엣지 컴퓨팅 환경에서 강화학습을 이용한 디바이스 제어

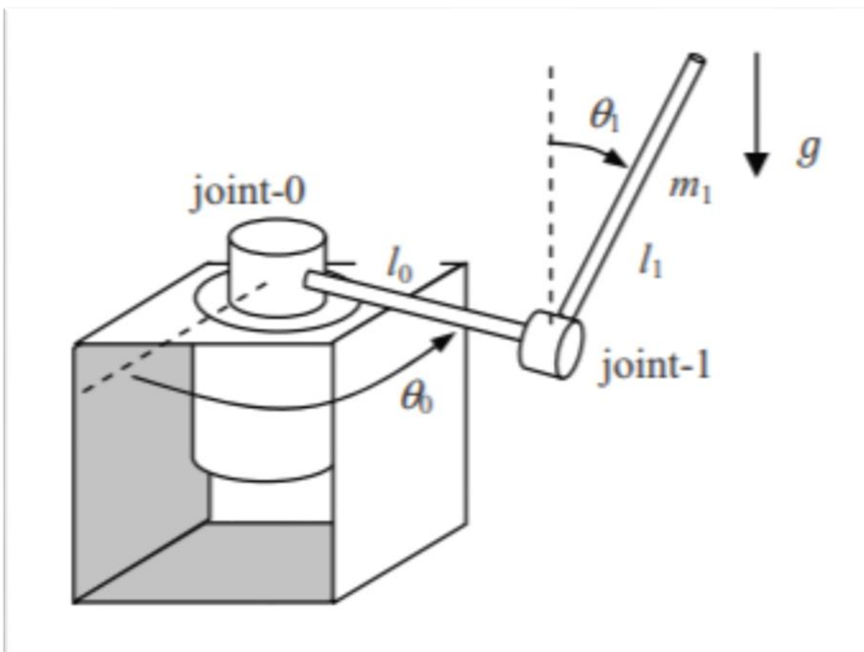
April 2019

Youn-Hee Han
yhhan@koreatech.ac.kr
http://link.koreatech.ac.kr

Rotary Inverted Pendulum

◈ Rotary Inverted Pendulum (RIP)

- a nonlinear dynamical system
- a pendulum mounted on an arm end
- one motor & an arm rotating in the horizontal plane
- 2 joints → 2-DoF (Degree of Freedom)



https://www.quanser.com/products/qube-servo-2/

대표적인 비선형 제어 시스템 → 기계 제어 이론을 접목하기 적합한 플랫폼

Classical Control Engineering

- ◈ RIP Controlled by Control Model in Matlab/Simulink
  - – System Modeling, Dynamic Equation, and Control Algorithm [1995~2010]
    - https://kr.mathworks.com/videos/physical-modeling-building-a-rotary-pendulum-118779.html
    - http://www.seas.upenn.edu/~jiyuehe/rotary-inverted-pendulum/SystemModeling.html

We have

$$\vec{r}_{B/A} = \frac{1}{2} l_2(-\sin\theta_2 \hat{i} + \cos\theta_2 \hat{j}) \Rightarrow \vec{v}_{B/A} = \dot{\vec{r}}_{B/A} = \frac{1}{2} l_2 \dot{\theta}_2(-\cos\theta_2 \hat{i} - \sin\theta_2 \hat{j})$$

And

$$\vec{v}_{A/O} = \dot{\theta}_1 l_1 \hat{i}$$

Thus

$$\vec{v}_{B/O} = \vec{v}_{B/A} + \vec{v}_{A/O} = (\dot{\theta}_1 l_1 - \frac{1}{2} l_2 \dot{\theta}_2 \cos\theta_2)\hat{i} -$$

Select point O the datum for potential energy. So, the potential energies are:

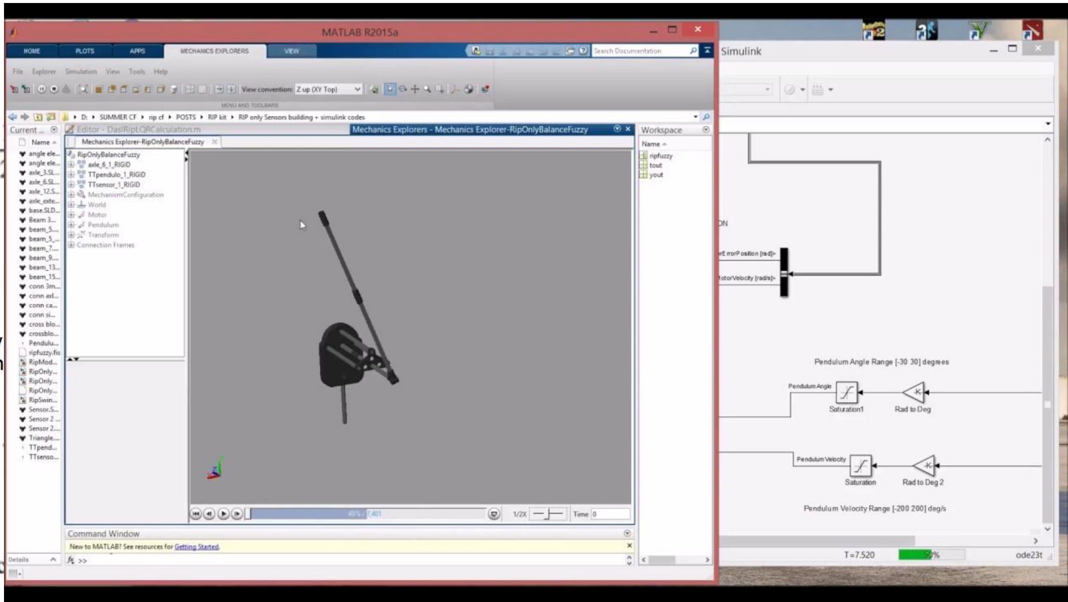
$$V_1 = 0, V_2 = m_2 g(\frac{1}{2} l_2 \cos\theta_2), V = V_1$$

Ignore the 3D printed structures of the arm since they are light. Thus, the arm's kinetic energy pulley, the rotational kinetic energy of the copper rod, and the translational kinetic energy of th mass):

$$T_1 = \frac{1}{2} I_1 \dot{\theta}_1^2 + \frac{1}{2} I_c \dot{\theta}_1^2 + \frac{1}{2} m_c v_{A/O}^2 = \frac{1}{2} \cdot \frac{1}{2} m_1 r_1^2 \cdot \dot{\theta}_1^2 + \frac{1}{2} \cdot$$

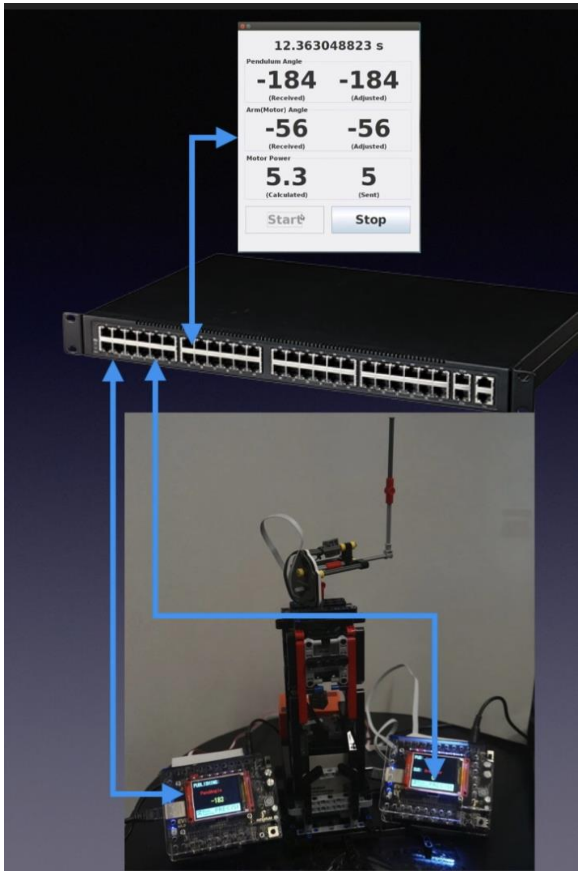
Kinetic energy for link 2 (the pendulum) is:

$$T_2 = \frac{1}{2} I_2 \dot{\theta}_2^2 + \frac{1}{2} m_2 v_{B/O}^2 = \frac{1}{2}\frac{1}{12} m_2 l_2^2 \dot{\theta}_2^2 + \frac{1}{2} m_2[(\dot{\theta}_1 l_1 - \frac{1}{2} l_2 \dot{\theta}_2 c$$



RIP Controlled by Remote Model

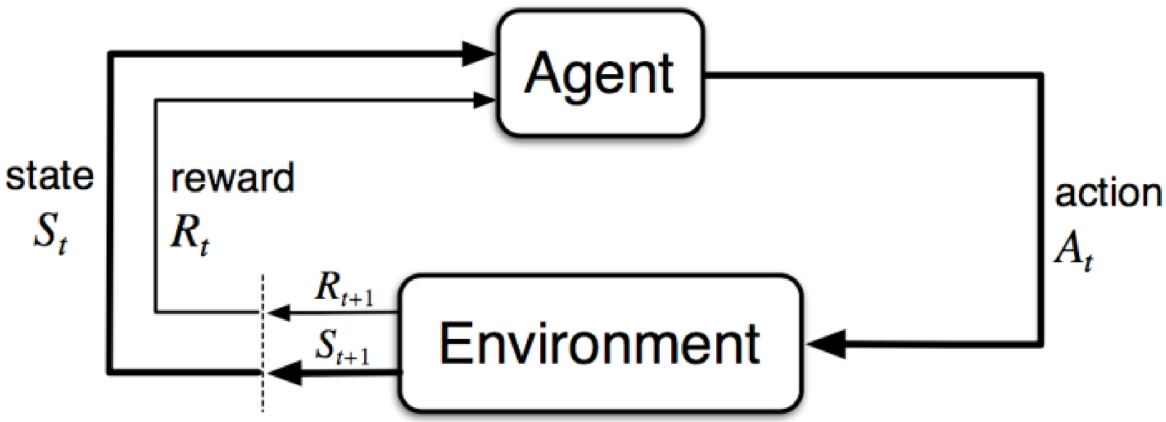
- RIP Controlled in an OpenFlow Network [2014 ~ Current]
  - Controller **located remotely** from the real pendulum system
  - MIDAS (MIDdleware Assurance Substrate)
    - https://repository.upenn.edu/cgi/viewcontent.cgi?article=1821&context=cis_papers



Deep Reinforcement Learning

◈ Deep Reinforcement Learning

심층 강화 학습(Deep Reinforcement learning)은 임의의 **환경** 안에서 정의된 에이전트가 현재의 **상태**를 인식하여, 선택 가능한 **행동**들 중 **보상**을 최대화하는 행동 혹은 행동 순서를 선택하기 위하여 **딥러닝 모델**을 활용하는 방법



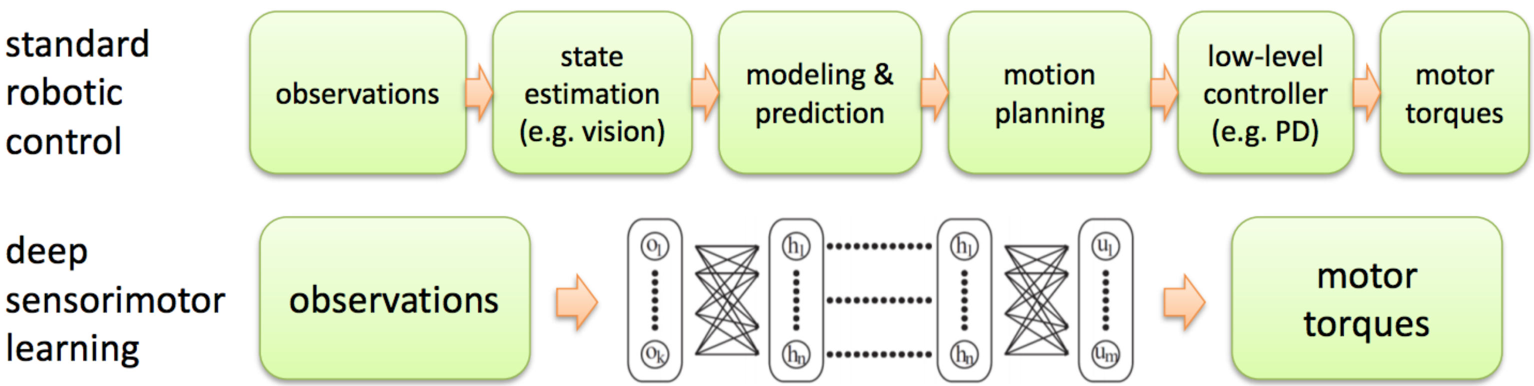
- Deepmind's Atari Breakout with DQN (2015)
  - https://www.youtube.com/watch?v=V1eYniJ0Rnk

Deep Reinforcement Learning for Device Control

Stanford Autonomous Helicopter

Pieter Abbeel et al. "**An Application of Reinforcement Learning to Aerobatic Helicopter Flight**" Advances in Neural Information Processing Systems Conference, 2006.

