Automatic Configuring Reinforcement Learning Powered Plug and Play Hardware System

Sharfa Nawab, Bushra Khan, Bisma Malik, Shivania, Hira Solangi
Department of Computer Science, Quaid-e-Awam University of Engineering Science & Technology, Nawabshah, Sindh
*Correspondence: sharfanawab52@gmail.com

Citation | Nawab. S, Khan. B, Malik. B, Shivania, Solangi. H, "Automatic Configuring Reinforcement Learning Powered Plug and Play Hardware System IJIST, Vol. 7 Issue. 10 pp 123-133, December 2025
Received | November 08, 2025 **Revised |** November 30, 2025 **Accepted |** December 04, 2025 **Published |** December 07 2025.

The introduction of new Internet of Things (IoT) devices in modern automation systems remains a challenge because developers must manually code, calibrate, and configure each device. each sensor and actuator require individualized programming, which contributes to longer development times and reduces the scalability and flexibility of smart systems. To overcome these constraints, the present research proposes an adaptive and self-learning control structure that allows hardware integration using Reinforcement Learning (RL) in a plug-and-play manner. The system automatically identifies, configures, and manages hardware devices via software agents, limiting the need for human intervention. The proposed system bridges the gap between hardware and software layers by integrating embedded systems, communication protocols, and RL-based adaptive control, creating an integrated system. This study enhances automation, scalability, and adaptability, enabling more efficient IoT ecosystems while significantly reducing manual effort required for programming and system calibration.

Keywords: Internet of Things (IoT), Embedded Systems, Plug-and-Play, Reinforcement Learning (RL), Automatic Automation, Real-Time Visualization, Adaptive System

Introduction:

IoT and embedded systems are significant components of emerging developments in the rapidly evolving automation and smart environments. They utilize microcontrollers, sensors, and actuators to gather data and perform tasks independently. They encompass robots and industrial automation, as well as smart homes and healthcare surveillance systems. Despite advancements in IoT devices and their interconnectivity, a major issue persists: each time a new device is introduced, it must be configured, coded, and calibrated. Developers spend considerable time writing new code to accommodate all sensors and actuators. For example, when a new motion or temperature sensor is implemented, one must adjust the communication settings, modify the code, and conduct a series of tests to ensure proper functionality. This physical effort limits system flexibility and slows development as IoT systems grow larger and more complex.

Another major issue is that devices are not produced by a single manufacturer, hence interoperability is not guaranteed. Connections between devices are irregular and difficult due to differences in pin configurations, data formats, and communication protocols [1].

Due to these integration issues, IoT systems cannot achieve true plug-and-play functionality similar to that of new software. According to research conducted by ACPNPHS, A smart plug-and-play hardware control system has been proposed in which Reinforcement Learning (RL) is utilized to address this problem. This system can detect new hardware automatically, learn to configure and control it, and execute operations without human programming, using an intelligent software mechanism. On first execution, when connecting a microcontroller such as an Arduino or ESP32 to a new device, such as a sensor, actuator, or motor, the RL agent explores the input and output behavior of the device, learns how to operate it, and subsequently executes the device autonomously. The system is capable of learning and adapting without human intervention [2].

The ACPNPHS framework integrates embedded systems, IoT communication protocols, and machine learning into a single system. It also features a graphical user interface (GUI) displaying real-time visuals of the system learning process as well as the hardware in operation. These characteristics ensure system flexibility and lay the foundation for smart, self-learning IoT networks capable of integrating new devices with ease.

Ultimately, this study aims to simplify automation by reducing the need for manual coding and configuration by enhancing system flexibility through learning. The plug-and-play hardware system, based on RL implementation, is intended to ensure that, in the future, embedded devices will be capable of connecting autonomously and achieving self-learning and universal IoT integration.

Literature Review:

Based on the three essential requirements of a plug-and-play environment, it has been reported that physics-based reinforcement learning approaches are unable to fulfill these conditions because they rely solely on a global transition model of the environment. The literature has described a communication layer used in cloud-based robotic systems in which multiple users communicate with various robots. In these systems, interoperability between heterogeneous devices, such as smartphones, tablets, computers, and different types of robots, is required [3].

