Deep Q-Network Based Rotary Inverted Pendulum System and Its Monitoring on the EdgeX Platform

Ju-Bong Kim
Dept. of Computer Science Engineering
Korea University of Technology and Education
Cheon-an, Korea
jubong1992@gmail.com

Hyun-Kyo Lim
Dept. of Interdisciplinary Program in Creative Engineering
Korea University of Technology and Education
Cheon-an, Korea
glenn89@koreatech.ac.kr

Do-Hyung Kwon
Dept. of Interdisciplinary Program in Creative Engineering
Korea University of Technology and Education
Cheon-an, Korea
dohk@koreatech.ac.kr

Min-Suk Kim
Dept. of Knowledge-converged
Super Brain Convergence Research
Electronics and Telecommunications Research Institute
Dae-jeon, Korea
Mskim16@etri.re.kr

Yong-Geun Hong
Dept. of Knowledge-converged
Super Brain Convergence Research
Electronics and Telecommunications Research Institute
Dae-jeon, Korea
yghong@etri.re.kr

Youn-Hee Han*
Dept. of Computer Science Engineering
Korea University of Technology and Education
Cheon-an, Korea
jubong1992@gmail.com

Abstract: **A rotary inverted pendulum is an unstable and highly nonlinear device and is used as a common model for engineering applications in linear and nonlinear control. In this study, we created a cyber physical system (CPS) to demonstrate that a deep reinforcement learning agent using a rotary inverted pendulum can successfully control a remotely located physical device. The device we created is composed of a cyber environment and physical environment using the Message Queuing Telemetry Transport (MQTT) protocol with an Ethernet connection to connect the cyber environment and the physical environment. The reinforcement learning agent controls the physical device, which is located remotely from the controller and a classical proportional integral derivative (PID) controller is utilized to implement imitation and reinforcement learning and facilitate the learning process. In addition, the control and monitoring system is built on the open source EdgeX platform, so that learning tasks performed near the source of data generation and real-time data emitted from the physical device can be observed while reinforcement learning is performed. From our CPS experimental system, we verify that a deep reinforcement learning agent can control a remotely located real-world device successfully.**

Keywords: **EdgeX, Deep Q-Network (DQN), Deep Reinforcement Learning, Rotary Inverted Pendulum**

I. INTRODUCTION

A cyber space is a digital world created and controlled by a computer program. A physical space exists in the real-world and is controlled by physical laws. A Cyber Physical System (CPS) is a system in which cyber spaces and physical spaces are integrated regardless of scale and level [1]. In this study, we have developed a CPS system where a deep reinforcement learning agent in cyberspace performs a real-time control for an Internet of Things (IoT) device located in physical space. In an IoT environment, all objects are connected to each other on a network. Various IoT devices continuously generate vast amounts of data, and typically, a remote central cloud system is used to store and calculate such large amounts of collected data. However, the use of a remote cloud system is time consuming for data storage and real-time processing. EdgeX is an open-source platform of edge computing technology that overcomes these problems [2], [3]. It solves problems of interoperability caused by using different protocols between IoT devices to connect. Furthermore, unlike the central cloud system, EdgeX does not reside remotely, resulting in a possible reduction in transmission time.

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2018R1A6A1A03025526), and also funded by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2016R1D1A3B03933355).

Corresponding Author: Prof. Youn-Hee Han

In this study, we used the rotary inverted pendulum (RIP), Quanser's QUBE-Servo2 [4] deployed in our edge network to provide real-time control in the CPS system. Since the RIP is a very unstable system and has nonlinear characteristics, it has been widely used as a testbed for nonlinear control systems. In order to control the RIP stably in the control system, many classical and complex mathematical control models, such as full state feedback (FSF), linear quadratic regulator (LQR), and proportional integral derivative (PID), were used [5], [6]. Control using classical control engineering models is difficult if there is no deep understanding of the nonlinear control system. The initial state and the setting of the constant values used in the control equations in these models are particularly important in achieving successful control. In this study, we solved this problem through deep reinforcement learning rather than classical control engineering. The deep reinforcement learning agent with a well-trained neural-network model can cause an RIP to become upright by iterating through a series of cycles to determine the optimal action and observe the state. We also applied imitation and reinforcement learning to improve the learning speed utilizing the PID controller as an expert. From this we induced optimal quality experiences resulting in our reinforcement learning agent facilitating the learning process.