End-to-end control

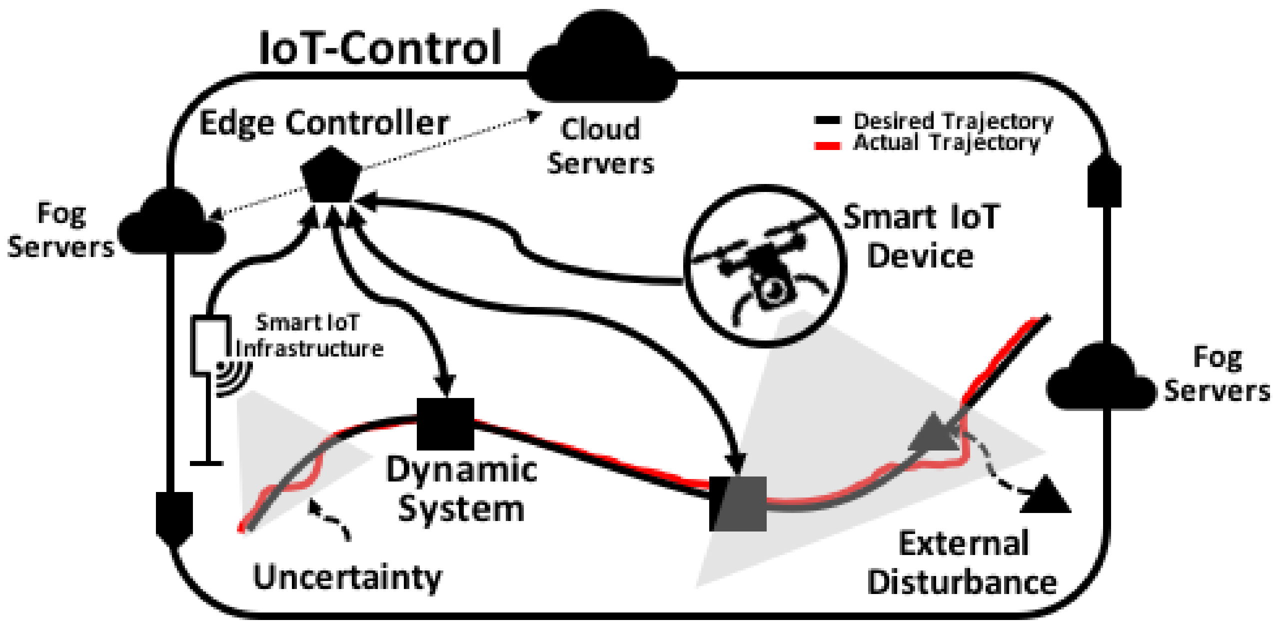


Reinforcement Learning at Edge or Cloud

IEEE Internet of Things

IoT-Control of Dynamic Systems Using Cloud-Fog Machine Learning

Mehdi Roopaei March 14, 2017 University of Texas



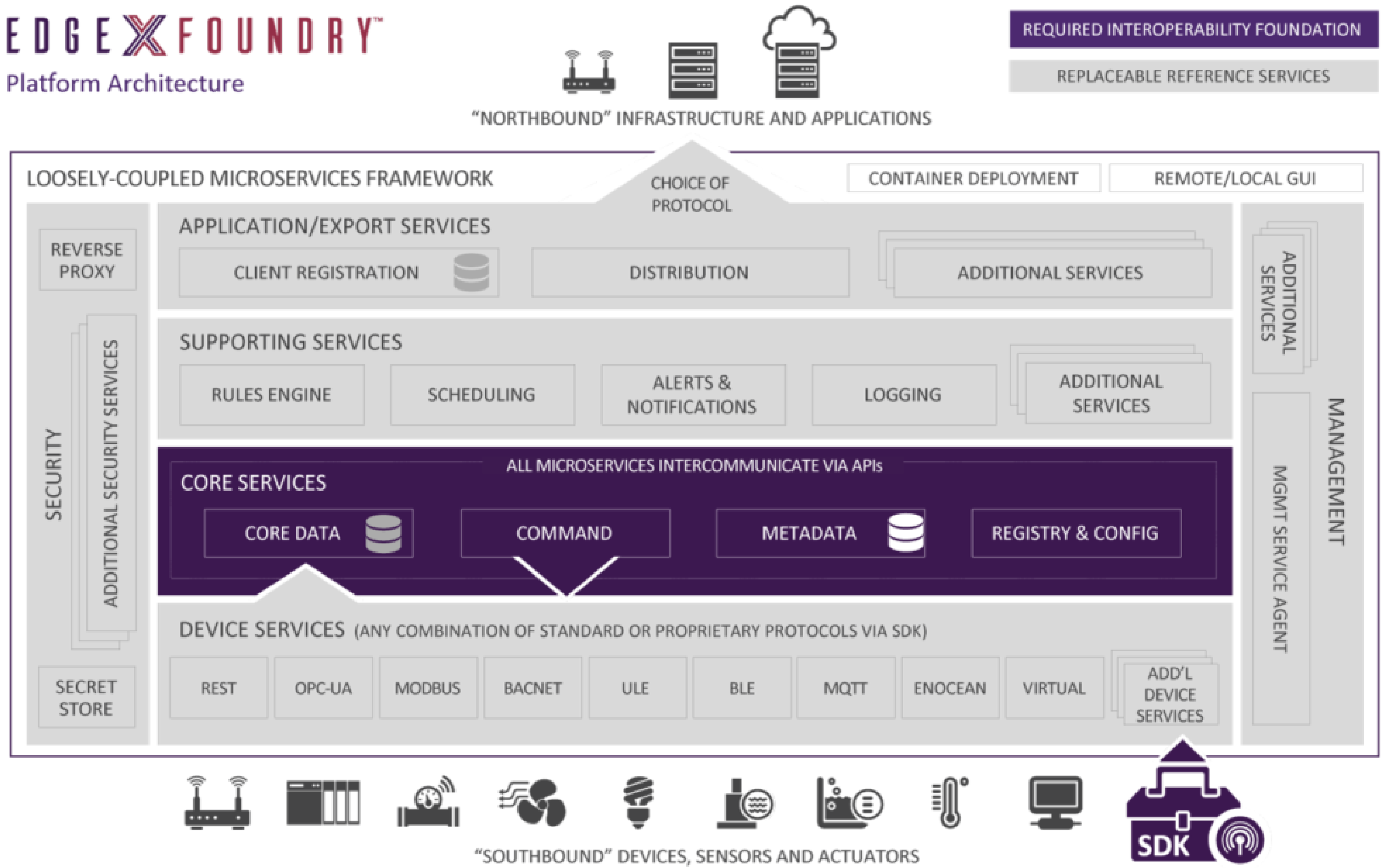
- Control at cloud has inherent challenge in real-time situation due to latency caused by congestion
- Control at edge can provide the stability of the dynamic system against the network fluctuations

https://www.edgexfoundry.org/

EdgeX

The Open Interop Platform for the IoT Edge

Vision: Create a common interoperability framework that enables an ecosystem of plug-and-play "EdgeX certified" components.



Our Target System

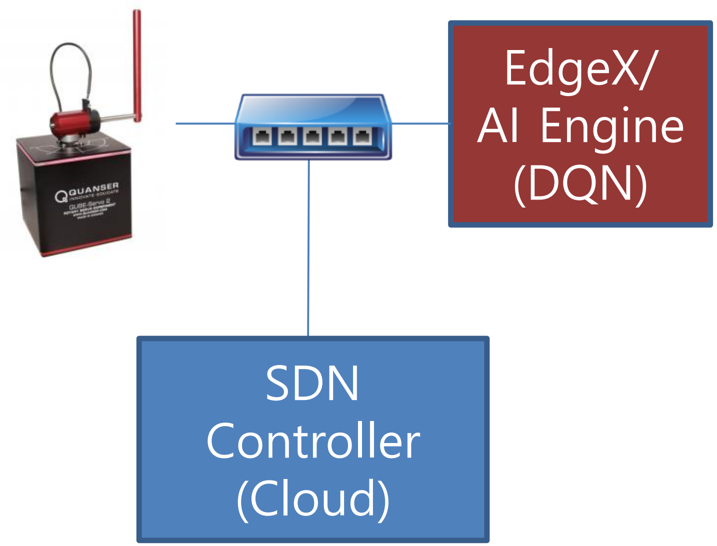
An **Edge-Controlled** Rotary Inverted Pendulum System using **Deep Reinforcement Learning** in an OpenFlow Network

◈ Three Outstanding Features

– 1. **Deep RL-based Control**
  - Model-Free Approach → DQN (Deep Q-Network)

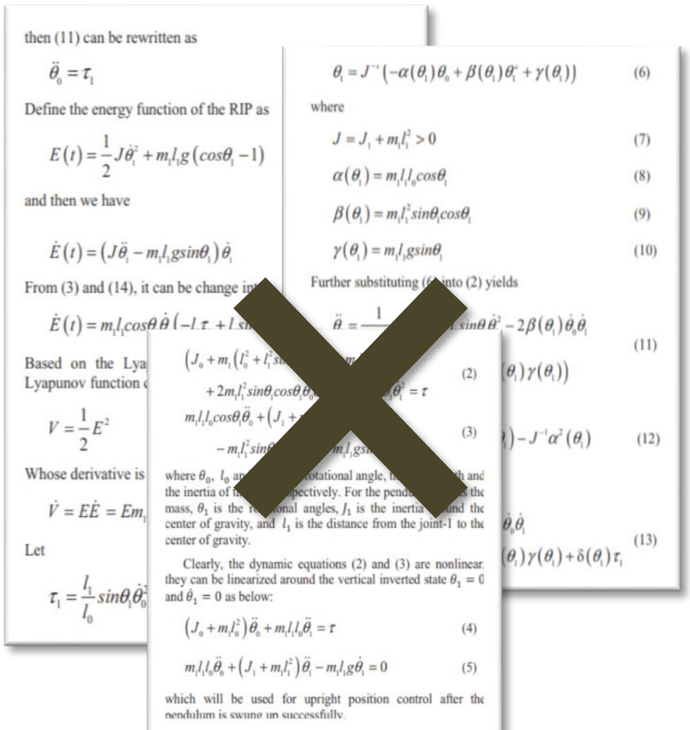
– 2. **Edge-Cotrolled**
  - EdgeX Open Platform

– 3. **Networked Remote Control**
  - SDN-based Switched Network



Deep Reinforcement Learning for RIP Control

◈ RIP Dynamic System Control


then (11) can be rewritten as

$$\ddot{\theta}_0 = \tau_1$$

Define the energy function of the RIP as

$$E(t) = \frac{1}{2} J\dot{\theta}_1^2 + m_1 l_1 g(\cos\theta_1 - 1)$$

and then we have

$$\dot{E}(t) = (J\ddot{\theta}_1 - m_1 l_1 g \sin\theta_1)\dot{\theta}_1$$

$$\theta_1 = J^{-1}(-\alpha(\theta_1)\theta_0 + \beta(\theta_1)\theta_1^2 + \gamma(\theta_1)) \quad (6)$$

where

$$J = J_1 + m_1 l_1^2 > 0 \quad (7)$$

$$\alpha(\theta_1) = m_1 l_1 l_0 \cos\theta_1 \quad (8)$$

$$\beta(\theta_1) = m_1 l_1^2 \sin\theta_1 \cos\theta_1 \quad (9)$$

$$\gamma(\theta_1) = m_1 l_1 g \sin\theta_1 \quad (10)$$

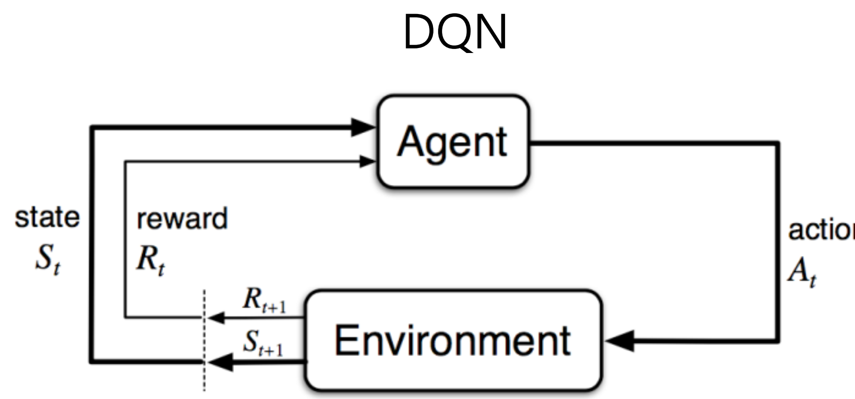
Clearly, the dynamic equations (2) and (3) are nonlinear. they can be linearized around the vertical inverted state $\theta_1 = 0$ and $\dot{\theta}_1 = 0$ as below:

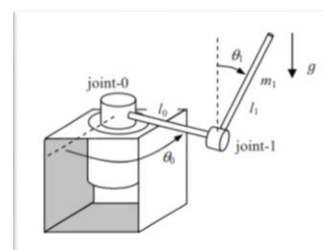
$$(J_0 + m_1 l_0^2)\ddot{\theta}_0 + m_1 l_1 l_0 \ddot{\theta}_1 = \tau \quad (4)$$

$$m_1 l_1 l_0 \ddot{\theta}_0 + (J_1 + m_1 l_1^2)\ddot{\theta}_1 - m_1 l_1 g \theta_1 = 0 \quad (5)$$

which will be used for upright position control after the pendulum is swung up successfully.

