Year: 2026, Volume: 1, Number: 2, Pages: 114-126

Journal of Advanced Vocational Information and Communication Technology

Homepage: https://journal.isas.or.id/index.php/javict, email: javict@isas.or.id

IoT and Cloud System Implementation Based on FIWARE for Smart Home Application

Angga Pradipta Kurnia Putra[1], Zaky Wahyu Oktavianto[1,], Reksa Prastama Putra[1]*

[1,*]Politeknik Elektronika Negeri Surabaya, Surabaya, Indonesia

*Corresponding Author: zayuto33@gmail.com

Article Info

Article History:
Submitted 14 Apr 2026
First Revised 19 May 2026
Accepted 23 May 2026
First Available Online 24 May 2026
Publication Date 25 May 2026

Keyword:
Smart Home;
Fiware;
Internet of Things;
Cloud Computing;
Data Privacy.

Abstract

Third-party smart home platforms raise significant security and privacy concerns due to their centralized hosting models. This study develops a decentralized, easily deployable smart home platform using the FIWARE IoT framework that enables users to host systems on private servers. The platform integrates FIWARE Generic Enablers (Orion, Keyrock, Wilma, IoT Agent) with custom web, mobile, and desktop applications, containerized using Docker Compose. Hardware nodes (NodeMCU ESP8266 with DHT11 sensors and relay modules) operate without a central gateway. The system successfully achieved full CRUD functionality across client applications, with automated Arduino sketch compilation and upload. However, performance testing revealed critical latency anomalies, with the Orion Context Broker exhibiting rare but severe response delays up to 10 seconds. While the platform successfully addresses privacy concerns and simplifies deployment, FIWARE's resource consumption and occasional latency spikes present optimization challenges requiring future work.

1. Introduction

Smart Home or Home Automatoin is one of the many implementations of IoT and Cloud Computing system for devices in a building or a house. Smart Home system provides the ability to control and montior those devices for user's efficiency, practicality, and comfort [1]. There are a lot of third party companies who provide easeness of Smart Home implementation. Those companies provide the devices and host the whole Smart Home

platform on their platform. This brings some security and privacy concern to some people, they wonder if their Smart Home system is spying on them. Other than that, Smart Home products are quite expensive, and to create and develop our own Smart Home system is not an easy thing to do[2], [3].

FIWARE, an open-source cloud based IoT Framework provides ready to use components that could be used freely. The architecture and implementation of FIWARE Cloud is made based on OpenStack, an open-source cloud middleware that is developed and used by a lot of people in the industry right now. FIWARE could make the development of Smart Home system easier since its components could be used and set up easily. FIWARE could manage IoT devices whethere it's an actuator or sensor, context data management, and it could also provide security on the system [1], [4].

The aim of this research is to solve that privacy and security issue on Smart Home system that is provided by third party companies, and to simplify or ease the struggle of creating a Smart Home system from scratch. The developed system could be set up and deployed by anyone easily without extensive knowledge about programming and other related stuff. The system will be consisted of a web application, mobile application, desktop application, backend server, FIWARE services, deployment script, and documentation web.

The web application will be used to manage various entities on the system such as rooms, devices, and users. The mobile application also will be used to manage rooms and devices, and it will only be developed on Android operating system. The desktop application will be used to compiling and upload sketch file of Arduino to respectable Arduino board to ease and simplify the hardware setup part of the development of a Smart Home system. The backend server is to serve various APIs with various functions and it is used to connect the client applications to the FIWARE services [5]. The FIWARE services that will be used here are Orion, Wilma, Keyrock, and IoT Agent. And lastly, the deployment script will be used to deploy the whole backend server and FIWARE services easily on any cloud or remote server just by running few commands. The documentation web is to guide users to setup the whole system from scratch on their own server.

While previous studies have successfully implemented smart home systems using various localized technologies, they often face critical limitations. For instance, Bluetooth and ZigBee-based systems generally suffer from strict spatial range limitations, whereas architectures relying on central physical gateways, such as a Raspberry Pi, inherently incur higher initial infrastructure costs. Furthermore, existing independent cloud-based solutions frequently lack standardized context data management and require complex, manual server configurations, rendering them highly inaccessible to non-technical users. To explicitly address these research gaps, this study proposes a distinct architectural approach. Unlike previous iterations, the proposed system uniquely combines the standardized NGSI-LD context data management of the FIWARE framework with a cost-effective, gateway-free ESP8266 node topology, supplemented by fully automated Docker-based deployment scripts [6]. This novel integration ensures that users can effortlessly deploy a secure, highly standardized, and private IoT ecosystem on their personal servers without requiring advanced programming expertise or expensive intermediary hardware.

