



UNIVERSITAS INDONESIA

RANCANG BANGUN SISTEM PENGENDALIAN LEVEL AIR PADA PROSES
SISO MENGGUNAKAN *REINFORCEMENT LEARNING* DENGAN AGEN *DEEP*
DETERMINISTIC POLICY GRADIENT BERBASIS PLC



SKRIPSI

Muzhaffar Ma'ruf Ibrahim

1706974832

Fakultas Matematika dan Ilmu Pengetahuan Alam

Program Studi Fisika

Depok

2021



UNIVERSITAS INDONESIA

RANCANG BANGUN SISTEM PENGENDALIAN LEVEL AIR PADA PROSES
SISO MENGGUNAKAN *REINFORCEMENT LEARNING* DENGAN AGEN *DEEP*
DETERMINISTIC POLICY GRADIENT BERBASIS PLC

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains

Muzhaffar Ma'ruf Ibrahim

1706974832

Fakultas Matematika dan Ilmu Pengetahuan Alam

Program Studi Fisika

Depok

2021

HALAMAN PERNYATAAN ORISINALITAS

**Skripsi/Tesis/Disertasi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar**

Nama	: Muzhaffar Ma'ruf Ibrahim
NPM	: 1706974832
Tanda Tangan	:
Tanggal	: 21 Juni 2021

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh

Nama Mahasiswa : Muzhaffar Ma'ruf Ibrahim

NPM : 1706974832

Nama Pembimbing : Dr. Prawito Prajitno

Judul Penelitian : Rancang Bangun Sistem Pengendalian *Level Air* Pada Proses SISO
Menggunakan *Reinforcement Learning* Dengan Agen *Deep Deterministic Policy Gradient* Berbasis PLC

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Indonesia Diajukan oleh

DEWAN PENGUJI

Pembimbing I	: Dr. Prawito Prajitno	()
Penguji I	: Dr. Arief Sudarmaji, MT.	()
Penguji II	: Dr. Sastra Kusuma Wijaya	()

Ditetapkan di	: Depok
Tanggal	: 21 Juni 2021

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Sesungguhnya puji hanya bagi Allah. Penulis memuji, meminta pertolongan, dan meminta ampun hanya kepada-Nya dengan penuh rasa harap dan cemas. Skripsi ini penulis persembahkan untuk Ayahanda dan Ibunda tercinta. Terima kasih penulis sampaikan kepada mereka atas kasih sayang mereka selama ini. Skripsi ini dibuat sebagai syarat untuk memperoleh gelar Sarjana Sains di Program Studi S1 Fisika Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Indonesia. Dalam proses penulisan dan penelitian, penulis banyak mendapatkan arahan, bimbingan, saran, serta koreksi dari berbagai pihak. Untuk itu penulis ucapan terima kasih kepada:

1. Dr. Prawito Prajitno sebagai dosen pembimbing akademis dan tugas akhir, yang bersedia meluangkan waktu, tenaga, dan pikiran untuk mendukung penulis dalam melakukan riset sampai selesai.
2. Dr. Dr. Arief Sudarmaji, MT. dan Dr. Sastra Kusuma Wijaya selaku penguji sidang skripsi yang telah memberikan masukan dan evaluasi kepada penulis dalam melakukan riset.
3. Dr. Baiq Hana Susanti, M.Sc selaku direktur PT. Artifisial Intelegensi Indonesia serta staff-staff, meliputi Bu Nurul, Kak Citra dan Mas Anang yang telah memfasilitasi kegiatan penelitian tugas akhir ini kepada penulis.
4. Tim Riset S1 TA 2021 yang berisikan Arifqi, Dandung, Ferdy, Rendi, dan Tiva yang telah sama-sama berbagi beban.
5. Keluarga besar Fisika Instrumentasi Pikachu UI 2021 atas kerjasamanya selama kuliah serta Instrumentasi 2015 dan 2016 yang sering berbagi pengalamannya mengenai perkuliahan.
6. Keluarga besar Departemen Fisika UI atas waktu dan kenangannya.
7. Orang tua dan keluarga yang telah memberi dukungan baik materiel dan moril.
8. Keluarga besar KSM EP UI 2020 dan Alumni KSM EP UI Script(S)he yang berisikan Christy, Deborah, Khodi, Prilly, dan Yudo atas bantuan dan *support*-nya selama ini.

9. Tim Bodrex AiCI yang telah memberikan bantuan dan dukungan secara personal kepada penulis meliputi Kak Fitri, Kak Charnel, dan Bang Zoel.
10. Futukh, Radit, Sore, Ryan, Rendy, Abdul dan Mas Bowo yang bersedia menjadi teman penulis dan teman nongkrong di warkop Boga Jaya sekaligus menjadi penghilang penat selama proses penggeraan skripsi dan masa perkuliahan.

Depok, 21 Juni 2021

Muzhaffar Ma'ruf Ibrahim

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Muzhaffar Ma'ruf Ibrahim
NPM : 1706974832
Program Studi : Fisika
Fakultas : Matematika dan Ilmu Pengetahuan Alam
Jenis Karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif** (*Non-exclusive Royalty-Free Right*) atas karya ilmiah saya yang berjudul:

**RANCANG BANGUN SISTEM PENGENDALIAN LEVEL AIR PADA PROSES SISO
MENGGUNAKAN REINFORCEMENT LEARNING DENGAN AGEN DEEP
DETERMINISTIC POLICY GRADIENT BERBASIS PLC**

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta. Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di: Depok
Pada tanggal: 21 Juni 2021
Yang menyatakan

(Muzhaffar Ma'ruf Ibrahim)

ABSTRAK

Nama : Muzhaffar Ma'ruf Ibrahim

Program Studi : S1 Fisika

Judul : Rancang Bangun Sistem Pengendalian *Level* Air Pada Proses SISO Menggunakan *Reinforcement Learning* Dengan Agen *Deep Deterministic Policy Gradient* Berbasis PLC

Pembimbing : Dr. Prawito Prajitno

Penelitian ini dilakukan dengan meyimulasikan pengendalian ketinggian air menggunakan sistem *Reinforcement Learning* dengan *agent* DDPG pada suatu tangki berkapasitas 36,4 liter menggunakan Simulink Matlab. Tujuan dari penelitian ini adalah untuk menjaga *level* air yang terukur pada tangki agar berada pada *setpoint* yang telah ditetapkan. Sistem akan diuji dengan melakukan perubahan *setpoint* secara *upscale* maupun *downscale*. *Plant* ini terdiri atas *control valve*, *flow transmitter*, *level transmitter*, dan *Programmable Logic Controller*. Komunikasi antara Simulink Matlab dan PLC dijembatani oleh OPC server. OPC merupakan perangkat lunak intreoperabilitas yang digunakan dalam pertukaran data menggunakan arsitektur *client/server*. Pada penelitian ini diberikan batasan bahwa ketinggian maksimal yang digunakan pada tangki adalah 60 cm. Penerapan Sistem kendali RL menggunakan agen DDPG bertujuan untuk mengoptimasi performa sistem kendali yang telah umum digunakan dengan melihat parameter-parameter seperti *rise time*, *settling time*, *maximum overshoot*, dan *steady-state error* sebagai data kualitatif. Berdasarkan hasil percobaan, RL DDPG memiliki nilai performasi dengan *maximum overshoot* sebesar 2,4 %, dan *steady-state error* maximum sebesar 2,1 %.

Key words:

Ketinggian air, PLC, *Reinforcement Learning*, Simulink Matlab, DDPG

ABSTRACT

Name	: Muzhaffar Ma'ruf Ibrahim
Study Program	: Physics
Title	: Design of <i>Programmable Logic Controller</i> (PLC) Based for Water Flow control in a single-input and single-output (SISO) system by Implementing Deep Deterministic Policy Gradient of Reinforcement Learning
Counsellor	: Dr. Prawito Prajitno

This research was conducted by simulating water level control using the Reinforcement Learning system with a DDPG agent in a 36,4-liter tank using Simulink Matlab. This study aimed to maintain the measured water level in the tank at a predetermined setpoint. The system will be tested by making upscale and downscale setpoint changes. This plant consists of a control valve, flow transmitter, level transmitter, and *Programmable Logic Controller*. The OPC server bridges the communication between Simulink Matlab and the PLC. OPC is an interoperability software used in data exchange using a client/server architecture based on COM/DCOM. In this study, there is a limitation that the maximum height used is 60 cm. Application of the RL control system using DDPG agents aims to optimize the performance of commonly used control systems by looking at rise time, settling time, maximum overshoot, and steady-state error as qualitative data. Based on the experimental results, RL DDPG has a performance value with a maximum overshoot of 2,4 % and a maximum steady-state error of 2,1%.

Keywords:

Water level, PLC, *Reinforcement Learning*, Simulink Matlab, DDPG

DAFTAR ISI

HALAMAN PERNYATAAN ORISINALITAS	iii
HALAMAN PENGESAHAN	iv
KATA PENGANTAR	v
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS	vii
ABSTRAK	viii
ABSTRACT	ix
DAFTAR ISI.....	x
DAFTAR TABEL.....	xii
DAFTAR GAMBAR.....	xiii
DAFTAR LAMPIRAN	xvi
BAB 1 PENDAHULUAN.....	4
1.1 Latar Belakang	4
1.2 Perumusan Masalah.....	7
1.3 Tujuan Penelitian	7
1.4 Batasan Masalah	7
1.5 Manfaat Penelitian	8
1.6 Sistematika Penulisan	8
BAB 2 TINJAUAN PUSTAKA	9
2.1 <i>Liquid Level Control</i>	9
2.2 <i>Reinforcement Learning</i>	12
2.2.1. Kerangka <i>Reinforcement Learning</i>	13
2.2.2. Algoritma <i>Deep Deterministic Policy Gradient</i>	14
2.3 <i>Programmable Logic Control (PLC)</i>	21
2.3.1. <i>Water Level Transmitter</i>	23
2.3.2. <i>Flow meter</i>	24
2.3.3. <i>Control Valve</i>	26

BAB 3 METODE PENELITIAN	28
3.1 Cara Kerja Sistem.....	28
3.2 Rancang Bangun Sistem	31
3.3 Rancangan <i>Software</i>	37
3.3.1. Perancangan <i>Ladder Diagram</i>	38
3.3.2. Perancangan Matlab – Simulink.....	38
BAB 4 HASIL DAN PEMBAHASAN	54
4.1. Karakteristik <i>Control Valve</i>	54
4.2. Pengukuran konstanta <i>flow out</i> (q_0)	55
4.3. Identifikasi Plant	57
4.4. Simulasi <i>Open Loop</i>	59
4.4.1. Simulasi <i>Real Plant</i>	59
4.4.2. Simulasi MATLAB.....	60
4.5. Training Algoritma DDPG	61
4.5.1. Proses Training	61
4.5.2. Optimalisasi training	64
4.6. Implementasi pada <i>Plant</i>	67
4.6.1. Kontrol Ketinggian Pada Simulasi	67
4.6.2. Kontrol Ketinggian Pada <i>Real Plant</i>	69
BAB 5 KESIMPULAN DAN SARAN	71
5.1. Kesimpulan	71
5.2. Saran.....	71
DAFTAR PUSTAKA	73

DAFTAR TABEL

Bab 3

Tabel 3. 1 Command Block pada Speed PLC	35
Tabel 3. 2 Address memory pada Speed PLC	38
Tabel 3. 3 Konfigurasi Parameter Agent Options.....	53
Tabel 3. 4 Konfigurasi Parameter Training Options.....	53

Bab 4

Tabel 4. 1 Korelasi antara bukaan control valve dengan flow rate secara upscale dan downscale	54
Tabel 4. 2 Variabel Fungsi Transfer Plant	58
Tabel 4. 3 Performa Kendali Pada Simulasi	68
Tabel 4. 4 Performa kendali pada real plant	70

DAFTAR GAMBAR

Bab 1

Gambar 1. 1 Komponen-komponen dari artificial intelligence	5
---	----------

Bab 2

Gambar 2. 1 Sistem konfigurasi sistem kendali a.) sistem open-loop tanpa aktuator b.) sistem open-loop c.) sistem closed-loop (Dorf & Bishop, 2008)	10
Gambar 2. 2 Proses penyimpanan liquid-level	12
Gambar 2. 3 Interaksi Reinforcement Learning a.) skenario perubahan state b.) proses transisi state	13
Gambar 2. 4 Skematika DDPG	18
Gambar 2. 5 Arsitektur I/O PLC	21
Gambar 2. 6 Sistem PLC (Bolton, 2017).....	22
Gambar 2. 7 Submersible Water Level Transmitter	23
Gambar 2. 8 Bagian-Bagian Submersible Water Level Transmitter	23
Gambar 2. 9 Deformasi Diafragma dengan sensor Piezo-resistive	24
Gambar 2. 10 Orifice Flow Meter.....	24
Gambar 2. 11 Dwyer SFI-801 Sight Flow Indicator dan A-712 Sensor.....	25
Gambar 2. 12 Power-Genex Electro-Pneumatic Positioner EPL series.....	26
Gambar 2. 13 Skematika Electro-Pnemuatic Positioner	26

Bab 3

Gambar 3. 1 Sistem pengukuran ketinggian dengan outlet flow konstan (Steven & R.Coughanowr, 2009)	28
Gambar 3. 2 Pemodelan secara matematis water tank system.....	29
Gambar 3. 3 Blok Komponen Dasar.....	30
Gambar 3. 4 Diagram Blok Kinerja Sistem	31

Gambar 3. 5 Gambar P&ID dari Rancang Bangun.....	32
Gambar 3. 6 PLC Fultek CPU 101F	33
Gambar 3. 7 Software Speed PLC	34
Gambar 3. 8 Konfigurasi Pemograman pada Programming Device.....	35
Gambar 3. 9 <i>Address Number</i> pada Data Block 1 Speed PLC	36
Gambar 3. 10 Konfigurasi Address Number pada OPC KEPServerEx.....	37
Gambar 3. 11 Model untuk mencari Fungsi Transfer Control Valve	39
Gambar 3. 12 OPC Configuration.....	39
Gambar 3. 13 Real-Time Sync.....	40
Gambar 3. 14 OPC Write.....	40
Gambar 3. 15 OPC Read.....	40
Gambar 3. 16 Constant	40
Gambar 3. 17 To Workspace	41
Gambar 3. 18 Scope	41
Gambar 3. 19 Model Water Tank Reinforcement Learning Environment Model.....	42
Gambar 3. 20 Generate Observation.....	42
Gambar 3. 21 Variabel-variabel pada blok Generate Observation	43
Gambar 3. 22 Calculate Reward	43
Gambar 3. 23 Sistem Kalkulasi Reward	43
Gambar 3. 24 Stop Simulation.....	44
Gambar 3. 25 Sistem Exceed Bound Di Dalam Blok Stop Simulation	45
Gambar 3. 26 RL Agent.....	45
Gambar 3. 27 Water Tank System.....	46
Gambar 3. 28 Model Matematis Water Tank System.....	46
Gambar 3. 29 Policy Network (Actor) (Siraskar, 2021)	49
Gambar 3. 30 Critic Network (Action-Value) (Siraskar, 2021)	49

