DOI:10.5937/IIZS25189D

SIMULATED IoT SENSOR MONITORING AND ALERTING USING PYTHON AND WEB APIs

Hristina Delibašić Marković[1], Violeta Petrović[1] and Ivan Petrović[2]
[1]University of Kragujevac, Faculty of Science, Kragujevac, Serbia
[2]Academy of Professional Studies Šumadija, Department in Kragujevac, Kragujevac, Serbia
e-mail: hristina.delibasic@pmgf.kg.ac.rs

Abstract: A software-only framework is presented for the simulation of Internet of Things (IoT) sensor streams and the coordination of real-time alerting via Python and web-based APIs. Synthetic data generators produce temperature, light, and humidity readings according to configurable statistical models and temporal trends. These readings are constantly checked against set limits, and when they go over those limits, automatic notifications are sent out through RESTful interfaces to services like Telegram and IFTTT. The suggested method eliminates the need for physical hardware, making it easier to quickly create prototypes, teach concepts, and test IoT alert systems in places with few resources. The framework's performance is tested using a simulation with 50 samples, showing that it can accurately detect events in under 2 seconds and reliably send alerts to outside channels. The results substantiate the utility of a virtualized IoT testbed for early-stage development and educational applications.
Key words: IoT simulation, Python-based alerting, synthetic sensor data, web API integration, virtual IoT testbed, real-time notifications.

INTRODUCTION

The rapid development of Internet of Things (IoT) technologies has reshaped residential, industrial, commercial, and healthcare sectors by enabling real-time monitoring, automation, and intelligent decision-making [1]. At the core of these systems are networks of sensors that capture essential parameters such as temperature, humidity, and luminosity, which are directly tied to thermal transfer, environmental control, and process efficiency [2]. In the context of energetics and process technique, the accurate measurement and analysis of these quantities are crucial for improving process stability, optimizing energy use, and ensuring environmental safety.

Despite their potential, the early development and testing of IoT applications are often constrained by the high cost and complexity of physical hardware [3]. To address these challenges, software-based IoT simulation frameworks have emerged as efficient and cost-effective alternatives for preliminary validation and education [4,5]. Python, with its readability and extensive scientific libraries such as Pandas, NumPy, and Matplotlib, provides a powerful platform for building such simulations [6, 7]. This paper presents a software-only framework that generates synthetic environmental data, detects threshold exceedances in real time, and dispatches alerts through APIs like Telegram and IFTTT [8]. The approach supports rapid prototyping, educational demonstrations, and process-oriented analysis in resource-limited environments.

RELATED WORK

Research on IoT simulation can be grouped into several main directions. Hybrid and hardware-in-the-loop simulation combines physical and virtual components, originally developed in structural engineering for studying nonlinear dynamic systems [9]. Later studies improved stability and accuracy of algorithms [10-12], while deep learning models were even applied to replace finite element solvers [13, 14]. Although designed for civil engineering, principles such as synchronized feedback and real-time logic are relevant for IoT simulators aiming to emulate

sensor behaviour and responses. A second direction is software-only sensor-network emulators, such as NS-3 [15] and CupCarbon [16], which model communication protocols and packet-level behaviour. While effective for network analysis, they focus less on sensor dynamics or alerting. Cloud-based approaches extend this by simulating sensors as virtualized services, supporting large-scale monitoring and control [17]. However, these solutions typically require access to real devices or testbeds, limiting their use in fully virtualized contexts. The third stream emphasizes Python frameworks as a flexible and accessible tool for scientific computing [18]. Libraries such as Pandas, NumPy, and Matplotlib allow efficient data handling, visualization, and API integration [6, 7]. In IoT, this enables software-only testbeds for generating sensor data, applying threshold-based logic, and sending notifications through RESTful services [19,20]. Despite this progress, a clear gap remains: lightweight frameworks that integrate virtual sensing, real-time evaluation, and automated alerting without reliance on hardware or proprietary platforms. The present work addresses this gap by introducing a Python-based environment tailored for education, prototyping, and process-oriented analysis.

SYSTEM ARCHITECTURE

The proposed framework reproduces the logic of an IoT sensing and alerting pipeline entirely in software. It consists of three main components: (1) a virtual sensor module, (2) a threshold evaluation engine, and (3) a notification and integration layer, with optional logging and visualization. All elements are implemented in Python to ensure transparency, modularity, and adaptability. The workflow, shown in Fig. 1, mirrors the sensing-processing-notification loop typical of real industrial and environmental monitoring systems.