2. Methods

2.1 Foundational Architectures and Design Rationale

The methodology of this research was formulated by analyzing various existing Smart Home architectures to optimize cost efficiency and system functionality [3], [7]. This study develops a decentralized system utilizing a node-based approach without relying on a physical gateway device, such as a Raspberry Pi, to significantly reduce development and operational costs [8], [9]. This architectural decision builds upon the research conducted by Suri et al. [7], which integrated and automated household electronic devices based on environmental variables without human intervention. The visual representation of this foundational node-based design is illustrated in Figure 1. In addition to cost efficiency, the security aspect of the proposed method refers to the study by Dantas et al. [10], which implemented a Smart Home system utilizing the WAMP protocol, WebSockets, and JSON data serialization. This approach enables robust security with low operational costs, as the system does not require dedicated public IP addresses or high-end servers, as depicted in Figure 2.

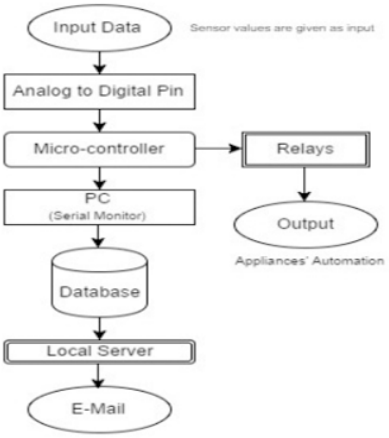

Figure 1. Node-based system design from related research [6]

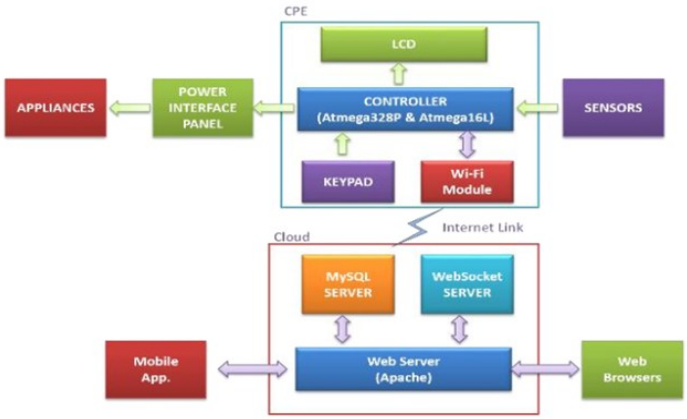

Figure 2. Web security protocol system design from related research [1]

The integration of the FIWARE IoT framework in this study is heavily influenced by its successful implementation in an e-Health Cloud system developed by Celesti et al. [11]. The utilization of FIWARE has been proven to simplify the development process, lower maintenance costs, and enable the deployment of the entire system on an independent server to guarantee data security within a localized environment, as shown in Figure 3.

DOI: *https://doi.org/10.52158/javict.v1i2.1533*
ISSN: *3124-0437 (media online)*

Furthermore, the environmental monitoring methodology incorporates principles from Wireless Sensor Network (WSN) technology to detect various hazards, including fire risks, temperature fluctuations, and carbon monoxide concentrations, a topology model visualized in Figure 4. Finally, the control structure concept was adapted from Prasetiyo's model, which utilized Arduino as nodes managed by a gateway controller to handle API requests, scheduling, and offline modes Figure 5. By synthesizing these prior methodologies, the current research proposes an optimized, gateway-free FIWARE integration.

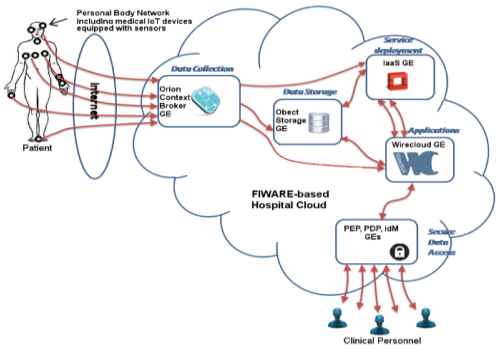

Figure 3. FIWARE-based system design from related research [4]