Bab 4

Gambar 4. 1 Benchmark Trajectory.....	47
Gambar 4. 2 Grafik hubungan antara flow rate dan % bukaan control valve.....	55
Gambar 4. 3 Grafik Water Level (cm) terhadap Δt (s)	56
Gambar 4. 4 Diagram Open-Loop	57
Gambar 4. 5 Step Response Open Loop % CV, % FM terhadap waktu (s)	58
Gambar 4. 6 Grafik Hubungan antara % bukaan control valve dengan ketinggian air (cm)	60
Gambar 4. 7 Grafik Hubungan antara % Bukaan CV & Ketinggian air (%).....	60
Gambar 4. 8 Blok sistem simulasi dengan % bukaan control valve sebagai input dan ketinggian Air sebagai output.....	61
Gambar 4. 9 Display proses training oleh agent dengan average reward maximum sebesar 4800	62
Gambar 4. 10 RL Episode manager dengan average reward maximum sebesar 4800.....	62
Gambar 4. 11 Display proses training oleh agent dengan average reward maximum sebesar 4500	63
Gambar 4. 12 Respon ketika menggunakan strategi training pada proses awal training....	65
Gambar 4. 13 Proses training ketika strategi training diganti dengan varian baru	66
Gambar 4. 14 Respon ketika mengganti strategi training dengan varian baru	66
Gambar 4. 15 Desain Simulink Untuk Uji Coba Simulasi	67
Gambar 4. 16 Gambar Grafik Proses Pengendalian Ketinggian Air Berbasis Simulasi	68
Gambar 4. 17 Blok Water Tank System Pada Uji Coba Real Plant	69
Gambar 4. 18 Gambar Grafik Proses Pengendalian Ketinggian Air Berbasis Real Plant..	69

DAFTAR LAMPIRAN

Lampiran 1. Plant Pengendali Ketinggian Level Air (tampak samping).....	74
Lampiran 2. Plant Pengendali Ketinggian Level Air (tampak depan).....	74
Lampiran 3. Konfigurasi PLC menggunakan koneksi ethernet.....	74
Lampiran 4. wiring system rancang bangun pengendali ketinggian air	74



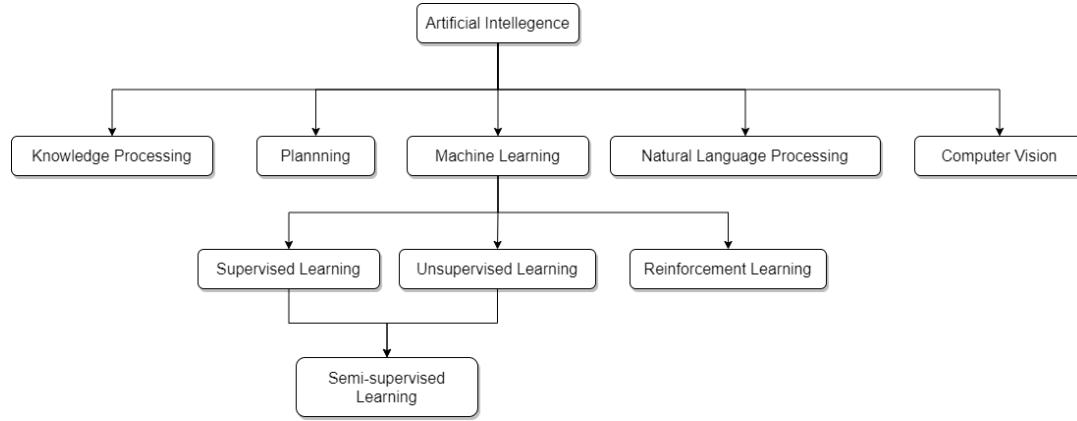
BAB 1

PENDAHULUAN

1.1 Latar Belakang

Artificial intelligence (AI) telah memunculkan pergantian paradigma diberbagai bidang industri di seluruh dunia, seperti transportasi hingga bidang kesehatan. Topik yang sangat jarang diketahui masyarakat ini telah menjadi pemantik gagasan serta ide bagi banyak pelaku industri serta akademisi. Kemajuan yang pesat pada perangkat keras, semakin terjangkaunya sistem penyimpanan data, serta dikombinasikan dengan kemampuan AI untuk melakukan ‘*self-learn*’ telah mendorong AI menjadi algoritma terdepan dalam pengaplikasianya, yang terbukti pada cepatnya perkembangan teknologi, seperti *Computer Vision* dan *Natural Language Programming* (NLP).

Saat ini, topik yang paling berpengaruh di AI adalah *Machine Learning* (ML). ML dapat dideskripsikan sebagai bidang ilmu yang mempelajari dan mengembangkan berbagai algoritma serta model statistik untuk memberikan mesin kemampuan *self-learning* secara eksplisit tanpa diprogram untuk melakukannya (Russel and Norvig, 2009). Bidang-bidang ML dapat dibedakan menjadi *supervised learning*, *unsupervised learning*, *semi-supervised learning*, dan *reinforcement learning*.



Gambar 1. 1 Komponen-komponen dari *artificial intelligence*

Reinforcement learning (RL) mencoba untuk mengatasi berbagai batasan yang dimiliki oleh *supervised learning*, *unsupervised learning*, dan *semi-supervised learning* dengan mengombinasikan metode-metode ML sebelumnya menjadi satu kesatuan algoritma melalui modifikasi proses *learning*-nya. Tujuan utama RL secara umum adalah untuk memberi mesin kemampuan *learning* melalui pemetaan optimal di suatu kondisi (*environment*) melalui *action* (disebut sebagai *policy*) dengan melakukan pencarian *trial-and-error* yang dipandu oleh sinyal *reward*. Dalam beberapa skenario khusus, *action* tidak hanya berpengaruh pada *reward* di *local area*, namun juga pada *global reward* yang akan datang. Kedua fitur ini – *guided trial and error* serta *delayed feedback* – membedakan RL dari berbagai pendekatan ML lainnya dan pada akhirnya mampu memperluas batasan baru dari pengetahuan saat ini (Sutton and Barto, 2018). Maka dari itu, RL dapat menjadi solusi bagi kontrol nonlinear serta adaptif yang diakibatkan besarnya nilai *dead time* atau parameter yang sangat bervariasi. Hal ini dapat menjadi alternatif dari sistem kendali yang umum dipakai seperti sistem kendali PID (Jian & Wenjian, 2001).

Dengan formulasi masalah yang tepat, RL telah berhasil menyimulasikan pengendalian proses dengan regulasi atau pendekripsi *set point* secara lengkap (Nian et al., 2020). Kontrol optimal juga telah terbukti layak untuk metode RL dan ADP (*Adaptive Dynamic Process*) dalam berbagai literatur. Salah satu langkah awal menuju implementasi RL skala industri adalah penggunaan RL pada *tuning* PID dengan mengkonfigurasi kembali

PID menjadi fungsi-fungsi parameter kontrolnya. Dan pada rencana yang lebih optimis, agen RL yang dipantau secara ketat akan layak digunakan sebagai kontrol optimal adaptif yang kontinu. Google adalah salah satu inisiator pertama yang menerapkan sistem kontrol ini dengan penghematan listrik hingga 40% di *data center* industri mereka (Richard Evans & Jim Gao, 2016).

Potensi lain dari RL adalah karakteristik sistem adaptasi yang *real-time*. Dibandingkan dengan kerangka kerja kontrol optimal adaptif konvensional lainnya di mana pengidentifikasi ulang model diperlukan, sistem *model-free* dapat menyesuaikan *policy* secara langsung. Sistem *model-free* mencari *policy* optimal tanpa perlu memodelkan *environment*-nya. *Model-free* memiliki kesamaan pola dengan konsep *end-to-end machine learning* dimana model *machine learning* ini dapat mengonversi data input menjadi prediksi *output* dengan melewati beberapa proses *intermediate* sehingga didapatkan hasil yang diinginkan secara spontan. Terakhir, RL dapat menyelesaikan *credit assignment problem* (CAP) temporal di mana *value* yang ditugaskan ke setiap *state* dapat mengindikasikan kecenderungan untuk mendapatkan akumulasi *value* tertinggi yang dilaksanakan melalui *action* yang diputuskan oleh *policy*. Informasi ini sangat penting dalam manajemen peringatan dini, analisis sumber kerusakan, dan aplikasi penerapan lainnya (Nian et al., 2020).

Tugas akhir ini berisi rancangan sistem kendali yang digunakan untuk mengontrol *level air* dengan menggunakan pendekatan *Reinforcement Learning*. Metode ini membuat *agent* mempelajari cara terbaik dalam menyelesaikan permasalahan pada sistem kendali ketinggian air melalui berbagai kumpulan *experience* yang berinteraksi secara terus menerus. *Control system* pada miniatur *plant* berupa pengendalian *pneumatic control valve* untuk mengalirkan debit air sesuai dengan ketinggian *level air* yang diinginkan dan akan dideteksi oleh *submersive water level transmitter*.

1.2 Perumusan Masalah

Pada penelitian ini, peneliti melakukan *processing control* pada variabel output, yaitu ketinggian air pada tangki. Proses ini dilakukan pada *miniature plant* dengan menggunakan sistem kendali *Reinforcement Learning* menggunakan pendekatan *Deep Deterministic Policy Gradient*. Pengujian akan dilakukan dengan memberikan gangguan eksternal berupa **perubahan *flow in*** serta perubahan *set point*. Hasil pengujian ini akan digunakan untuk menganalisis performa sistem dan kesesuaian dengan teori.

1. Bagaimana bentuk permodelan sistem kendali *Reinforcement Learning* serta penerapannya pada alur SISO *plant*?
2. Bagaimana hasil dari sistem pengendalian oleh *Reinforcement Learning*?
3. Bagaimana kinerja sistem kendali *Reinforcement Learning* dalam mengontrol *action* dari *control valve*?



1.3 Tujuan Penelitian

1. Merancang sistem pengendali *valve* berbasis model *Reinforcement Learning* dengan pendekatan *Deep Deterministic Policy Gradient*.
2. Mendapatkan grafik *transient response* dari *plant* melalui simulasi *open loop system*.
3. Mendapatkan hasil dengan performa yang baik menggunakan melalui model *Reinforcement Learning* dengan pendekatan *Deep Deterministic Policy Gradient* untuk mencapai ketinggian air yang diinginkan pada *plant*.

1.4 Batasan Masalah

1. Kontrol *Reinforcement Learning* dengan pendekatan *Deep Deterministic Policy Gradient* dilakukan pada *miniature plant* yang menyimulasikan proses dalam skala laboratorium.
2. Fluida yang digunakan pada percobaan ini adalah air.
3. *Miniature plant* diintegrasikan dengan sensor, aktuator, dan PLC untuk mengendalikan dan melakukan pengawasan pada sistem.

1.5 Manfaat Penelitian

Hasil dari penelitian ini diharapkan dapat memberikan pengetahuan terkait dengan pengaplikasian *Reinforcement Learning* dengan pendekatan *Deep Deterministic Policy Gradient* terhadap rancang bangun sistem *Programmable Logic Control (PLC)* terutama di bidang industri.



1.6 Sistematika Penulisan

Sistematika dalam penulisan ini terbagi menjadi 5 bab yang tiap-tiap babnya berisikan subbab yang menjelaskan penelitian yang dilakukan secara detail. Susunan bab-bab tersebut adalah sebagai berikut:

BAB 1 PENDAHULUAN

Bab ini memberikan gambaran umum mengenai penelitian yang dilakukan, seperti latar belakang penelitian, tujuan penelitian, batasan masalah dan sistematika penulisan

BAB 2 TINJAUAN PUSTAKA

Bab ini berisi tentang teori-teori penunjang penelitian yang terdiri dari teori dasar mengenai alat dan parameter yang digunakan, serta permasalahan yang akan dibahas.

BAB 3 METODE PENELITIAN

Bab ini berisi penjelasan lengkap mengenai alat dan bahan, cara kerja dan proses selama penelitian dilakukan.

BAB 4 HASIL DAN PEMBAHASAN

Bab ini terdiri dari data hasil penelitian baik yang telah diolah maupun yang belum diolah, serta analisis hasil yang telah diperoleh

BAB 5 KESIMPULAN DAN SARAN

Bab ini berisi penarikan kesimpulan berdasarkan tujuan dari hasil penelitian yang dilakukan dan saran yang akan berguna untuk pengembangan dan keberlanjutan penelitian.

BAB 2

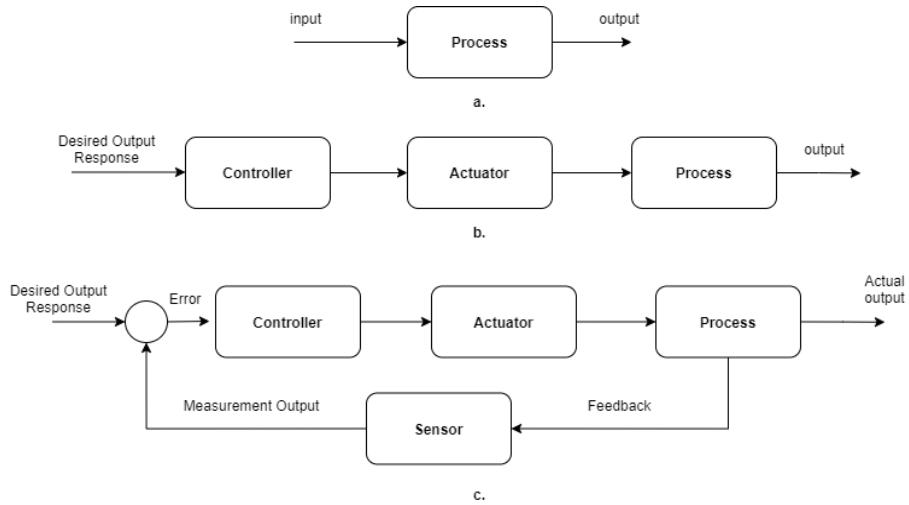
TINJAUAN PUSTAKA

Penelitian ini menerapkan RL sebagai pengganti sistem kontrol dalam mengendalikan *flow air* yang mengalir *miniature plant*. Jenis *agent* yang digunakan pada penelitian ini adalah *Deep Deterministic Policy Gradient* (DDPG) yang telah tersedia pada *Reinforcement Learning Toolbox* milik Matlab. Sistem RL ini akan dipasangkan dengan aspek kontrol eksperimental lainnya, seperti PLC dan *water level miniature plant*. Maka dari itu, perlu adanya studi literasi yang mumpuni untuk menyinkronisasi berbagai aspek yang dibutuhkan menjadi satu kesatuan sistem kontrol pengendali ketinggian air.