A DRL agent was trained using historical transitions of decision-making through a batch reinforcement learning technique called Batch-Constrained Q-learning (BCQ), which requires no further system interaction. This method enables the introduction of a plug-and-play agent that, after offline training, can be deployed instantly without experiencing learning deficits or performance issues. As a result, the plug-and-play agent functions as a universal offline DRL training solution, independent of the specific system, platform, or problem domain. Using Dynamic Adaptive Streaming over HTTP (DASH) as a case study, the

functionality of the proposed plug-and-play DRL system has been demonstrated. Preventing system disruption was a primary goal in the development of the plug-and-play DRL agent. The PnP-DRL method provides a plug-and-play approach to real-world systems, as shown in Figure 8, enabling deployment without learning gaps or operational disruptions. It has been noted that the PnP-DRL agent's performance is consistent and reliable [4].

When applied to IoT, plug-and-play devices are defined in contrast to traditional systems that isolate devices into silos. Plug-and-play devices are linked via social interaction, enabling coordinated operation. For example, a smartwatch may automatically instruct a medicine dispenser to prepare doses based on an elderly user's blood pressure, with the relationship configured for the treatment duration and extendable as needed [5].

The term "plug-and-play" is also used to describe systems where new robotic platforms can be integrated simply by adding a record to a database. This approach allows non-technical users to modify system configurations remotely and enables flexible scaling of the number of agents. Additionally, a plug-and-play solution has been proposed where configurations are automatically retrieved from a public database, and reverse tunneling enables communication across network types, including local WiFi, public WiFi, and mobile data networks [3].

The detailed algorithm of HQS-based plug-and-play IR incorporating a deep denoiser prior, referred to as DPIR, is summarized in Algorithm [6].

One of the primary challenges of PnP-AI relates to system reliability. Consistent and accurate selection of input datasets, proper training, and parameter configuration are required by intelligent machines to achieve precise results. Furthermore, many ML algorithms are application-specific and require user involvement for configuration. Consequently, the reliability of a PnP solution is often evaluated based on user feedback, which reduces overall system autonomy [7].

Each iteration of the resulting PnP algorithm involves some kind of inversion of the forward model followed by denoiser-induced regularization [8].

Encapsulating Bayesian prior knowledge in an algorithmic denoiser is a fundamental concept of PnP-inspired techniques. Since data-fitting updates and denoisers can be produced independently, and multiple combinations of updates and denoisers are possible, this approach promotes code modularity. An end-to-end trained system can outperform a general-purpose PnP system under ideal conditions, but the drawback of this generality is a potential reduction in reconstruction quality [9].

The generalized policy update approach allows learning new policies for multiple tasks based on previously learned policies, whereas conventional reinforcement learning provides a framework for learning a single policy to accomplish a specific task. In particular, a standard RL technique evaluates the current policy for a single task as determined by its reward function during policy evaluation [10].

RL algorithms can be regarded as methods that transform infeasible dynamic programming approaches into practical algorithms into practical algorithms, enabling application to large-scale problems [11].

RePNP is a PnP framework for image restoration based on deep reinforcement learning. Considering an imprecise observation model, a more realistic and challenging scenario was evaluated. This framework employed a fully convolutional network trained with PPO as the image denoising prior, and image denoising was modeled as a Markov decision process [12].

Three algorithms were selected from the major families of offline RL approaches: policy regularization, implicit regularization, and Q-function regularization. These families cover most widely recognized offline RL methods (Levine et al., 2020), making them suitable representatives for demonstrating the general applicability of the proposed framework [13].

RL Factory, consisting of three primary modules: Tool Use, RL-Training, and WebUI, has demonstrated the effectiveness of reinforcement learning in repeated rounds of tool invocation. Multiple tool invocation rounds, multimodal data interactions, and tool registration are all managed by the Tool Use module. [14].

It has been acknowledged that deep RL is not yet sufficiently applicable to solve robotics problems with real-world complexity. Nevertheless, current advancements represent a significant process. Configuring robots to learn via reinforcement learning in real-world scenarios may be beneficial for services such as geriatric care, continuous cleaning and disinfection, and robust industrial processes, particularly where human input is minimal (e.g., during pandemics). However, an RL agent maximizes only the reward function it receives; therefore, improperly defined reward functions may result in undesired behaviors. Accordingly, alongside algorithm performance improvements, parallel research in RL safety is essential [15].

In reinforcement learning, the underlying dynamics and reward function are typically the two main characteristics of an environment. The transition function p, connected to the PnP-ADMM framework operations, represents environment dynamics. The PnP-ADMM structure and reward formulation used during policy training have been defined in the literature. A differentiable environment is desirable to enable efficient policy learning. To this end, a CNN-based denoiser is employed to process images before their use in the system. [16].