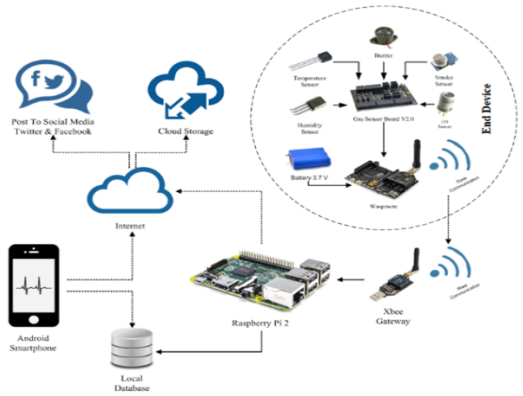

Figure 4. Wireless Sensor Network-based system design from related research

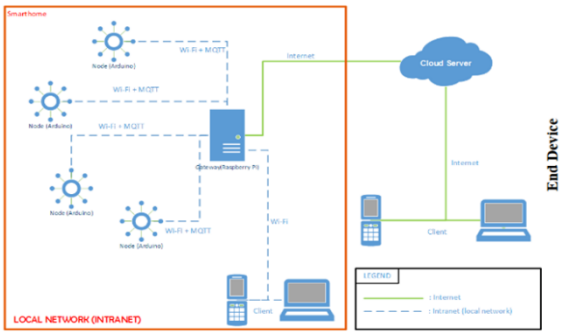

Figure 5. Gateway-based control system design from related research

2.2 Proposed System Architecture and FIWARE Integration

Building upon the established rationale, this research designs a comprehensive smart home platform that facilitates independent and automated system deployment on a user's private server. The system architecture adopts the FIWARE framework to autonomously manage database configurations, context data storage logic, and system authentication

flows through its Generic Enablers (GEs). The overall architecture comprises a cloud-based backend service ecosystem, client applications (web and mobile), a desktop application for hardware initialization, and automated deployment scripts. The architectural design of the proposed system is presented in Figure 6.

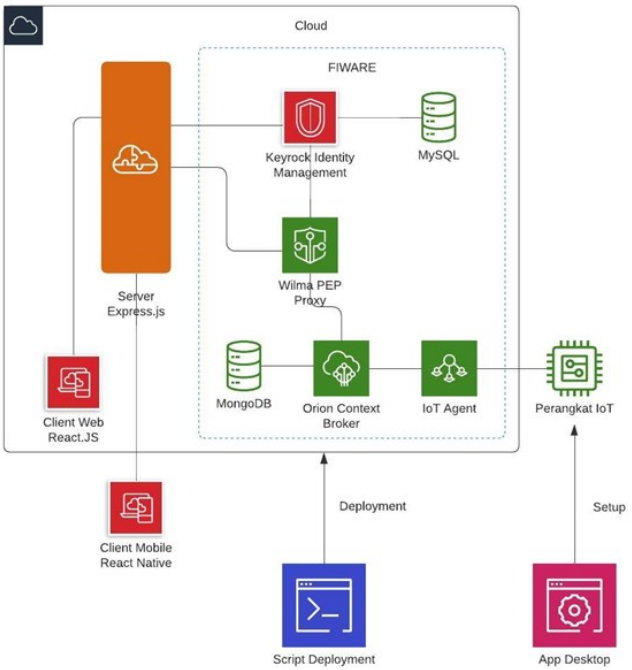

Figure 6. System architecture of the proposed FIWARE-based smart home platform

The core cloud infrastructure is built upon four primary FIWARE services. The Orion Context Broker functions as the central management hub to process storage, updates, and queries for system entities using a MongoDB database. Keyrock manages identity and user authentication using a MySQL database, while the Wilma PEP Proxy secures the ecosystem through token authentication and API access authorization. Lastly, the IoT Agent bridges data communication between Orion and the physical devices. These services are interconnected by a Node.js and Express.js backend server, utilizing Socket.io for real-time device status synchronization. All software modules are containerized using Docker Compose to ensure standardized deployment.

2.3 Client Software Development and Automation

The client interface is divided into three primary modules. The first is a web application developed using React.js with Redux state management to facilitate the management of room entities and hardware control. The second is a React Native mobile application providing equivalent monitoring and control functions optimized for Android environments. The third module is an Electron-based desktop application for Windows, designed to automate hardware setup. It runs the Arduino CLI in the background to compile and upload code sketches directly to the microcontroller memory. Furthermore, a command-line interface (CLI) deployment script is provided to automate the server dependency installation, ensuring ease of initialization for users.