2.1 Liquid Level Control

Sistem kendali adalah bagian penting dari masyarakat modern. Sistem kendali berhubungan dengan pemahaman serta pengendalian bagian-bagian dari keadaan lingkungan atau biasa disebut sebagai dengan sistem. Dapat ditemukan berbagai pengaplikasian sistem kendali di sekitar kita: peluncuran roket, *smart-car*, hingga *smartphone*. Dan manusia bukanlah satu-satunya makhluk hidup yang mampu menciptakan sebuah sistem kendali, namun secara alami sistem ini juga terdapat dalam bentuk kehidupan lain. Salah satunya adalah berbagai sistem kendali yang terdapat di dalam manusia sendiri, seperti pankreas yang berfungsi dalam sistem regulasi gula darah. Hal ini juga ditemui pada *fight-or-flight response*, dimana dengan meningkatnya adrenalin diikuti oleh kenaikan detak jantung, menyebabkan lebih banyak oksigen dapat didistribusikan ke seluruh sel tubuh. Dampaknya, mata terus mengikuti pergerakan objek, tangan menggenggam objek tersebut, dan menempatkan objek tersebut ke posisi yang telah ditentukan (Nise, 1990). Sehingga dapat disimpulkan bahwa tujuan utama dari sistem kendali adalah untuk mendesain serta membangun sebuah sistem yang dapat menyelesaikan suatu tugas yang diberikan dan secara luas mampu mencapai *automatic control*, *remote control*, serta *power amplification* (Chen, 1993).





Gambar 2. 1 Sistem konfigurasi sistem kendali a.) sistem *open-loop* tanpa aktuator b.) sistem *open-loop* c.) sistem *closed-loop* (Dorf & Bishop, 2008)

Sistem kendali adalah interkoneksi antara komponen-komponen yang membentuk sebuah sistem konfigurasi dan menghasilkan sistem respons yang diinginkan (Dorf & Bishop, 2008). Pada gambar 2 hubungan antara *Input-output* dapat direpresentasikan dalam bentuk relasi *cause-and-effect* dari sebuah proses yang mewakili sebuah fenomena dari sinyal input yang menghasilkan variabel sinyal *output* dengan amplifikasi energi. Pada gambar 2.b), sebuah *open-loop control system* menggunakan sebuah *controller* dan aktuator untuk mendapatkan respons yang diinginkan tanpa sistem *feedback* didalamnya. Sebaliknya, *closed-loop control system* memanfaatkan pengukuran tambahan dari *actual output* sehingga dapat dibandingkan dengan respons *output* yang diinginkan. Pengukuran *output* tersebut disebut sebagai sinyal *feedback* yang ditunjukkan pada gambar 2.c).

Pada konfigurasi *closed-loop*, setidaknya ada empat jenis variabel yang berperan dalam pengendalian, jenis-jenis variabel tersebut antara lain:

1. *Process Variable* (PV) adalah besaran fisis yang menunjukkan kondisi sistem kendali agar nilainya tetap atau berubah mengikuti alur tertentu (variabel terkendali).
2. *Manipulated Variable* (MV) adalah variabel yang digunakan untuk melakukan koreksi atau mengendalikan PV (variabel kontrol).
3. *Set point* (SP) adalah nilai variabel proses yang diinginkan (nilai acuan).

4. Gangguan/*disturbance* adalah variabel masukan yang dapat memengaruhi nilai *Process Variable*.

Pada umumnya, industri menggunakan tangki penyimpanan cairan yang bekerja sebagai tangki lonjakan untuk meredam fluktuasi di aliran masuk. Terdapat 3 tujuan utama dalam sistem kontrol pada input tangki penampungan, yaitu (1) laju aliran keluar dari perubahan tangki harus berubah secara bertahap, bukan secara tiba-tiba, untuk menghindari gangguan pada unit proses hilir, (2) ketinggian cairan harus dipertahankan dalam batas atas dan bawah yang telah ditentukan, dan (3) kesetimbangan *steady-state mass* yang stabil harus dicapai sehingga aliran masuk dan keluar ekuivalen. Ketiga tujuan ini dapat dicapai dengan mengatur ketinggian cairan sebagai respons terhadap aliran masuk. Strategi ini disebut sebagai *averaging level control* (Dale E. Seborg et al., 2011).

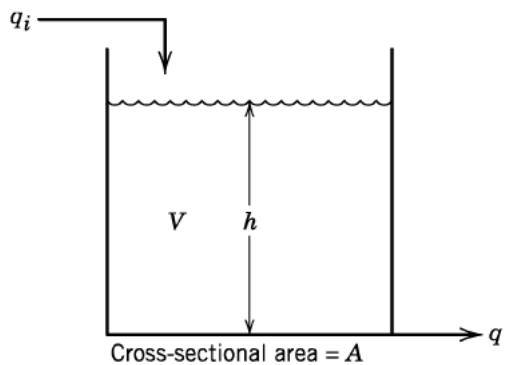
Sistem kendali dari *level* cairan pada sebuah wadah penampung pada umumnya melibatkan 2 aspek penting. Pertama, perubahan pada rasio aliran keluar dari wadah penampungan harus dapat terjadi sehalus mungkin sehingga tidak mengganggu proses hilirisasi. Kedua, ketinggian cairan tidak boleh menyimpang terlalu jauh dari tingkat operasi normal yang diinginkan. Kedua hal ini menempatkan permintaan yang saling bertentangan pada sistem kendali (Cheung & Luyben, 1979).

Secara umum, wadah penampungan cairan berbentuk silinder dengan saluran tunggal *flow in* dan *flow out*. Pada gambar 2.2, q_i dan q adalah rasio aliran masukan dan keluaran. Sehingga didapatkan persamaan kesetimbangan massa pada sistem

$$\frac{d(pV)}{dt} = \cancel{pq}_i - \cancel{pq} \quad (1.1)$$

Dan diasumsikan massa jenis \cancel{p} dapat dikatakan konstan serta luas penampang silinder, A . *Volume* dari cairan pada tangki dapat direpresentasikan sebagai $V = Ah$, dimana h adalah ketinggian cairan pada tangki. Sehingga didapatkan penurunan dari persamaan kesetimbangan *volume*

$$A \frac{dh}{dt} = q_i - q \quad (1.2)$$



Gambar 2. 2 Proses penyimpanan *liquid-level*

2.2 Reinforcement Learning

Reinforcement Learning dalam pengertian khusus adalah pembaharuan dari kontrol optimal dan merupakan studi lanjutan dari teori kontrol (Beysolow II, 2019). *Optimal control* diinisiasi pada tahun 1950-an dan digunakan untuk mendeskripsikan permasalahan sehingga didapatkan sebuah indikator pasti dari kriteria “optimal”. Secara umum, kontrol optimal merupakan kumpulan dari persamaan diferensial. Persamaan-persamaan ini kemudian menetapkan sebuah *value* yang akan memperkecil besarnya nilai pada fungsi *error*. Basis dasar dari kontrol optimal adalah pemrograman dinamis yang dibuat oleh *Richard Bellman* pada tahun 1950. Pemrograman dinamis merupakan metode optimalisasi untuk menekankan penyelesaian individu skala besar dengan memisahkannya menjadi beberapa bagian kecil.

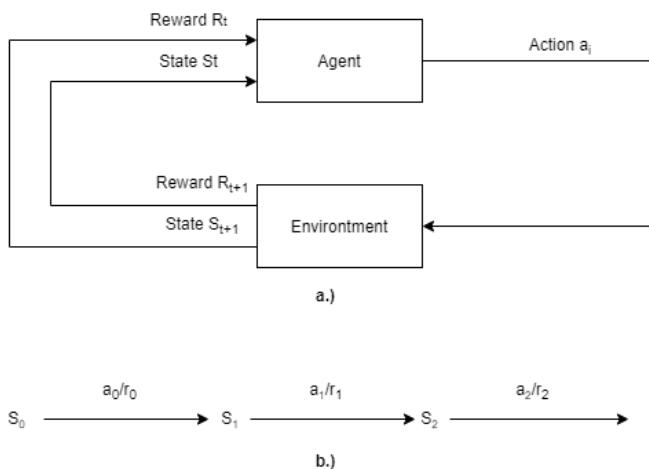
Reinforcement Learning menggunakan gagasan yang berasal dari teori sistem dinamis, secara khusus kontrol optimal dari *Markov Decision Process* (MDP). Dasar pemikiran sistem ini secara sederhana adalah untuk mencari pola dari pembelajaran *agent* yang secara terus menerus berinteraksi dengan *environment* untuk mencapai tujuan yang diinginkan. Pembelajaran *agent* harus mampu merespons *state* yang terdapat pada *environment* sampai batas tertentu dan melakukan *action* yang terkait dengan aspek *state* tersebut. *Agent* adalah program yang dirancang sebagai subjek dari proses RL. *Agent* melakukan *action* melalui interaksi dengan *environment* dan menerima *reward* berdasarkan *action* yang diambil.



2.2.1. Kerangka Reinforcement Learning

Pada penelitian ini, *agent* merupakan PLC yang bertugas sebagai pengambil keputusan dari sinyal input yang diterima dan diolah menjadi keluaran berupa bukaan *control valve*. *Environment* merupakan segala sesuatu yang berinteraksi dengan *agent* dan segala sesuatu diluar *agent* dimana secara matematis tersusun atas 2 fungsi, yaitu *transition probability* dan *reward* (Wang & Hong, 2020). *Environment* pada penelitian ini merupakan keseluruhan *miniature plant* yang berinteraksi dengan PLC melalui *controller*, yaitu *control valve*. Sehingga dapat disimpulkan bahwa *environment* ini dikategorikan sebagai *stochastic* dan *continuous*. *Action* adalah prosedur yang memungkinkan *agent* untuk berinteraksi dan memanipulasi *environtment*-nya sehingga dapat melakukan transfer informasi antar-state. Keputusan untuk memilih *action* yang paling tepat dilakukan oleh *policy* (Shaked Zychlinski, 2019). *Reward* adalah indikator skalar dari baik atau buruknya suatu performa RL dalam menyelesaikan masalah (Saito et al., 2018).

MDP bertujuan untuk mengolah tiga aspek *state* — *sensation*, *action*, dan *goal* — dalam bentuk sesederhana mungkin tanpa mengabaikan salah satu dari aspek



Gambar 2. 3 Interaksi Reinforcement Learning a.) skenario perubahan *state* b.) proses transisi *state*

tersebut (Sutton & Barto, 2018). Setiap metode yang cocok untuk memecahkan suatu masalah dapat dianggap sebagai metode RL.

Pada setiap transisi *state*, *reward* memiliki nilai yang berbeda-beda, oleh karena itu *reward* dapat digambarkan dengan nilai yang bervariasi di setiap *step*, seperti r_0, r_1, r_2, \dots seperti pada gambar 5 (Abhishek & Biswas, 2012).

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \quad \text{dimana } 0 < \gamma < 1$$

Gamma (γ) yang juga disebut *discount factor* menentukan jenis *reward* di masa mendatang:

- Nilai gamma 0 berarti *reward* dikaitkan dengan *state* pada saat ini saja.
- Nilai gamma 1 berarti *reward* bersifat *long-term*.

2.2.2. Algoritma Deep Deterministic Policy Gradient

Penelitian ini menggunakan salah satu agen RL yang digunakan dalam *continuous action space* serta Algoritma *Deep Deterministic Policy Gradient* (DDPG) adalah metode RL yang bersifat *model-free*, *online*, *off-policy* dari RL.



- *Model-free*

Model-free bermakna bahwa sistem hanya berfokus untuk mencari fungsi *value* secara langsung melalui interaksi dengan *environment* serta tidak membutuhkan estimasi dari *transition model* atau ML *model* untuk mencapai stabilitas (Madhu Sanjeevi, 2018). *Value* didefinisikan sebagai akumulasi 'keuntungan' dari beberapa *step* mendatang. Sebaliknya, *reward* didefinisikan sebagai 'keuntungan' langsung dari *action* yang dipilih pada *step* waktu saat ini. Dengan kata lain, *value* adalah akumulasi *reward* dari beberapa *step* yang akan datang hingga *step* akhir (Wang & Hong, 2020).

- *Online*

Algoritma *online learning* bekerja dengan data, tepat saat data tersebut tersedia. Secara khusus, Algoritma *online learning* memperbaharui secara bertahap setiap bagian data yang baru tiba, lalu membuang data tersebut ketika tidak digunakan lagi. Hal tersebut merupakan karakteristik dari algoritma *online* untuk

melupakan *experience* lama dari waktu ke waktu, sehingga sistem dapat beradaptasi pada populasi non-stasioner atau dinamis.

- *Off-policy*

Ketika iterasi *learning* meningkat, *learning policy* (*policy* yang melakukan *learning*) diperbaharui berdasarkan perkiraan fungsi *value*. Sementara itu, *sampling policy* memilih *state* berikutnya untuk dikunjungi selama simulasi. Ketika *learning policy* dan *sampling policy* ekuivalen, maka hal ini disebut sebagai *on-policy learning*, sedangkan ketika mereka berbeda, hal ini disebut sebagai *off-policy learning* (Powell et al., 2011). Jika bersikeras membuat keputusan berdasarkan *policy -on-policy learning-*, sistem memiliki kemungkinan untuk terjebak dengan hanya mengunjungi sejumlah kecil *state* yang terlihat baik berdasarkan buruknya pegamatan *learning policy*, sambil menghindari *state* yang mungkin tidak terlihat baik. Hal ini dapat menyebabkan estimasi yang buruk dari fungsi *value* yang mengatur perilaku dari *policy* yang dihasilkan dan pada akhirnya mengarah ke optimum lokal. Hal ini adalah alasan utama untuk menggunakan algoritma *off-policy* dimana algoritma ini menyusun strategi eksplorasi yang efisien (*sampling policy*) dengan mengunjungi *state* yang mungkin ‘tidak atraktif’ untuk saat ini karena kurangnya informasi, namun berpotensi ‘sangat atraktif’.

DDPG adalah agen berjenis *actor-critic* dari RL yang menyusun kebijakan optimal dengan cara memaksimalkan *reward* jangka panjang (Mathworks, 2020). Algoritma yang baru dikembangkan dalam *Reinforcement Learning* ini, digunakan untuk menyelesaikan permasalahan yang kompleks dari input yang belum diproses, *high-dimensional*, serta sensori (Lillicrap et al., 2016). Algoritma ini mewarisi kelebihan dari algoritma sebelumnya, yaitu Q-learning dan Deep Q-Network (Mnih et al., 2013). Dibandingkan Q-learning, metode ini bekerja pada bidang *continuous action space* dan dibandingkan dengan metode *Deep Q-Network*, algoritma ini memiliki *policy network* sehingga dapat menyajikan *action* yang bersifat deterministik. Agen DDPG dapat di-training dalam *environment* dengan *observation space* yang bersifat kontinyu maupun

diskrit. Agen DDPG ini juga telah banyak diterapkan di beberapa penelitian meliputi efisiensi pemakaian energi pada *three-train network shanghai metro line 1* (Yang et al., 2019), penjadwalan optimal pada *microgrid* (Fan et al., 2021), dan *penerapannya* diberbagai games ATARI pada *Arcade Learning Environment* (Mnih et al., 2013).