We used the RIP as a networked device. The RIP device is composed of a cyber environment and physical environment, and the Message Queue for Telemetry Transport (MQTT) protocol [7] is used by the Ethernet connection to connect the cyber environment and the physical environment. Through the MQTT broker, sensor data and control data are delivered continuously between the reinforcement learning agent and the RIP device. If the agent is located in a cloud system and all data are exchanged between the RIP device and the agent in the cloud, inherent problems are caused in real-time interaction due to latency caused by congestion. To overcome these problems, we deployed the reinforcement learning agent in an edge computing device in the local network where the RIP device was located. In this way, the learning tasks were performed near the data generation source (the RIP device). We used EdgeX to gather the sensor data emitted from the RIP system and monitor how well deep reinforcement learning was occurring. From the CPS experimental system deployed in the edge network, we verified that a deep reinforcement learning agent in cyberspace can control a remotely located physical device.

In the following sections we describe the overall system; explain the deep reinforcement learning process; propose an imitation and reinforcement learning model that facilitates the learning process; describe the learning process monitoring system based on the EdgeX platform and present experimental results; and finally provide our conclusions.

II. System Overview

In Fig. 1, our overall CPS system is described. The system has two major components, the RIP device and the EdgeX-based control and monitoring system. The sensor data and control data generated from the two components are delivered through the Ethernet switching network. On the EdgeX platform, the deep reinforcement learning agent performs the learning task to control the RIP device and to try to cause the pendulum to become upright. A monitoring system is also implemented on the EdgeX platform and the deep reinforcement learning task is continuously monitored.

The RIP device is equipped with optical encoders that provide feedback on the angular position and angular velocity of the pendulum and the motor. The resolution of the angles of the pendulum and the motor is 512 counts per revolution. It is appropriate to derive the state information corresponding to the motor power input value. A Raspberry Pi 3 Model B (Raspberry Pi) is connected to the RIP device via the serial peripheral interface (SPI) communication method and is also connected to the Ethernet switch. The Raspberry Pi receives the motor power data (i.e., control input) from the reinforcement learning agent and sends it to the RIP device. Additionally, it also receives the angular position and angular velocity of the pendulum and the motor (i.e., state information) from the RIP device, and sends these data to the agent located in the EdgeX platform.

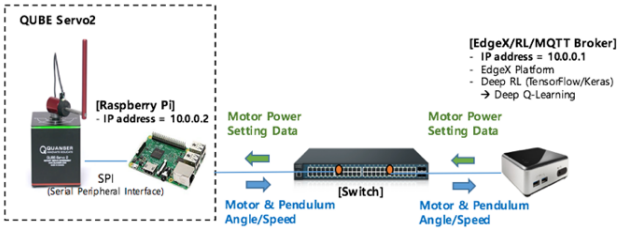

Figure 1. The overall system configuration

Deep reinforcement learning has been predominantly studied in arcade game simulation environments [8]. In such environments, all components are located in a cyber space and implemented as a kind of software. However, as shown in Fig. 2, the RIP device is deployed as a tangible product in the physical space and the reinforcement learning agent is located in the cyber space. Therefore, it is important to make the two components interoperable in a real-time manner. In addition, it is also important that the reinforcement learning agent should not be extensively modified to control the physical device. That is, the reinforcement learning agent should perform the control task in the same way regardless of whether the object to be controlled exists in the cyber space or in the physical space.

We configured a cyber environment and made it correspond to the physical environment so that the reinforcement learning agent observed the state of the RIP device and controlled it only through the cyber environment. The agent does not know that the RIP device is located in the physical space when observing and controlling it. We used MQTT as the communication protocol between the physical environment and the cyber environment. MQTT is a publish/subscribe protocol created for interoperability with machine to machine (M2M) or IoT devices. In our system, the MQTT broker is also placed on the same device as the EdgeX platform, and all sensing and control data are delivered via this MQTT broker.

The cyber environment provides the deep reinforcement learning agent with the three functions: 1) reset, 2) step; and 3) close. The three functions are used for the reinforcement learning task and are described as follows:

1) reset: The reinforcement learning agent calls this function when it starts a new cycle during which the PID device is controlled and the pendulum attempts to become upright. When this function is called, the RIP device in the physical environment is reset and its initial state is returned to the agent via the cyber environment.