DOI: *https://doi.org/10.52158/javict.v1i2.1533*
ISSN: *3124-0437 (media online)*

2.4 Hardware Topology Design

The hardware design implements the aforementioned decentralized node topology without a central gateway. The end-devices designed for this simulation include an environmental temperature sensor, a lighting actuator, and an electronic solenoid door lock. All devices are directly integrated and controlled using NodeMCU ESP8266 development boards equipped with built-in Wi-Fi modules. A DHT11 sensor is utilized for precise temperature and humidity data acquisition. For the actuators, high-voltage AC power for the lamp and DC power for the door lock are electronically isolated using mechanical relay modules to ensure the stability of the microcontroller's logic circuits. The schematic diagrams for the sensor-lamp module and the door lock module are illustrated in Figure 7 and Figure 8, respectively.

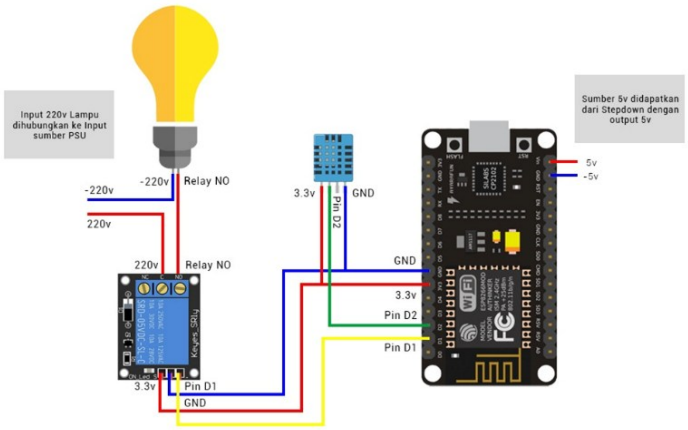

Figure 7. Schematic diagram of the temperature sensor and lamp device

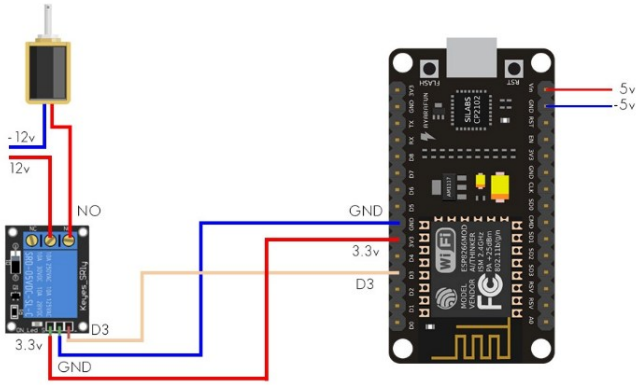

Figure 8. Schematic diagram of the electronic door lock device

2.5 System Evaluation and Testing Methodology

To rigorously validate the proposed decentralized smart home platform, a systematic testing framework was executed, focusing on deployment reliability, functional integrity, and performance efficiency. The testing environment was established on a Digital Ocean Virtual Private Server (VPS) featuring 2 vCPUs and 4GB of RAM, operating on Ubuntu 20.04 LTS. This specific hardware configuration was selected as a baseline to satisfy the resource-intensive build requirements of the web application services.

The evaluation process comprised three distinct scenarios. First, the automated deployment flow was assessed using the custom CLI-based installation scripts to verify the orchestration of Dockerized FIWARE services. Second, cross-platform functionality

testing was performed through manual end-to-end scenarios, verifying the stability of CRUD operations across the React-based web dashboard, the Android mobile application, and the Windows-based desktop hardware configurator. Third, an API performance benchmark was conducted on the core backend services, including Keyrock, Orion, and the IoT Agent. This stress test involved executing 100 continuous request iterations per service using a custom Node.js-based benchmarking tool to ensure statistical consistency. The primary evaluation metrics recorded include the successful deployment status, physical hardware actuation latency (measured in seconds), and API response durations specifically capturing the minimum, maximum, and average response times (measured in milliseconds) to assess system stability under high-frequency context updates.

3. Results and Discussion

This section presents a comprehensive evaluation of the developed FIWARE-based smart home platform. The assessment is categorized into three primary phases: deployment flow evaluation on a cloud server, comprehensive functionality testing across all client and hardware interfaces, and application programming interface (API) performance analysis.

3.1 Deployment Flow Evaluation

The initial phase evaluated the automated deployment mechanism on a Digital Ocean Virtual Private Server (VPS). The execution utilized the pre-configured CLI script to install dependencies and initialize the FIWARE ecosystem via Docker Compose. The empirical results demonstrated that the script successfully initialized the platform without critical failures. However, it was observed that the deployment necessitates a minimum specification of two vCPU cores, as the web application's build process consumes substantial computational resources that a single-core configuration cannot support. The successful service initialization is documented in Figure 9.