Methodology:

To build a smart and dynamic plug-and-play system, this research employs a systematic approach to integrating Reinforcement Learning (RL) with embedded hardware and IoT platforms. The ACPNPHS research framework consists of three main components.

Problem Identification:

The methodology begins with a comprehensive evaluation of available embedded systems, IoT models, and RL-based automation mechanisms. Based on the literature, most existing solutions rely on predetermined device profiles, manual configuration, simulated environments, or cloud-based processing. These limitations stem from the lack of autonomous, hardware-dependent plug-and-play systems capable of dynamically learning to operate with new devices. Consequently, the core issue is the absence of a smart system that can automatically identify, train, and manage newly interconnected IoT devices without manual programming or prior knowledge of the device.

Design Framework:

The approach of connecting a new hardware device such as a sensor or actuator to the system via a microcontroller interface is the starting point of the method, as shown in Figure 1. This system automatically loads and begins collecting input/output data to define the operational behavior and communication properties of a device upon connection. The Reinforcement Learning (RL) module, which is the central component of the suggested system, processes the gathered data. The RL agent learns to associate device responses with appropriate actions based on the environment.

The algorithm uses a trial-and-error learning method during this process; If the system correctly identifies a device or takes the right action, it receives a positive reward; otherwise, it is penalized or receives a negative reward. This continuous reward–penalty cycle improves system performance and decision-making over time to improve the performance and decision-making abilities of the system, thus decreasing the necessity of human intervention in the configuration and calibration process.

To provide a visual representation of the RL mechanism, device identification, and outcomes of the learning process in real time, the system includes a graphical user interface (GUI). The GUI provides the user with a clear view of how the system adapts to every

newly connected device by showing sensor detection, learning status, and system logs. The system can identify the device and automatically assign appropriate functionality—such as turning on a light, switching on a motor, or setting an alarm—once the RL agent has gathered sufficient information and reached an optimal reward state.

After successful device recognition, the system uses the microcontroller to execute the learned action autonomously. Each newly connected device can start operating right away without the need for manual code modification with the help of the ACPNPHS approach. The flexibility of the RL model allows the system to be scaled to support complex IoT settings since it can support a wide range of devices and sensor types.

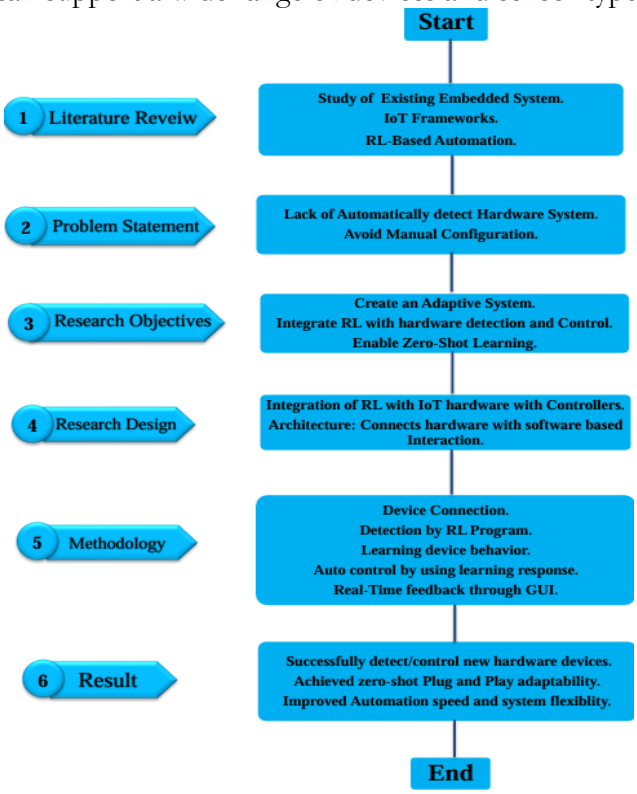

Figure 1. ACPNPHS research framework showing the integration of reinforcement learning with IoT devices and embedded systems

Testing and verification:

Overall, the ACPNPHS methodology shows how true plug-and-play automation can be achieved by integrating RL with IoT and embedded systems, as illustrated in Figure 2. It enhances system intelligence and flexibility, reduces development time, and eliminates repetitive code. The suggested framework advances a universal self-learning control system with real-time device recognition and operation, as demonstrated in the results section, as shown in the results section. The proposed ACPNPHS system focuses on real-time hardware-level autonomy, in contrast to the existing plug-and-play and reinforcement learning-based solutions that primarily take into account simulation environments, cloud-based, or application-specific device settings. ACPNPHS allows on-device learning by direct interaction of physical sensors and actuators, compared to the pre-defined devices or profiles, offline training, or databases used in earlier research. The originality of this work lies in its ability to independently detect new IoT devices, understand their dynamics in real time using reinforcement learning, and perform the appropriate actions without the need to write code or adhere to predetermined rules. Additionally, real-time incentive display, live device action logs, and GUI-based feedback are incorporated, which greatly improve the validation's practicality and are essentially lacking in earlier research. Because of this,

ACPNPHS is more adaptable, scalable, and useful in an actual IoT environment than other systems.

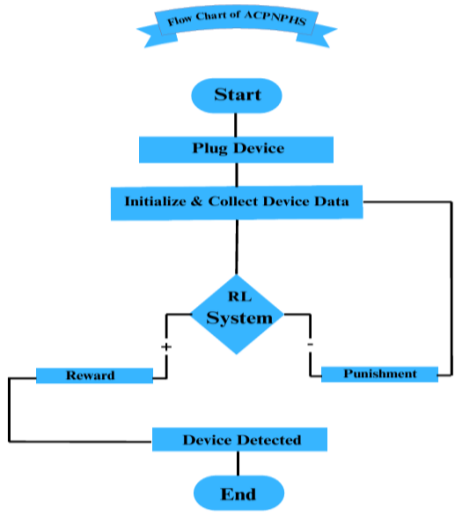

Figure 2. Flowchart of the ACPNPHS system showing device detection, reinforcement learning process, and action execution

Result:

The suggested ACPNPHS framework shows excellent experimental performance through an RL-driven adaptive learning process, proving that the system can successfully identify and configure newly connected microcontroller-based IoT devices. The system detected device activity without human programming. As the various sensors and actuators were connected to the microcontroller, the system automatically determined the input/output patterns of the various sensors and actuators, and the optimal course of action was selected.

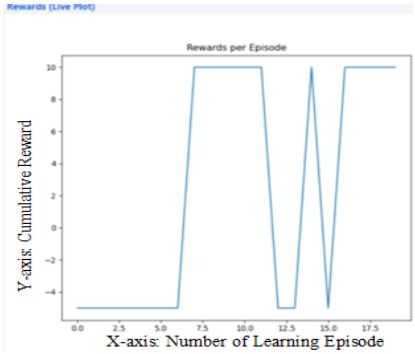

Figure 3. Cumulative reward versus learning episodes for the ultrasonic sensor during reinforcement learning training.

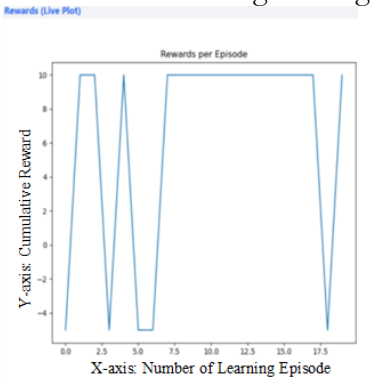

Figure 4. Cumulative reward versus learning episodes for the ultrasonic sensor during reinforcement learning training.

The reward graph in Figures 3 and Figure 4 illustrates that the Y-axis represents the overall reward, indicating how well the agent performed throughout each episode, while the X-axis represents the number of learning episodes, each of which included several device action trials. The reward curve was unstable at the beginning because of random exploration, which reflects the RL agent's initial exploration of the device behavior. The curve increased over time before stabilizing, indicating improved Q-value estimates and increased confidence in selecting the correct action.

In the applied reward scheme, a positive reward value of +10 is used when a system correctly recognizes a device and takes the relevant action, and a negative reward value of -5 is used when exploring and choosing the wrong action. Random exploration leads the RL agent to receive multiple negative rewards during the initial learning events. Nonetheless, with continued learning, the rate of positive rewards increases slowly, meaning that the ability of the agent to make a decision and learn the way to use the device is successful.