2) step: During an episode, the agent calls this function repeatedly in order to send the control input to the RIP device and receive the state information from the RIP device. The agent in the cyber environment also generates reward information when it receives the state information from the RIP device. The reward information is crucial information for the agent's learning task.

3) close: The agent calls this function when a cycle comes to end, so that the state of the RIP device is no longer delivered to the agent.

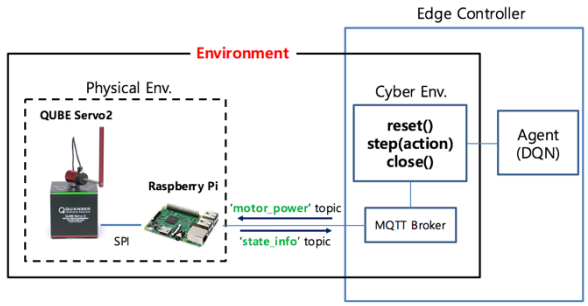

Figure 2. The reinforcement learning environment as a CPS system

III. Deep Reinforcement Learning for Controlling the RIP Device

At each step, the reinforcement learning agent observes a state (s) of the environment (i.e., the RIP device), inputs an action (a) based on its policy network (i.e., a neural network model) into the environment, and then receives the reward (r) for the action and the next state (s') from the environment. The goal of the agent is to interact with the environment and maximize the cumulative reward value of the actions. In this process, the action-value function is used to select an appropriate action and usually is derived using a nonlinear function approximation such as neural network.

Since the system attempts to approximate a nonlinear function, it is inherently unstable and divergent. Double Deep Q-Network (DQN) has been previously studied and has been found to resolve this problem [9], [10]. In Double DQN, the transitions $<s, a, r, s'>$ obtained at every step are stored in the experience memory and then a mini-batch is randomly sampled and used for optimizing the parameters in the main neural network. By using the experience memory, we can break the similarity of the sequential input and increase learning. In addition, a target network is built by copying its structure and parameters from the main network. The parameters of the target network are updated at regular intervals from the main network. The target network can help prevent non-stationary targets from occurring. The experience memory and the target network help to stabilize the approximation of the action-value function.

A. State

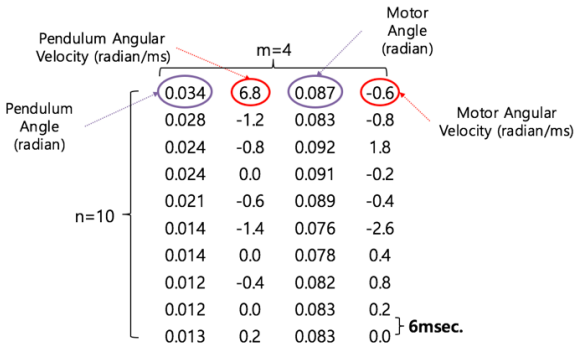

Figure 3. Time series state information

Fig. 3 depicts the time series state information delivered from the RIP device. The state consists of n time series units. A unit contains four pieces of information: 1) pendulum angle, 2) pendulum angular velocity, 3) motor angle; and 4) motor angular velocity. The n time series units represent a historic state of the RIP device. In our study, the gap between each unit is 6 msec. and n is 10, therefore a state describes the change of a unit during 60 msec. This change information is used as input for our deep neural network.

B. Deep Q-Network

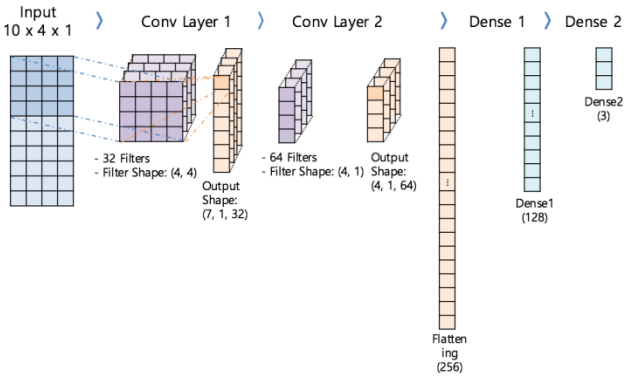

Figure 4. Deep Q-network based on convolutional neural network

Fig. 4 describes the convolutional neural network (CNN) architecture used in our reinforcement learning process. The input state can be expressed as a 10×4 pixel image with one channel. The first hidden layer contains thirty-two 4×4 filters and the stride is 1. The second hidden layer contains 64 4×1 filters and the stride is 1. The next hidden layer is a fully connected linear layer and contains 256 rectifier units. The last hidden layer contains 128 rectifier units and it is also fully connected to the previous layer. The output layer contains just three units and the resulting values for the possible three actions are calculated from the output layer.