DDPG adalah metode pembelajaran *policy* yang mengintegrasikan *deep learning neural network* ke dalam *Deterministic Policy Gradient* (DPG) (Fan et al., 2021). DPG adalah metode *policy learning* yang ditingkatkan berdasarkan *policy gradient* dalam RL. *Policy gradient* menjelaskan kebijakan optimal masing-masing *step state* melalui fungsi distribusi probabilitas, dan pemilihan *action* yang didasarkan pada distribusi tersebut, sementara DPG secara langsung memperoleh nilai pasti dari keputusan *action* secara *real time* melalui fungsi *policy*

$$a = \mu(s) \quad (2.1)$$

Struktur *network* DDPG yang ditampilkan dalam Gambar 2.4 terdiri atas dua bagian: *actor network* dan *critic network*. DDPG menggunakan *actor network* $\mu(s|\theta^A)$ dan *critic network* $\mu(s, a|\theta^C)$ untuk memperkirakan fungsi *policy* $\mu(s)$ dan fungsi nilai *state action* $Q(s, a)$. θ^A dan θ^C adalah bobot *actor network* dan *critic network*. Ide utamanya adalah untuk menghasilkan *action* di bawah pedoman *actor network* dan *critic netwrok* menggunakan fungsi nilai *state action*. Hal ini bertujuan untuk menaksir *action*, kemudian memandu pembaruan bobot *network* dan *actor network*-nya sendiri melalui evaluasi.

Selama proses *training*, agen DDPG akan memperbarui properti *actor-critic* setiap *step* selama *training* serta menyimpan *experience* masa lalu menggunakan *circular experience buffer*. Agen memperbarui *actor-critic* menggunakan *mini-batch* dari *experience* yang diambil secara acak dari *buffer* dan memicu *action* yang dipilih oleh *policy* menggunakan model *stochastic noise* di setiap *training step*.

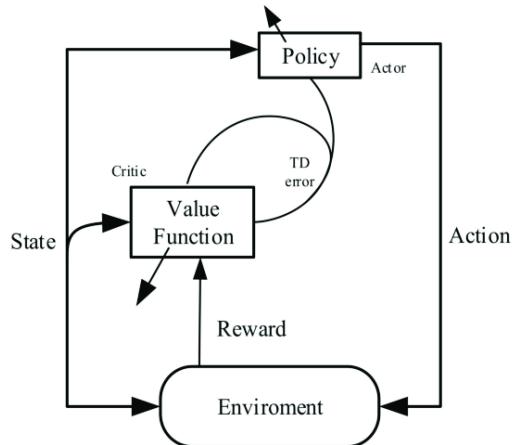
Berikut tahapan algoritma dari DDPG (Sutton et al., 2018):

1. Secara acak menginisialisasi *critic network* $Q(s, a|\theta^Q)$, *actor network* $\mu(s|\theta^\mu)$ dengan bobot θ^Q dan θ^μ .
2. Menginisialisasi *target network* Q' dan μ' dengan bobot $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$.
3. Menginisialisasi *Replay buffer* R
4. Untuk *episode* 1 sampai M, lakukan
 - Menginisialisasi proses N .
 - Menerima *initial state* s_1 dari *environment* E .
5. Untuk $t=1$ hingga T , lakukan
 - Memilih *action* $a_t = \mu(s_t|\theta^\mu) + N_t$ bedasarkan *actor network* saat ini.
 - Melakukan *action* a_t pada *environment* E , lalu menerima *reward* r_t dan berlanjut pada *state* baru s_{t+1} .
 - Menyimpan *transition probability* (s_i, a_i, r_i, s_{i+1}) pada *buffer* R .
 - Mengambil sampel secara acak dari *minibatch* dari transisi N (s_i, a_i, r_i, s_{i+1}) dari R .
 - Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$.
 - Melakukan *update* pada *critic* dengan meminimalisasi *loss*:

$$L = \frac{1}{N} \sum_i [y_i - Q(s_i, a_i|\theta^Q)]^2.$$
 - Melakukan *update* pada *actor policy* menggunakan *sampled gradient*.

$$\nabla_{\theta^\mu} \mu|_{s_i} \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_i}$$
 - Melakukan *update* pada *target network*.

$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}, \theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$



Gambar 2. 4 Skematika DDPG

2.2.2.1. DDPG *Agent Options*

Algoritma DDPG memiliki unit-unit pengaturan yang berkorelasi pada proses *training* yang akan dijalankan oleh *agent*. Pengaturan ini digunakan untuk mempercepat *training*, optimalisasi *policy*, dan meningkatkan performa *agent*. Algoritma DDPG memiliki beberapa *options*, meliputi:

- TargetSmoothFactor

Target Smooth Factor berfungsi sebagai *smoothing factor* dalam pembaruan/*update target actor* dan *target critic*.

- TargetUpdateFrequency

Target Update Frequency berguna dalam mengatur berapa banyak step antara update *target actor* dan *target critic*.

- ResetExperienceBufferBeforeTraining

Options ini berguna dalam membersihkan *experience buffer*. Hal ini bertujuan untuk memastikan bahwa pada sesi simulasi, *buffer* tidak terdapat data yang masuk.

- SaveExperienceBufferWithAgent

Options ini bertujuan untuk menyimpan *experience buffer* ketika menyimpan *agent*.

- SequenceLength

Sequence Length merupakan *options* untuk menentukan nilai *maximum batch-training trajectory* ketika menggunakan (*Recurrent Neural Network*).

- NumStepsToLookAhead

Options ini berguna untuk menentukan jumlah *future reward* yang digunakan untuk mengestimasi *value* dari *policy*.

- ExperienceBufferLength

Options ini berguna untuk mengatur ukuran dari *experience buffer* yang akan digunakan dalam *training*. Selama *training*, agent akan mengkalkulasi *update* menggunakan *mini-batch of experiences* yang dipilih secara acak dari *buffer*.

- SampleTime

Options ini berfungsi melakukan *time sampling* atau pengaturan waktu pengambilan sampel oleh *agent* pada waktu **simulais**

- DiscountFactor

Discount factor berfungsi memberikan bobot pada *future reward* dalam rangka menurunkan atau meningkatkan pertimbangan *future reward* tersebut (Siraskar, 2021). *Options ini* memiliki rentang input 0 hingga 1. Jika *discount factor* bernilai 0, maknanya *action-value* akan mengalami *immediate reward*, sedangkan jika bernilai 1 maka *agent* akan mempertimbangkan secara penuh *future reward*.

- NoiseOption

NoiseOption merupakan *option* pada DDPG yang menerapkan prinsip *Ornstein Uhlenbeck Action Noise*. *Noise* berguna dalam mengontrol perilaku eksplorasi dan eksplorasi pada *training* dengan mengatur besaran nilai dari *standard deviation* dan *standard decay rate*. Jika *agent* terjebak dalam *local optima*, maka input nilai dari *standard deviation* dan menurunkan nilai *standard deviation decay rate* dalam rangka meningkatkan eksplorasi.

2.2.2.2. DDPG Training Options

Reinforcement Learning Toolbox pada matlab juga menyediakan *training options* sehingga *user* dapat menentukan parameter-parameter *training* yang ingin ditargetkan. Terdapat beberapa *option* dalam proses training, meliputi:

- MaxEpisodes

Options ini bertujuan mengatur batas maksimum jumlah episode pada proses *training*.

- MaxStepsPerEpisode

Options ini bertujuan untuk menentukan banyaknya step dalam proses *training* 1 episode.

- ScoreAveragingWindowLength

Options ini berfungsi menghitung rata-rata dari *scores*, *reward*, dan banyaknya step pada *agent* yang sedang atau telah di-*training*.

- StopTrainingCriteria

Options ini berfungsi dalam menentukan kondisi pemberhentian *training* dengan syarat tertentu.

- StopTrainingValue

Options ini berfungsi dalam menentukan nilai target yang diinginkan untuk memberhentikan proses *training*.

- SaveAgentCriteria

Options ini berfungsi dalam menyimpan *agent* yang telah ditraining dengan melewati kondisi tertentu.

- SaveAgentValue

Options ini berfungsi menentukan nilai target dari *agent* yang telah ditraining sehingga *agent* dapat disimpan.

- Verbose

Options ini berfungsi menampilkan *training progress* pada *command line*.

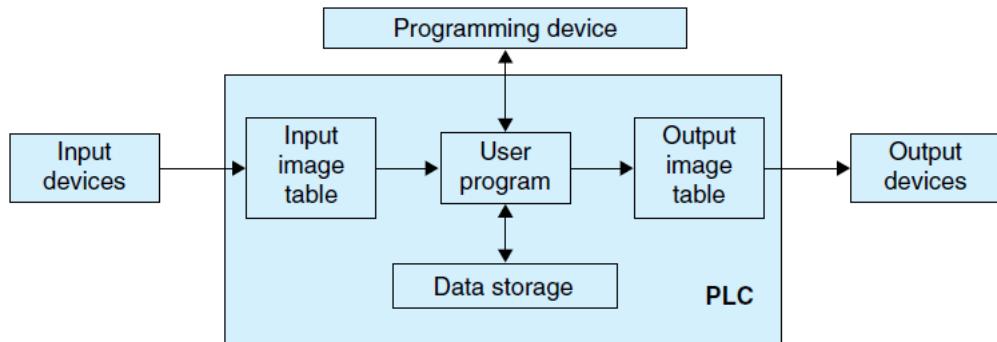
- Plots

Options ini berfungsi menampilkan *training progress* melalui Episode Manager.

2.3 Programmable Logic Control (PLC)

Programmable logic controller (PLC) adalah komputer industri yang memiliki fungsi menerima *input* dari berbagai perangkat masukan, mengevaluasi *input* berdasarkan logika pemrograman, dan memberikan *output* berupa kontrol pada perangkat keluaran (Khaled Kamel & Kamel, 2013). Modul dan blok-blok fungsional PLC dapat dilihat pada

Gambar 3. Sampel serta *input* tabel gambar diambil oleh perangkat input PLC yang terus diperbarui secara *real time*. Program *user* dimuat dalam memori penyimpanan PLC melalui perangkat pemrograman memiliki fungsi menyelesaikan logika aplikasi yang telah ditentukan sebelumnya dan memperbarui tabel logika *output internal*. Perangkat *output* digerakkan secara *real time* sesuai dengan pembaharuan pada nilai tabel *output*.

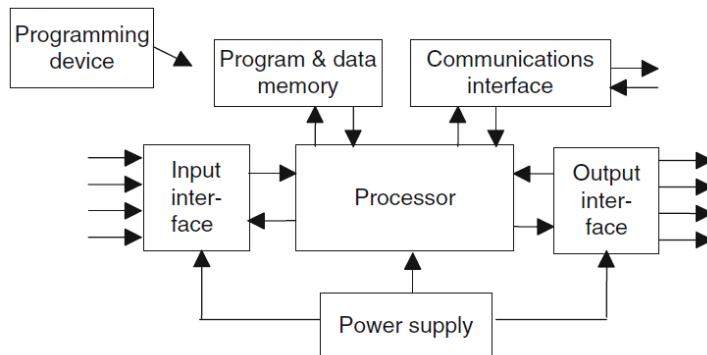


Gambar 2. 5 Arsitektur I/O PLC

Interface pada PLC dapat dicocokkan dengan semua jenis perangkat PLC, terlepas dari vendor yang dipilih. Sensor dan aktuator memungkinkan PLC untuk berinteraksi dengan semua jenis perangkat analog serta perangkat ON/OFF melalui modul I/O digital, konverter *analog-to-digital*, konverter *digital-to-analog*, dan sirkuit isolasi. Selain masukan *power supply* dan *I/O interface*, seluruh sinyal pada PLC merupakan sinyal digital yang bertegangan rendah.

PLC pada umumnya terdiri atas *Central Processing Unit* (CPU), *power supply*, memori data, modul komunikasi, perangkat *programming* dan bagian sirkuit untuk menangani data I/O (Bolton, 2017).

- *Central Processing Unit* (CPU) adalah unit sistem yang tersusun atas mikroprosesor. Sistem ini menafsirkan sinyal *input* dan melakukan kontrol sesuai dengan program yang tersimpan di dalam memori, dan mengomunikasikan keputusan yang diambil sebagai sinyal *output*.
- *Power supply* adalah unit untuk mengonversi sumber tegangan AC menjadi tegangan DC rendah (5V) yang penting bagi *processor*, serta sirkuit *input* maupun output modul *interface*.



Gambar 2.6 Sistem PLC (Bolton, 2017)

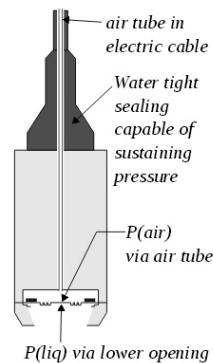
- Perangkat *programming* berfungsi untuk memasukkan program yang dibutuhkan ke memori penyimpanan pada *processor*. Program tersebut diolah di dalam perangkat dan dikirimkan ke unit penyimpanan pada PLC . Program tersebut diolah di dalam perangkat dan dikirimkan ke unit penyimpanan pada PLC.
- Unit memori berfungsi sebagai tempat penyimpanan program sistem kontrol yang akan dieksekusi oleh mikroprosesor dan tempat penyimpanan data input bagi pemrosesan sistem output. Pada penelitian ini, input devices yang digunakan adalah sensor ketinggian level air dan arus air/flow yang mana befungsi sebagai penerima sinyal untuk diproses pada CPU sedangkan output devices yang digunakan adalah *control valve*.

2.3.1. Water Level Transmitter



Gambar 2. 7 Submersible Water Level Transmitter

Water Level Transmitter adalah alat yang digunakan untuk mengukur ketinggian air di suatu wadah penampungan. Instrumen ini terdiri atas sensor pada *probe* yang terhubung dengan *coated cable* yang diturunkan hingga ke bagian dasar tangki. Pada sistem rancang bangun ini, merk yang digunakan adalah Wisner WLT 605 series dengan jenis *Submersible Water Level Transmitter* yang menggunakan prinsip tekanan hidrostatik dengan bahan silikon yang diletakkan pada *probe* bagian bawah.



Gambar 2. 8 Bagian-Bagian Submersible Water Level Transmitter

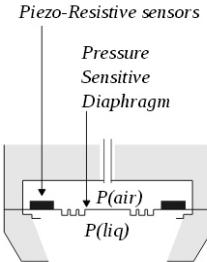
Tekanan hidrostatik adalah tekanan yang diberikan oleh cairan didalam tangki yang ditentukan oleh 2 variabel, yaitu massa jenis cairan dan ketinggian/kedalaman level. Dengan massa jenis cairan yang konstan, perubahan tekanan hidrostatik dapat koresponden dengan perbedaan ketinggian/kedalaman cairan.

$$P(liq) = \rho gh + P(air)$$



Dimana ρ dan g bernilai konstan K , maka

$$P(liq) - P(air) = Kh$$

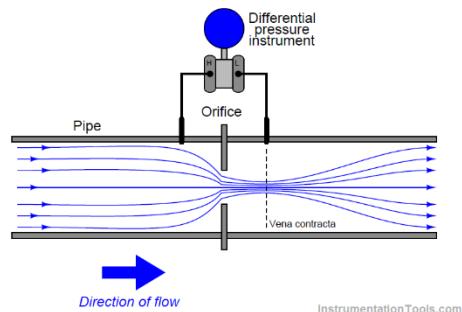


Gambar 2. 9 Deformasi Diafragma dengan sensor *Piezo-resistive*

Berdasarkan persamaan diatas, dapat dilihat bahwa perbedaan tekanan akan sebanding dengan ketinggian dari level pada tangki. Persamaan ini memiliki unit ukur yang disebut sebagai mHO (Besar tekanan ketika dimasukkan kedalam air **apada ketinggian tertentu**). *Submersible water level transmitter* bekerja dengan tekanan yang diberikan oleh cairan $P(liq)$ dan menguranginya dengan tekanan udara $P(air)$ didalam *probe* menggunakan diafragma tekanan tunggal yang sensitif dan ventilasi udara di dalam kabel koneksi. Saat *level transmitter* dicelupkan lebih dalam ke dalam cairan, $P(liq)$ menjadi lebih tinggi dari $P(air)$ dan diafragma mengalami deformasi. Deformasi bahan *piezo-resistive* ini akan mengonversi deformasi ini menjadi hambatan listrik.