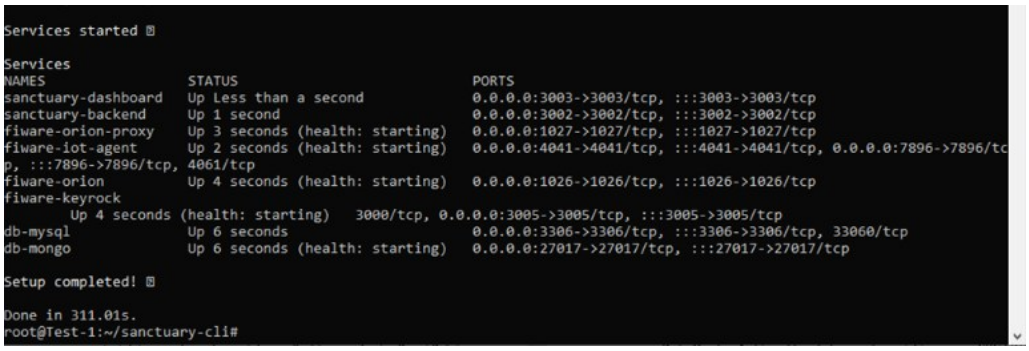

Figure 9. Successful deployment initialization of the smart home services on the cloud server

3.2 System Functionality Assessment

To ensure operational reliability, manual end-to-end functionality testing was conducted across all client interfaces. Both the React-based web dashboard and the Android mobile application flawlessly executed user authentication and complete Create, Read, Update, and Delete (CRUD) operations for room and device entities. Concurrently, the Windows desktop application successfully automated the generation, compilation, and uploading of Arduino sketches to the NodeMCU ESP8266 boards Figure 10.

DOI: *https://doi.org/10.52158/javict.v1i2.1533*
ISSN: *3124-0437 (media online)*

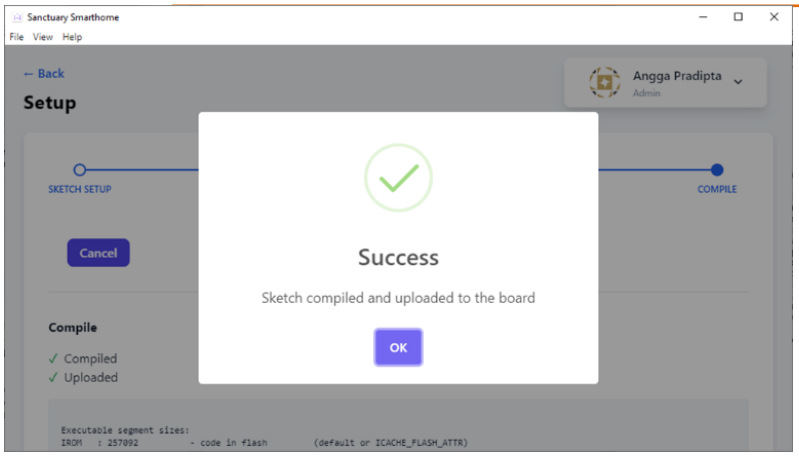

Figure 10. Successful sketch compilation and upload process via the desktop application

Subsequent hardware functionality tests verified that the physical components specifically the lamp, solenoid door lock, and temperature sensor responded accurately to remote commands dispatched from the client applications. The observed response latency between command initiation and hardware actuation was approximately one second. This minor delay is a direct consequence of the polling logic implemented within the microcontroller's sketch file, which requests state updates from the server every second to conserve bandwidth. The operational synchronization between the virtual interface and the physical hardware is depicted in Figure 11, Figure 12, and Figure 13.

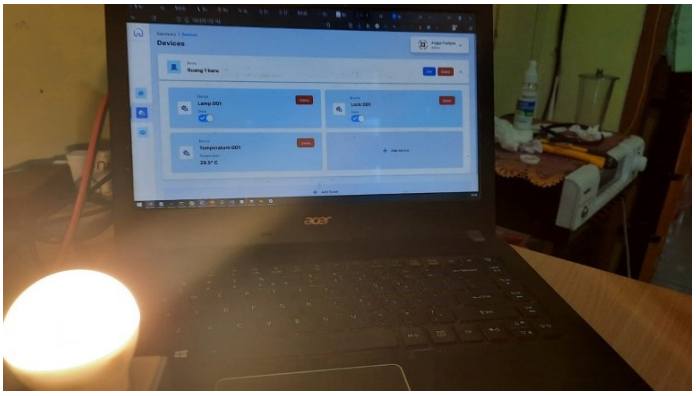

Figure 11. Functional testing of the smart lamp device synchronized with the web interface