C. Actions and Reward

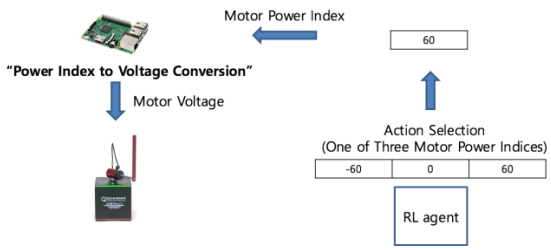

Figure 5. Action selection and the delivery to the RIP device

Fig. 5 describes the action selection and the delivery to the RIP device. The action is selected by the agent from the convolutional neural network and the selected action information is used to control the motor of the RIP device in the physical environment. The three possible motor power indices are -60, 0, and 60. These absolute values correspond to the force that drives the motor. The negative sign indicates the turning direction of the motor was from right to left, while the positive sign means the turning direction of the motor was from left to right. Motor power indices are converted to motor voltage values inside the Raspberry Pi. These voltage values turn the motor of the RIP device.

The reward information is crucial for the agent's reinforcement learning task. The reward information is determined by the cyber environment at every step. The reward value is dependent on the success or failure of each step. If the pendulum is in an upright range of ± 7.5 degrees (i.e., the pendulum is vertically erected) at a step, the step is successful. Otherwise, the step has failed. If the step is successful, the reward value is 1 and the episode continues. If the step fails, the reward value is -100, and the episode comes to end. When an episode comes to end, a score is assigned from the sum of the reward values of all the steps during an episode.

D. Imitation Learning

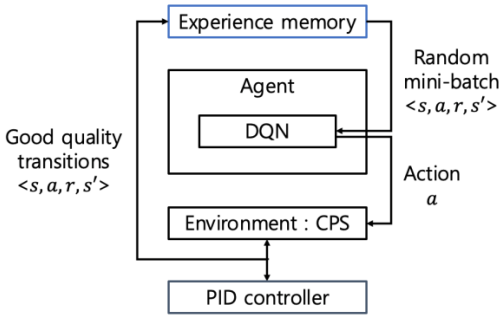

Figure 6. Process to populate the experience memory with good quality transitions

At the beginning of every episode, the initial state of the RIP device changes from time to time due to the inertia of the pendulum and noisy disturbance. It makes the Double DQN learning task difficult. In addition, the learning speed is slow since there is variation in the condition of the physical environment in real time during learning. In particular, since our system's agent is located away from the actual RIP device, time series state information may lag. This makes it difficult to control the RIP device through the learning task. As a solution to this, an imitation reinforcement learning method is applied into our Double DQN strategy.

We attach a stable PID controller into the Double DQN learning process in our cyber environment and let it control the RIP device for a period at the beginning of learning task. During this period, good quality transitions $<s, a, r, s'>$ are obtained and filled into the Double DQN's experience memory. When the experience memory is completely filled, the PID controller is removed and it is replaced with the Double DQN agent. From that time, the Double DQN agent begins to learn the environment and make its model more robust than the stable PID controller.

The communication sequence is classified into three phases according to the state of the pendulum. The first is the upswing control phase. It is the process of applying force to the motor until the pendulum reaches the position where its angle is perpendicular to the ground. The second is the imitation learning phase. At this phase, the experience memory is filled with good quality transitions to balance the pendulum upright. The final step is the actual reinforcement learning phase. During this phase, the agent attempts to balance the pendulum upright by selecting an action from the Double DQN's action-value function (i.e., the convolutional neural network).