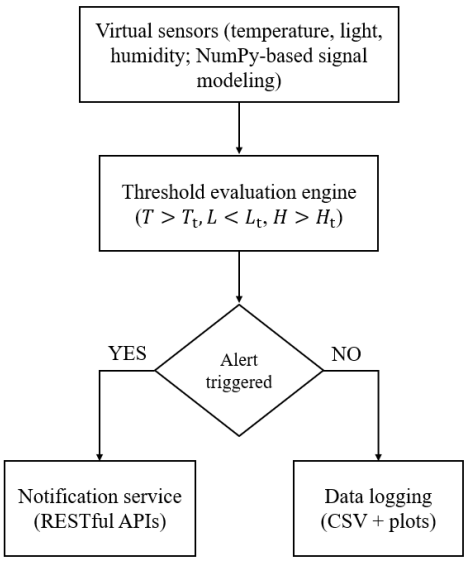

Fig. 1. *Logical flow of synthetic sensor data, threshold evaluation, alert dispatching, and optional logging in the proposed IoT simulation framework.*

The virtual sensor module generates synthetic readings of temperature, light intensity, and humidity-parameters that represent fundamental physical quantities in energetics and process engineering. Temperature values follow sinusoidal or stochastic trends, reflecting heat transfer governed by radiative and convective processes. Light intensity simulates diurnal illumination cycles and sudden attenuation (e.g., cloud cover), which directly influence thermal balance through absorption and re-radiation. Humidity is modelled as a gradual decrease or oscillatory trend, representing evaporation, ventilation, or moisture transport in air. Each sensor stream is configurable in terms of baseline, amplitude, and noise, making it possible to approximate realistic environmental dynamics. The threshold evaluation engine continuously compares simulated data against user-defined limits that mark critical process states (e.g., overheating, insufficient illumination, or low humidity). Such thresholds represent practical boundaries in

energetics and process technique - for instance, maximum operating temperature of materials, minimum light levels for efficient energy conversion, or humidity levels required for stability of sensitive processes. The evaluation engine, implemented through Python conditional logic, can be adapted for compound or time-dependent conditions, offering flexibility for different process scenarios. Once a threshold is crossed, the notification layer formats the event into an alert that includes sensor type, value, timestamp, and triggered condition. This alert is dispatched in real time via RESTful APIs (e.g., Telegram, IFTTT), mimicking the role of industrial supervisory systems that issue warnings when safety or efficiency margins are exceeded. In cases where thresholds are not exceeded, the framework records the data into structured CSV logs and optionally produces plots using Matplotlib. These visualizations highlight both normal operation and deviations, serving as an analogue to diagnostic charts in process monitoring.

IMPLEMENTATION

The framework was realized entirely in Python, relying on open-source libraries that support numerical modelling, control logic, communication through RESTful APIs, and visualization. All components operate locally on a standard computer, which eliminates the need for specialized hardware or cloud services and makes the system easy to reproduce in different environments.

To emulate the behaviour of environmental variables relevant for energetics and process technique, three types of virtual sensors were implemented: temperature, light intensity, and humidity. Their evolution in time is described by mathematical functions that capture realistic physical trends. Temperature follows sinusoidal variations that resemble diurnal heating and cooling cycles, while additional random noise introduces local fluctuations similar to microclimatic effects. Humidity can be modelled as an exponential decrease, representing evaporation or ventilation, whereas light intensity is described by periodic changes that emulate solar irradiation modulated by cloud cover or shading. By adjusting parameters such as baseline values, amplitudes, and noise levels, it becomes possible to reproduce both stable and highly dynamic conditions, creating a versatile testbed for analysing process behaviour.

The simulated data streams are continuously evaluated against predefined thresholds that correspond to critical points in physical systems. For example, maximum allowable temperature for material stability, minimal light levels required for energy conversion efficiency, or humidity levels necessary for maintaining environmental balance. When one of these thresholds is exceeded, the framework immediately triggers an alert, thereby reproducing the way industrial supervisory systems signal the crossing of operational limits. These alerts are dispatched in real time through web-based services such as Telegram or IFTTT, which act as communication channels between the simulated process and the operator.