2.3.2. *Flow meter*

Flow meter adalah instrumen yang dipakai untuk mengetahui aliran material (gas, cairan) dalam suatu jalur aliran pada waktu tertentu. *Flow meter* terdiri atas perangkat primer dan perangkat sekunder. Perangkat primer pada *flow meter* berbentuk



Gambar 2. 10 *Orifice Flow Meter*

sebagai *orifice* yang mengganggu laju aliran yang menyebabkan terjadinya penurunan tekanan dan hal ini akan menghasilkan produk berupa sinyal analog. Output sinyal akan diterima oleh bagian sekunder yang berfungsi dalam menampilkan, merekam, dan/atau mentransmisikannya sebagai hasil pengukuran dari laju aliran pada pipa.

Pada penelitian ini, flow meter yang digunakan adalah SFI-801 Sight flow indicator dengan menerapkan prinsip dasar *Hall effect* yang diakibatkan reaksi dari perputaran rotor yang termagnetisasi. *Hall effect* adalah suatu peristiwa berbeloknya aliran listrik dalam pelat konduktor dalam hal ini rotor karena pengaruh medan magnet. Flow transmitter mengubah instensitas medan magnet yang terdapat disekelilingnya menjadi sinyal listrik. Sinyal **input** yang **masuk** berupa tegangan dengan interval 0-10 V yang membutuhkan supply tegangan sebesar 24VDC. 



Gambar 2. 11 Dwyer SFI-801 Sight Flow Indicator dan A-712 Sensor

Pemilihan *flow meter* ini juga disesuaikan dengan penggunaan pipa *PolyVinyl Chloride* (PVC) $\frac{1}{2}$ inch serta memiliki bahan material dasar polysulfone berstandar FDA/NSF yang memiliki batas temperatur minimum -28°C dan maksimum 100°C dengan akurasi $\pm 5\%$. *Flow meter* ini juga dilengkapi dengan sensor A-712 yang dapat digunakan dalam *me-monitoring* aliran air dan mentransmisikan sinyal input ke PLC.

2.3.3. Control Valve

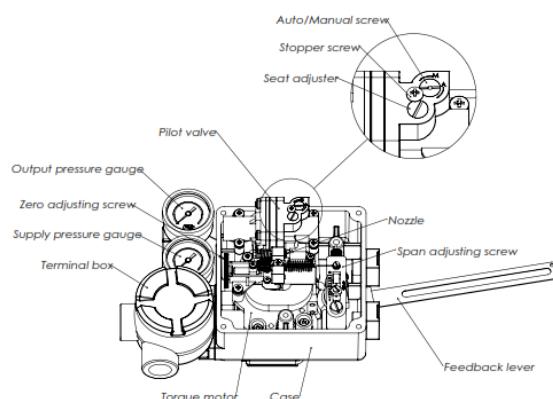
Salah satu komponen dasar dari berbagai jenis sistem kendali adalah rangkaian kontrol akhir yang memiliki berbagai spesifikasi serta bentuk bergantung pada penerapan sistem kendali yang diinginkan. Salah satu jenis dari rangkaian kontrol akhir yang paling umum digunakan adalah *pneumatic control valve*. Instrumen ini berfungsi mengatur aliran dari suatu fluida melalui bukaan yang diperintahkan. Pada proses kendali, siklus kontrol menghasilkan output disetiap sebagai hasil akhirnya dimana setiap siklus menerima dan secara internal menghasilkan gangguan pada variabel proses. Gangguan ini akan diatasi melalui desain sistem kendali untuk mempertahankan variabel



Gambar 2. 12 Power-Genex *Electro-Pneumatic Positioner EPL series*

proses, dimana dalam penelitian ini merupakan ketinggian air pada tangki penampung.

Pada penelitian ini, *control valve* yang digunakan menggunakan prinsip dasar *electro-pneumatic positioner*. Jenis ini berkerja dengan memanfaatkan energi pneumatik yang tersambung dengan *compressor* sebagai penggerak dan sinyal elektrik sebagai media kontrol yang tersambung dengan perangkat pemograman. Sinyal elektrik dialirkan ke kumparan yang terpasang pada katup pneumatik dengan mengaktifkan saklar, sensor,



Gambar 2. 13 Skematika *Electro-Pneumatic Positioner*

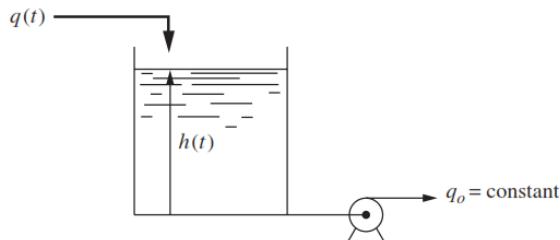
atau saklar pembatas. Sinyal akan diterima kumparan dan akan menghasilkan medan elektromagnetik serta mengaktifkan/mengaktivasikan katup pengatur arah sebagai rangkaian kontrol akhir.

BAB 3

METODE PENELITIAN

3.1 Cara Kerja Sistem

Sistem kendali yang dilakukan pada proses ini merupakan pengendalian ketinggian air melalui pengaturan bukaan *valve* sesuai dengan ketinggian yang terukur pada tangki. Hal ini dilakukan dengan mengatur *flow in* (q_i) serta *flow out* (q_o) pada tangki dengan produk berupa nilai ketinggian air yang telah ditentukan/*setpoint* hingga sistem berjalan secara stabil dan kontinyu. Ketika ketinggian air yang terukur tidak sama dengan *setpoint* yang ditetapkan, maka *level transmitter* akan secara otomatis memberikan sinyal kepada *control valve* untuk membuka *valve* hingga selisih/error antara *set point* serta ketinggian aktual bernilai minimal atau 0. Dan sistem akan mempertahankan posisi ini hingga perubahan setpoint terjadi/*steady state*. Proses ini termasuk jenis *single-input single-output* (SISO) dan lebih dikenal dengan proses *water level control*.



Gambar 3. 1 Sistem pengukuran ketinggian dengan *outlet flow* konstan (Steven & R.Coughanowr, 2009)

Ketika *flow in* dan *flow out* bernilai sama, maka ketinggian pada tangki tidak akan berubah sehingga sering disebut sebagai keadaan *steady state*. Hal ini dapat direpresentasikan dalam persamaan kesetimbangan massa

$$\frac{d(\rho V)}{dt} = \rho q_i - \rho q_o \quad (3.1)$$

Massa jenis pada persamaan ini bernilai konstan sehingga dapat dieliminasi dan menyisakan variabel *volume*. Jika dijabarkan, variabel *volume* merupakan multiplikasi antara luas area A dan ketinggian h dari tangki.

$$\frac{d(Ah)}{dt} = \rho q_i - \rho q_o \quad (3.2)$$

Variabel luas alas tangki dapat dipindahkan ke **RHS** dan didapatkanlah persamaan baru

$$\frac{dh}{dt} = \frac{q_i}{A} - \frac{q_o}{A} \quad (3.3)$$

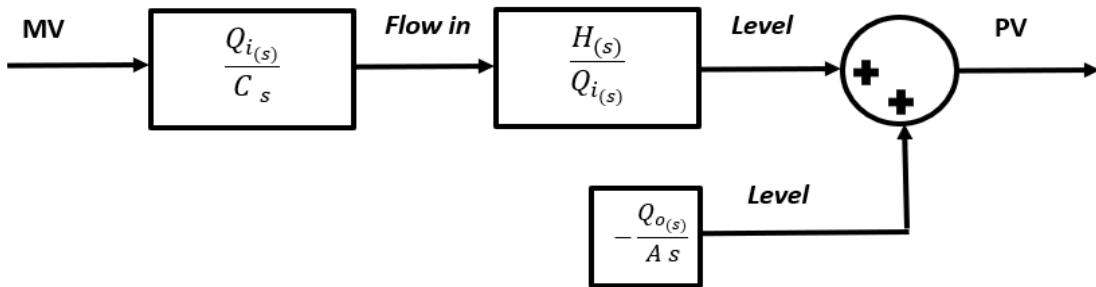
Berdasarkan persamaan diatas, dapat dilihat bahwa fungsi transfer terbagi menjadi 2, yaitu fungsi transfer yang bergantung pada *flow in* (q_i) dan *flow out* (q_o). Jika diubah dalam bentuk **laplace**, maka formulasi terkonversi menjadi

$$H_{(s)} = \frac{Q_{i(s)}}{A s} - \frac{Q_{o(s)}}{A s} \quad (3.4)$$

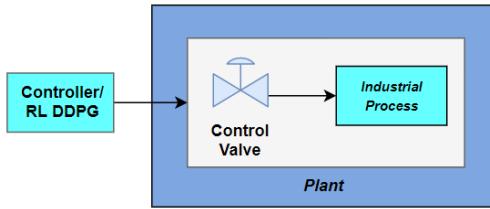
Dan diketahui bahwa *flow out* menggunakan pompa, sehingga memiliki konstanta *flow* tersendiri. Pada *flow in*, Pada *flow in*, *flow* dipengaruhi oleh bukaan *control valve* sehingga dapat didefinisikan kembali

$$\frac{Q_{i(s)}}{C_s} \times \frac{H_{(s)}}{Q_{i(s)}} = \frac{H_{(s)}}{C_s} \quad (3.5)$$

Sehingga dapat direpresentasikan seperti pada gambar



Gambar 3. 2 Pemodelan secara matematis *water tank system*

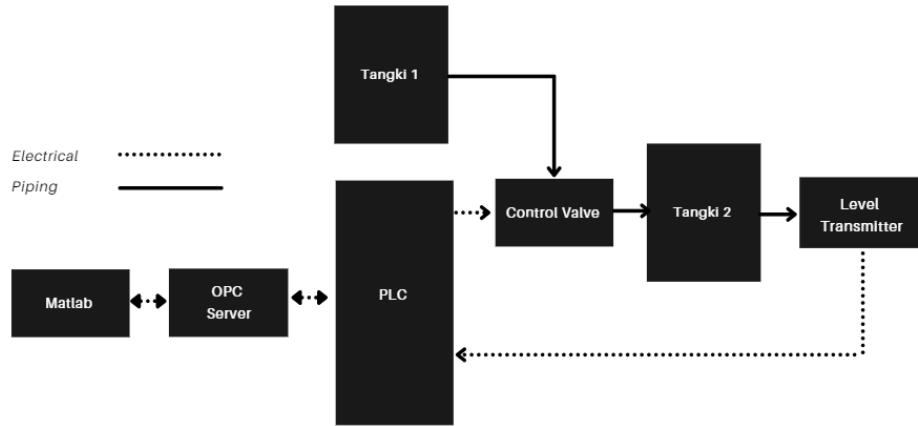


Gambar 3. 3 Blok Komponen Dasar

Proses ini akan disimulasikan menggunakan Simulink Matlab secara dinamis hingga sistem mencapai titik kesetimbangan. Desain keseluruhan sistem ini agak berbeda dengan beberapa penelitian yang dilakukan oleh beberapa peneliti lainnya yang menggunakan besaran fisis *flow* sebagai besaran yang ditinjau, sedangkan penelitian menggunakan ketinggian air pada tangki. Namun, desain sistem RL DDPG pada penelitian secara garis besar merupakan adaptasi dari penelitian oleh (Siraskar, 2021) dan desain perangkat keras diadaptasi dari penelitian oleh (Ahmad, 2019) yang perbedaanya terletak pada sistem kontrol yang digunakan. (Siraskar, 2021)

Pada gambar 3.2, *Reinforcement Learning* bertugas sebagai Controller yang mengatur proses bukaan *control valve* sehingga *flow* air yang masuk dapat dikendalikan hingga ketinggian yang diinginkan pada *industrial process*. Plant meliputi *flow transmitter*, *level transmitter*, pompa, *ball-valve*, dan *control valve*. *Flow transmitter* berfungsi untuk mengukur laminaritas atau kestabilan aliran yang masuk sehingga peneliti dapat menjaga jalannya *training* sehingga tidak mengganggu proses pengukuran fungsi transfer di setiap % bukaan valve. *Level transmitter* berfungsi mengukur ketinggian air pada tangki yang digunakan sebagai besaran fisis yang akan digunakan dalam kendali ketinggian air. Ketinggian air dipengaruhi oleh debit air yang masuk melewati flow in (q_i) dimana % bukaan control valve serta ball valve mengatur debit air yang masuk ke tangki.



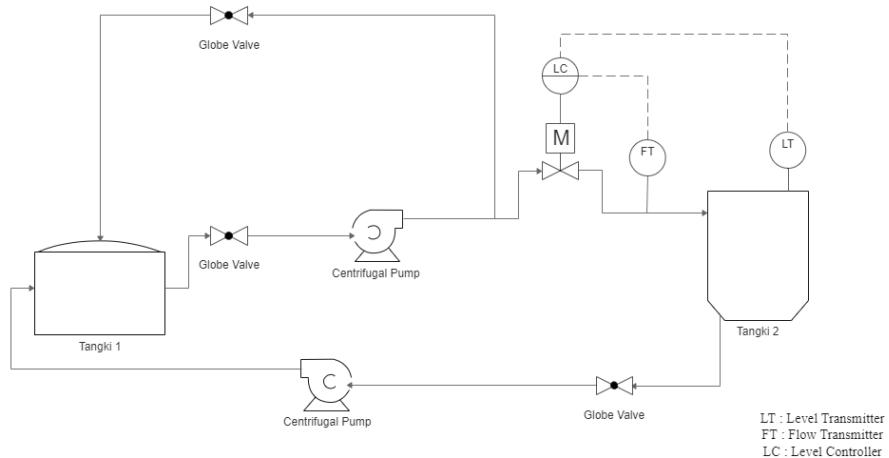


Gambar 3.4 Diagram Blok Kinerja Sistem

Pada gambar 3.3, *Programmable Logic Controller* digunakan untuk mengatur kinerja *plant* sesuai dengan variabel-variabel yang telah ditetapkan oleh Matlab. Didalam Matlab dibuatlah program yang berisikan komputasi berbasis matriks dan membuat kerangka serta permodelan simulasi didalam Simulink. Komunikasi antara PLC dan Matlab dilakukan menggunakan *Open Platform Communication Server* (OPC Server). Pada PLC sendiri, dibuatlah skema khusus yang digunakan untuk menggambarkan secara grafis diagram rangkaian elektronika dan perangkat keras komputer berdasarkan logika berbasis-relay yang disebut sebagai *Ladder diagram* menggunakan software *Speed PLC*.