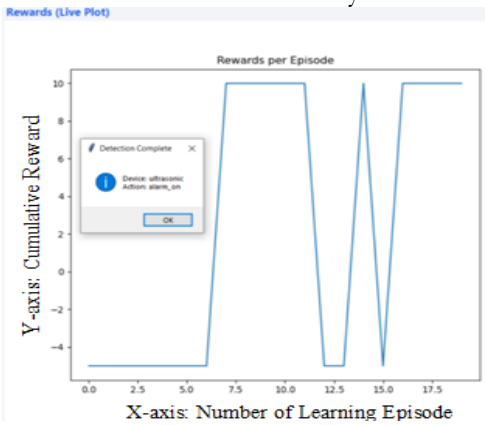

Figure 5. GUI-based detection and automatic action execution for the ultrasonic sensor during real-time IoT device interaction.

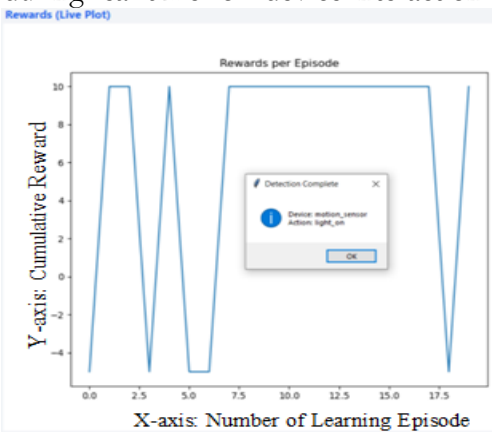

Figure 6. GUI-based detection and automatic action execution for the motion sensor during real-time IoT device interaction.

In Figures 5 and Figure 6 show that two IoT devices, such as an ultrasonic sensor and a motion sensor, enable the RL agent to learn and execute different actions on both devices, demonstrating hardware-level plug-and-play learning. Furthermore, the system immediately displays a pop-up message showing the device name and the executed action whenever a device is detected. Figures 5 and Figure 6 record GUI messages that verify the system's accurate identification of the hardware and autonomous execution of the corresponding action, show this real-time feedback. The integration of the reward graph and live device-action notifications demonstrates both the plug-and-play capability of ACPNPHS in real hardware scenarios and the learning efficiency of the RL agent. The RL agent

performs exploration during the early training episodes to determine the correct actions for each detected device.

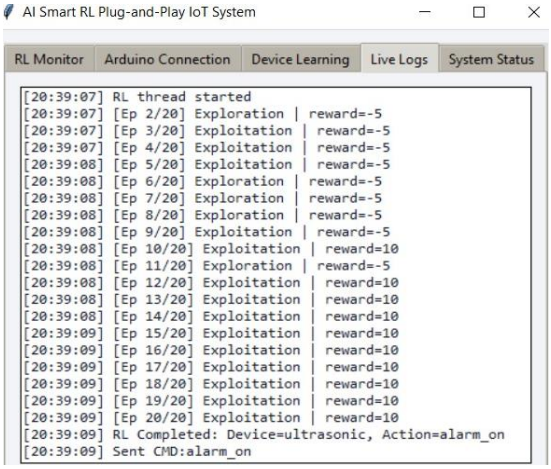

Figure 7. Live system log showing episode number, selected action, and reward values during RL training for the ultrasonic sensor

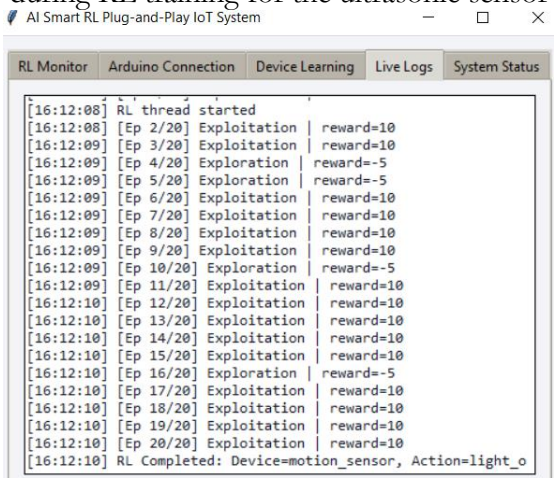

Figure 8. Live system log showing episode number, selected action, and reward values during RL training for the motion sensor.