In addition to alerting, the system records all sensor values and events into structured log files, enabling further analysis of temporal behaviour. The data can be visualized through time-series plots with marked thresholds and alert instances, providing an intuitive picture of process evolution and deviations. Depending on the needs of the user, the simulation can be executed in two different modes: in batch mode, where a finite number of samples is generated for offline inspection, or in real-time mode, where data streams evolve continuously with delays that mimic real sensor readings. Both approaches support applications in education, early-stage research, and practical process analysis.

RESULTS AND DISSCUSION

The simulation results highlight the physical nature of the monitored parameters (temperature, light intensity, and humidity) and demonstrate how their variations can be meaningfully translated into alerts. Rather than being treated as abstract data, these variables reflect processes such as radiative heating, evaporative cooling, and atmospheric attenuation, all of which are central to energetics and process technique. By mapping their temporal behaviour against defined thresholds, the framework captures moments when conditions deviate from

safe or efficient regimes and triggers alerts accordingly. The time series in Fig. 2 illustrates these dynamics.

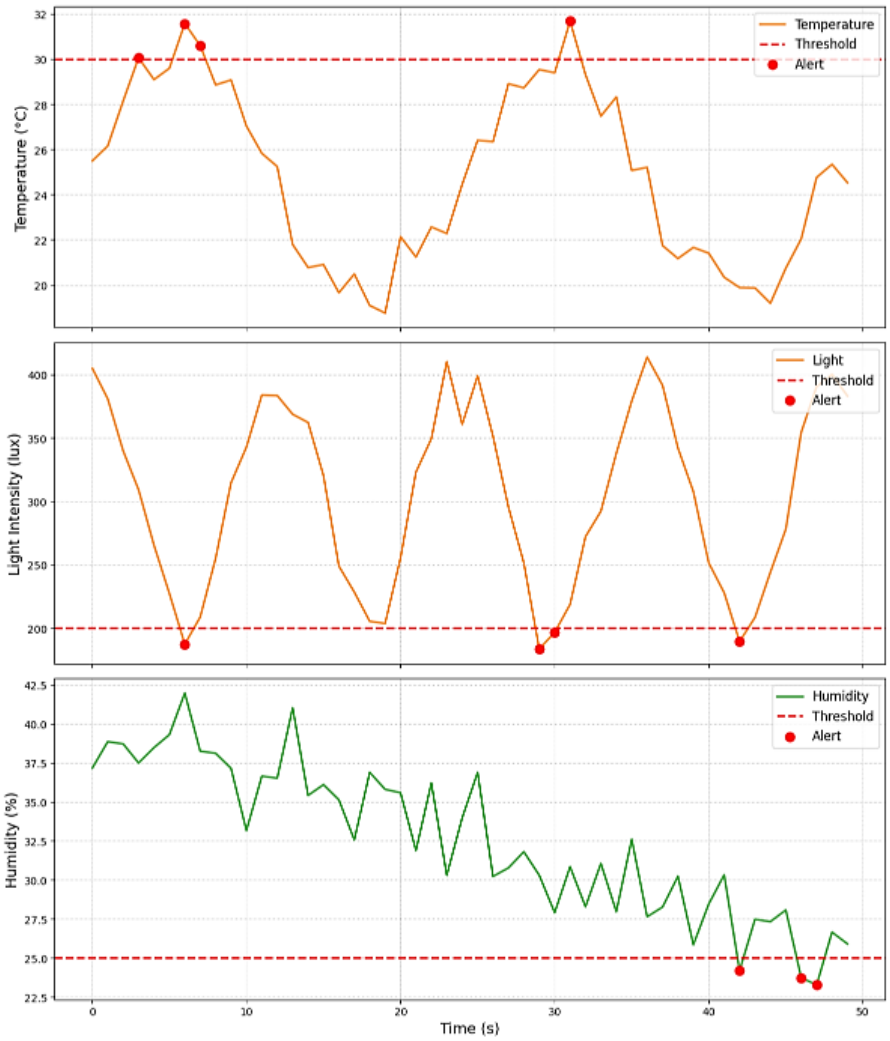

Fig. 2. *Simulated time series of environmental parameters over a 50-second interval, illustrating dynamic behavior of temperature, light intensity, and relative humidity. Each subplot includes a dashed red line indicating the predefined threshold for alert activation: 30 °C for temperature, 200 lux for light, and 25% for humidity. Red circular markers denote the precise instances of threshold exceedance, triggering automated alerts.*