Classical Control Model
based on PID and LQR

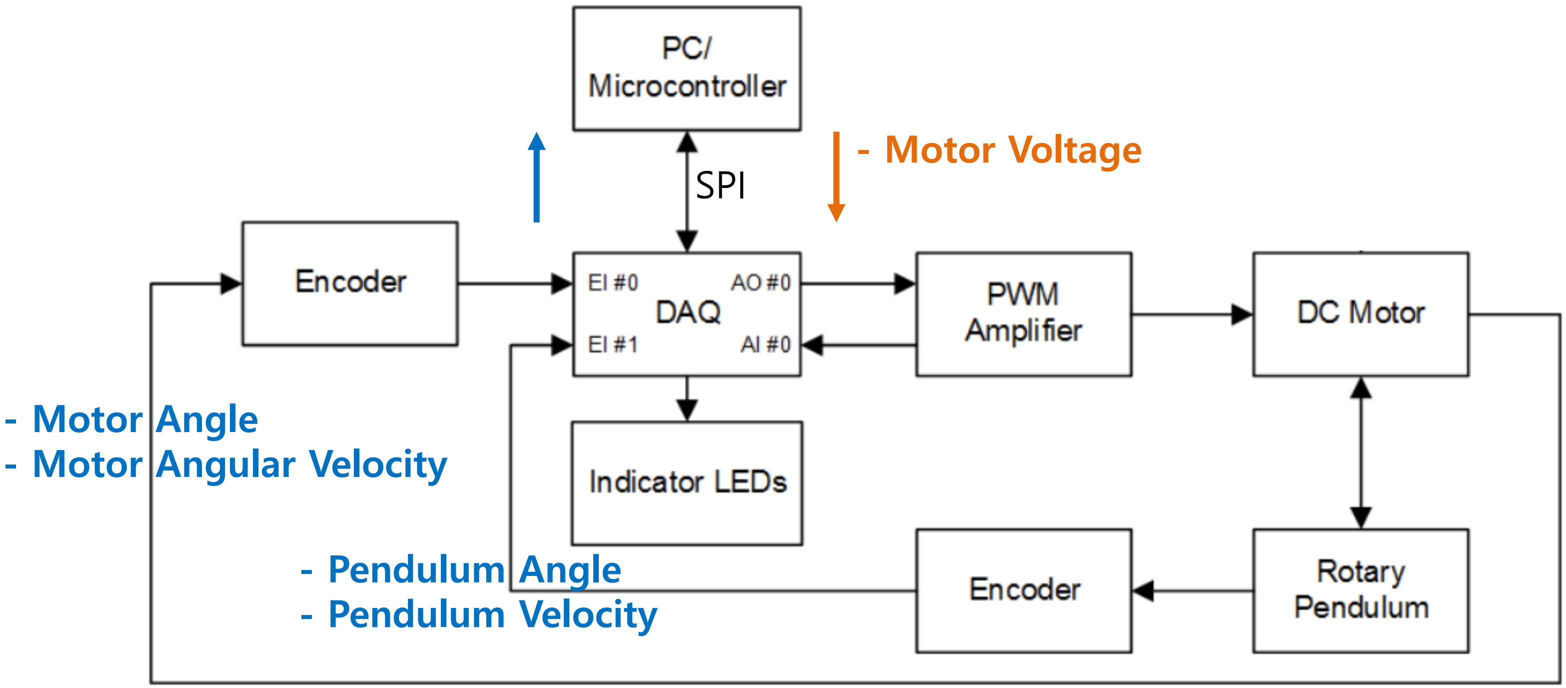




Model-free
Deep Q-Network

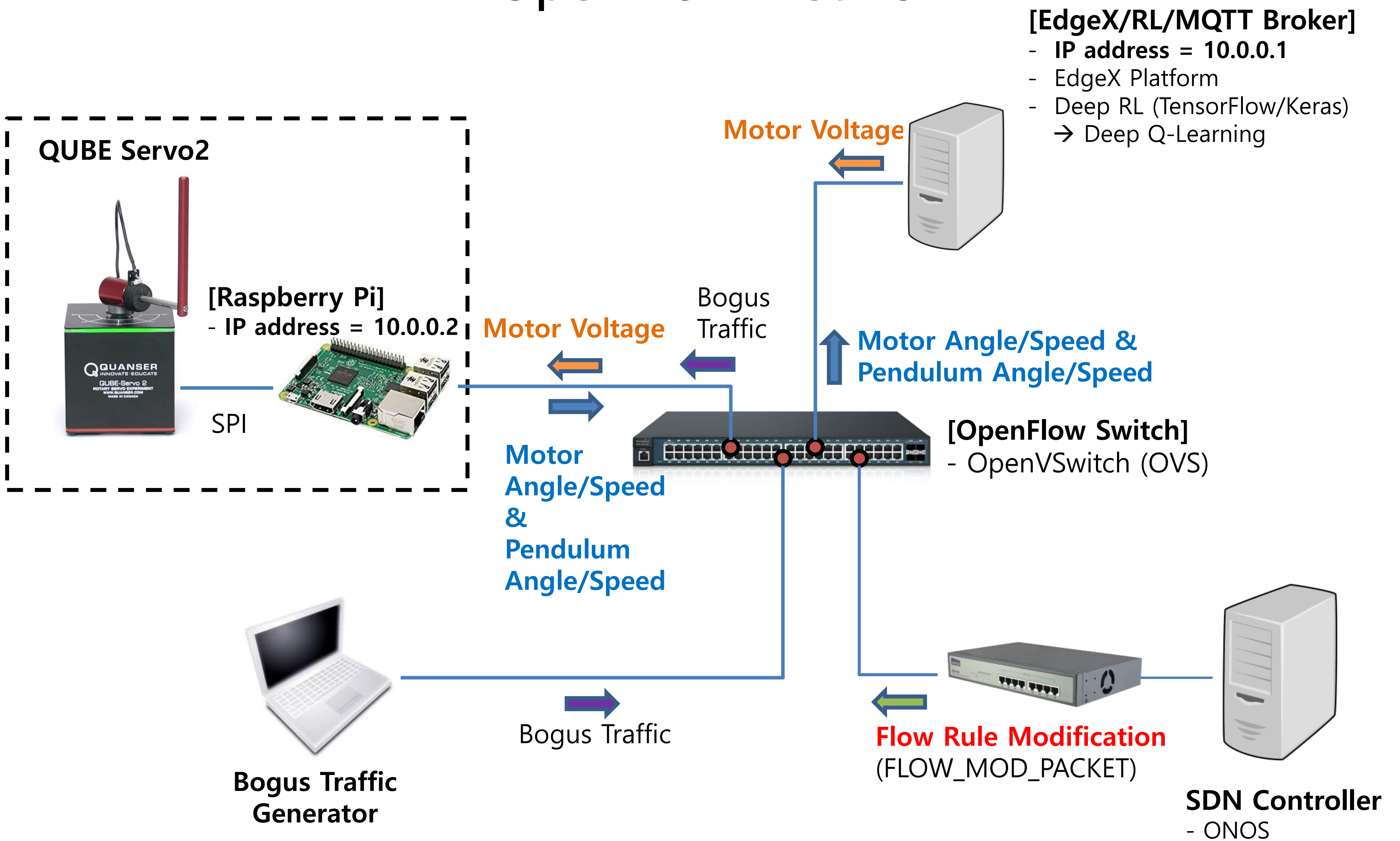
RIP System – QUANSER QUBE Servo2

https://www.quanser.com/products/qube-servo-2/

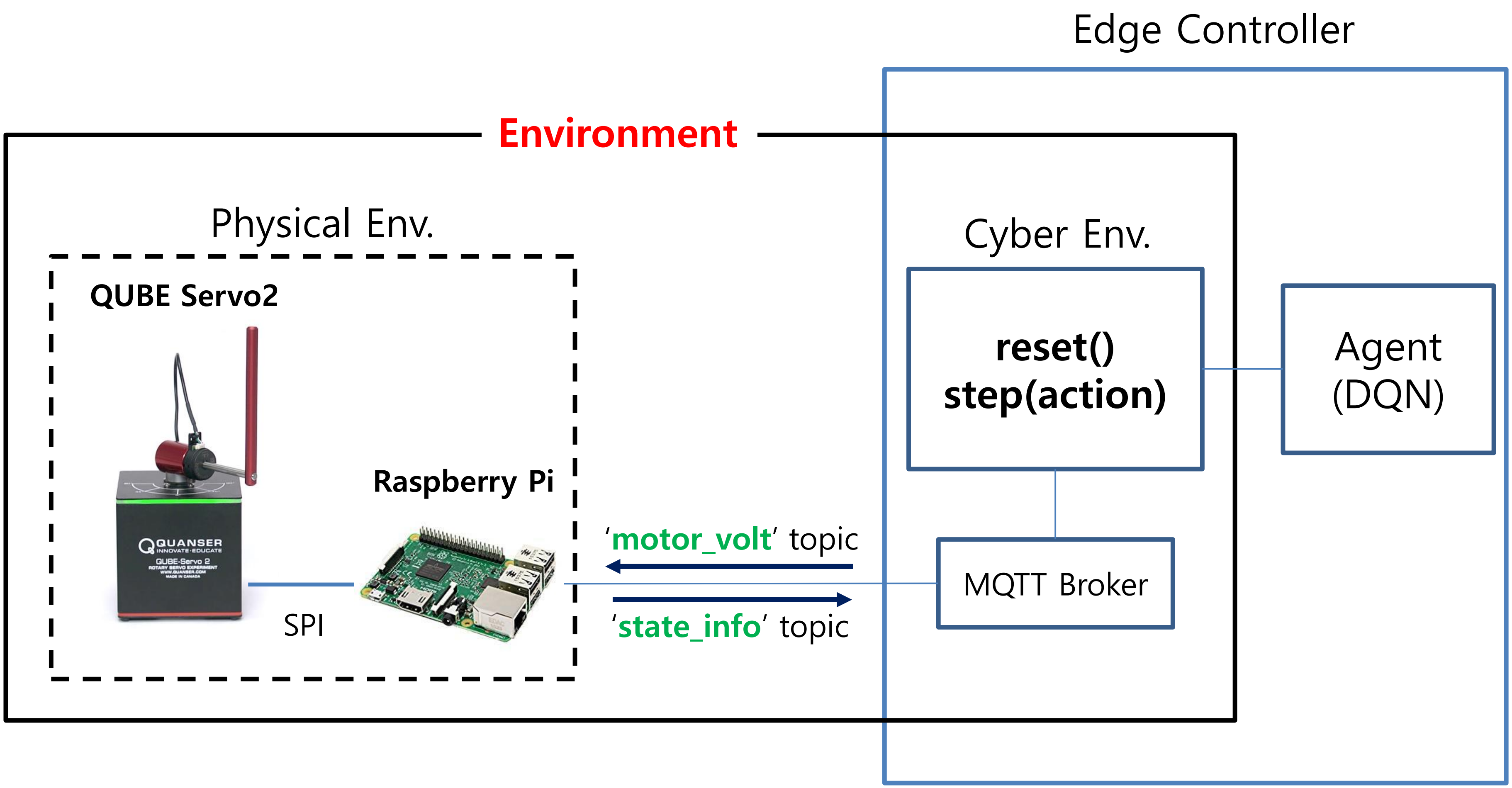


- DAQ (Data Acquisition)
- PWM (Pulse Width Modulation
- SPI (Serial Peripheral Interface)

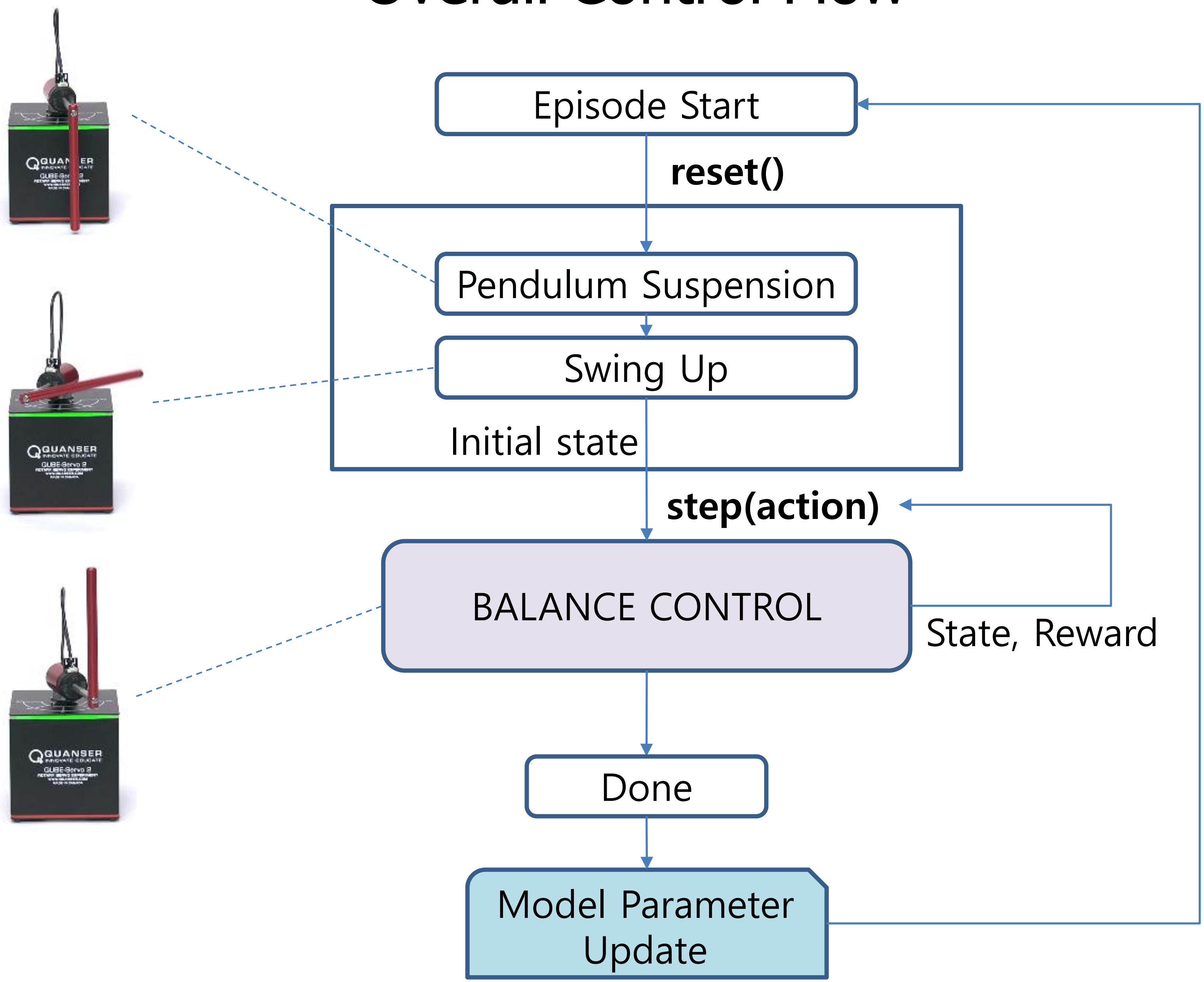
Proposed Distributed System in OpenFlow Network



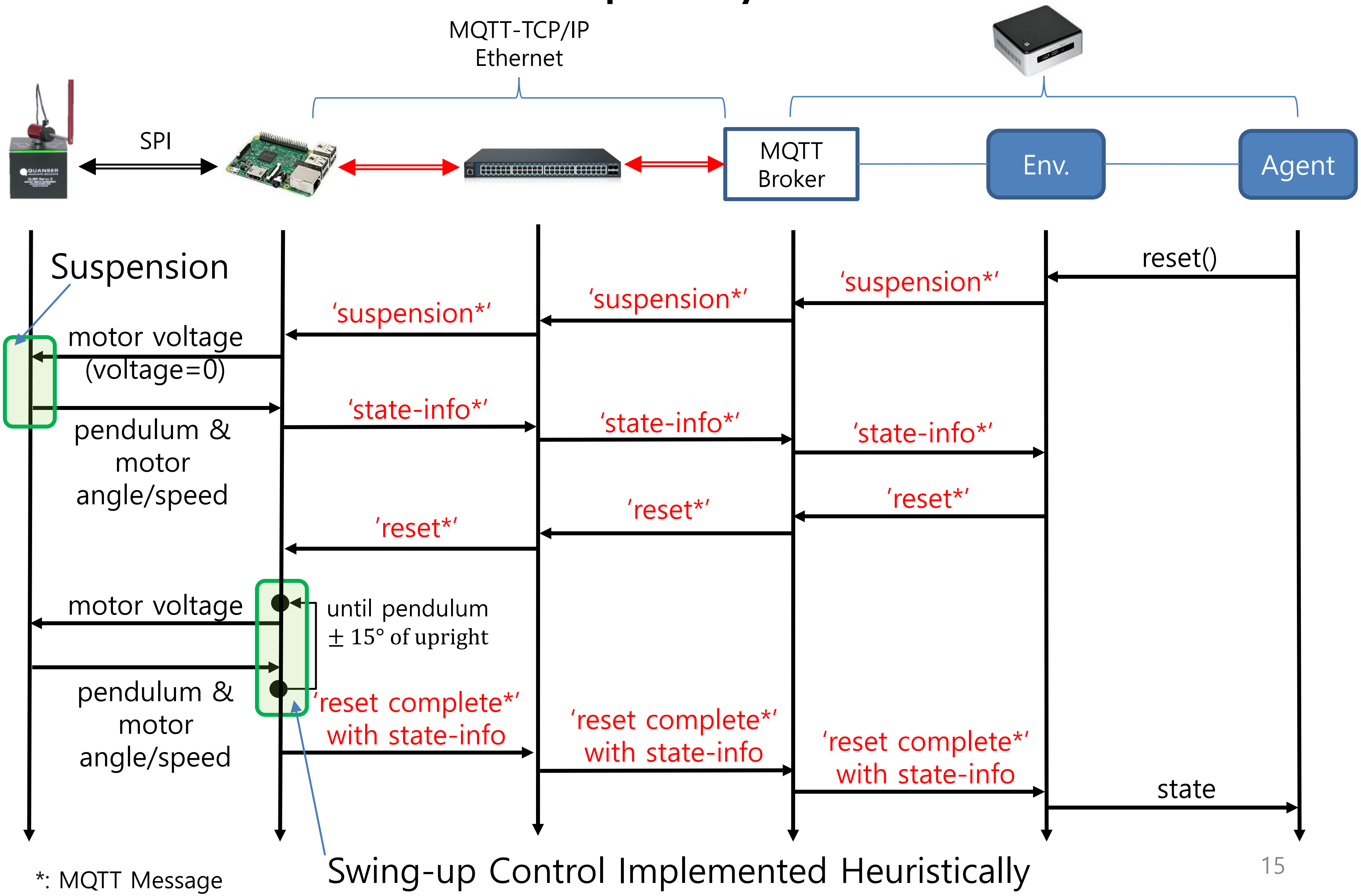
RL Environment: Cyber-Physical System (CPS)



Overall Control Flow



Pendulum Suspension & Swing-Up Control @ Raspberry Pi



*: MQTT Message

State Management at Environment

◈ State Information from the RIP Environment

Pendulum Angle & Velocity

- θ_{k1} ➔ θ'_{k1} $(= \frac{d\theta_{k1}}{dt})$

Motor Angle & Velocity

- θ_{k2} ➔ θ'_{k2} $(= \frac{d\theta_{k2}}{dt})$

단위 상태 정보 (4-elements)

	State Elements
0	θ_{k1}
1	θ'_{k1}
2	θ_{k2}
3	θ'_{k2}

상태 정보 (Time-Series)

n-History	
	4-elements state
0	$(\theta_{01}, \theta'_{01}, \theta_{02}, \theta'_{02})$
1	$(\theta_{11}, \theta'_{11}, \theta_{12}, \theta'_{12})$
…	
n	$(\theta_{n1}, \theta'_{n1}, \theta_{n2}, \theta'_{n2})$

강화학습
엔진
(DQN)

State Management at Environment

◈ States Information (Time-series)

Pendulum Angle (radian)

Pendulum Angular Velocity (radian/ms)

Motor Angle (radian)

Motor Angular Velocity (radian/ms)

m=4

n=10

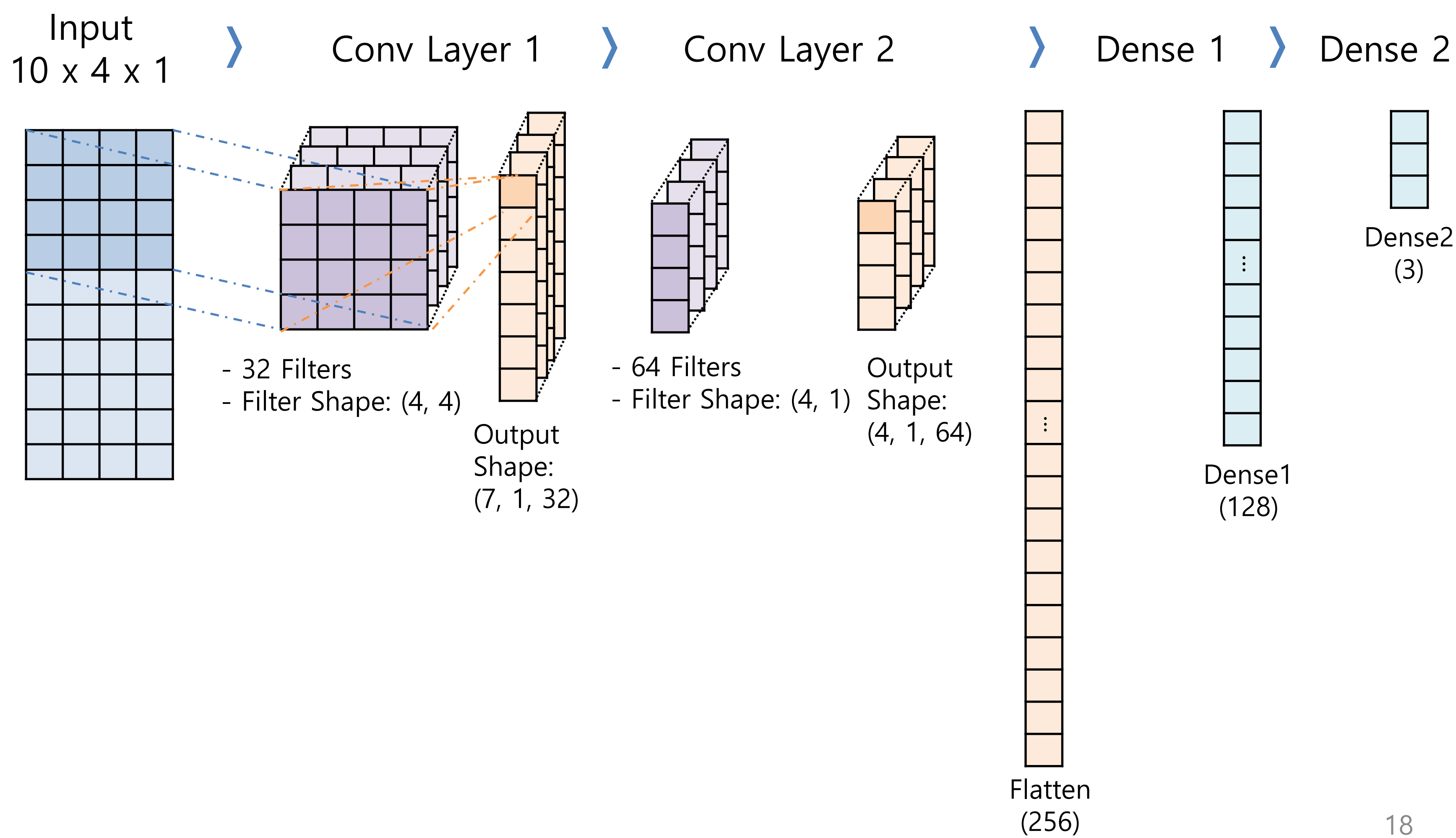
0.034	6.8	0.087	-0.6
0.028	-1.2	0.083	-0.8
0.024	-0.8	0.092	1.8
0.024	0.0	0.091	-0.2
0.021	-0.6	0.089	-0.4
0.014	-1.4	0.076	-2.6
0.014	0.0	0.078	0.4
0.012	-0.4	0.082	0.8
0.012	0.0	0.083	0.2
0.013	0.2	0.083	0.0

7msec.