The live logs capture this process in real time, as shown in Figures 7 and Figure 8. They display the episode number, the identified device, the action selected by the agent, and the reward that was received. The records of the initial exploration experiences indicate the use of different actions and rewards, indicating the agent's efforts to learn how to operate the device by means of trial and error. As the training progresses, the agent progressively relies on learned Q-values, and the live logs begin to show the correct actions with increasingly stable rewards. A reward of +10 is assigned for correct device recognition and successful action execution, while a reward of -5 is assigned for incorrect actions during exploration. The frequency of positive rewards increases across learning episodes, which means that there is a progressive improvement of the policy. Q-values stabilize as the learning algorithm converges toward an optimal policy to an optimal action-selection strategy goes hand in hand with this trend. The live logs provide an interactive display of the learning process, showing how ACPNPHS maps device features to optimal actions using reinforcement learning with real-time user feedback. These patterns were confirmed by the GUI logs, which indicated the events of real-time device detection, chosen actions, the computation of rewards, and updates in the Q-table. Once the system had reached the stable learning point, it reliably showed accurate responses, including turning on the buzzer, turning on the light, or responding to sensor data, depending on the hardware that was

detected. In comparison to previous studies, which are primarily based on simulation or teleoperation environments, the proposed ACPNPHS system offers practical and hardware-based validation. The visual results were not compared with the previous studies, and thus, the evaluation here is completely done based on visual results generated by the system, which includes the live reward graph, detection logs, Q value updates, and GUI screenshots. In general, the findings validate that the ACPNPHS framework provides effective plug-and-play hardware automation. The reward pattern stabilization, exact device detection, and proper action execution are all that prove the RL-powered system can be effective to minimize manual configuration and enable scalable integration to the real IoT settings.

Table 1. Displays the qualitative comparison with previous studies

Features	Previous Studies	ACPNPHS
Experimental Environment	Simulation/Teleoperation	Real IoT hardware
Device Knowledge	Predefined / Profiles	Unknown devices
Learning Mode	Offline / pre-trained	Online, real-time RL
Manual Configuration	Required	Not required
Hardware-Level Validation	Limited / Not shown	Fully demonstrated
Live Reward Visualization	Not available	Real-time reward graph
GUI-based Feedback	Minimal	Popups, live log monitoring

Table 1 is a qualitative comparison of the proposed ACPNPHS framework and previous plug-and-play systems of the Internet of Things. In contrast to previous techniques that depend on simulation conditions or previously used device profiles, ACP N PHS enables learning real hardware behavior in real time by means of reinforcement learning and automatic configuration of devices. The comparison highlights key differences in in the areas of the experimental environment, awareness of the device, approach to learning, and validation of the system. ACPNPHS uses reinforcement learning to learn the unknown device behavior in real time, unlike previous methods that largely rely on simulation, teleoperation, or predefined device profiles. Another point highlighted in the table is that ACPNPHS does not require any manual configuration and offers hardware-level validation with live reward visualization and GUI-based feedback, which is largely absent in prior studies.

Conclusion and Future Work:

This study shows that reinforcement learning enables IoT systems to achieve plug-and-play functionality. Unlike systems based on fixed code or manual configuration, the ACPNPHS framework learns how devices behave independently and selects the best action based on experience. After observing the patterns of various sensors and actuators, the system learns the expected behavior of individual devices. This simplifies the use of IoT hardware and makes it more adaptive and considerably more flexible compared to traditional rule-based systems.

Overall, the work demonstrates that IoT automation can reduce the need for complex installation or specialized capabilities. A learning-based system can autonomously control devices and improve decision-making over time, and continue to improve as it operates in real time. This enables IoT environments to become fully autonomous and self-managing, capable of servicing smart homes, laboratories, and industrial systems with minimal human involvement.

The system can be expanded to support many more devices simultaneously in the future, as the number of devices it supports increases. Improved device recognition and classification techniques can be added to identify new or unknown devices more accurately. Energy-efficient RL policies can be implemented to reduce power consumption for battery-

powered IoT devices. Faster local processing through edge computing can further accelerate the system by processing data near the devices. Additional security features can be incorporated so the system can automatically detect and block any unusual or unsafe device behavior.