Temperature follows an oscillatory trend with peaks occasionally surpassing the critical value of 30°C, representing conditions where thermal stress may impair materials or biological systems. Light intensity exhibits periodic fluctuations consistent with illumination cycles, with rapid dips simulating shading effects; when values fall below 200 lux, the system identifies insufficient illumination for reliable operation. Humidity decreases gradually to represent drying, with the 25% threshold marking the onset of unfavourable environmental conditions. The close alignment of alerts with physical threshold crossings demonstrates the robustness of the detection mechanism and its resistance to noise. When examined together, the three parameters reveal strong coupling. As shown in Fig. 3, increases in light intensity often drive temperature rises, while reduced humidity amplifies thermal stress by limiting evaporative cooling. This multivariate view underscores the importance of cross-parameter analysis: adverse conditions are rarely the result of a single variable but emerge from their combined influence.

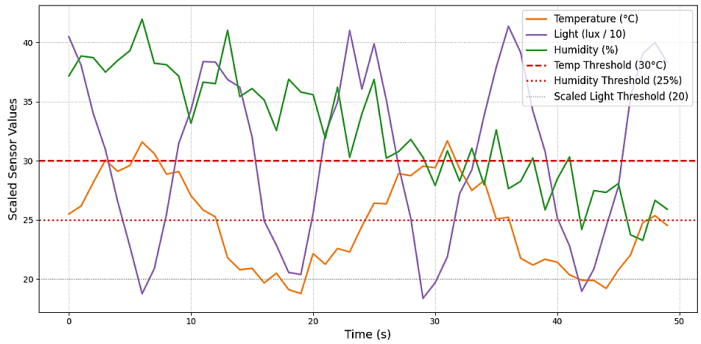

Fig. 3. *Overlay of normalized sensor readings illustrating the concurrent evolution of temperature (orange), light intensity scaled by a factor of 10 (purple), and humidity (green) over the simulation duration. Horizontal dashed lines mark respective alert thresholds for temperature (30 °C), light intensity (200 lux), and humidity (25%).*

The temporal distribution of alerts in Fig. 4 further illustrates this point. Temperature and light thresholds are often exceeded simultaneously, signalling periods of intense radiative forcing, while humidity alerts appear later, reflecting slower but cumulative drying processes. Such concurrency of alerts mirrors real-world compound stress events, for example heat waves coupled with low humidity, which can degrade materials, reduce efficiency, or threaten human safety.

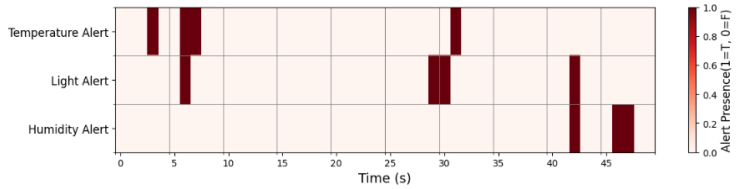

Fig. 4. *Heatmap representation of alert states across the three monitored parameters. Rows correspond to sensor channels, temperature, light, and humidity, while columns represent successive time points. A value of 1 (dark red) indicates that the respective parameter has breached its critical threshold at that moment; 0 (pale) denotes nominal conditions.*

Figures 5 and 6 extend the analysis by projecting the data into phase spaces. The temperature-humidity plot identifies regions where elevated heat coincides with insufficient moisture, marking zones of compounded risk. The light-temperature-humidity interaction map highlights how illumination drives heating differently under moist and dry conditions, with the most severe stresses occurring when high light, high temperature, and low humidity overlap. These compound risk zones are of particular importance in process monitoring, as they pinpoint operating regimes where failure is most likely to occur.

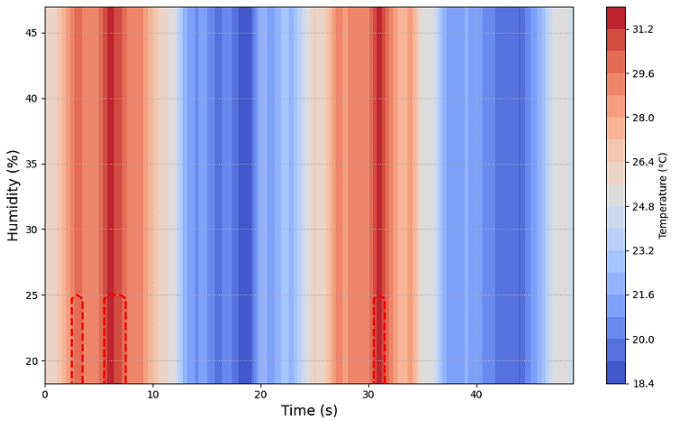

Fig. 5. *Two-dimensional contour map of temperature (color-coded) plotted against time (x-axis) and humidity (y-axis). The dashed red horizontal line marks the humidity alert threshold at 25%.*