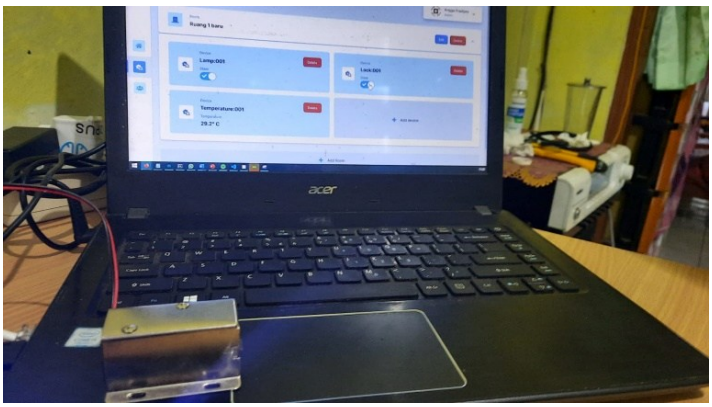

Figure 12. Functional testing of the electronic door lock device

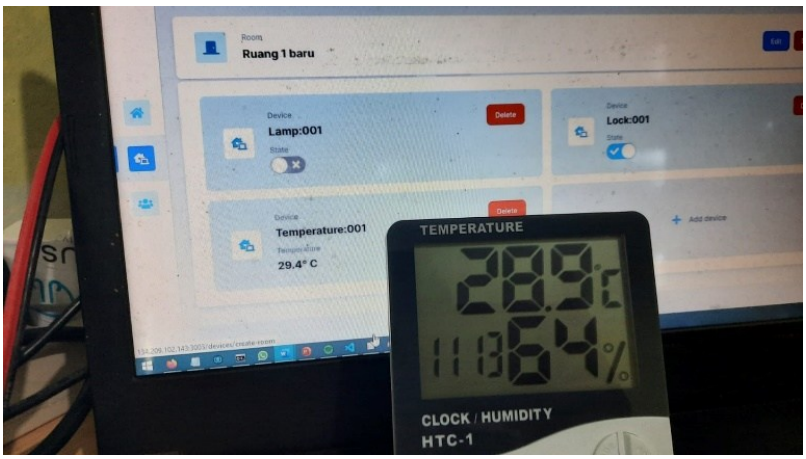

Figure 13. Temperature sensor data acquisition compared with a standard digital thermometer

3.3 API Performance and Response Time Analysis

The final experimental phase analyzed the response time and stability of the backend APIs under simulated stress conditions. Table 1 summarizes the quantitative data obtained from executing 100 continuous request iterations using the custom automated benchmarking tool for three primary services: Keyrock (authentication routing), Orion Context Broker (entity management), and the IoT Agent (direct device control).

Table 1. API Performance and Response Time Evaluation Metrics

API Component	Total Requests	Min. Response Time (ms)	Max. Response Time (ms)	Avg. Response Time (ms)	Error Rate (%)
Keyrock	100	60	1200	162	0
Orion Context Broker	100	70	4304	177	0
IoT Agent	100	88	933	145	0

As shown in Table 1 and further visualized in the performance charts Figures 14, 15, and 16, Keyrock and the IoT Agent demonstrated relatively stable performance with average response times of 162 ms and 145 ms, respectively. However, the Orion Context Broker exhibited significant latency anomalies. While its automated test average was 177 ms, it recorded a severe maximum delay spike of 4304 ms.

Furthermore, during extended manual stress testing scenarios, Orion occasionally experienced severe response degradation, reaching extreme latencies of up to 10 seconds (10,000 ms) per request. Although the baseline error rate remained at 0% without system crashes, these extreme latency spikes highlight a critical architectural bottleneck. This phenomenon indicates that FIWARE's default Orion configuration struggles with efficient resource allocation and database locking when handling continuous, high-frequency context data updates from multiple IoT nodes simultaneously [12].