3.2 Rancang Bangun Sistem

Penelitian ini terdiri atas beberapa perangkat keras serta sistem *piping* yang dibuat berkelok-kelok serta lurus untuk mengatur stabilitas *flow air* yang mengalir. Pipa memiliki diameter $\frac{1}{2}$ inch dengan panjang lintasan keseluruhan pipa ± 15 -meter dengan sistem yang akan berakhir pada tangki 1 kembali secara terus menerus. Dari gambar diatas akan dijelaskan mengenai prinsip kerja dari masing perangkat keras yang digunakan meliputi, PLC, *Level Transmitter*, *Flow Transmitter*, serta *Control Valve* pada subbab 3.2.1 dan 3.2.2.



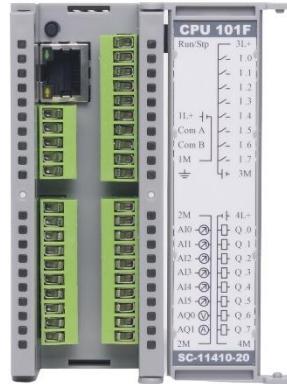
Gambar 3. 5 Gambar P&ID dari Rancang Bangun



3.2.1. PLC Fultek CPU 101 F

Programmable Logic Controller (PLC) adalah instrumen khusus dari *controller* berbasis mikroprosesor dengan memori yang dapat diprogram untuk menyimpan instruksi serta mengimplementasikan fungsi seperti *logic*, *sequencing*, *timing*, *counting*, dan perhitungan aritmatika sehingga mampu melakukan kendali pada proses (Siraskar, 2021). Pada awalnya PLC digunakan untuk menggantikan relai logika, namun jangkauan fungsinya yang terus meningkat membuat pengaplikasian instrumen ini semakin kompleks.

Pada rancang bangun sistem kendali ini, peneliti menggunakan PLC Fultek CPU 101F dengan spesifikasi 32 KB penyimpanan program, 1-unit MBbit Ethernet Full Duplex, Modbus TCP, Web Server dengan ruang penyimpanan file sebesar 512 KB, 1 buah port serial RS485, Modbus RTU, 8 koneksi input digital 200 KHz, 8 koneksi output digital 655 KHz 0,1 A, 6 koneksi 0-10/0-20 mA 12-bit input analog, 1 buah 0-10V dan 1 buah 0-20 mA 12 output analog.

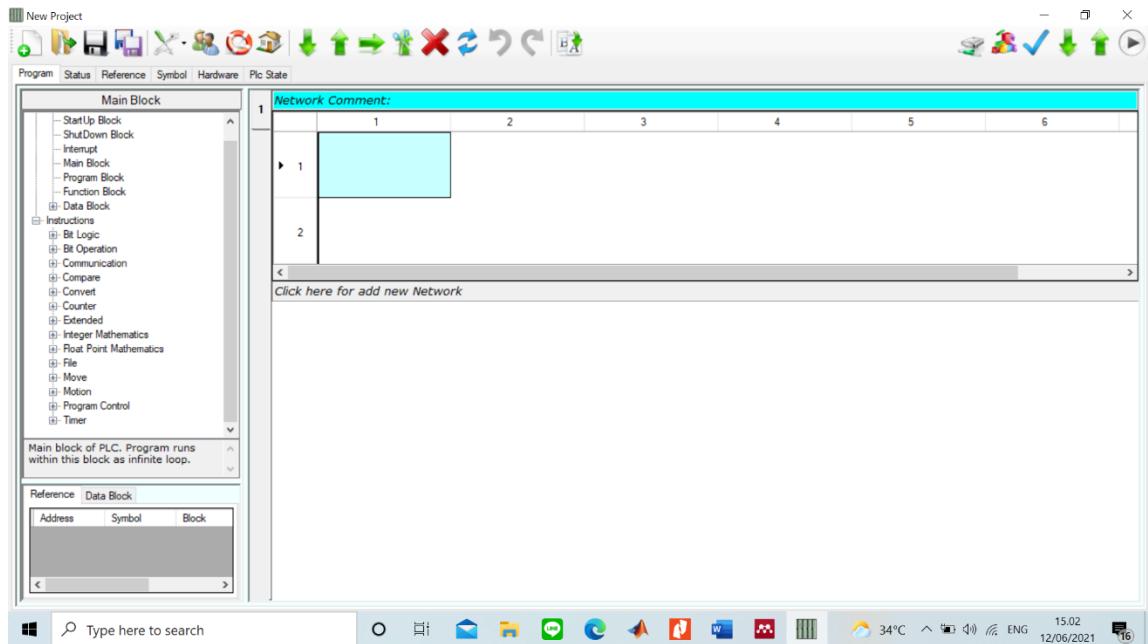


Gambar 3. 6 PLC Fultek CPU 101F

Kelebihan dari PLC tipe ini adalah fungsi *fast counter* yang mampu membaca frekuensi hingga 200 KHz ketika satu input digital diaktifkan. Hal ini berpengaruh pada *sampling time* ketika PLC membaca besaran analog pada koneksi pin input analog. Pin *Analog Input* memiliki resolusi data sebesar 12-bit dengan jangkauan data sebesar diantara 0 hingga 4095. PLC ini juga memiliki pin yang dapat diatur jenis masukan analog yang ingin diukur, meliputi input tegangan 0-10V ataupun input arus sebesar 0-20 mA.

3.2.2. *Software Speed PLC*

Software ini digunakan untuk membuat algoritma pemrograman menggunakan *ladder diagram* atau kode Mneuemonic yang dituliskan ke dalam PLC. Untuk memulai penggunaan Speed PLC, diperlukan pembuatan file *project* dengan mengklik “New Project” lalu memilih tipe PLC yang digunakan yang mana pada penelitian ini menggunakan PLC Fultek CPU 101F.



Gambar 3.7 Software Speed PLC

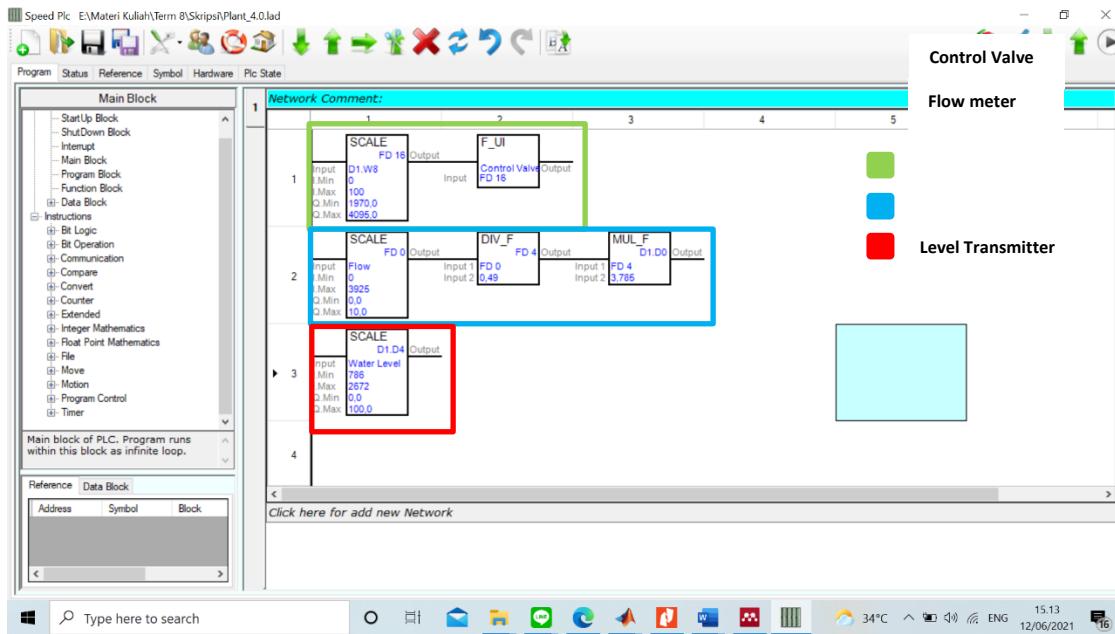
Peneliti menggunakan beberapa instruksi *ladder diagram* untuk melakukan pengukuran melalui sinyal input analog yang diterima dan dikonversi menjadi format penulisan baru, meliputi tabel dibawah ini

Scale		Melakukan konversi dari input berupa <i>signed numbers</i> ke <i>floating-point numbers</i> dengan skala yang telah dispesifikasi
Multiplication		Instruksi melakukan pengkalian dengan input berupa <i>floating point</i>

Division		Intruksi melakukan pembagian dengan input berupa <i>floating point</i>
Float to Unsigned Integer		Melakukan konversi dari input berupa <i>floating-point number</i> ke <i>unsigned number</i>

Tabel 3. 1 Command Block pada Speed PLC

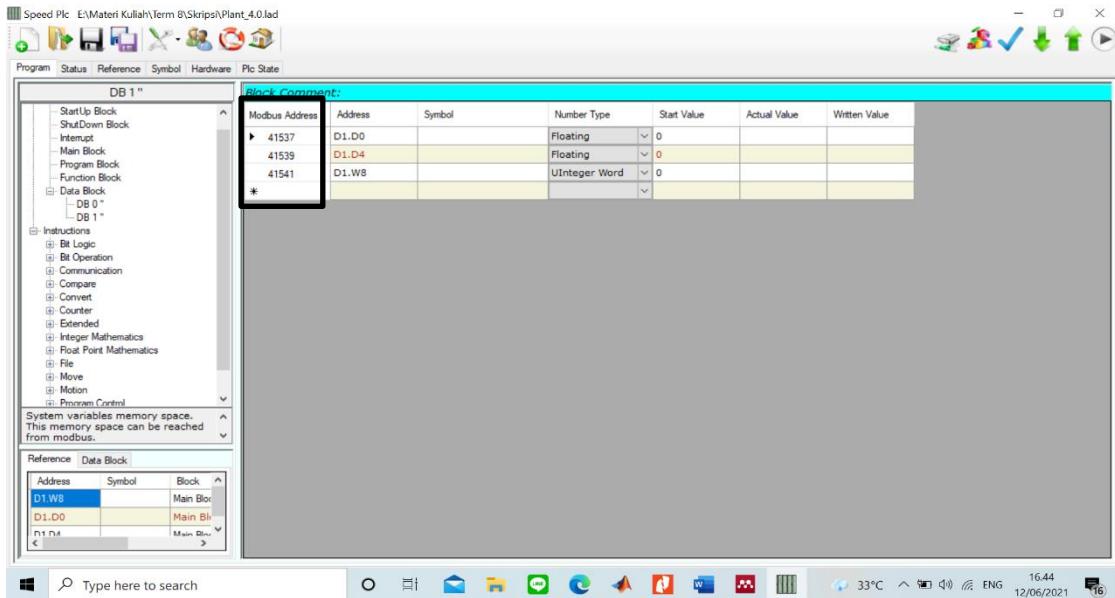
Ladder Diagram yang digunakan pada sistem pengendali ketinggian memiliki 3 variabel penting, yaitu *Control Valve*, *Flow Meter*, dan *Level Transmitter*. Maka dari itu, perlu dibuat sistem pemrograman I/O untuk memproses data inputan berupa sinyal analog dari ketiga perangkat ini dimana unit input terdiri atas *Flowmeter* dan *Level Transmitter*, sedangkan unit output terdiri atas *Control Valve*.



Gambar 3. 8 Konfigurasi Pemrograman pada Programming Device

3.2.3. Open Platform Communication

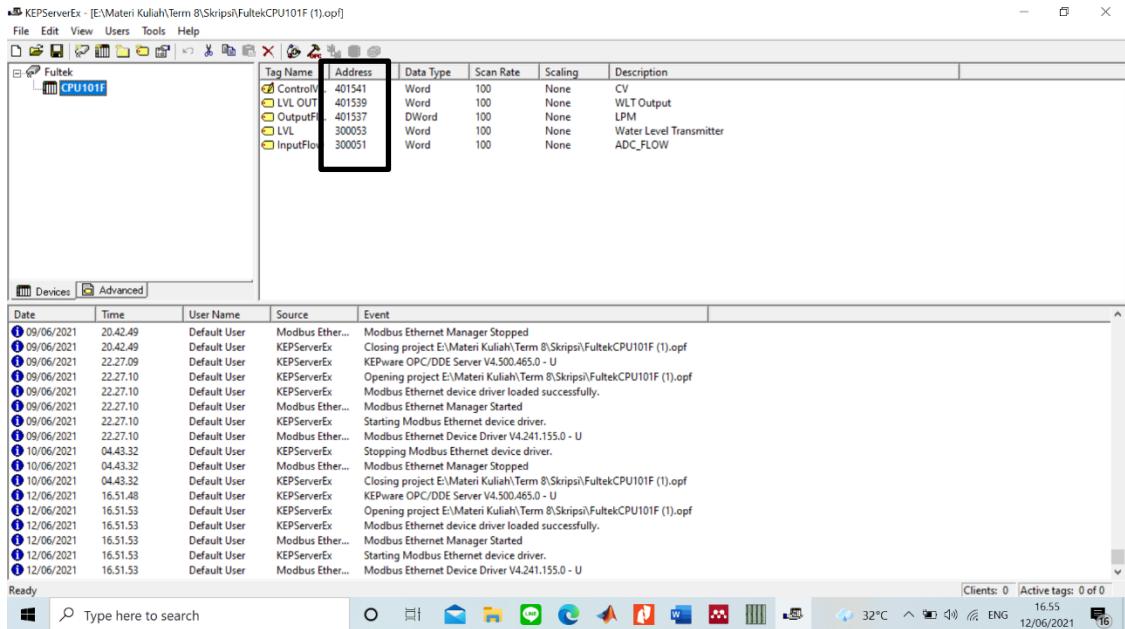
Pada gambar 3.7, *address number* yang disesuaikan pada OPC diambil dari Data Block 1 (DB 1) Speed PLC. Setelah mendapatkan informasi mengenai *address number* melalui kolom *modbus address*, maka dilakukan penulisan ulang pada OPC server. Hal ini bertujuan untuk komunikasi yang *di-input* kedalam KepServer sebagai tag untuk pengambilan nilai pada perangkat terkait, meliputi *Control Valve* (*ControlValve*), input *level transmitter* (*LVL*), output *level transmitter* (*LVL Out*), input *flowmeter* (*InputFlow*), dan output *flowmeter* (*OutputFlow*).



Gambar 3.9 Address Number pada Data Block 1 Speed PLC

Penulisan ini dilakukan dengan memberikan *name tag*, *address number*, tipe data dan tipe *client access*. Dikarenakan dasar penulisan tipe data PLC terdiri atas Word dan Bit, maka tipe data yang dipilih pada OPC adalah tipe data Word atau DWord. Lalu, komunikasi melalui Simulink Matlab dilakukan dengan memakai Library Browser OPC

Read dan OPC write yang dikoneksikan terlebih dahulu menggunakan OPC Data Access Explorer.



Gambar 3. 10 Konfigurasi Address Number pada OPC KEPServerEx

Pada toolbar ini, data input dan output dari komunikasi yang dilakukan oleh OPC di-*load* dan secara otomatis akan tersambung dengan MATLAB. Setelah melakukan *load* pada OPC Data Access Explorer, blok OPC Read dan OPC write dapat digunakan dengan meng-*add address* berupa tag-tag yang terdapat pada OPC server. Pada blok OPC Read, tag yang dipilih adalah tag Control Valve, sedangkan blok OPC Write terdiri atas tag ControlValve, LVL, dan InputFlow.