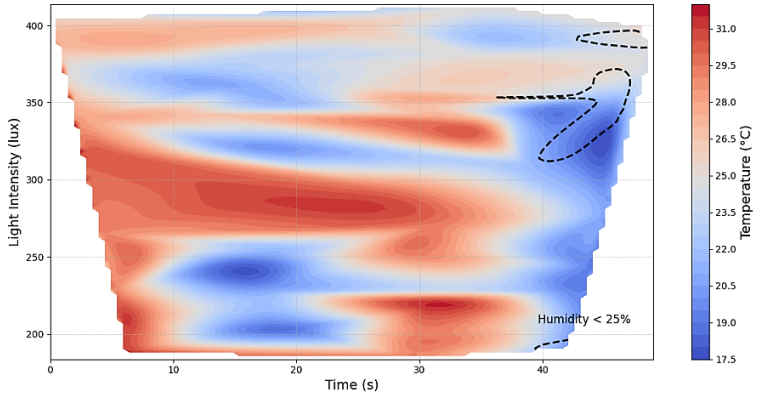

Fig. 6. *Three-dimensional contour visualization of temperature variation (color scale) over time and light intensity. Dashed black contours indicate regions where relative humidity falls below the critical 25% threshold.*

Taken together, the results presented in this Section demonstrate that a purely software-based simulation can capture both the physics of environmental variability and the logic of IoT-driven detection. The framework therefore provides a practical and low-cost testbed for analysing process dynamics, anticipating critical states, and designing alert strategies relevant to energetics and process technique.

CRITICAL ASSESSMENTS OF THE PROPOSED FRAMEWORK

The proposed framework offers clear advantages for studying IoT-based monitoring in energetics and process technique. By simulating temperature, light, and humidity entirely in software, it enables the exploration of key physical quantities that govern heat transfer, illumination efficiency, and moisture control, all without the cost or complexity of real sensors. Its modular Python design provides a reproducible environment for both research and teaching, while API-based notifications mirror the logic of industrial supervisory systems. At the same time, the framework has limitations. It does not reproduce imperfections of real transducers (e.g., drift or hysteresis) or network effects such as latency and packet loss. Its reliance on simple threshold rules, though transparent, cannot capture the adaptability of modern process monitoring strategies that increasingly rely on predictive or machine learning approaches. These constraints, however, suggest clear directions for further work. Hardware-in-the-loop extensions could bring the simulation closer to real process validation, probabilistic alerting would better capture environmental variability, and integration with network emulators could combine physical modelling with communication analysis. In this way, the framework can evolve into a more powerful tool for designing and testing resilient monitoring strategies in energetics and process technique.

CONCLUSION

This study presented a Python-based framework that emulates IoT sensing and alerting workflows entirely in software. By simulating temperature, humidity, and light intensity the framework provides a reproducible and low-cost platform for analysing how environmental variables evolve, cross critical thresholds, and trigger automated alerts. The results show that both univariate and multivariate behaviours can be represented in a way that highlights the physical coupling of heat transfer, illumination, and moisture control.
The approach is valuable for both teaching and early-stage research: it allows algorithm validation and systems-level thinking without the need for physical hardware, while API-based notifications reproduce the logic of industrial supervisory systems. Future work will focus on enhancing realism by introducing stochastic hardware effects, adaptive alerting, and integration with cloud infrastructures. In doing so, the framework can grow into a versatile tool

not only for IoT prototyping and education, but also for exploring and optimizing monitoring strategies in energy and process-oriented applications.

Acknowledgements. Authors would like to acknowledge the support received from the Science Fund of the Republic of Serbia, #GRANT 6821, Atoms and (bio)molecules-dynamics and collisional processes on short time scale - ATMOLCOL. Our appreciation also goes to the Serbian Ministry of Education, Science and Technological Development (Agreement No. 451-03-137/2025-03/ 200122). H. Delibasic Markovic would also like to express gratitude to COST Actions CA21159 - "Understanding interaction of light with biological surfaces: possibility for new electronic materials and devices" for their support and CA22148 - "An international network for Non-linear Extreme Ultraviolet to hard X-ray techniques (NEXT)".