Dimension & Shape of a State Data: (n, m, 1) → (10, 4, 1)

CNN based Deep Q-Network

◈ CNN (Convolutional Neural Network)



RL Actions

◈ Three Motor Power Indices

Motor Power Index

60

"**Power Index to Voltage Conversion**"

Motor Voltage

Action Selection
(One of Three Motor Power Indices)

-60	0	60

강화학습
엔진
(DQN)

Reward & Score

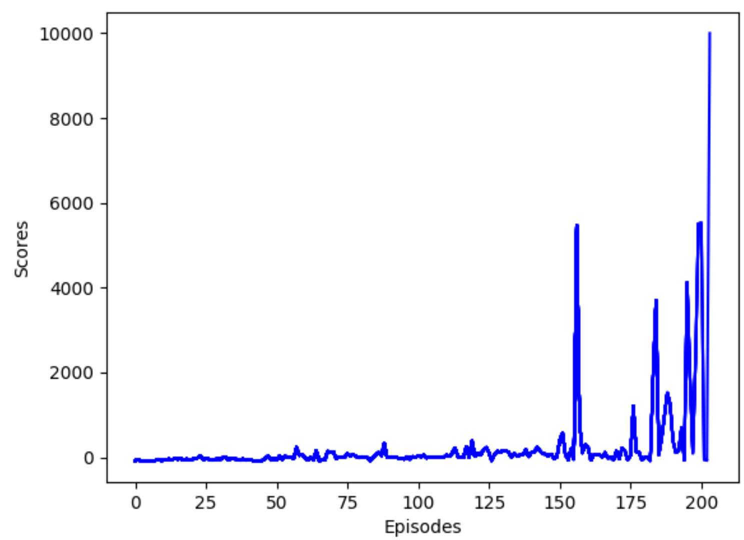
◈ Reward (every step)

- 정상적인 Step 수행한 경우: +1
- Step 수행이 Fail인 경우: -100
  - 1) Pendulum is out of ±7.5° of upright
  - 2) Motor is out of ±90° of inside

◈ Score (every episode)

- 임의의 에피소드 내에서 각 스텝별 Reward의 합

◈ Score Graph



Deep Q-Learning

◈Episode

– Sequence of states, actions and rewards

$$s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T, r_T$$

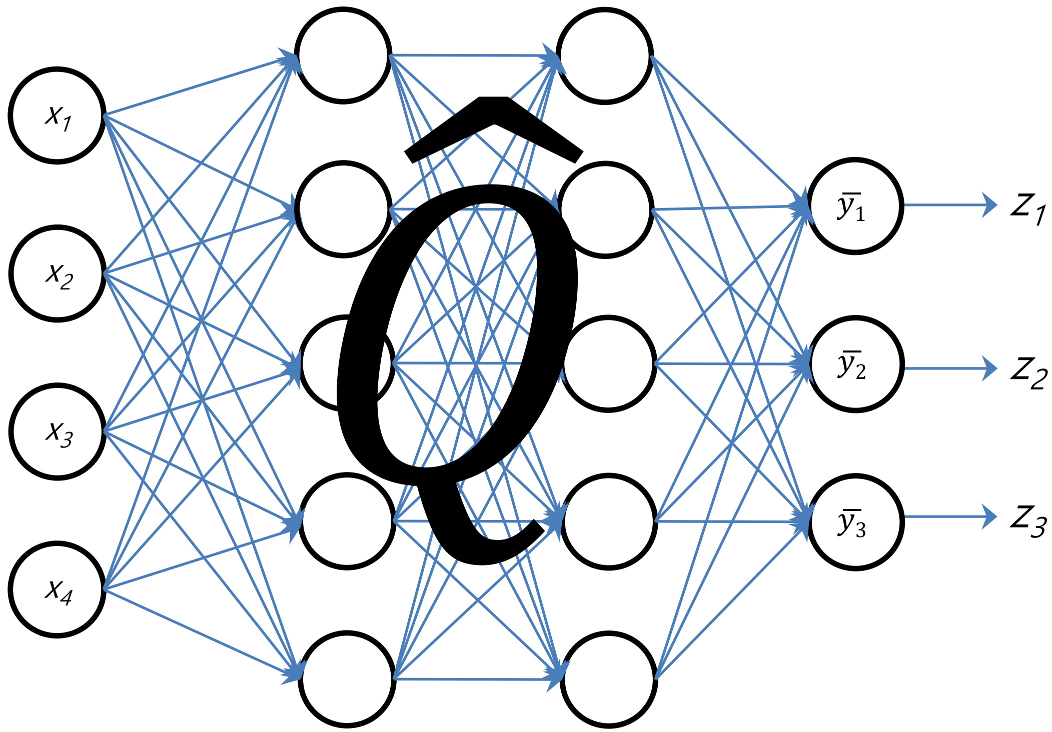
◈Q-Function

– 행동 가치 함수 (Action Value Function)

– 임의의 상태에서 어떤 행동이 얼마나 좋은지 알려주는 함수

Q-function Approximation ➔ Q-Network

State s



$NN(s)$

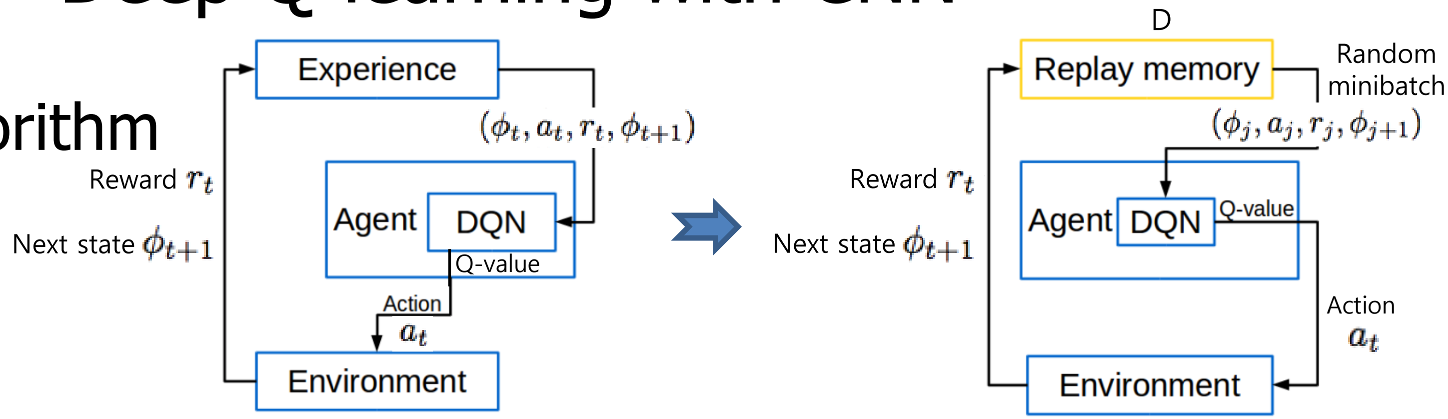
$$Loss = (NN(s) - y)^2$$

$$\boxed{NN(s) \approx \hat{Q}(s)}$$

$$\boxed{\begin{aligned} y &= Q(s) \\ &\approx r + \gamma \max_a \hat{Q}(s', a|\theta) \end{aligned}}$$

Deep Q-learning with CNN

◈ DQN Algorithm



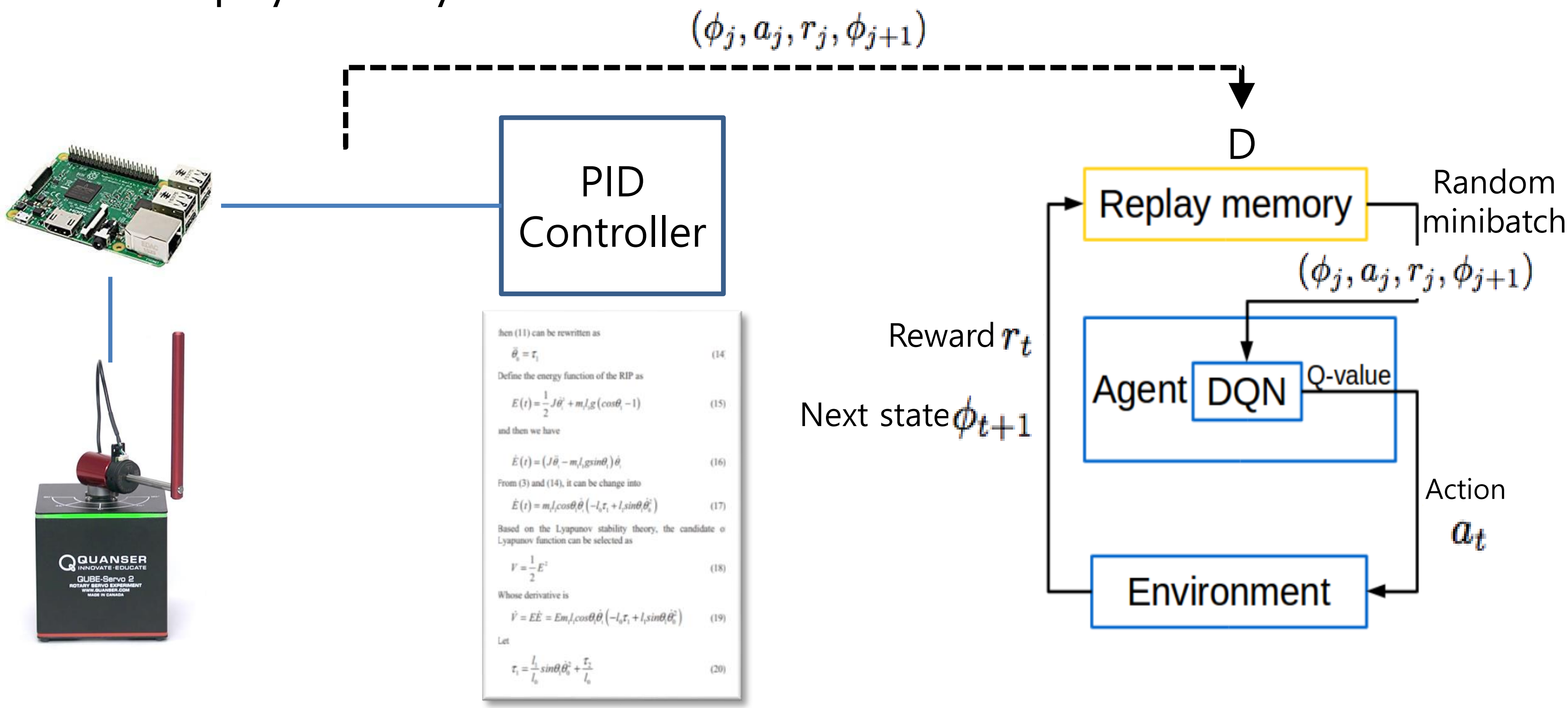
Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Imitational RL (모방 강화 학습)

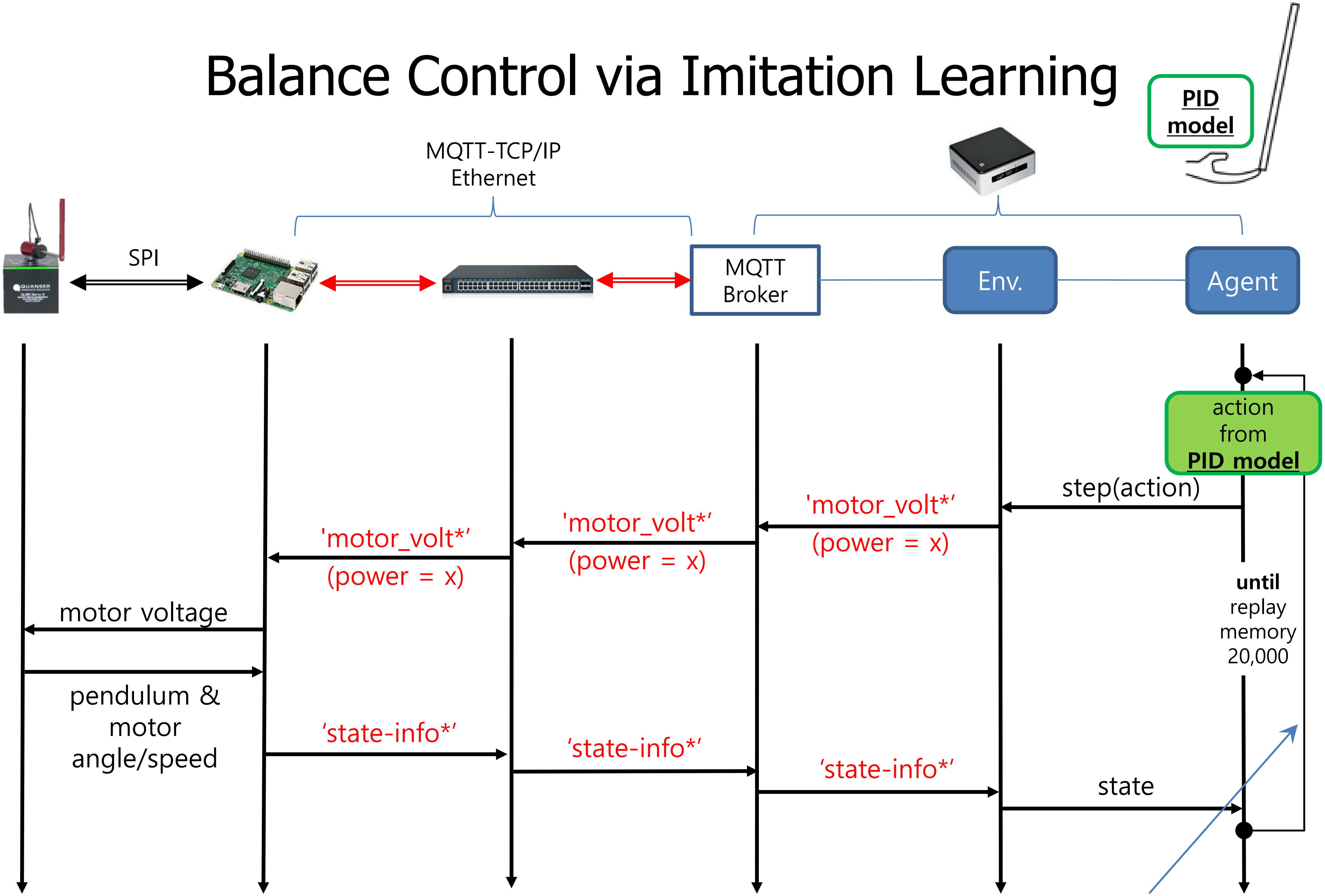
◈ Learning Acceleration

- with help of classical PID control model
- Fill up a large number (20,000) of good transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ into the replay memory



PID (Proportional-Integral-Differential)