3.3 Rancangan Software

Perancangan Software pada penelitian ini terdiri atas 2 bagian, yaitu perancangan Speed PLC dan perancangan Matlab-Simulink. Kedua sistem saling terhubung satu sama lain melalui OPC dikarenakan Matlab-Simulink membutuhkan akses alamat pada Speed PLC dalam rangka pengakuisian data yang terukur oleh unit input (*flow* meter dan *level*



transmitter) serta unit output (*control valve*). Data ini akan dipakai dalam bentuk *error* yang akan diproses melalui training dan dikendalikan oleh *Reinforcement Learning*.

3.3.1. Perancangan Ladder Diagram

Pada penelitian ini, *Ladder Diagram* dibuat menggunakan Speed PLC. Penyusunan *Ladder Diagram* berasal dari penggunaan unit *input* dan unit *output* yang dibutuhkan

No	Address Memory	Modbus Address	Data Type	Command block	Description
1	D1.W8	41541	Word	SCALE	Input Control Valve
2	IW 100	30053	Word	SCALE	Input Flow Meter
3	IW 104	30051	Word	SCALE	Input Level Transmitter
4	D1.D0	41537	Word	MUL_F	Output Flow Meter
5	D1.D4	41539	Word	SCALE	Output Level Transmitter

Tabel 3. 2 Address memory pada Speed PLC

Setelah menentukan *modbuss address* melalui aktivasi *data block*, maka peneliti menyusun sistem algoritma pemrograman yang telah direncanakan. Hal ini tersusun atas beberapa tugas fungsional, meliputi pengaturan *control valve*, akuisisi data dan konversi besaran fisis.

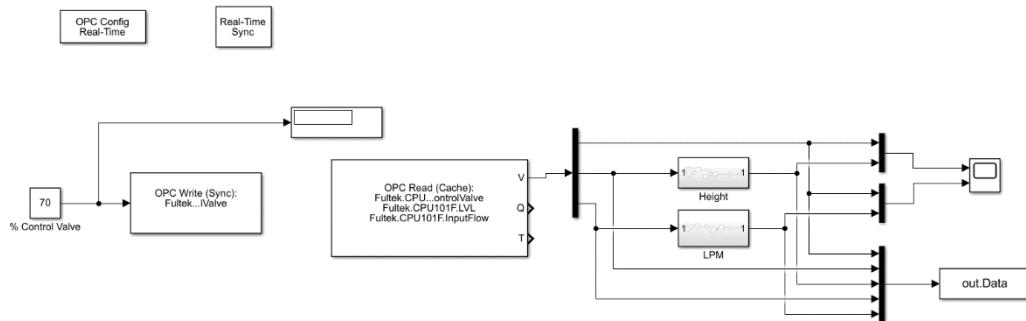
3.3.2. Perancangan Matlab – Simulink

Simulink merupakan fitur tambahan atau **ekstensi** dari Matlab yang memungkinkan *user* secara cepat dan akurat membangun model komputasi dari sistem yang dinamis menggunakan notasi diagram blok (Dabney & Harman, 1998). Dalam penelitian ini, simulink akan digunakan dalam memberikan perintah ke PLC, akuisisi serta visulisasi data, dan perancangan Reinforcement Learning. Fungsi-fungi ini akan digunakan dalam 2 tahap penyusunan sistem, yaitu pemodelan *control valve* dan pemodelan *Reinforcement Learning*.

3.3.2.1. Pemodelan Control Valve

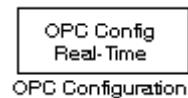
Pemodelan *control valve* tersusun atas sistem komunikasi OPC dan konversi yang digunakan dalam pengakuisisian data pencarian fungsi transfer melalui data *flow* dan ketinggian air pada tangki. Fungsi transfer didapatkan dengan memvariasikan nilai *manipulated variable* atau % bukaan *control valve* sehingga didapatkan besaran fisis dari ketinggian atau *level* pada tangki.

Struktur ini dibentuk dengan berbagai blok fungsional dari Simulink seperti yang terlihat pada gambar 3.10.



Gambar 3. 11 Model untuk mencari Fungsi Transfer *Control Valve*

- Blok OPC Configuration



Gambar 3. 12 OPC Configuration

Blok ini berfungsi dalam mendefinisikan OPC Client yang akan digunakan pada model, mengkonfigurasi keadaan *pseudo real-time* pada model, dan mengonfigurasi kondisi pada OPC *error* dan *event*. Blok ini hanya memiliki 1 output yang digunakan untuk menampilkan tingkat latensi model (waktu yang dibutuhkan untuk menunggu setiap *step* simulasi hingga mencapai kondisi *pseudo real-time*).

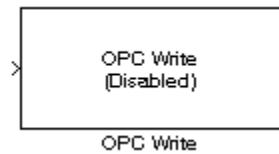
- Blok Real-Time Sync



Gambar 3. 13 Real-Time Sync

Blok ini berfungsi dalam menentukan *time sampling* serta *missed tick values* yang diterapkan pada simulasi. Blok ini tidak memiliki port sama sekali.

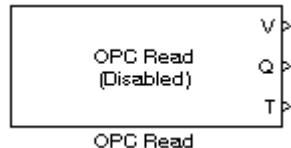
- Blok OPC Write



Gambar 3. 14 OPC Write

Blok ini berfungsi dalam penulisan data ke satu atau lebih item pada OPC server. Operasi ini dapat dilakukan secara sinkron maupun asinkron dengan cara menginput ID item yang telah ditentukan pada OPC Write.

- Blok OPC Read



Gambar 3. 15 OPC Read

Blok OPC Read berfungsi dalam pembacaan data dari satu atau lebih item pada OPC server baik secara sinkron ataupun asinkron. Output blok ini terdiri atas *values* (V) yang merupakan data item yang telah ditulis, *quality ID* (Q) dan *time stamps* (T) yang terkait dengan setiap nilai dari data tambahan.

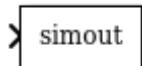
- Blok Constant



Gambar 3. 16 Constant

Blok *Constant* digunakan untuk memberikan sinyal bernilai *complex* atau *real*.

- Blok To Workspace



Gambar 3. 17 To Workspace

Blok ini berfungsi untuk menulis data dari sinyal input ke dalam *workspace*. Selama simulasi berlangsung, blok ini meng-*input* data pada *internal buffer* dan akan dimasukkan ke dalam *workspace* ketika simulasi telah selesai.

- Blok Scope



Gambar 3. 18 Scope

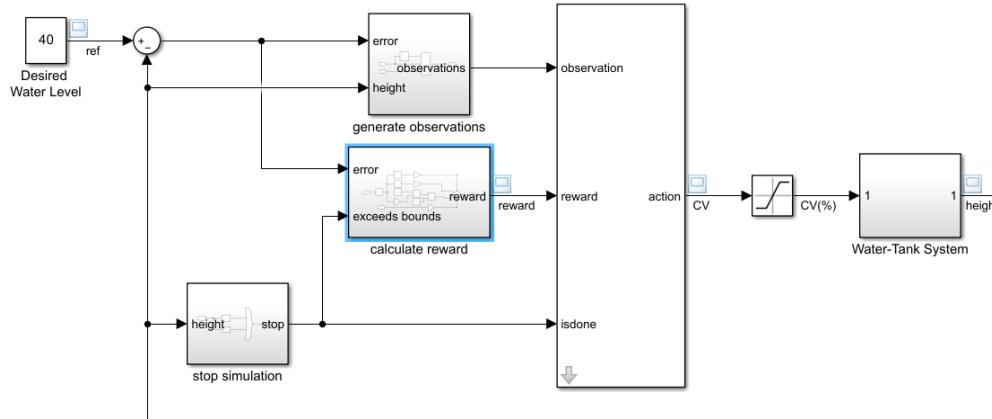
Blok Scope digunakan untuk menampilkan sinyal yang dihasilkan oleh simulasi yang sedang atau telah dijalankan.

Keseluruhan sistem dimulai dengan pembacaan input oleh blok OPC write yang dihubungkan ke *control valve* pada *plant* sehingga sinyal input masuk ke *control valve* dan melakukan proses bukaan. Lalu, OPC read mentransmisikan data input dari sensor-sensor yang dipakai, meliputi *level transmitter* dan *flow transmitter*. Data-data ini kemudian akan ditampilkan oleh scope dan ketika simulasi telah selesai maka data akan disimpan ke dalam *workspace* melalui blok To Workspace. Data yang telah dikumpulkan oleh blok ini pada *workspace* akan diolah serta dianalisis.

3.3.2.2. Pemodelan Reinforcement Learning

Pemodelan RL didalam Simulink dengan berbagai blok yang sudah tersedia pada library browser. Model Simulink akan digunakan dalam rangka melakukan *training agent* RL sehingga pengendalian proses dapat dilakukan sesuai dengan *target reward* yang diinginkan. Simulasi dilakukan dengan menggunakan algoritma *Deep Deterministic Policy Gradient* yang telah tersedia pada

Reinforcement Learning toolbox di Matlab. Simulasi algoritma *agent* terdiri atas pembuatan *environment*, model matematis serta reward yang akan digunakan dalam proses *agent training*.

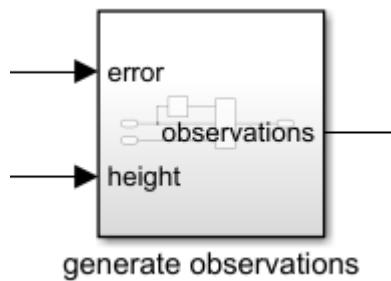


Gambar 3.19 Model Water Tank Reinforcement Learning Environment Model

Terdapat beberapa blok yang digunakan dalam memodelkan sistem pengendali ketinggian air pada Simulink.

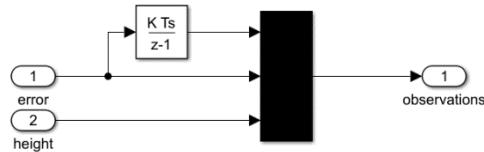
- Generate Observations

Environment dari RL menerima sinyal *action* dari *agent* dan menghasilkan sinyal observasi sebagai outputnya. Sinyal *action* pada *environtment* ini adalah sinyal kontrol ketinggian air yang akan dikirimkan pada plant



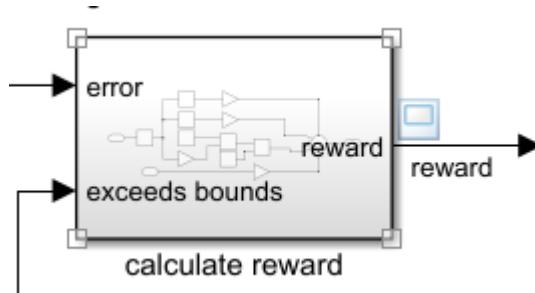
Gambar 3.20 Generate Observation

Terdapat 3 variabel pada blok ini, meliputi *integrated error*, *error* dan ketinggian yang terukur. Variabel-variabel ini akan disimpan dalam rangka proses *training agent*.



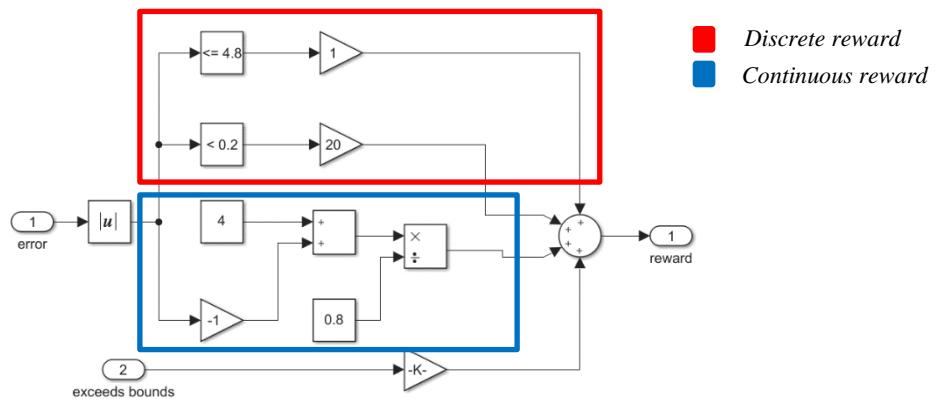
Gambar 3. 21 Variabel-variabel pada blok *Generate Observation*

- Calculate Reward



Gambar 3. 22 Calculate Reward

Block Calculate reward digunakan sebagai indikator keberhasilan *training* yang dilakukan oleh agent dalam bentuk *scalar reward signal*.



Gambar 3. 23 Sistem Kalkulasi Reward

Pada penelitian ini, terdapat 2 jenis reward yang digunakan, yaitu *discrete reward* dan *continuous reward*. Penggabungan kedua jenis reward ini sering disebut sebagai

sistem *hybrid*. *Discrete reward* berbentuk *reward* yang diberikan jika suatu kondisi terpenuhi.

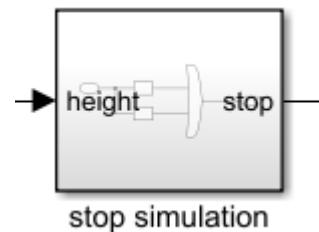
$$Reward = \begin{cases} 1, & \text{if } |e| \leq \Delta_1 \\ 20, & \text{if } |e| < \Delta_2 \\ -100, & \text{if } (y \leq 0, y > 6) \end{cases} \quad (3.4)$$

Δ_1 dan Δ_2 adalah *margin* yang **dizinkan**, dimana Δ_1 bernilai 4,8 dan Δ_2 bernilai 0,2. Dan akan mendapat *penalty (negative reward)* sebesar 100 jika melewati *exceed bound* y .

Continuous reward dapat didefinisikan sebagai sistem *reward* yang melibatkan *error* didalam fungsinya pada gambar 3.22. Nilai 4 pada *continuous reward* didapatkan dari probabilitas *error* terbesar dari ketinggian air yang memiliki rentang 1-5 dm. Dan angka 0.8 befungsi sebagai pembagi dalam rentang nilai 1-5.

$$Reward = \begin{cases} -100, & \text{if } (y \leq 0, y > 6) \\ \frac{-e + 4}{0,8}, & \text{otherwise} \end{cases} \quad (3.5)$$

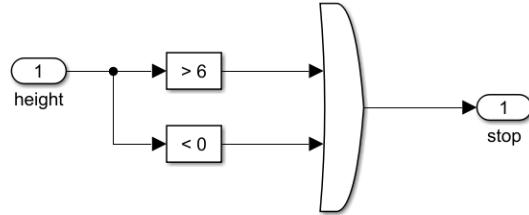
- Stop Simulation



Gambar 3. 24 Stop Simulation

Blok ini berfungsi untuk membatasi *training* sehingga tidak melewati batas atas/bawah (*exceed bounds*) dari ketinggian tangki saat proses simulasi berlangsung. Sistem akan menghentikan *training* jika mendeteksi adanya ketinggian air yang melewati batas yang telah ditentukan, sehingga memberikan nilai *penalty* yang

cukup besar pada *agent* melalui penghitungan yang dilakukan oleh blok calculate reward.