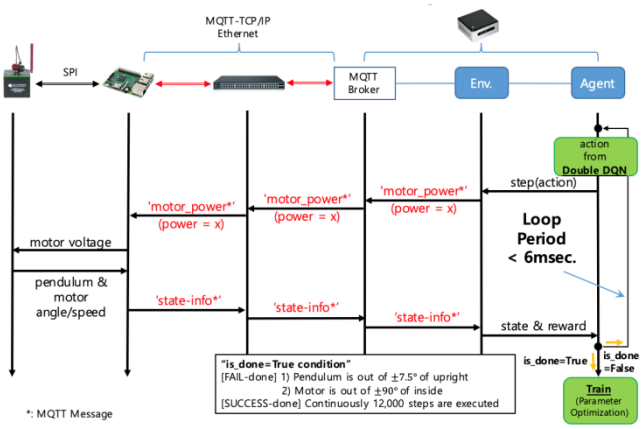

Figure 7. Balance control via the Double DQN agent

Fig. 7 depicts a communication sequence diagram of the final actual learning phase. This phase contains several steps one of which is an iterative loop. The loop is processed every 6 msec. in this phase. The steps of the loop are: 1) the agent selects an action by inserting the previous state values into its model (i.e., convolutional neural network), 2) the selected action is delivered to the RIP device (i.e., the physical environment) via the MQTT broker, 3) the RIP device's motor is turned according to the action value, 4) the state information of the RIP device is delivered to the cyber environment via the MQTT broker, 5) the reward value is decided inside the cyber environment; and 6) the state and reward values are delivered to the agent. The state information is repeatedly used to select the next action.

There are two ending conditions of the loop, fail-done and success-done. The fail-done condition indicates that the pendulum is $\pm$ 7.5 degrees from perpendicular or the

displacement of motor has deviated by ± 90 degrees. Additionally, if 10,000 steps are continuously processed, the loop comes to an end with a success-done condition.

IV. Device State Monitoring via EdgeX Platform

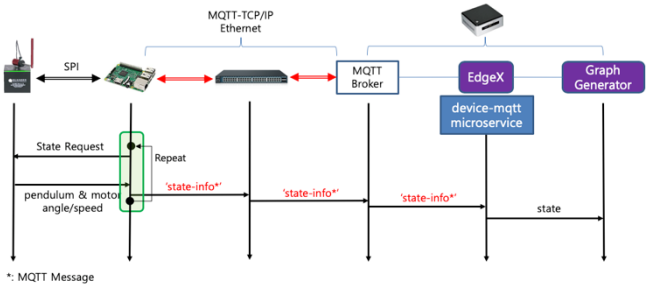

Figure 8. Device monitoring via EdgeX open framework

We used the EdgeX platform to monitor the two sensor values of the RIP device. The sensor values to be monitored were the angular value of the motor and the angular value of the pendulum. In our system, the EdgeX platform gathers the sensor values using the MQTT protocol through the Raspberry Pi connected to the RIP device. The Raspberry Pi is also connected to the one network switch. As an EdgeX client, a monitoring program (i.e., graph generator) receives the sensor values via EdgeX's microservices and renders them into two separate graphs in real-time.

The communication flow is shown in Fig. 8. The sensor values are constantly sent by the RIP device to Raspberry Pi via the SPI interface. Raspberry Pi then delivers the collected sensor values to the MQTT broker every 0.5 seconds through the Ethernet switch. When the MQTT broker receives the sensor values, it passes the values to the EdgeX platform via EdgeX's MQTT device microservice. EdgeX's export microservice is then used to export the sensor value to the monitoring program (i.e., graph generator).

V. Experimental Results

This section presents the results of the learning process of the reinforcement learning agent interacting with the RIP system and the monitoring results via the EdgeX platform.

A. Score and Loss

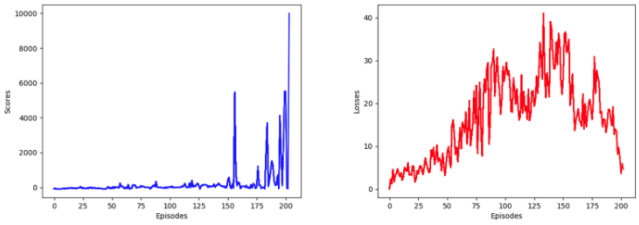

Figure 9. Episodic score and loss graph

In this study, the Double DQN agent had an experience memory of 20,000 transitions $<s, a, r, s'>$, the mini-batch size was 256, the learning rate was 10^{-4}, and the discount factor was 0.99. At the beginning of training, the DQN agent acquired good quality transitions through imitation learning. The score (i.e., the sum of the rewards of all the steps during an episode) increased gradually during an episode. When the experience memory was full, imitation learning came to end and at the end of each episode, CNN optimization was performed by the reinforcement learning agent.