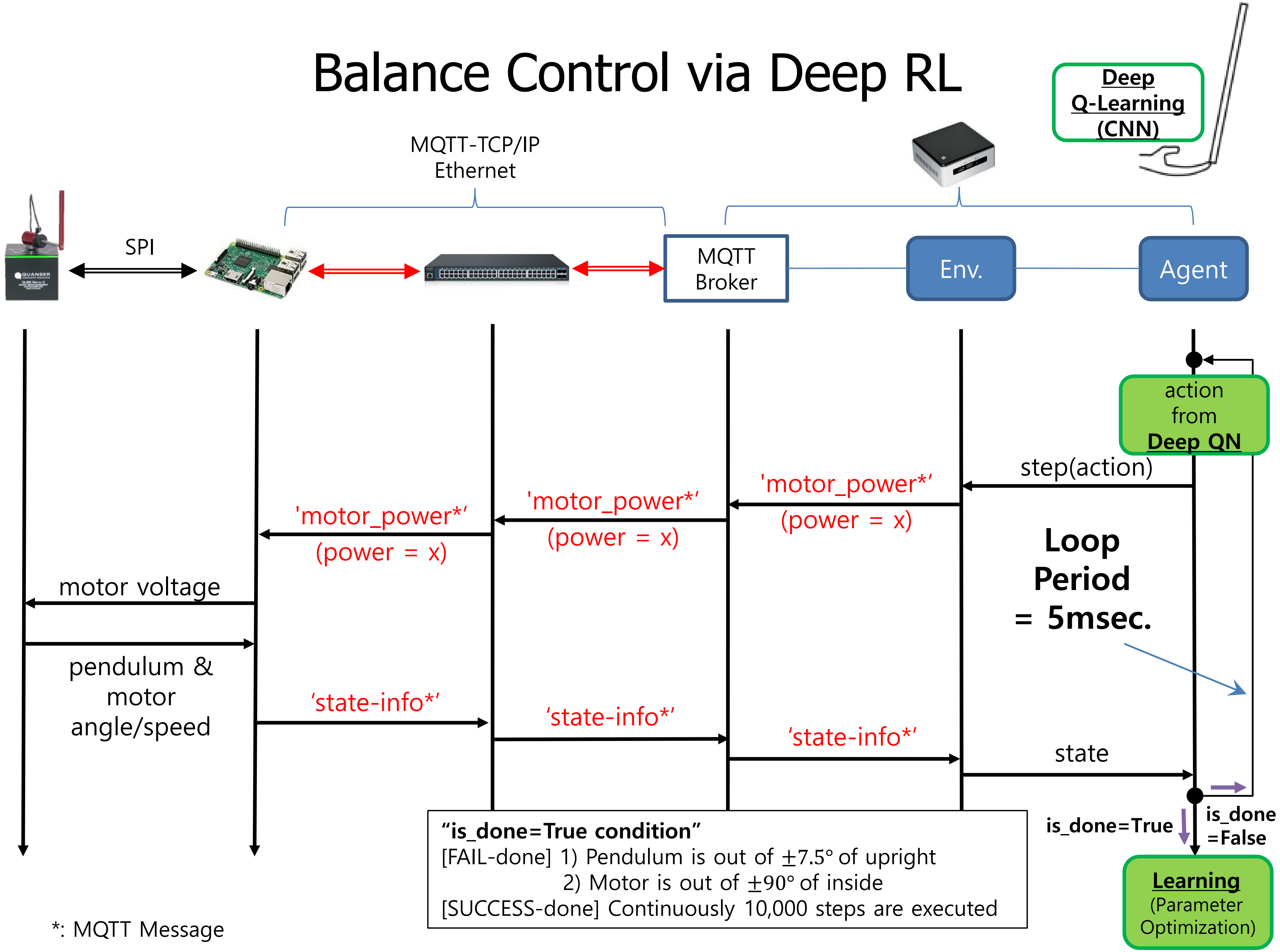
Balance Control via Imitation Learning



Loop Period = 5msec.

*: MQTT Message

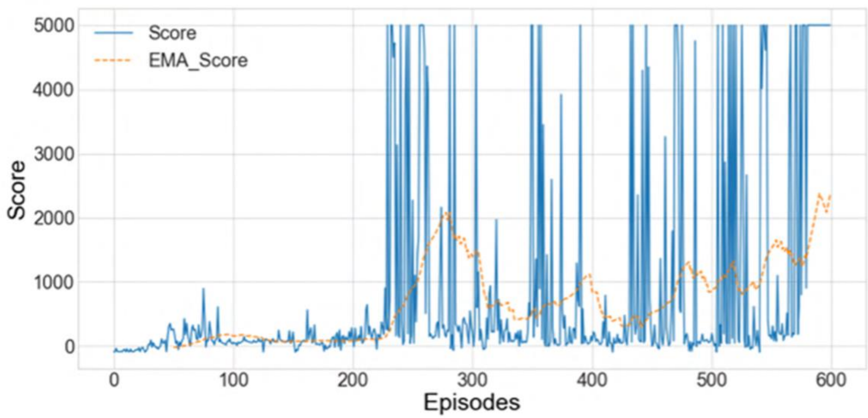
Balance Control via Deep RL

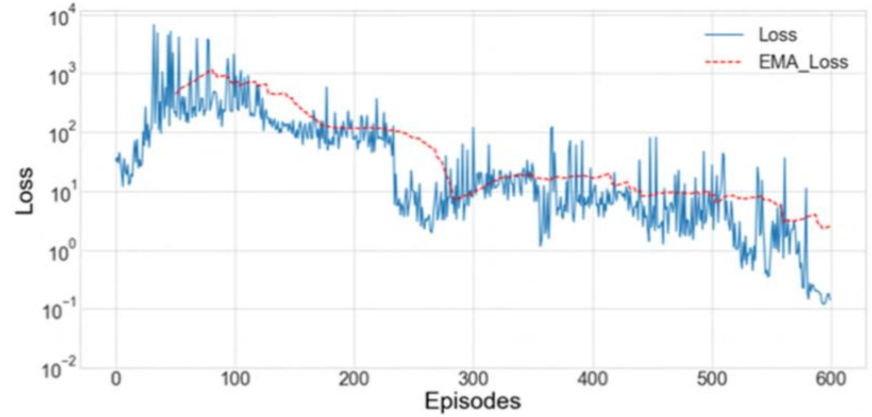


Experimental Results

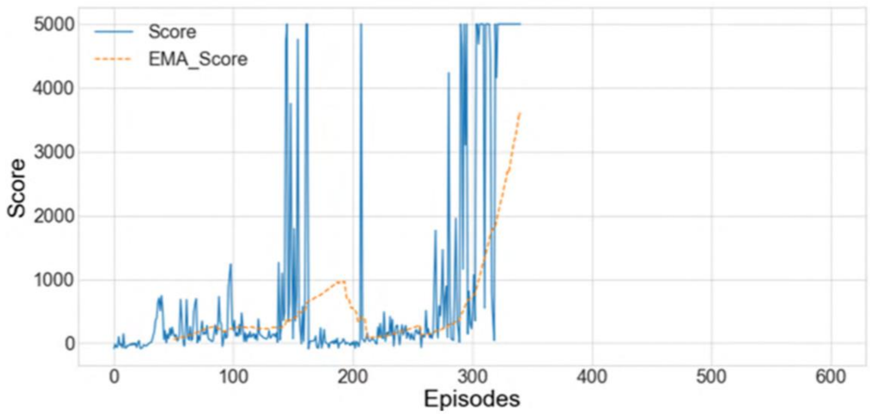
◈ Episodic Scores & Losses

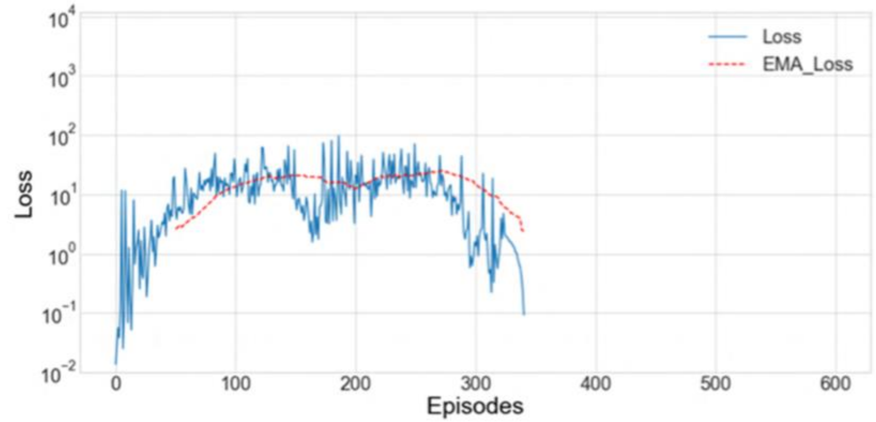
– Success-done condition (continuous 5.000 steps within an episode) repeats 20 times.





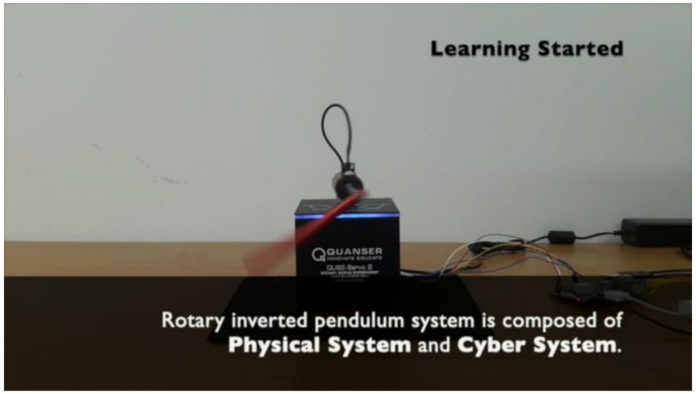
◈ Episodic Score & Losses (with imitation learning)

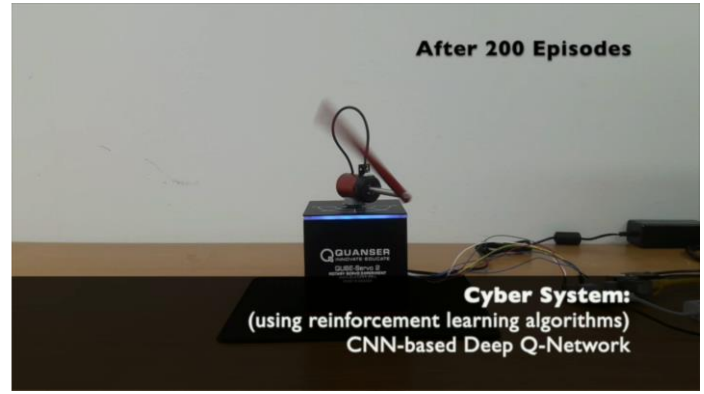


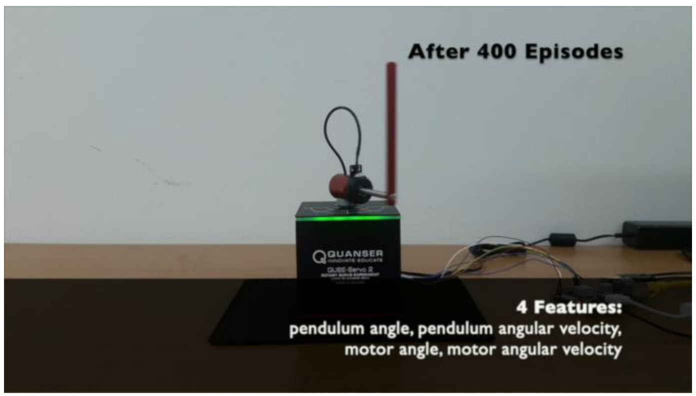


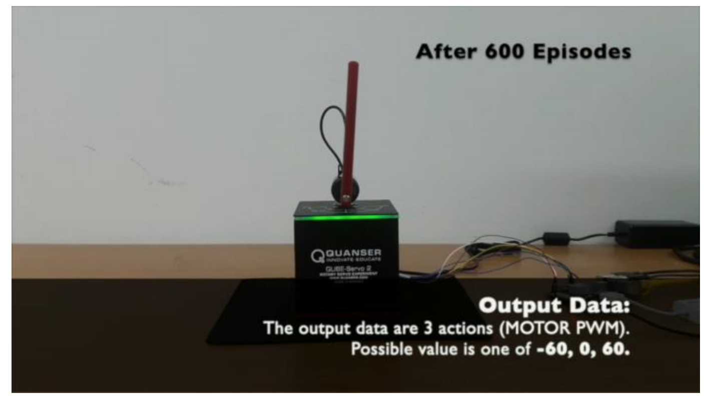
https://www.youtube.com/watch?v=vHd7vtadwdc&t=3s