References:

[1] Bessam Abdulrazak, Suvrojoti Paul, "IoT Architecture with Plug and Play for Fast Deployment and System Reliability: AMI Platform," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 2022, [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-09593-1_4

[2] S. V. Majid Abdolshah, Hung Le, Thommen Karimpanal George, Sunil Gupta, Santu Rana, "Plug and Play, Model-Based Reinforcement Learning," *arXiv:2108.08960*, 2021, [Online]. Available: https://arxiv.org/abs/2108.08960

[3] Alessandra Sorrentino, Filippo Cavallo, "A Plug and Play Transparent Communication Layer for Cloud Robotics Architectures," *Robotics*, vol. 9, no. 1, p. 17, 2020, [Online]. Available: https://www.mdpi.com/2218-6581/9/1/17

[4] Z. Xu, K. Wu, W. Zhang, J. Tang, Y. Wang, "PnP-DRL: A Plug-and-Play Deep Reinforcement Learning Approach for Experience-Driven Networking," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2476–2486, 2021, [Online]. Available: https://ieeexplore.ieee.org/document/9454317

[5] F. Y. Zakaria Maamar, Mohamed Sellami, Fatma Masmoudi, Muhammad Asim, Abdul Haseeb, Thar Baker, "A Plug&Play approach for modeling and simulating applications in the era of internet of social things," *IET Smart Cities*, 2021, [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/smc2.12005

[6] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, Radu Timofte, "Plug-and-Play Image Restoration with Deep Denoiser Prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020, [Online]. Available: https://arxiv.org/abs/2008.13751

[7] I. Al Ridhawi, S. Otoum, M. Aloqaily, and A. Boukerche, "Generalizing AI: Challenges and Opportunities for Plug and Play AI Solutions," *IEEE Netw.*, vol. 35, no. 1, pp. 372–379, Mar. 2021, doi: 10.1109/MNET.011.2000371.

[8] R. G. Gavaskar, C. D. Athalye, and K. N. Chaudhury, "On Plug-and-Play Regularization Using Linear Denoisers," *IEEE Trans. Image Process.*, vol. 30, pp. 4802–4813, 2021, doi: 10.1109/TIP.2021.3075092.

[9] U. S. Kamilov, C. A. Bouman, G. T. Buzzard, and B. Wohlberg, "Plug-and-Play Methods for Integrating Physical and Learned Models in Computational Imaging: Theory, algorithms, and applications," *IEEE Signal Process. Mag.*, vol. 40, no. 1, pp. 85–97, Jan. 2023, doi: 10.1109/MSP.2022.3199595.

[10] Yutaka Matsuo, Yann LeCun, "Deep learning, reinforcement learning, and world models," *Neural Networks*, vol. 152, pp. 267–275, 2022, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608022001150

[11] C. Szepesvári, "Algorithms for Reinforcement Learning," *Springer Nat.*, 2010, doi: 10.1007/978-3-031-01551-9.

[12] Chong Wang, Rongkai Zhang, Saiprasad Ravishankar, Bihan Wen, "REPNP: Plug-and-Play with Deep Reinforcement Learning Prior for Robust Image Restoration," *Proc. - Int. Conf. Image Process. ICIP*, 2022, [Online]. Available: https://arxiv.org/abs/2207.12056

[13] D. K. Denis Tarasov, Kirill Brilliantov, "Is Value Functions Estimation with Classification Plug-and-play for Offline Reinforcement Learning?," *arXiv:2406.06309*, 2024, [Online]. Available: https://arxiv.org/abs/2406.06309

[14] Jiajun Chai, Guojun Yin, Zekun Xu, Chuhuai Yue, Yi Jia, Siyu Xia, Xiaohan Wang, Jiwen Jiang, Xiaoguang Li, Chengqi Dong, Hang He, Wei Lin, "RLFactory: A Plug-

and-Play Reinforcement Learning Post-Training Framework for LLM Multi-Turn Tool-Use," *arXiv:2509.06980*, vol. 8, 2025, [Online]. Available: https://arxiv.org/abs/2509.06980

[15] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, Aravind Srinivas, "Reinforcement Learning with Augmented Data," *Adv. Neural Inf. Process. Syst.*, vol. 11, 2020, [Online]. Available: https://arxiv.org/abs/2004.14990

[16] Kaixuan Wei, Angelica Aviles-Rivero, Jingwei Liang, Ying Fu, Carola-Bibiane Schönlieb, Hua Huang, "Tuning-free Plug-and-Play Proximal Algorithm for Inverse Imaging Problems," *37th Int. Conf. Mach. Learn. ICML*, 2020, [Online]. Available: https://arxiv.org/abs/2002.09611

Copyright © by authors and 50Sea. This work is licensed under the Creative Commons Attribution 4.0 International License.