REFERENCES

[1] Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M., Internet of Things (IoT): A vision, architectural elements, and future directions, Future Generation Computer Systems, Vol.29, No.7, pp.1645-1660, 2013.
[2] Lee, I., Lee, K., The Internet of Things (IoT): Applications, investments, and challenges for enterprises, Business Horizons, Vol.58, No.4, pp.431-440, 2015.
[3] Atzori, L., Iera, A., Morabito, G., The Internet of Things: A survey, Computer Networks, Vol.54, No.15, pp.2787-2805, 2010.
[4] Mineraud, J., Mazhelis, O., Su, X., Tarkoma, S., A gap analysis of Internet-of-Things platforms, Computer Communications, Vol.89, pp.5-16, 2016.
[5] Yassein, M.B., Mardini, W., Khalil, A., Smart homes automation using Z-wave protocol, Proc. International Conference on Engineering & MIS (ICEMIS), IEEE, pp.1-6, 2016.
[6] McKinney, W., Data structures for statistical computing in Python, SciPy, Vol.445, No.1, pp.51-56, 2010.
[7] Hunter, J.D., Matplotlib: A 2D graphics environment, Computing in Science & Engineering, Vol.9, No.3, pp.90-95, 2007.
[8] Bröring, A., Seeger, J., Papoutsakis, M., Fysarakis, K., Caracalli, A., Networking-aware IoT application development, Sensors, Vol.20, No.3, pp.897, 2020.
[9] Takanashi, K., Udagawa, K., Seki, M., Okada, T., Tanaka, H., Nonlinear earthquake response analysis of structures by a computer-actuator on-line system, Bulletin of Earthquake Resistant Structure Research Center, Vol.8, pp.1-17, 1975.
[10] Chen, C., Ricles, J.M., Marullo, T.M., Mercan, O., Real-time hybrid testing using the unconditionally stable explicit CR integration algorithm, Earthquake Engineering & Structural Dynamics, Vol.38, No.1, pp.23-44, 2009.
[11] Kolay, C., Ricles, J.M., Marullo, T.M., Mahvashmohammadi, A., Sause, R., Implementation and application of the unconditionally stable explicit parametrically dissipative KR-α method for real-time hybrid simulation, Earthquake Engineering & Structural Dynamics, Vol.44, No.5, pp.735-755, 2015.
[12] Chae, Y., Kazemibidokhti, K., Ricles, J.M., Adaptive time series compensator for delay compensation of servo-hydraulic actuator systems for real-time hybrid simulation, Earthquake Engineering & Structural Dynamics, Vol.42, No.11, pp.1697-1715, 2013.
[13] Bas, E.E., Moustafa, M.A., Real-time hybrid simulation with deep learning computational substructures: System validation using linear specimens, Machine Learning and Knowledge Extraction, Vol.2, No.4, pp.469-489, 2020.
[14] Zhang, X., Xie, X., Tang, S., Zhao, H., Shi, X., Wang, L., Wu, H., Xiang, P., High-speed railway seismic response prediction using CNN-LSTM hybrid neural network, Journal of Civil Structural Health Monitoring, Vol.14, No.5, pp.1125-1139, 2024.
[15] Alam, T., Cloud-based IoT applications and their roles in smart cities, Smart Cities, Vol.4, No.3, pp.1196-1219, 2021.

[16] Chernyshev, M., Baig, Z., Bello, O., Zeadally, S., Internet of Things (IoT): Research, simulators, and testbeds, IEEE Internet of Things Journal, Vol.5, No.3, pp.1637-1647, 2017.
[17] Li, S., Tryfonas, T., Li, H., The Internet of Things: A security point of view, Internet Research, Vol.26, No.2, pp.337-359, 2016.
[18] Sun, Q., Berkelbach, T.C., Blunt, N.S., Booth, G.H., Guo, S., Li, Z., Liu, J., McClain, J.D., Sayfutyarova, E.R., Sharma, S., Wouters, S., PySCF: the Python-based simulations of chemistry framework, Wiley Interdisciplinary Reviews: Computational Molecular Science, Vol.8, No.1, p.e1340, 2018.
[19] Krishnamurthy, J., Maheswaran, M., Programming frameworks for Internet of Things, Internet of Things, Morgan Kaufmann, pp.79-102, 2016.
[20] Li, S., Xu, L.D., Zhao, S., The Internet of Things: A survey, Information Systems Frontiers, Vol.17, pp.243-259, 2015.