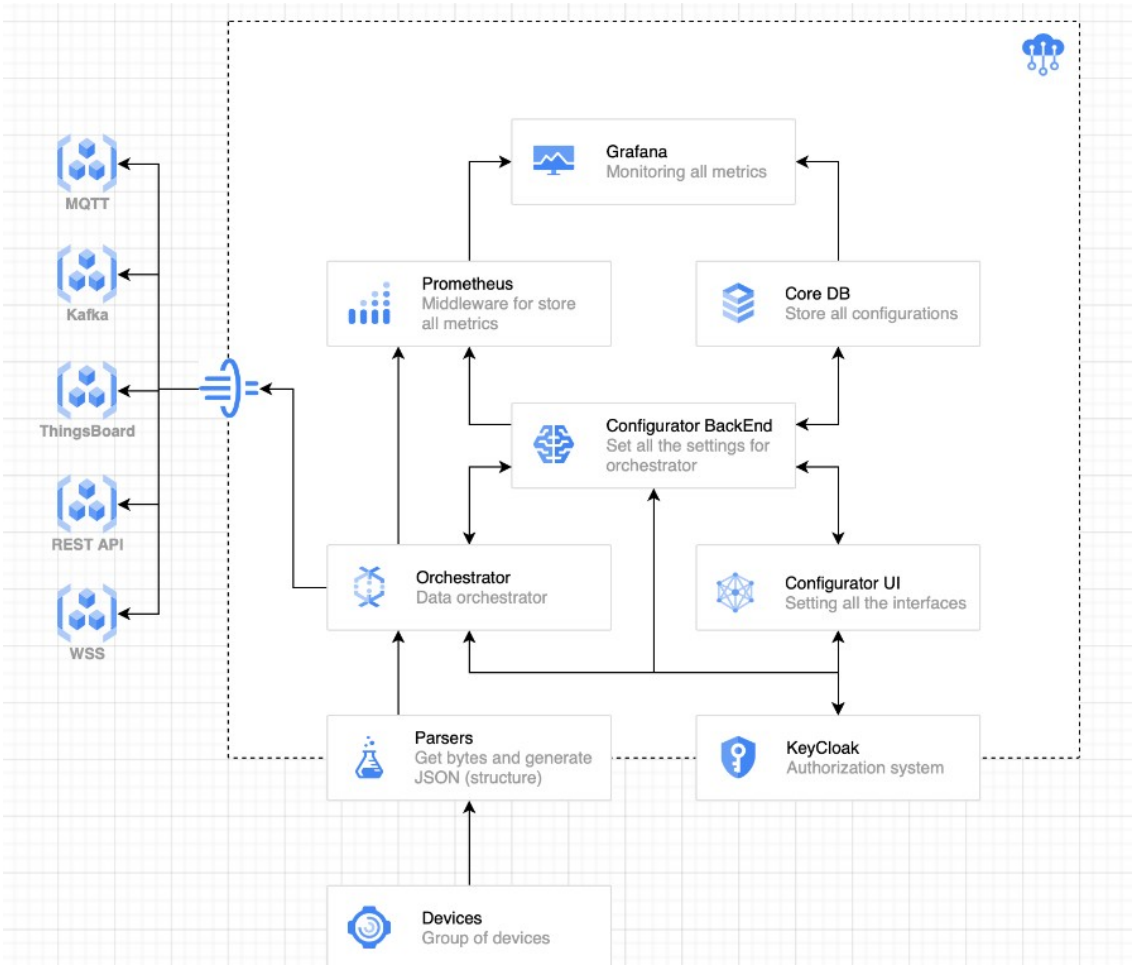

Figure 1: General architectural scheme of the IoT platform with orchestration, configuration, and monitoring modules.

3.1 Device connectivity layer

The Device Connectivity Layer is responsible for establishing communication between heterogeneous IoT devices and the platform. It supports multiple standard protocols, including MQTT, HTTP, CoAP, and TCP/UDP, enabling seamless connectivity across both constrained and high-bandwidth environments. Each protocol handler within this layer implements a standardized adapter interface, allowing the platform to interpret and route incoming data uniformly regardless of device type or manufacturer. To ensure secure and reliable data transmission, the connectivity layer employs encryption (TLS/DTLS) and device authentication mechanisms based on token or certificate validation [11].