DOI: *https://doi.org/10.52158/javict.v1i2.1533*
ISSN: *3124-0437 (media online)*

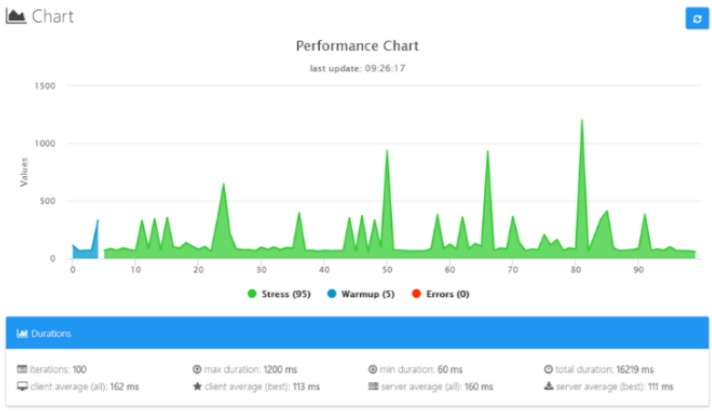

Figure 14. Performance chart and response time analysis of the Keyrock API

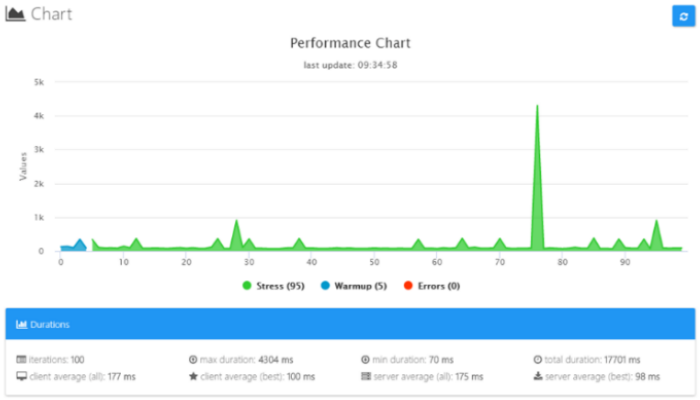

Figure 15. Performance chart and response time analysis of the Orion Context Broker API

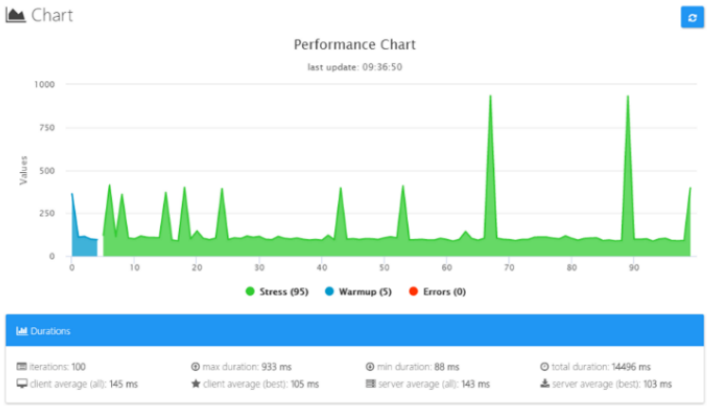

Figure 16. Performance chart and response time analysis of the IoT Agent API

4. Conclusion

This study successfully developed a decentralized, easily deployable smart home platform utilizing the FIWARE IoT framework, directly addressing the privacy and security concerns prevalent in third-party commercial hosting. By integrating Keyrock, Wilma, the Orion Context Broker, and the IoT Agent, the proposed architecture significantly lowers the technical barriers to independent home automation. The implementation proved successful in executing full CRUD operations and automated hardware configurations across web, mobile, and desktop interfaces without relying on an expensive central hardware gateway. However, rigorous performance testing revealed

notable architectural limitations. The reliance on Docker containerization demands substantial computational resources, rendering the baseline FIWARE infrastructure somewhat bloated for lightweight environments. Most critically, stress testing exposed severe latency anomalies within the Orion Context Broker, where response times sporadically degraded to 10 seconds per request under continuous, high-frequency updates, indicating inherent database locking and resource allocation bottlenecks. Therefore, future research must focus on optimizing FIWARE's baseline resource consumption or exploring lighter-weight microservice alternatives for context data management to ensure sustained, high-speed scalability in extensive residential deployments.

5. Acknowledgment

The authors would like to express their profound gratitude to all parties who contributed to the successful completion of this research. Special thanks are extended to the Department of Informatics and Computer Engineering at Politeknik Elektronika Negeri Surabaya for providing the essential technical support, laboratory facilities, and academic environment necessary for conducting this study. The authors also deeply appreciate the invaluable guidance, constructive feedback, and thorough evaluations provided by the supervisors and reviewers, which significantly contributed to the refinement and structural improvement of this manuscript.