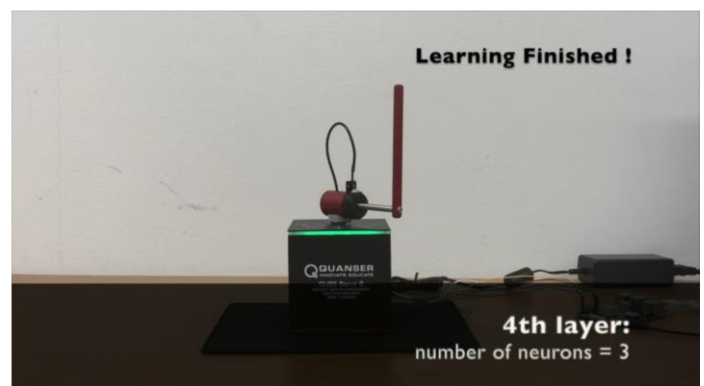
Learning Progress





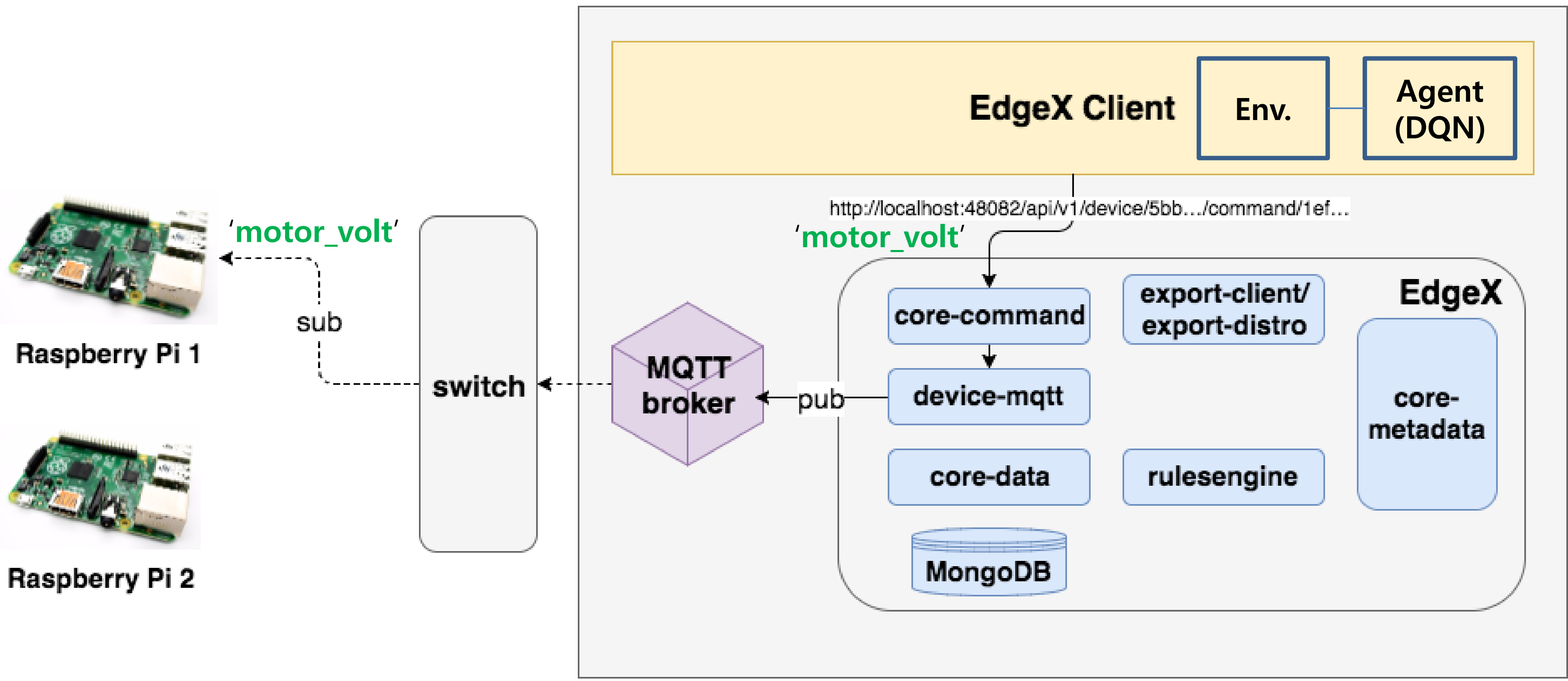






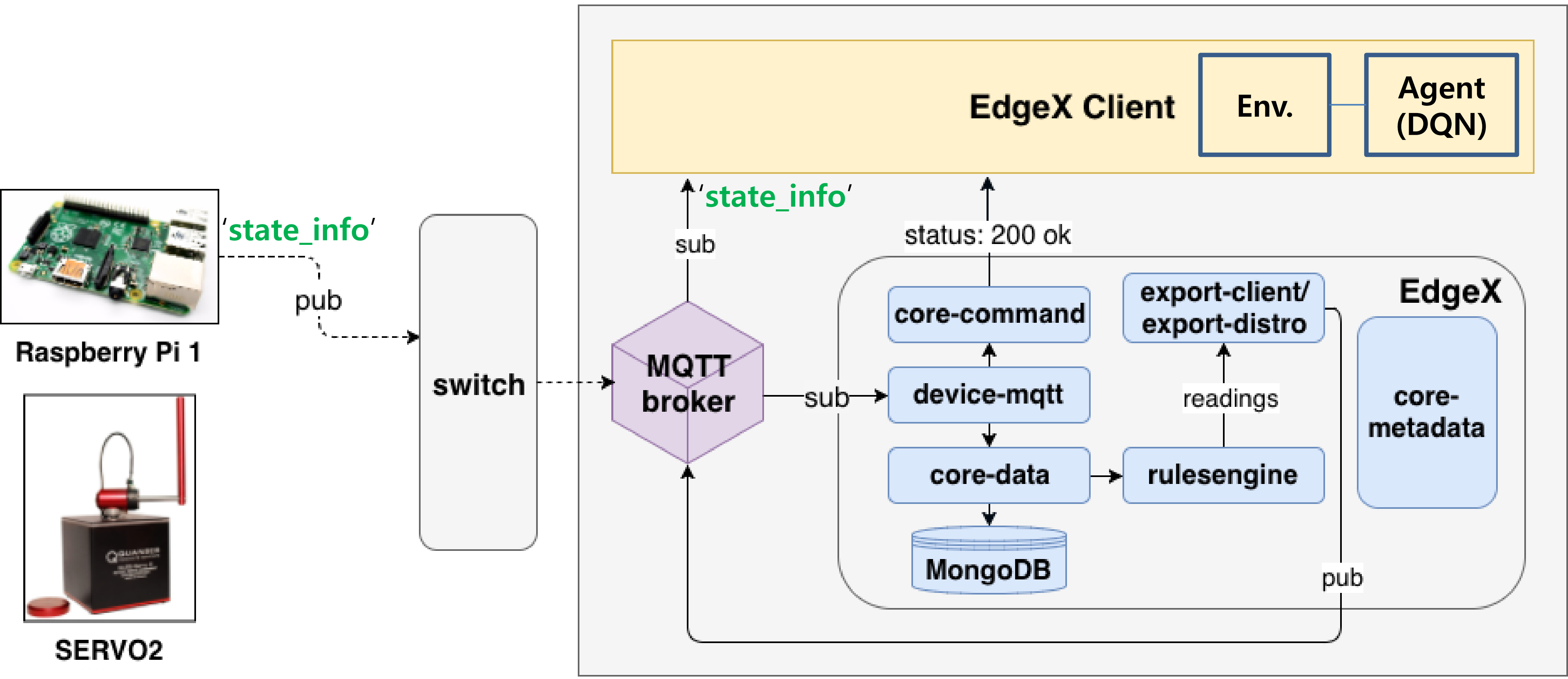
Control/Feedback on EdgeX platform

◈ Command via EdgeX (device-mqtt microservice)



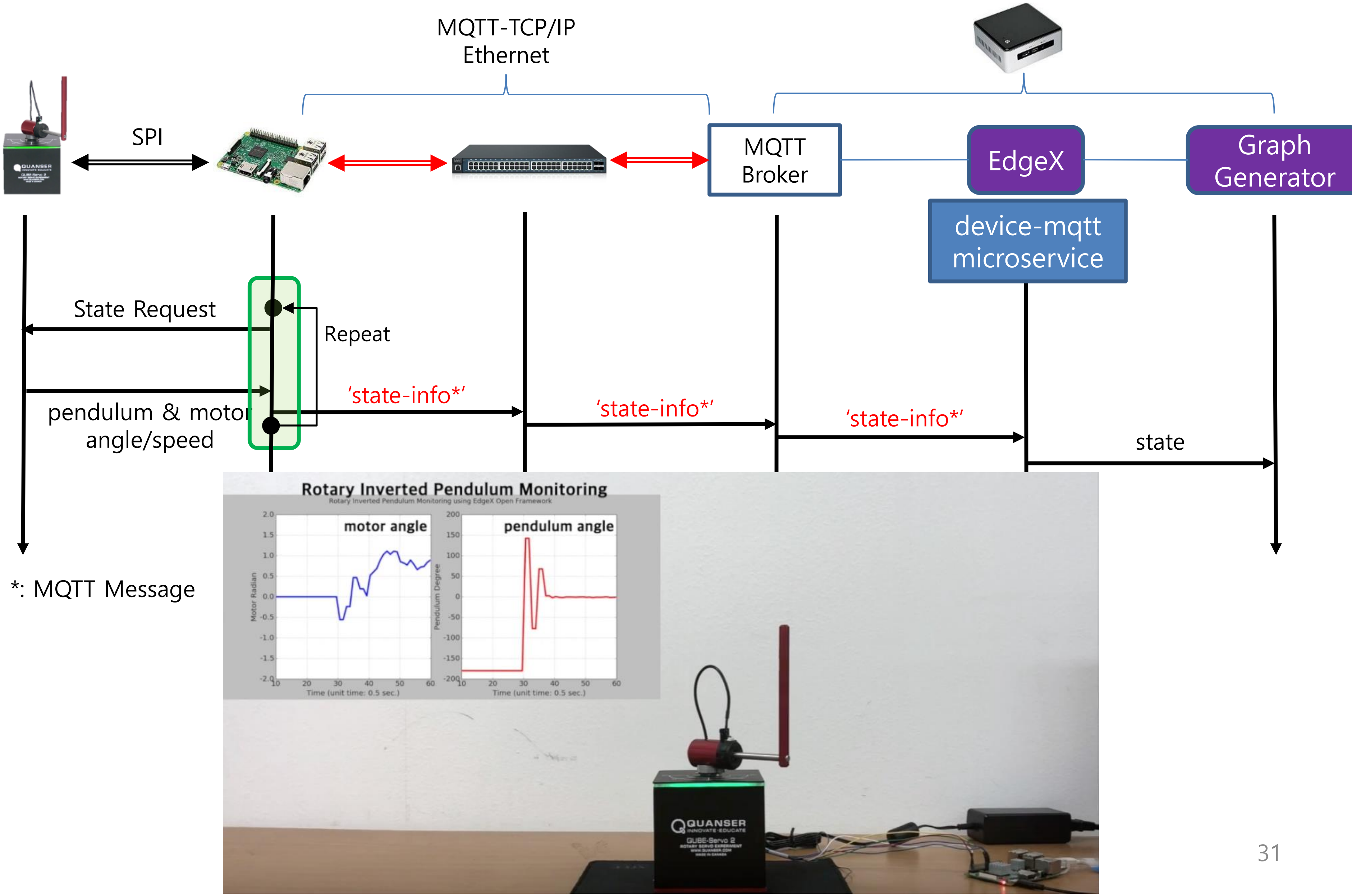
Control/Feedback on EdgeX platform

◈ Response via EdgeX (device-mqtt microservice)

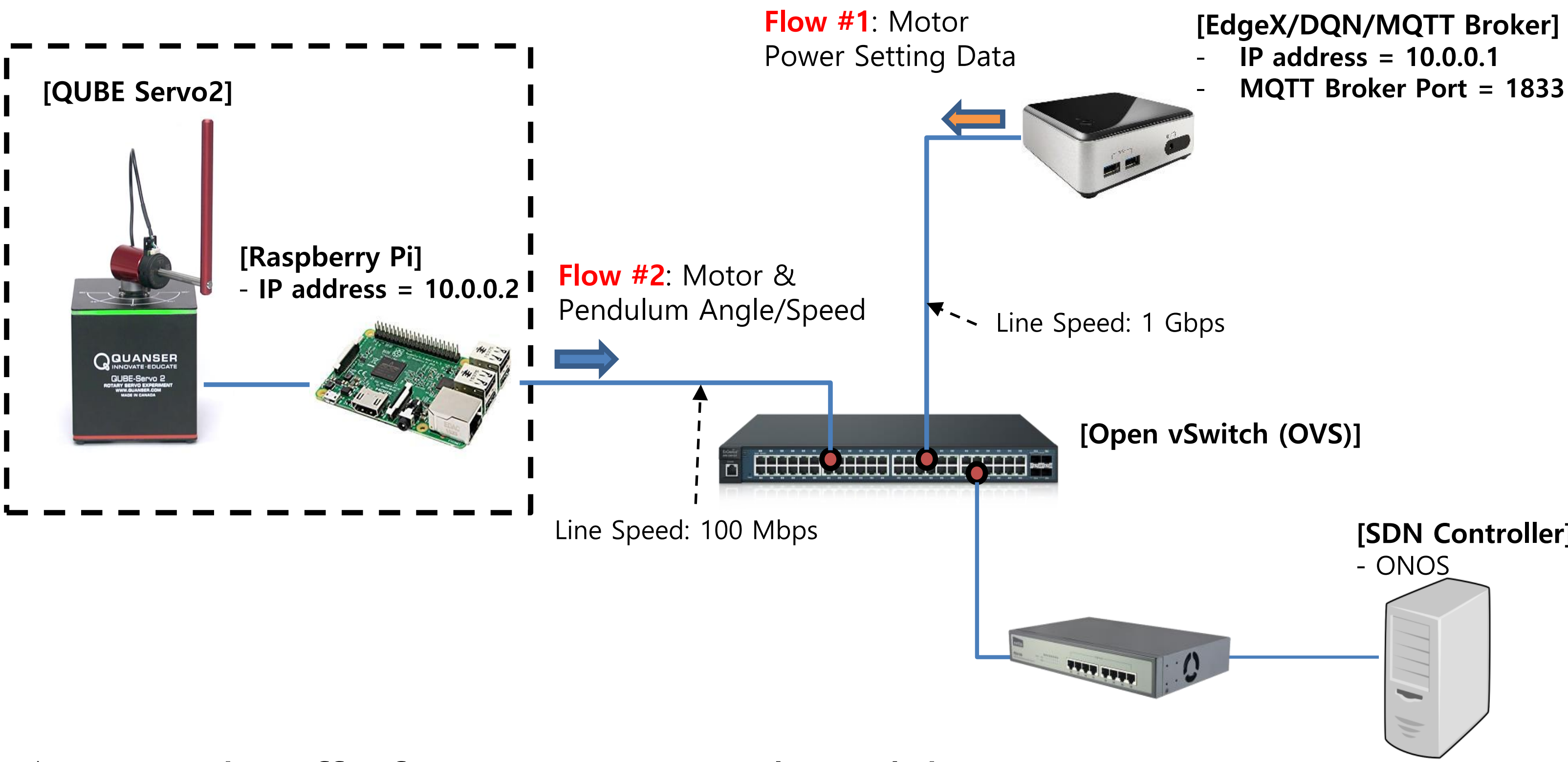


Control/Feedback RTT Time → 1.1 ~ 1.2 seconds ⬅ **It's too long!!!**

Device Monitoring via device-mqtt microservice in EdgeX Platform

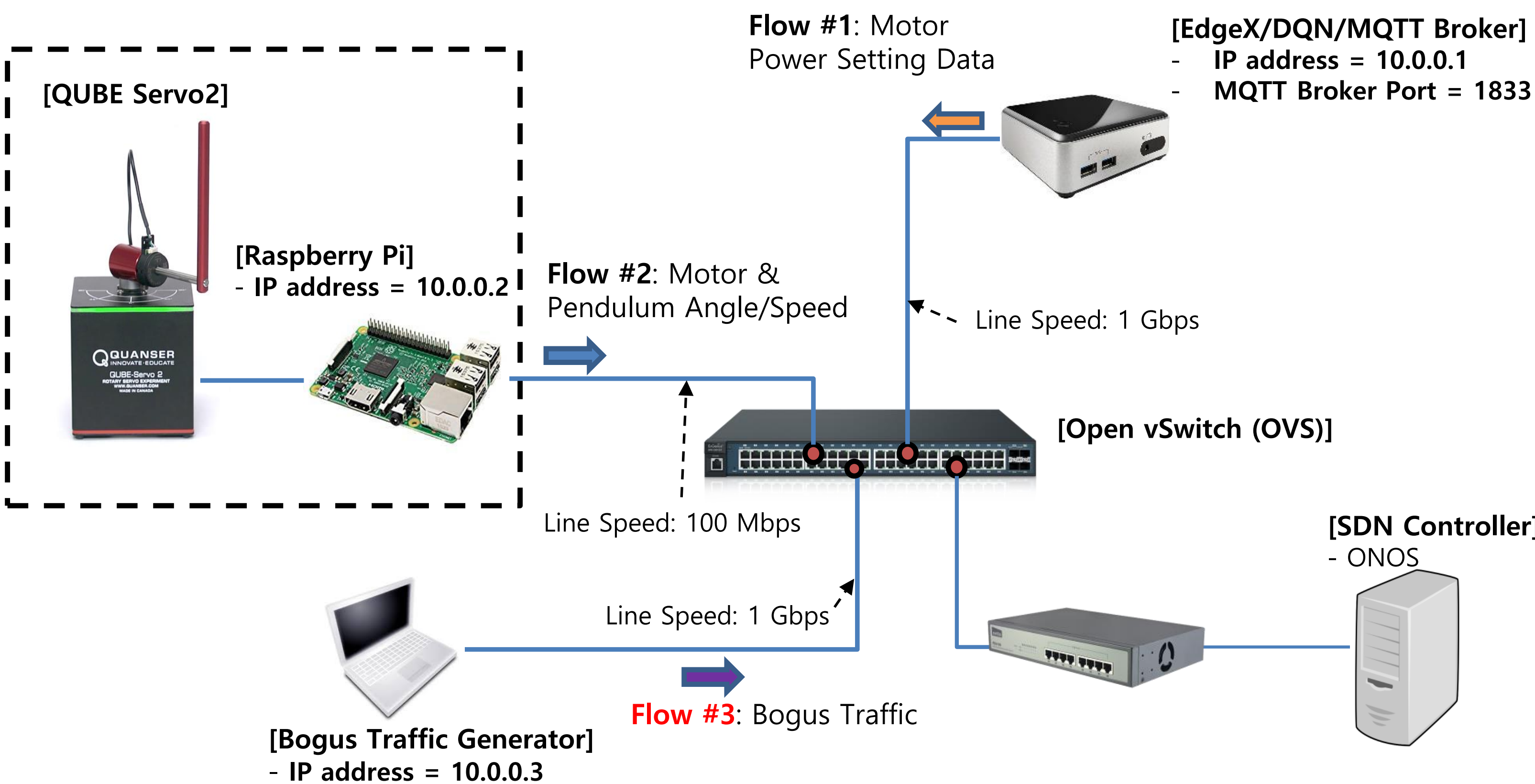


Remote RL - Network Traffic Flow



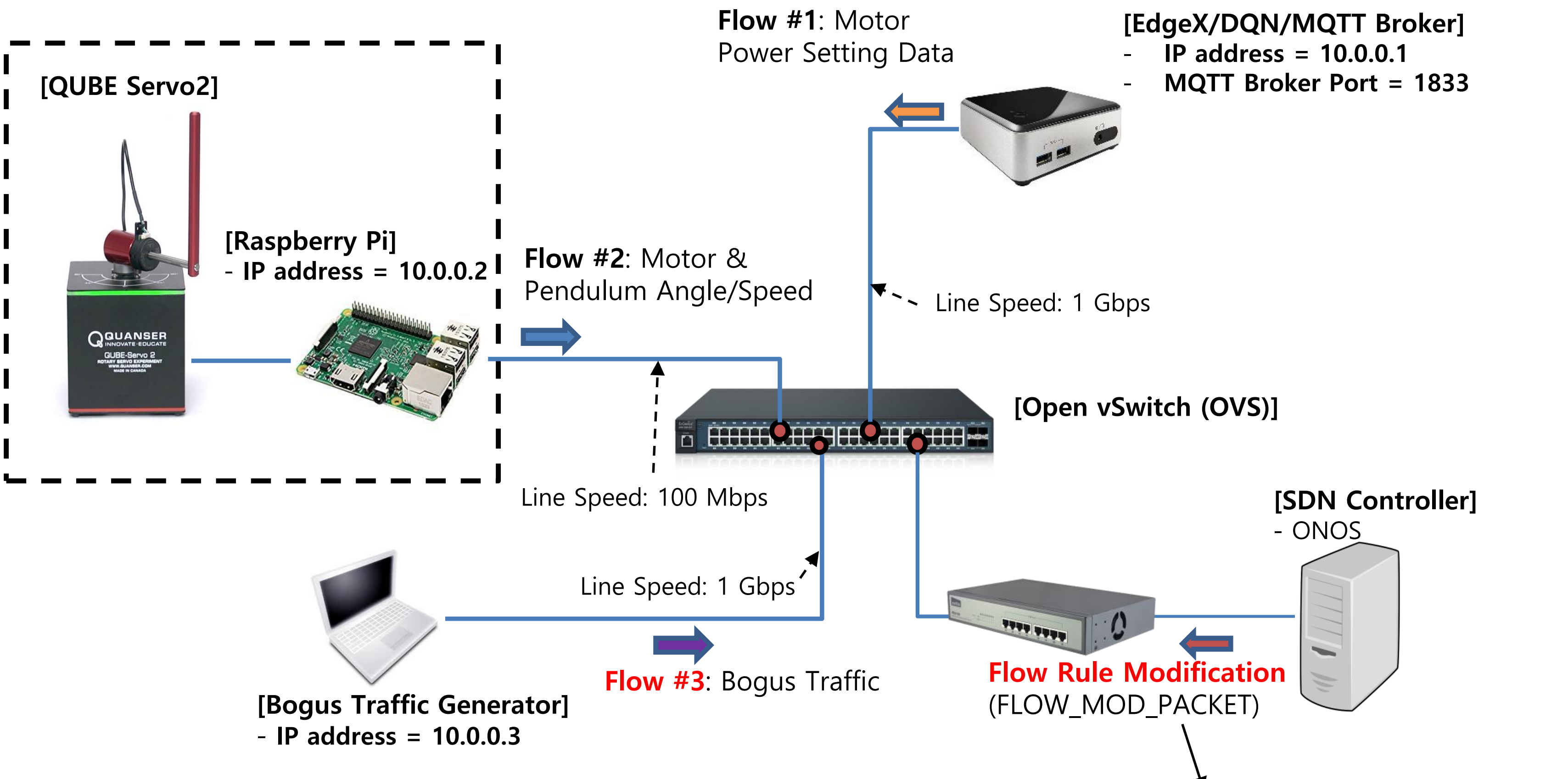
- Normal Traffic for Rotary Inverted Pendulum
  - Flow #1 (MQTT/TCP): EdgeX/DQN/MQTT Broker → RASPI/QUBE Servo2
    - Payload - Motor Power Setting Values
  - Flow #2 (MQTT/TCP): RASPI/QUBE Servo2 → EdgeX/DQN/MQTT Broker
    - Payload - Motor/Pendulum Angle and Speed Values