The loss value is a quantitative measure of how bad the performance of the CNN model is. In our experiment, the loss values were low in the initial phase due to the use of imitation learning but increased whenever an optimization was performed at the end of an episode. The CNN parameters were quickly learned by good quality transitions of imitation learning. However, the Double DQN agent generated a score lower than expected. This is explained by the random nature of reinforcement learning and the disturbance caused by a variety of physical noise. We also found that the loss value decreased when the score became high. Finally, the loss value was the lowest when the expected the sum of rewards generated by the action value function became similar to the actual calculated score.

B. Monitoring results on the EdgeX platform

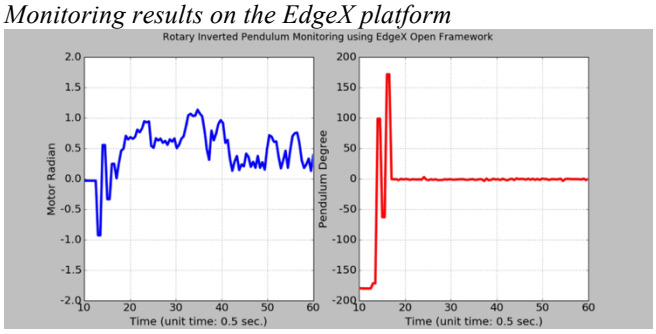

Figure 10. RIP monitoring using the EdgeX open framework

The graphical representation of the two sensor values was drawn on the EdgeX platform and it is shown in Fig. 10. The graph on the left figure shows the angular value (radians) of the motor. The angular value of the motor has a range of the -3.14 to 3.14. By using this information, the scale of the values on the graph was adjusted. The graph on the right shows the angular value (degrees) of the pendulum. The minimum value is -180, while the maximum value is 180. In the initial phase of the experiment, the pendulum angular value was largely shifted due to the upswing control and then tended to vibrate slightly and remained at zero. This indicated that the pendulum was upright.

VI. Conclusion

In order to control the actual nonlinear RIP control system by reinforcement learning, an experimental CPS environment was constructed in our edge network. Through the MQTT protocol, the cyber environment and the physical environment were connected while the state data of the RIP device was directly monitored through the EdgeX platform. After about 200 episodes were completed, the reinforcement learning agent kept the pendulum upright continuously.

There are few studies that use a physical device as the reinforcement learning environment. In this paper, we verified that the reinforcement learning agent can control a physical RIP device successfully. Notably, the device was located remotely from the agent. As a future work, we plan to utilize

asynchronous deep reinforcement learning with multiple remote RIP devices simultaneously.

REFERENCES

[1] E. A. Lee, "Cyber Physical Systems: Design Challenges," *The 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pp. 363-369, May 2008.

[2] H. El-Sayed et al., "Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment," *IEEE Access*, vol. 6, pp. 1706-1717, Dec. 2017.

[3] EdgeX foundry, https://www.edgexfoundry.org.

[4] QUBE_Servo_2_Product_Info_Sheet_v1.0, https://www.quanser.com/products/qube-servo-2/, accessed Sep 20. 2018.

[5] P. Masajedi, A. Ghanbarzadeh, and L. Fatahi, "The Optimization of a Full State Feedback Control System for a Model Helicopter in Longitudinal Movement," *International Journal of Intelligent Information Processing*, *vol. 3*, no. 4, pp.11-20, Dec. 2012.

[6] L. B. Prasad, B. Tyagi, and H. O. Gupta, "Optimal Control of Nonlinear Inverted Pendulum Dynamical System with Disturbance Input using PID Controller & LQR," *IEEE International Conference on Control System, Computing and Engineering*, 2011, pp. 540-545.

[7] M. Singh, M. A. Rajan, V. L. Shivraj and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)," *2015 Fifth International Conference on Communication Systems and Network Technologies*, Gwalior, 2015, pp. 746-751.

[8] V. Mnih, "Playing Atari with Deep Reinforcement Learning," NIPS Deep Learning Workshop, Dec. 2013.

[9] V. Mnih, "Human-level Control Through Deep Reinforcement Learning," *Nature*, *vol. 518*, no. 7540, pp. 529–533, Feb. 2015.

[10] H. v. Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," *The 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016, pp. 2094-2100.