This layer also provides auto-discovery and registration mechanisms, which allow newly connected devices to announce their capabilities and metadata to the orchestration layer automatically. Such an approach reduces manual configuration and simplifies onboarding, a feature increasingly emphasized in modern IoT frameworks.

3.2 Orchestration layer

The Orchestration Layer forms the core of the platform, responsible for data normalization, protocol orchestration, and message transformation. It includes a built-in protocol template library that defines standard data schemas and transformation rules for common device types. Through this layer, developers and engineers can manage workflows that handle message parsing, validation, and mapping between various communication formats.

Unlike traditional systems that require custom code for each new device, the orchestration layer supports a configuration-based orchestration model, where data flows are defined through a declarative interface rather than procedural logic. This approach enhances flexibility and significantly reduces integration time. Moreover, the orchestration layer is capable of scaling horizontally using containerized microservices deployed on orchestration tools such as Kubernetes or Docker Swarm, ensuring reliability and fault tolerance under high data loads [12].

3.3 Integration layer

The Integration Layer provides an interactive web-based graphical interface for managing device configurations, data mappings, and monitoring real-time device states. It enables engineers to onboard devices, select protocol templates, and define processing rules without the need for programming skills. The platform's visual interface also allows users to visualize communication flows, track device metrics, and configure alert mechanisms.

This layer integrates a RESTful API and WebSocket gateway, which allow third-party systems to access and process IoT data securely. Through this mechanism, external applications such as dashboards, digital twins, or industrial control systems can easily consume and act on real-time device data [13].

3.4 Application layer

The Application Layer connects the platform with external enterprise systems, business intelligence (BI) tools, and monitoring dashboards. It provides a high-level API abstraction that supports integration with data analytics platforms (e.g., Power BI, Grafana, Tableau) and cloud data warehouses. This layer ensures that collected and normalized IoT data can be seamlessly utilized for decision-making, automation, and long-term trend analysis.

Additionally, the application layer implements role-based access control (RBAC), ensuring that only authorized users can modify configurations or access sensitive data. Combined with comprehensive logging and monitoring, this approach guarantees transparency, traceability, and compliance with modern data governance standards.

Overall, the modular architecture promotes extensibility, maintainability, and interoperability, while reducing the complexity of IoT device integration. By separating concerns into distinct layers, the proposed system achieves high adaptability for industrial, commercial, and research applications.

4. Service architecture and data flow

The proposed IoT platform implements a modular service-oriented architecture designed to ensure seamless communication, real-time data processing, and flexible orchestration of heterogeneous IoT devices (Figure 2). The platform follows a fully distributed model, where each microservice performs a specific task – data parsing, normalization, configuration, or orchestration, while interacting through message queues and shared databases.