Bogus Network Traffic Flow



- Bogus Traffic
  - Flow #3 (iPerf/UDP): Bogus Traffic Generator → RASPI/QUBE Servo2
    - Payload - Arbitrary Data
    - Packet Generation: Transfer Bandwidth - 95Mbps (Total Packet Size: 0.95GBits)

SDN Control Message

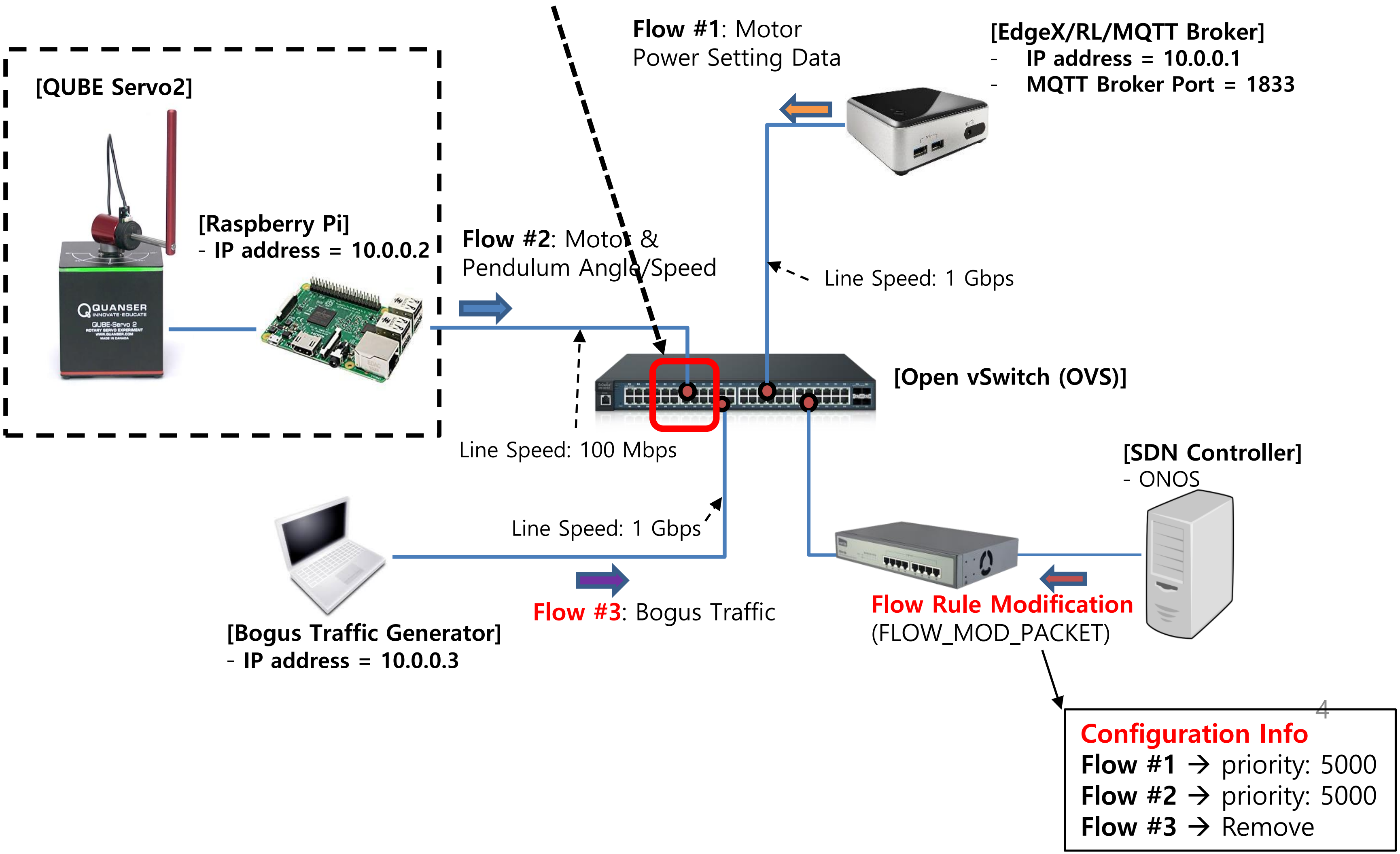


Configuration Info
Flow #1 → priority: 5000
Flow #2 → priority: 5000
Flow #3 → Remove

- ◈ Bogus Traffic Blocking
  - SDN Controller blocks the port on bogus traffic generator of OVS-switch

Network Traffic Monitoring

- Use "Wireshark"
  - Traffic monitoring at the "edgex_enp2s0" port in the OVS-switch



Network Traffic Monitoring

◈ Throughput Changes for Each (Data) Flow

Flow	Direction	Protocols	Condition		
			Normal	Bogus Traffic Generated	Bogus Traffic Blocked
#1	RL/MQTT Broker → RASPI/Serve-2	MQTT/TCP	146Kbps ⇨	22Kbps ⇨	135Kbps
#2	RASPI/Serve-2 → RL/MQTT Broker	MQTT/TCP	419Kbps ⇨	6.8Kbps ⇨	392Kbps
#3	Bogus → RASPI/Serve-2	iPerf/UDP	-	95Mbps	-

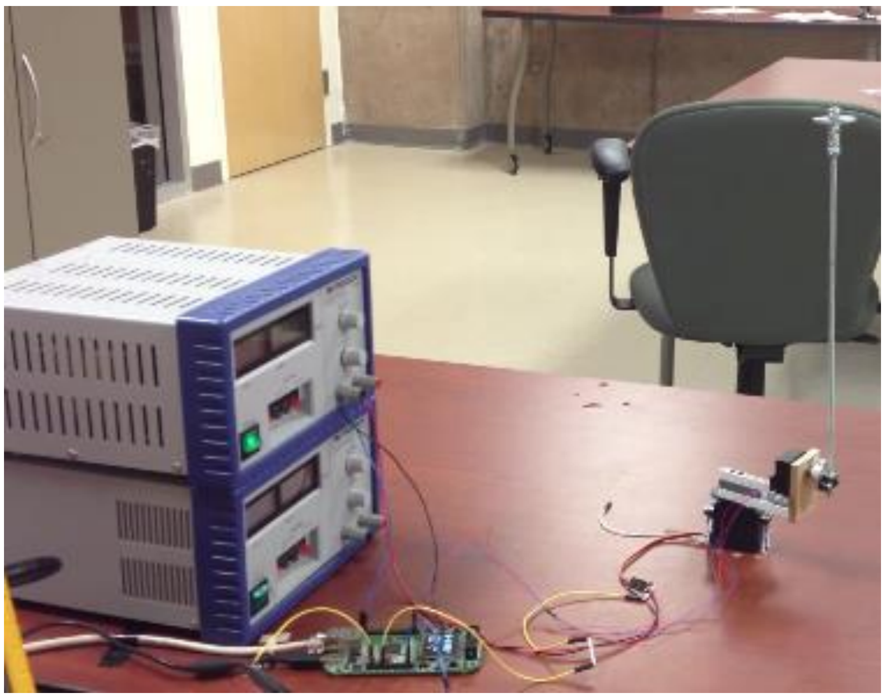
Digital Twinning & Model Transfer

◈ **Model Transfer**: Digital Twin System → Real System

– http://cps.ics.uci.edu/research/rotary-inverted-pendulum-example/

The rotary inverted pendulum control example is a case study to demonstrate **model-based design of cyber-physical systems**.

Simulation Model based on Simulink and OpenModelica

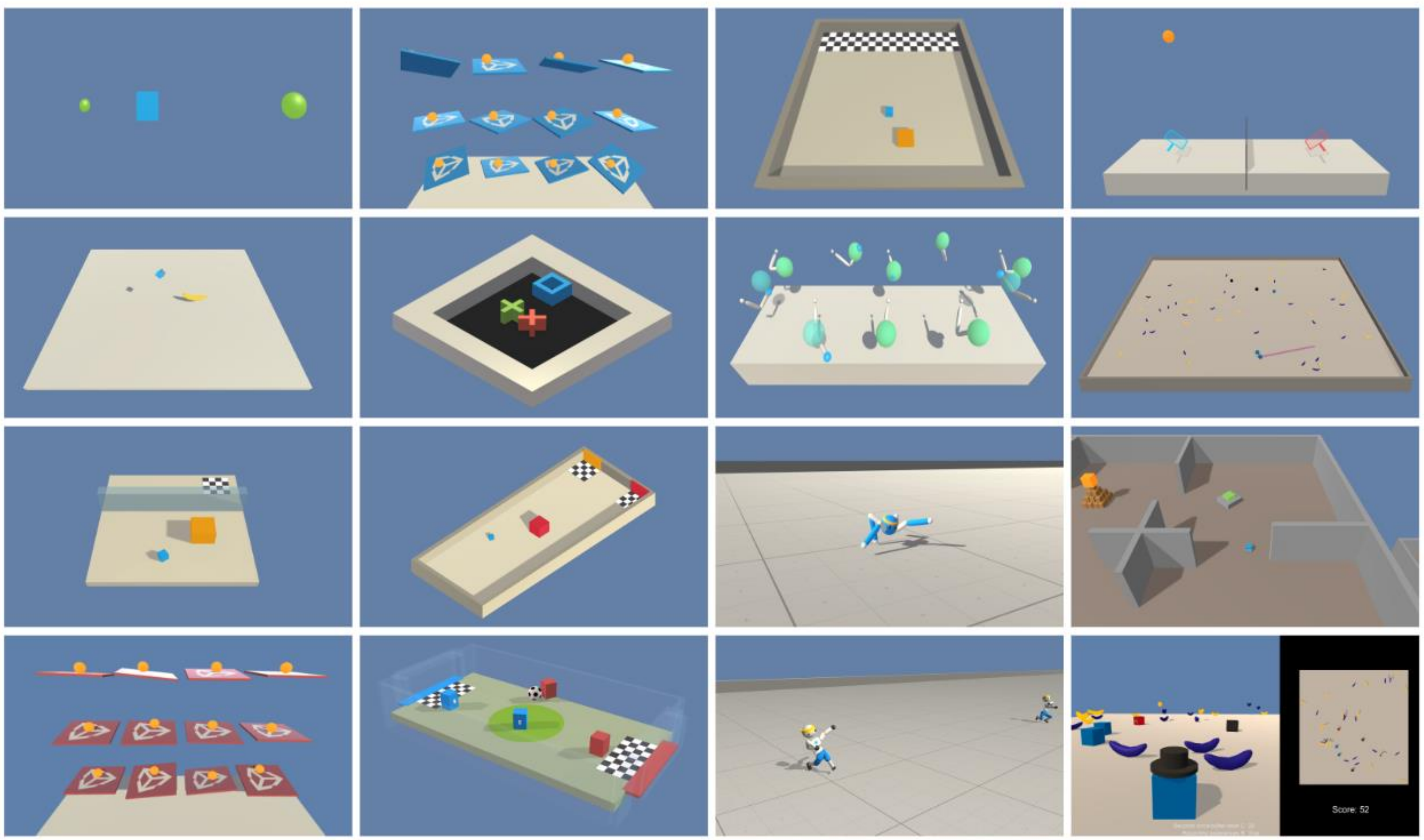


The Real System using the Batan S1213 R/C Servo and EKC-LM3S6965 TI ARM Cortex-M3