6. Author's Note

The authors hereby declare that there is no conflict of interest related to the publication of this article. Furthermore, the authors confirm that the manuscript is original and free from any form of plagiarism.

***DOI:** https://doi.org/10.52158/javict.v1i2.1533*
***ISSN:** 3124-0437 (media online)*

7. References

[1] A. Onay, G. Ertürk, C. Kıranlı, H. Ateş, and Y. E. Isikdemir, "A Smart Home Energy Monitoring System Based on Internet of Things and Inter Planetary File System for Secure Data Sharing," pp. 64–81, 2023, doi: 10.4236/jcc.2023.1110005.

[2] P. Mtshali and F. Khubisa, "A smart home appliance control system for physically disabled people," *2019 Conf. Inf. Commun. Technol. Soc. ICTAS 2019*, pp. 1–5, 2019, doi: 10.1109/ICTAS.2019.8703637.

[3] A. John, "Security of Smart Homes in Cloud-Based IOT Environment," vol. 14, no. 04, 2025.

[4] B. Tank and V. Gandhi, "A Comparative Study on Cloud Computing, Edge Computing and Fog Computing," *Adv. Transdiscipl. Eng.*, vol. 32, no. 8, pp. 665–670, 2023, doi: 10.3233/atde221329.

[5] D. Chikurtev, "Development of a smart IoT device for direct FIWARE context broker communication," *IFAC Pap.*, vol. 59, no. 27, pp. 90–95, 2025, doi: 10.1016/j.ifacol.2025.12.084.

[6] M. E. D. Aguirre and T. D. Fernández, "An IoT architecture for smart cities based on the Fiware platform," pp. 20–27, 2022.

[7] B. Suri, M. S. Taneja, and N. Sahni, "Smart Home Automation using IoT& Virtual Assistant," *Bhagwan Parshuram Inst. Technol.*, pp. 1167–1169, 2018.

[8] Z. Sharif, L. T. Jung, M. Ayaz, M. Yahya, and D. Khan, "Smart Home Automation by Internet-of-Things Edge Computing Platform," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 4, pp. 474–484, 2022, doi: 10.14569/ijacsa.2022.0130455.

[9] S. Faiazuddin, "system using Raspberry Pi4," pp. 714–719, 2020.

[10] L. Dantas, "A Development Environment for FIWARE-based Internet of Things Applications," pp. 21–26, 2019, doi: 10.1145/3366610.3368100.

[11] P. Zervoudakis, N. Karamolegkos, E. Plevridi, P. C., and A. Fragkiadakis, "EPOPTIS: A Monitoring-as-a-Service Platform for Internet-of-Things Applications," pp. 1–28, 2024, doi: 10.3390/s24072208.

[12] M. Elkhodr and S. Khan, "A Novel Semantic IoT Middleware for Secure Data Management : Blockchain and AI-Driven Context Awareness," pp. 1–31, 2024.

Authors Biographies

Angga Pradipta Kurnia Putra received his degree in Informatics Engineering from Politeknik Elektronika Negeri Surabaya (PENS), Indonesia. During his studies, he was an active member of Mobile Sensing and Edge Computing Technology (MSECT) research group at PENS. His research interests include Cloud Computing, Internet of Things (IoT), and Smart Home Systems.

Email: anggaa.pradipta@gmail.com
ORCID: -

Zaky Wahyu Oktavianto received his bachelor's degree in Informatics Engineering from Universitas 17 Agustus 1945 Surabaya, Indonesia. He is currently pursuing his master's degree in the Department of Informatics and Computer Engineering at Politeknik Elektronika Negeri Surabaya (PENS), Indonesia. He is also a member of Mobile Sensing and Edge Computing Technology (MSECT) research group at PENS. His research interests include Smart Home Systems, Embedded Systems, Edge Computing, and Image Processing.
Email: zayuto33@gmail.com
ORCID: **-**

Reksa Prastama Putra received his bachelor's degree in Informatics Engineering from Universitas Dian Nuswantoro, Indonesia. He is currently pursuing his master's degree in the Department of Informatics and Computer Engineering at Politeknik Elektronika Negeri Surabaya (PENS), Indonesia. He is also a member of Mobile Sensing and Edge Computing Technology (MSECT) research group at PENS. His research interests include Embedded Systems, Edge Computing, and Smart Plantation Systems.
Email: reksaprastama1905@gmail.com
ORCID: **-**

***DOI:** https://doi.org/10.52158/javict.v1i2.1533*
***ISSN:** 3124-0437 (media online)*