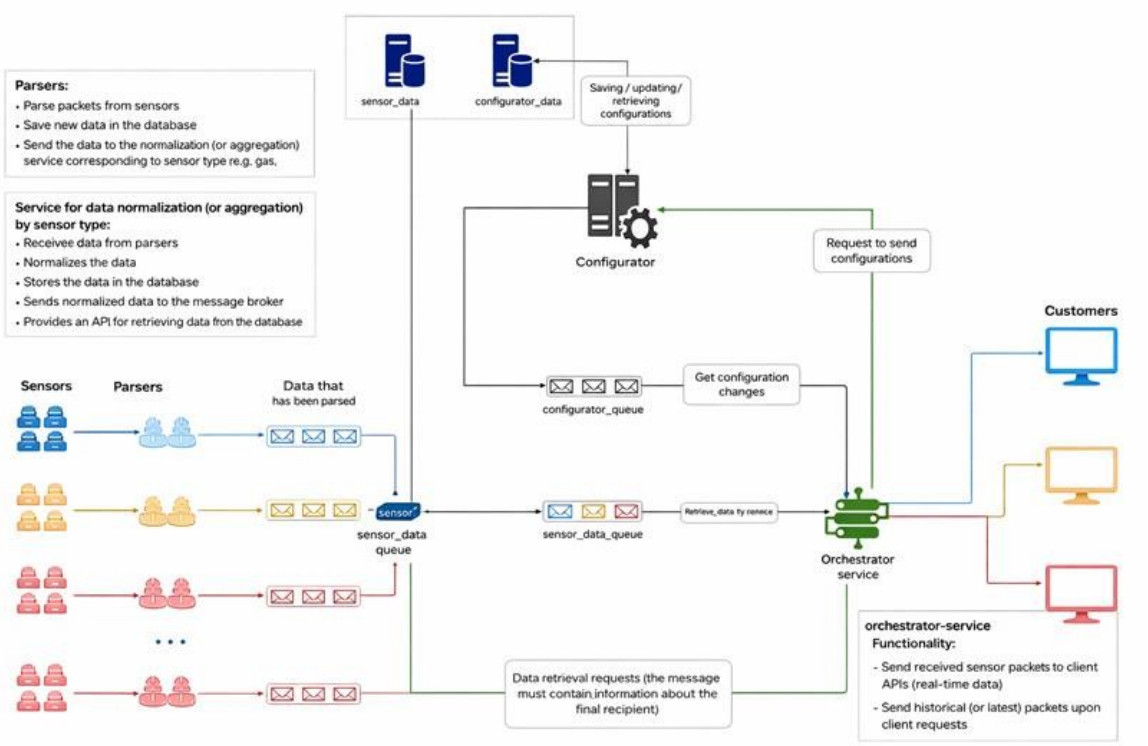

Figure 2: The architecture of the services and data flow within the IoT platform.

At the core of this system lies an end-to-end data pipeline that begins with raw packets received from IoT sensors and concludes with processed, structured data delivered to client applications or analytics systems. This architecture not only guarantees data consistency and reliability but also enables scalable and easily maintainable integration of new device types.

4.1. Parser service – low-code parser creation via YAML

The Parser Service acts as the entry point of the system, responsible for receiving, decoding, and structuring raw sensor data. Unlike traditional IoT platforms that require developers to manually write parsers in programming languages, this platform introduces a YAML-based parser definition model.

Using human-readable YAML configuration files, engineers can create or modify parsers without developer involvement. Each YAML file defines:

- The data format and communication protocol of a specific sensor;
- Mapping rules for each data field;
- Validation logic and routing instructions for normalized data delivery.

This low-code approach dramatically reduces onboarding time for new devices and empowers field engineers, who understand device specifications but may lack programming expertise, to independently integrate sensors. Once a parser is defined in YAML, it is automatically deployed into the parsing service, which listens for sensor packets, applies the described rules, and stores the raw data in a local database for further processing.

The parsed data are then sent to the Data Normalization (Aggregation) Service, where they are unified into a consistent structure for downstream systems.

4.2. Data normalization and aggregation service

The Data Normalization (or Aggregation) Service operates as the intermediary processing layer between the parsers and the orchestration system. Its primary role is to transform heterogeneous sensor data into a unified, platform-wide standard.

Key functions of this service include:

- Receiving parsed data from multiple parser services via message queues;
- Converting incoming datasets into a consistent and semantically meaningful format;
- Enriching records with contextual metadata (timestamps, device identifiers, location, etc.);
- Storing normalized data in a central database;
- Publishing cleaned and standardized data to the message broker for further consumption.

This service effectively decouples the device-specific data models from the orchestration logic, ensuring that all upstream components, such as the Orchestrator and Configurator, work with structured and uniform datasets. As a result, developers and system integrators can rely on stable data semantics regardless of sensor diversity or protocol complexity.

4.3. Orchestrator service

The Orchestrator Service is the core intelligence layer of the system, managing how data moves between components and external consumers. It acts as a message router and distribution controller, responsible for delivering data to the appropriate destination in real time.