Digital Twinning & Model Transfer

◈ **<u>Model Transfer</u>**: Digital Twin System → Real System

– <u>Unity3D ML-Agents Toolkit</u>

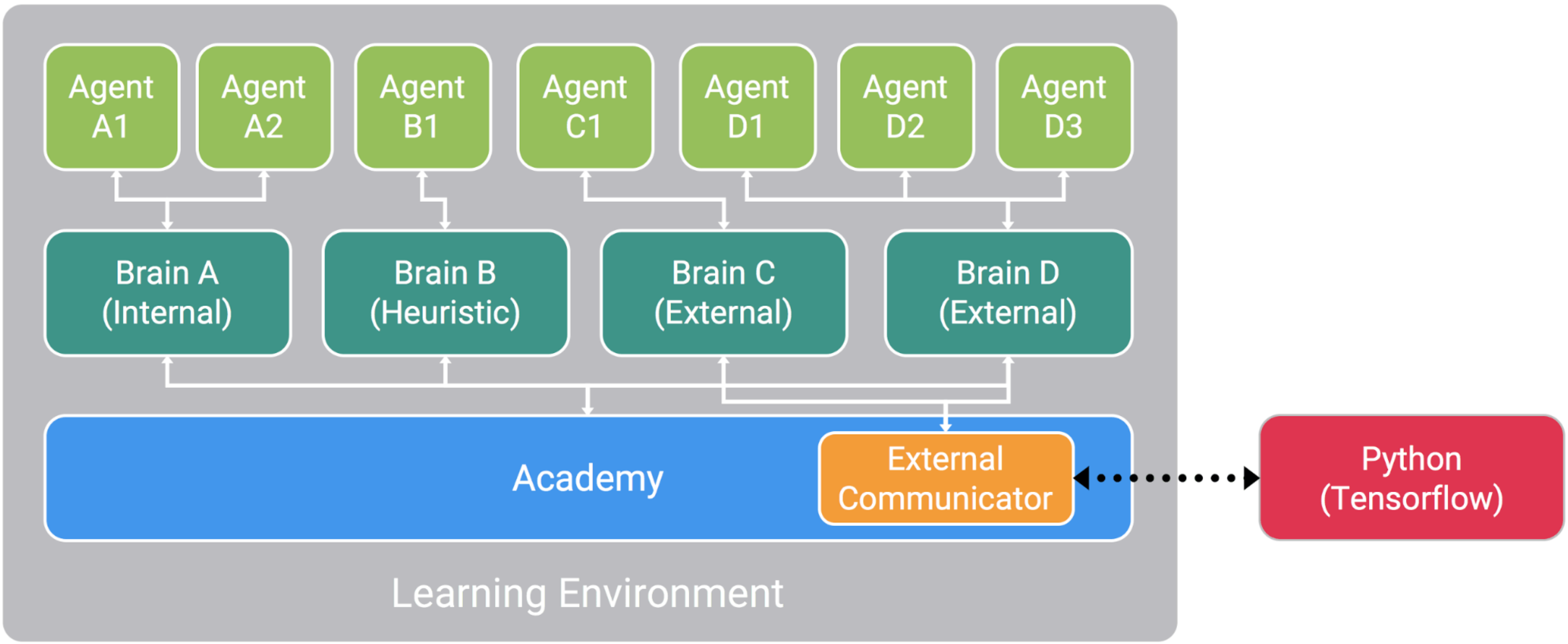


- https://www.youtube.com/watch?v=Hg3nmYD3DjQ

Digital Twinning & Model Transfer

◈ **Model Transfer**: Digital Twin System → Real System

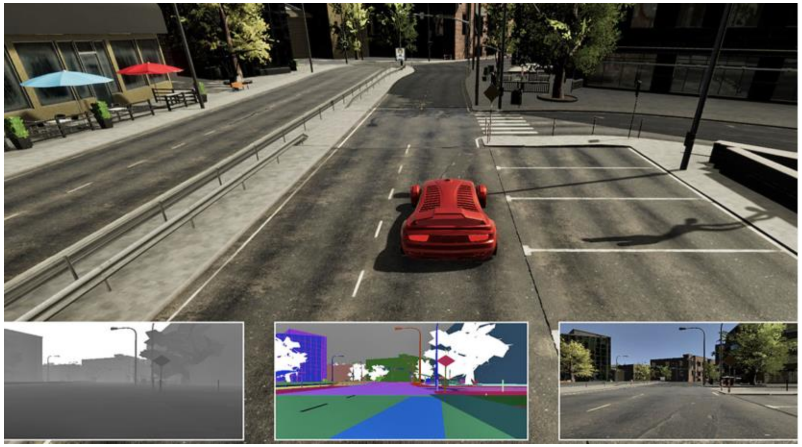
– Unity3D ML-Agents Toolkit

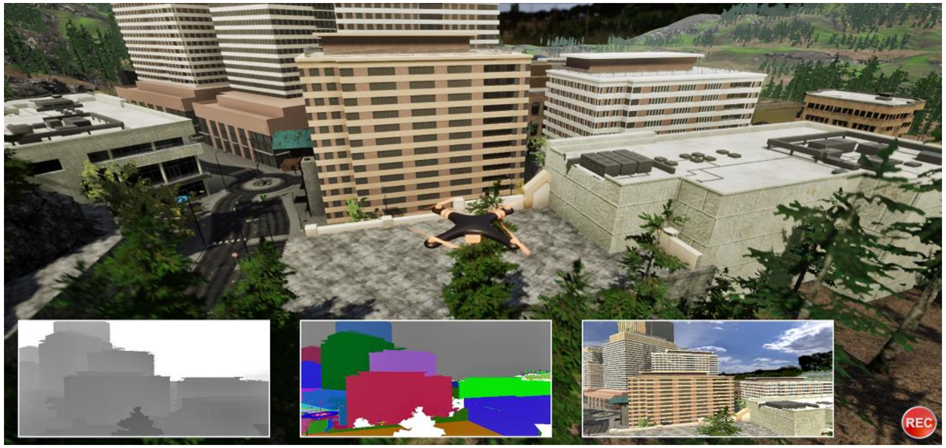


Digital Twinning & Model Transfer

◈ **Model Transfer**: Digital Twin System → Real System

- AirSim on Unity
  - https://github.com/Microsoft/AirSim
  - https://github.com/Microsoft/AirSim/tree/master/Unity



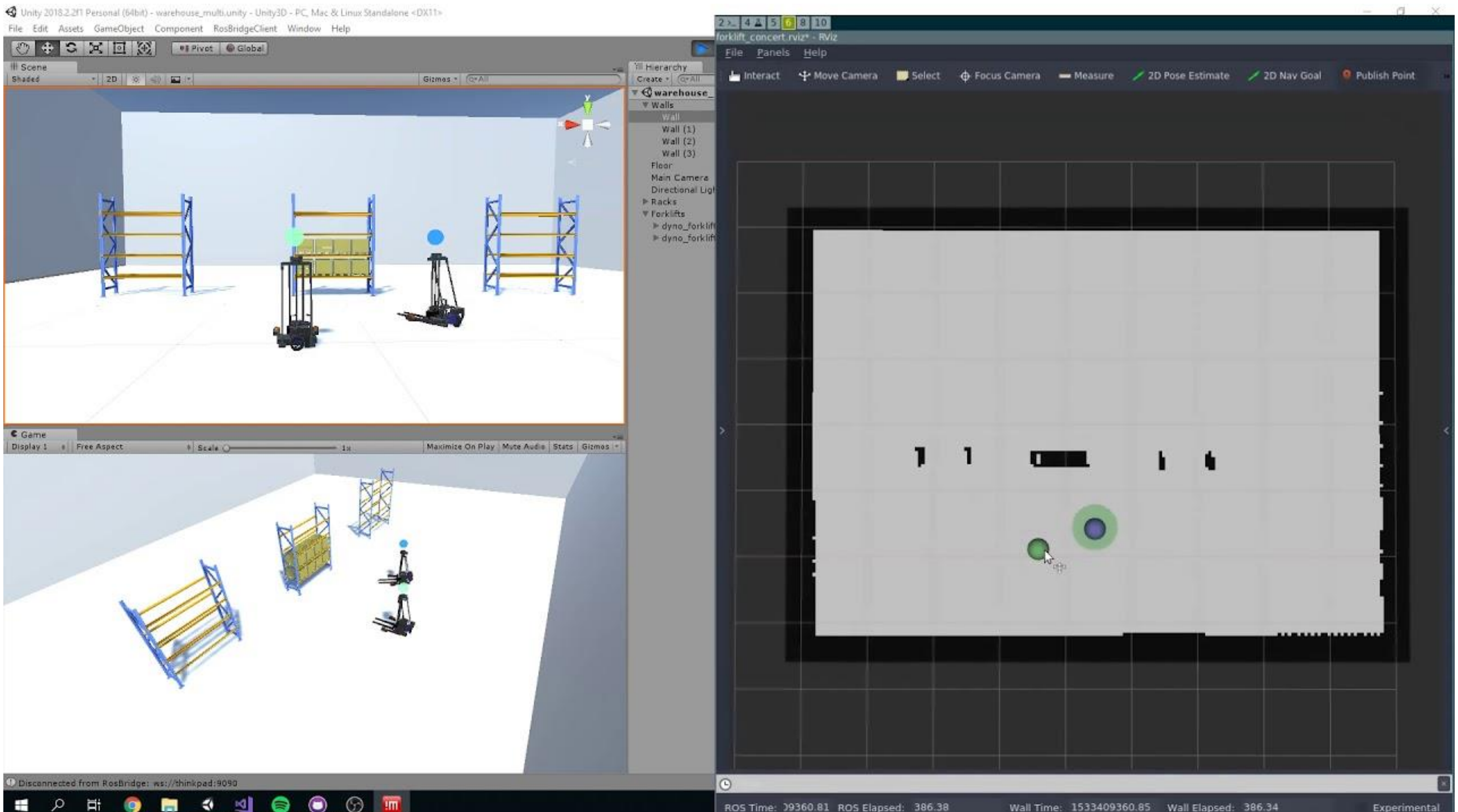


- https://www.youtube.com/watch?v=-WfTr1-OBGQ

Digital Twinning & Model Transfer

◈ **<u>Model Transfer</u>**: Digital Twin System → Real System

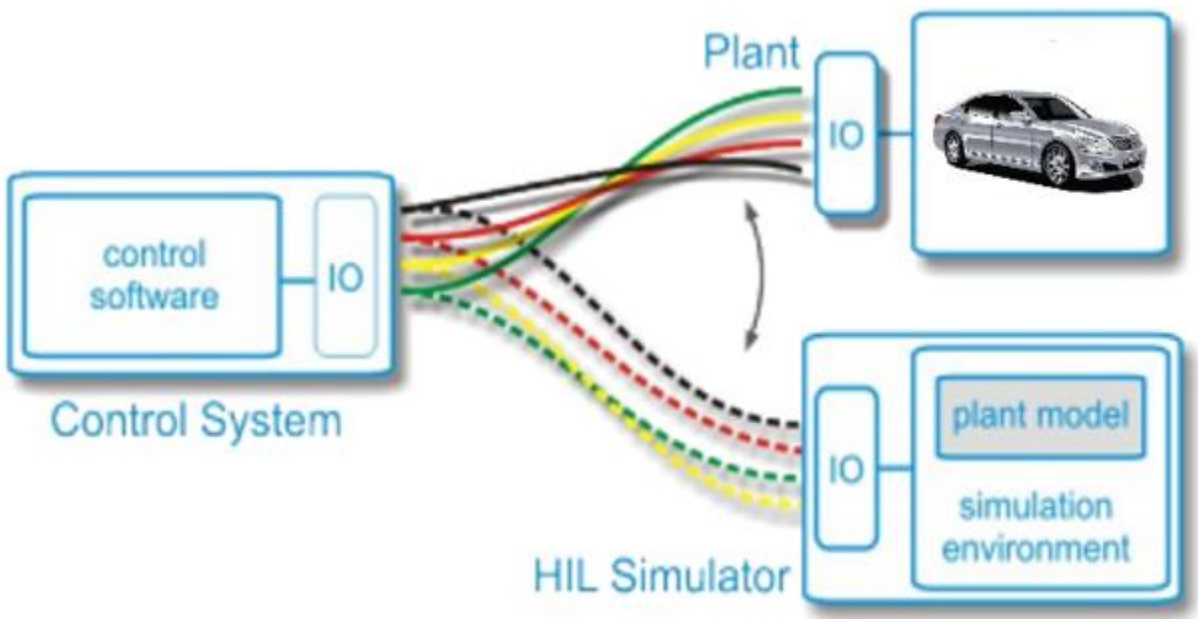
– <u>Unity3D ←→ ROS</u>



Digital Twinning & Model Transfer

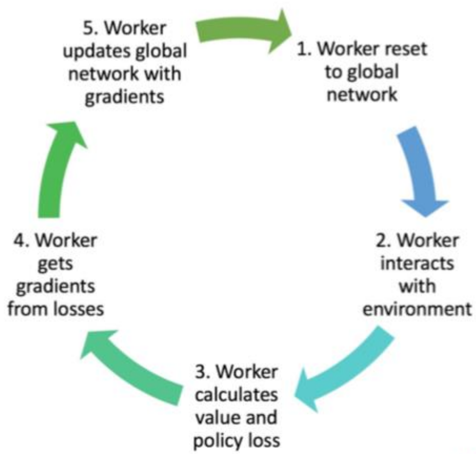
◈ **Model Transfer**: Digital Twin System → Real System

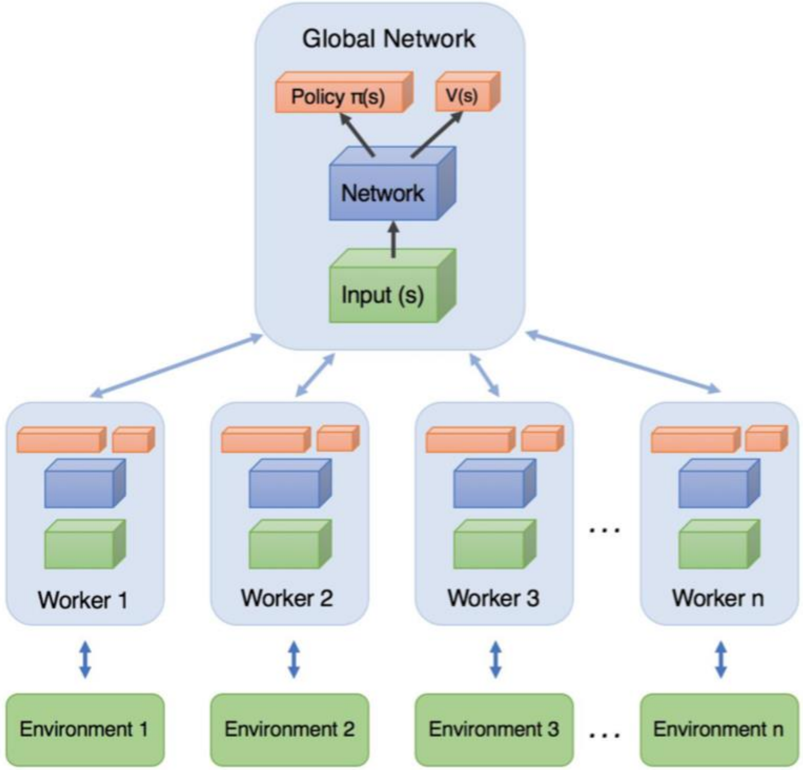
– HILS (Hardware in the Loop Simulation)



Asynchronous Advantage Actor-Critic (A3C)

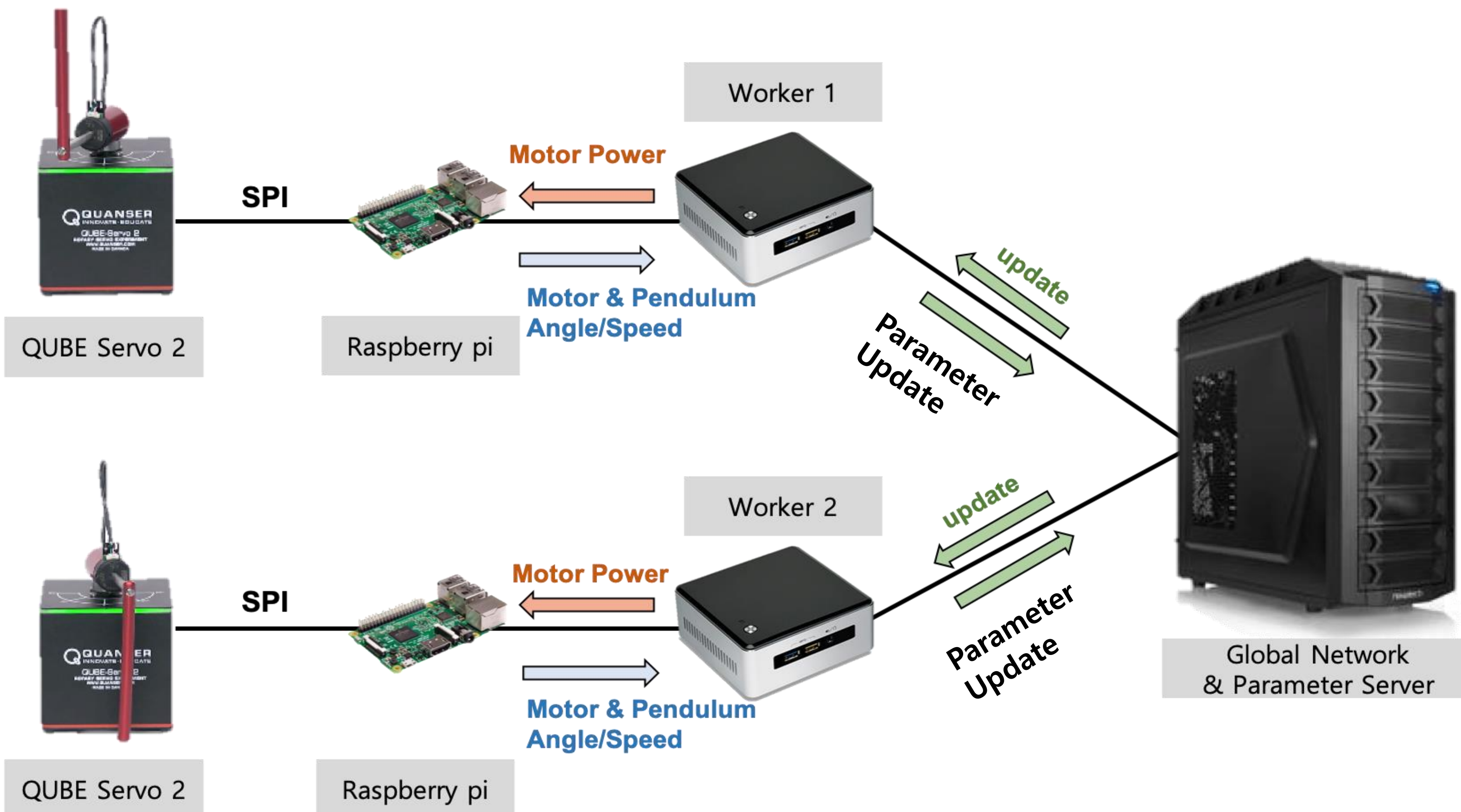
- A3C utilizes multiple Worker agents
- Speedup & Diverse Experience
- Combines benefits of Value & Policy Iteration
- Continuous & Discrete action spaces





Images(L-R): [A3C: Training workflow of each worker agent (L) and High-level architecture (R)]

Distributed A3C with Multiple Devices



Comments & Questions