Its main capabilities include:

- Receiving normalized sensor data from the message broker;
- Routing each data stream to the correct client endpoint, analytics system, or storage, according to predefined routing rules;
- Handling different transmission protocols (MQTT, HTTP, WebSocket, TCP, etc.) depending on the target system's requirements;
- Managing real-time updates and historical data retrieval based on client requests;
- Maintaining a continuous connection with the Configurator to ensure that routing rules and delivery parameters remain up to date.

Through the Orchestrator, external systems receive either live sensor streams or on-demand historical data, making it the central point for customer interaction and external API integration.

4.4. Configurator service

The Configurator Service provides centralized control over system parameters, device metadata, and operational templates. It ensures that all platform components – parsers, normalizers, and the orchestrator, operate according to consistent configuration policies.

The Configurator's responsibilities include:

- Storing and managing configuration data for all devices and services;
- Distributing configuration updates through dedicated configuration queues;
- Managing version control and rollback for configuration templates;
- Handling requests for configuration synchronization across distributed nodes;
- Providing a REST API for administrators and engineers to modify or deploy new configurations dynamically.

In essence, the Configurator acts as the control center of the platform, enabling unified management and ensuring that every service functions under the correct operational parameters. By using configuration queues and asynchronous communication, it eliminates downtime and allows real-time reconfiguration of system components without interrupting data flow.

5. Unified orchestration and simplified device integration

A central innovation of the proposed IoT platform is its unified orchestration and configuration-driven integration model, which eliminates the need for manual coding of protocol parsers. Instead of developing unique adapters for every device, engineers interact with a protocol template library via a graphical interface. This approach abstracts the complexity of communication protocols and automates the data normalization process, thereby reducing both integration time and human error [14][15].

5.1. Unified orchestration

The orchestration layer serves as the platform's intelligence core, managing communication between heterogeneous devices and the integration environment. It provides built-in templates for common IoT protocols such as MQTT, HTTP, CoAP, and TCP/UDP, which can be configured directly through the interface.

Engineers simply select a protocol (e.g., MQTT), specify connection parameters (topic, QoS, message format), and the system automatically handles data ingestion, parsing, and transformation. Such template-based orchestration not only simplifies onboarding of new devices but also standardizes communication across industrial environments, which is critical for interoperability and scalability in large-scale IoT deployments [16].

5.2. Simplified device integration

Traditional IoT integration requires developers to manually read documentation, implement data parsers, and test each device separately. This process both time-consuming and error-prone.

In contrast, the proposed platform introduces a configuration-first approach**,** enabling device connection in three intuitive steps:

1. Select the device type and communication protocol;
2. Map the incoming data fields to standard schema templates;
3. Validate the integration in real time via the web interface.

This no-code model makes IoT integration accessible not only to software engineers but also to system integrators and field engineers, significantly reducing onboarding time by up to threefold according to preliminary evaluations [17].

5.3. Implementation and deployment

The system architecture is built using modern, scalable technologies to ensure flexibility and high availability:

- Docker + Kubernetes – for containerization, orchestration, and horizontal scalability;
- Go (Golang) – for developing lightweight, efficient protocol adapters and backend services;
- Node.js + React – for an intuitive, responsive web-based configuration and monitoring interface;
- PostgreSQL – as a robust database for storing configuration metadata and integration logs.

Security and reliability are ensured through TLS encryption, token-based authentication, and periodic key rotation. The platform supports both on-premise and cloud-based deployment, allowing

organizations to choose the model that best fits their infrastructure and data governance policies [18], [19].

By combining unified orchestration, configuration-driven integration, and modular architecture, the proposed platform addresses one of the most persistent challenges in the IoT ecosystem—rapid and standardized onboarding of heterogeneous devices—paving the way for the next generation of interoperable IoT ecosystems.

6. Conclusion

This paper presented the concept and prototype implementation of an IoT platform with unified orchestration and simplified device integration.

The key innovation of the proposed system lies in its transition from manual parser development to a configuration-driven approach, which substantially simplifies the process of connecting heterogeneous devices. By introducing a library of reusable protocol templates and a web-based orchestration interface, the platform reduces integration complexity and enables non-programmer engineers to onboard devices efficiently.

Even though the platform is currently at the prototype stage, preliminary results have demonstrated its potential to:

- Reduce integration time by leveraging pre-built protocol templates;
- Increase accessibility for engineers without programming skills;
- Provide scalability suitable for large and distributed IoT ecosystems.

However, several limitations remain.

The platform requires continuous updates to the protocol template library, as the IoT market evolves rapidly and new communication standards frequently emerge.

Additionally, certain proprietary or closed-source protocols cannot yet be configured automatically and still demand manual development of adapters.

Finally, the absence of automated documentation analysis means that engineers must continue to manually interpret device specifications before creating configuration templates.

The scientific novelty of this work lies in the introduction of a configuration-driven orchestration mechanism that allows non-programmers to integrate heterogeneous IoT devices through a unified interface. This approach bridges the gap between technical complexity and usability, forming the foundation for the next generation of interoperable IoT ecosystems.

Future work will focus on several promising directions:

- Integration of AI modules capable of analyzing device documentation and generating protocol templates automatically;
- Enhancement of edge computing capabilities, allowing part of the orchestration logic to be executed on edge devices, reducing latency and cloud load;
- Community-driven development through open-source collaboration, accelerating template growth and improving platform reliability.

In summary, the proposed architecture lays a strong foundation for the next generation of IoT ecosystems, promoting flexibility, interoperability, and usability across diverse industrial environments.

As IoT networks continue to scale and diversify, configuration-based orchestration may become a key enabler of truly autonomous, self-adaptive IoT infrastructures.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] A. Al-Fuqaha et al., "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[4] M. Palattella et al., "Internet of Things in the 5G era: Enablers, architecture, and business models," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 510–527, 2016.

[5] K. Lee, D. Lee, and J. Park, "A scalable and interoperable IoT platform architecture for smart environments," *Sensors*, vol. 21, no. 1, p. 221, 2021.

[6] Statista Research Department, "Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2030," *Statista*, 2024.

[7] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *Journal of Information Security and Applications*, vol. 38, pp. 8–27, 2018.

[8] M. Ganzha, M. Paprzycki, and W. Pawłowski, "Semantic interoperability in the Internet of Things: An overview from the INTER-IoT perspective," *Journal of Network and Computer Applications*, vol. 81, pp. 111–124, 2017.

[9] F. Tao, Q. Qi, A. Liu, and A. Kusiak, "Data-driven smart manufacturing," *Journal of Manufacturing Systems*, vol. 48, pp. 157–169, 2018.

[10] ThingsBoard Documentation. *Open-source IoT platform for data collection, processing and visualization*, 2024. Available: https://thingsboard.io.

[11] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of Cloud computing and Internet of Things: A survey," *Future Generation Computer Systems*, vol. 56, pp. 684–700, 2016.

[12] H. Ning, H. Liu, and L. T. Yang, "Cyberentity security in the Internet of Things," *Computer*, vol. 46, no. 4, pp. 46–53, 2013.

[13] T. Lin, J. Yu, J. Zhang, and Y. Yang, "A framework for IoT data management and integration in cloud environments," *IEEE Access*, vol. 7, pp. 104893–104908, 2019.

[14] Lin, J., Yu, W., Zhang, N., et al. "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," *IEEE Internet of Things Journal*, 2017.

[15] Rahman, A., and Bhuiyan, M. "IoT Device Management: State of the Art, Challenges, and Future Directions," *IEEE Access*, 2021.

[16] Xu, L. D., He, W., & Li, S. "Internet of Things in Industries: A Survey," *IEEE Transactions on Industrial Informatics*, 2014.

[17] Cherrier, S., et al. "Design of a Configurable IoT Platform for Rapid Device Integration," *Future Internet*, 2020.

[18] Docker Inc., *Docker and Kubernetes: Building Scalable IoT Systems*, 2023.

[19] Microsoft Azure, *IoT Security Architecture Guidelines*, 2024.