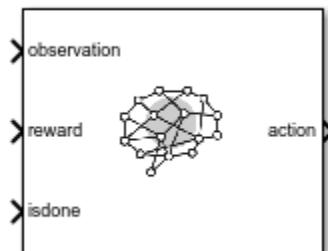
Gambar 3. 25 Sistem Exceed Bound Di Dalam Blok Stop Simulation

Jika dijabarkan secara matematis, dapat ditulis

$$reward = \begin{cases} -100 & \text{if } height \leq 0 \text{ dm, } height \geq 6 \text{ dm} \end{cases} \quad (3.6)$$

Dimana sistem ini menggunakan **logic gate** OR untuk mendefinisikan makna matematis dari 2 kategori yang telah ditetapkan.

- RL Agent

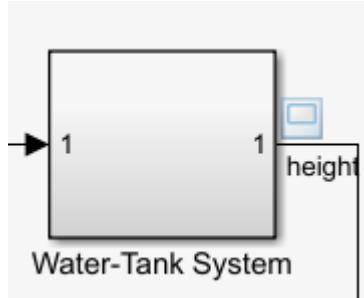


Gambar 3. 26 RL Agent

Blok ini merupakan toolbox dari MATLAB yang digunakan untuk simulasi dan melakukan *training* pada agent reinforcement learning, dimana pada penelitian ini *agent* yang digunakan adalah DDPG. Blok ini terdiri atas 3 input yang meliputi observation, reward, dan is done. Input observation bertugas menerima sinyal observasi yang diberikan oleh *environment*, input reward bertugas menerima sinyal reward yang dihitung melalui obeservasi data, dan is done yang bertugas menerima

sinyal kategori dalam rangka mengehentikan simulasi jika masuk pada kondisi tertentu.

- Water Tank System

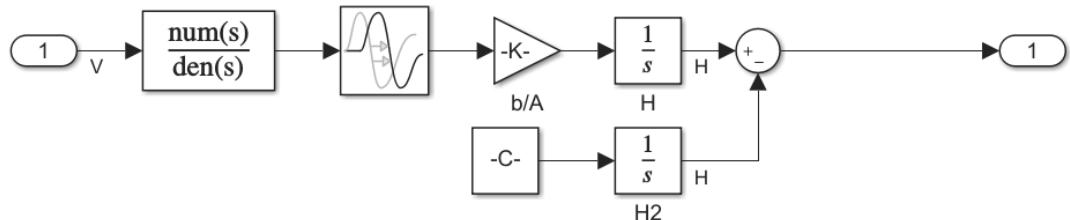


Gambar 3. 27 Water Tank System

Blok ini berisi informasi mengenai karakteristik dari *plant* yang digunakan yang mana merupakan model matematika dari sistem rancang bangun. Pembuatan model ini sering disebut sebagai proses *First-Order Plus Time Delay* (FOPTD) yang menggunakan blok *transfer function* dan *time delay*.

$$G(s) = \frac{k}{(1 + Ts)} e^{-Ls} \quad (3.7)$$

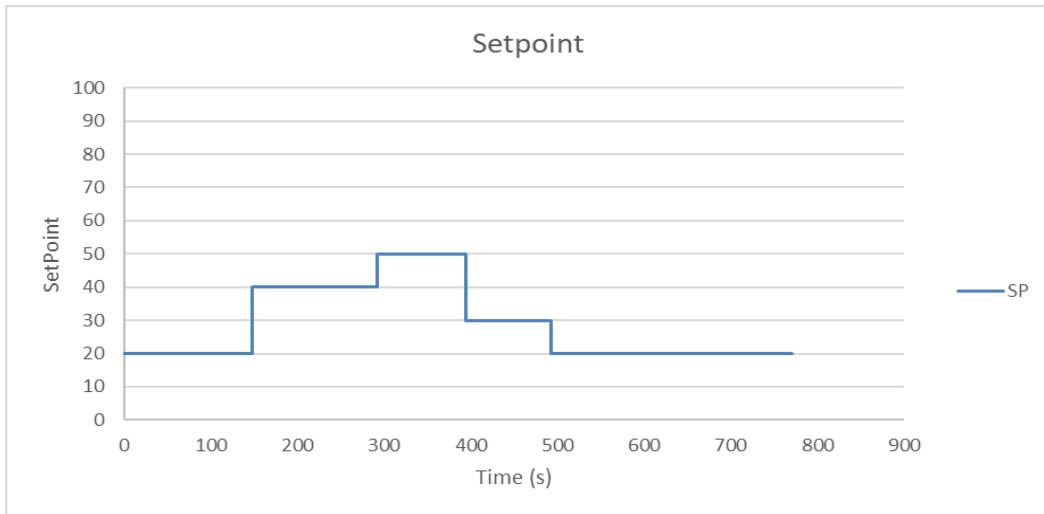
Dimana $k = 0.2737$, $T = 0.1$, dan $L = 0.09$



Gambar 3. 28 Model Matematis Water Tank System

3.3.2.3. Strategi Training

Tujuan dari *training* adalah untuk mendapatkan performa *agent* yang dapat mengikuti persis dengan *benchmark trajectory* yang telah ditentukan, pada penelitian ini bisa dilihat pada gambar 4.8. Maka dari itu, *agent* dilatih untuk mengikuti beberapa *level* dari sinyal lurus/konstan yang acak sehingga *agent* dapat menentukan keputusan terbaik di setiap *level* yang akan ditetapkan oleh *setpoint*.



Gambar 4. 1 Benchmark Trajectory

Pada penelitian ini, *agent* diajak untuk mempelajari langkah awal dalam nilai ketinggian air yang di-set acak dengan penggunaan *syntax randi* dan *rand*. *Randi* (3) menghasilkan *range* nilai acak berkisar antara 1 hingga 3 dan *syntax rand* menghasilkan nilai acak antara 0 hingga 1. Kombinasi satuan-satuan *syntax* ini akan menghasilkan nilai ketinggian level secara acak tanpa perlu diubah secara manual oleh *user*. Pada *script* ini juga dilakukan penambahan kondisi pemakaian disaat $h \leq 0 \text{ || } h \geq 6$. Hal ini terkait dengan ketinggian tangki sebesar 6 dm, sehingga keadaan *real* dapat diaproksimasi. *Benchmark trajectory* pada *training* yang dilakukan oleh *agent* berkisar antara 1 dm hingga 5 dm.

```
function in = localResetFcn_modeldm(in)
% Randomize reference signal
blk = sprintf('rlwatertank_simulation_model6_check1/Desired \nWater
Level');
h = randi(3) + rand + 1;
```

```

while h <= 0 || h >= 6
    h = randi(3) + rand + 1;
end
in = setBlockParameter(in,blk,'Value',num2str(h));

% Randomize initial height
h = randi(3) + rand + 1;
while h <= 0 || h >= 6
    h = randi(3) + rand + 1;
end

blk = 'rlwatertank_simulation_model6_check1/Water-Tank System/H';
in = setBlockParameter(in,blk,'InitialCondition',num2str(h));
end

```

Matlab memiliki Toolbox pada RL yang menyediakan *reset function* sehingga dapat mengimplementasikan strategi *training* yang telah dibuat.

```
env.ResetFcn = @(in)localResetFcn_modeldm(in);
```

3.3.2.4. Desain *Observation*

Dalam rangka membentuk *action* pada RL, diperlukan pendefinisian *observation* dengan membatasinya melalui *syntax* dari pemrograman. Proses pelatihan *agent* yang diperlukan adalah batas *action* yang dilakukan, respon yang dihasilkan setelah *agent* memberi keputusan dan sistem yang dikendalikan. Informasi-informasi ini meliputi, *integrated error*, *error*, dan ketinggian yang diinginkan. *Error* didapatkan dari selisih antara ketinggian yang diinginkan dan ketinggian yang diukur pada tangki. Nilai error akan terus diproses oleh *agent* untuk diperkecil hingga *agent* mampu mengarahkan *process variable* pada ketinggian yang diinginkan.

Pada tahap ini juga, *agent* dibatasi pilihan-pilihan *action* yang akan dimobil dalam jangka tertentu, sehingga *agent* akan sesuai dengan rancangan bangun. *Action* yang akan diambil adalah % bukaan *control valve* untuk mengendalikan ketinggian

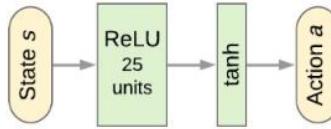
air pada tangki. Batasan *action* yang diberikan dimulai dari 0 hingga 100 dengan besaran persentase/%. Dapat dilihat pada *syntax* dibawah

```
%Create Observation (State)
obsInfo = rlNumericSpec([3 1], 'LowerLimit', [-inf -inf 0 ], 'UpperLimit', [
inf inf inf]);
obsInfo.Name = 'observations';
obsInfo.Description = 'integrated error, error, and measured height';
numObservations = obsInfo.Dimension(1);

actInfo = rlNumericSpec([1 1], 'LowerLimit', 0, 'UpperLimit', 100);
actInfo.Name = 'CV';
numActions = actInfo.Dimension(1);
```

3.3.2.5. Desain *Environment*

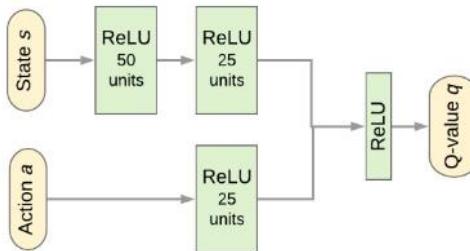
Pada penelitian ini, *agent* yang digunakan adalah DDPG dimana kerangka dasar dari algoritmanya menggunakan metode *actor critic*. Metode ini yang menentukan rasio *learning* dari proses pembelajaran RL DDPG.



Gambar 3. 29 Policy Network (Actor)

(Siraskar, 2021)

Actor network terdiri atas 2 *hidden layer*, yaitu 25 *fully connected* dan *tanh* layer dan *critic network* juga memiliki 2 *hidden layer*, yaitu 50 *fully connected* dan



Gambar 3. 30 Critic Network (Action-Value)

(Siraskar, 2021)

20 *fully connected*. *Learning rate* pada actor critic harus diatur seimbang, jika salah satu diatur lebih cepat dibandingkan yang lain, maka *training* akan lebih mirip dengan metode Q-learning (*critic centric view*) atau *policy gradient* (*actor centric view*) (Guillaume Matheron, Nicolas Perrin, 2020). Hal ini juga berpengaruh pada eksplorasi dan eksloitasi *state* dimana jika terlalu rendah maka *learning* akan berjalan lebih lama sedangkan jika terlalu besar maka *learning* akan terjebak pada *local optima*. Script mengenai penerapan metode *actor critic* pada penelitian ini dapat dilihat dibawah ini. Pada penelitian ini, *learning rate* pada *actor* bernilai 1×10^{-4} dan pada *critic* bernilai 1×10^{-3} .

```

statePath = [
    featureInputLayer(numObservations, 'Normalization', 'none', 'Name', 'State')
        fullyConnectedLayer(50, 'Name', 'CriticStateFC1')
        reluLayer('Name', 'CriticRelu1')
        fullyConnectedLayer(25, 'Name', 'CriticStateFC2')];
actionPath = [
    featureInputLayer(numActions, 'Normalization', 'none', 'Name', 'Action')
        fullyConnectedLayer(25, 'Name', 'CriticActionFC1')];
commonPath = [
    additionLayer(2, 'Name', 'add')
    reluLayer('Name', 'CriticCommonRelu')
    fullyConnectedLayer(1, 'Name', 'CriticOutput')];

criticNetwork = layerGraph();
criticNetwork = addLayers(criticNetwork, statePath);
criticNetwork = addLayers(criticNetwork, actionPath);
criticNetwork = addLayers(criticNetwork, commonPath);
criticNetwork = connectLayers(criticNetwork, 'CriticStateFC2', 'add/in1');
criticNetwork =
connectLayers(criticNetwork, 'CriticActionFC1', 'add/in2');

%Create Critic Representation
criticOpts = rlRepresentationOptions('LearnRate', 1e-
03, 'GradientThreshold', 1);
critic =
rlQValueRepresentation(criticNetwork, obsInfo, actInfo, 'Observation', {'Sta-
te'}, 'Action', {'Action'}, criticOpts);

```

```

actorNetwork = [
    featureInputLayer(numObservations, 'Normalization', 'none', 'Name', 'State')
        fullyConnectedLayer(3, 'Name', 'actorFC')
        tanhLayer('Name', 'actorTanh')
        fullyConnectedLayer(numActions, 'Name', 'Action')
    ];
]

%Create Actor Representation
actorOptions = rlRepresentationOptions('LearnRate', 1e-
04, 'GradientThreshold', 1);

actor =
rlDeterministicActorRepresentation(actorNetwork, obsInfo, actInfo, 'Observa-
tion', {'State'}, 'Action', {'Action'}, actorOptions);

```

3.3.2.6. Parameter Training

Pada proses ini, peneliti menentukan parameter-parameter dari algoritma DDPG dalam melakukan *training* pada agent. Parameter-parameter menjadi sangat penting karena menentukan bagaimana proses *training* akan dijalankan. Dalam proses *training*, *agent* diberikan waktu sampel sebesar 1 sekon yang bermakna *agent* akan mengambil *action* dalam waktu 1 sekon. Setiap 1 *episode*, peneliti memberikan *agent* sebanyak 200 step untuk melakukan proses pengendalian ketinggian air. Sehingga, dibutuhkan waktu sekitar 200 sekon untuk setiap episode untuk *agent* melakukan *learning* dengan target (*measured value*) yang terus berubah-ubah.

Lalu, *action* yang diambil *agent* akan berpengaruh terhadap akumulasi *reward* yang dihasilkan oleh *agent*. Pada gambar 3.22, produk yang dihasilkan oleh *continuous reward* dan *discrete reward* akan berbeda. Pada *discrete reward*, jika *agent* diasumsikan mampu melakukan *action* secara lengkap maka *maximum reward* yang didapatkan akan bernilai 21. Sedangkan pada *continuous reward*, *maximum reward* yang didapatkan adalah 5. Akumulasi dari *discrete* dan *continuous reward* akan menghasilkan nilai *reward* maksimum sebesar 26 dalam 1

step dan dalam 1-episode menghasilkan *reward* maksimum sebesar 5200. Sehingga, peneliti mengambil nilai 4800 sebagai target kebijakan optimum yang dihasilkan oleh *agent* sebagai indikator keberhasilan *training*.

Tahap selanjutnya, peneliti melakukan penentuan parameter *agent options* dan *training options*. Peneliti melakukan pemilihan pemilihan parameter *training* dan *agent options*, meliputi maksimum episode, maksimum *step per episode*, *score averaging window length*, *stop training criteria*, dan *stop training value*. Peneliti memilih maksimum episode sebesar 20000 episode, sehingga training akan berhenti jika *policy* optimum belum ditemukan jika *stop training value* belum dicapai. Peneliti juga memilih parameter *average reward* sebagai target/indikator keberhasilan dari training dengan nilai *average reward* sebesar 4800 dari 64 data sebelumnya. Peneliti juga melakukan pengaturan *noise option* pada *agent option* dalam rangka menyeimbangkan proses eksplorasi dan eksloitasi. Penerapan parameter-parameter ini menerapkan penelitian menggunakan *agent DDPG* oleh (Siraskar, 2021).

```
%DDPG Agent Options
agentOpts = rlDDPGAgentOptions(...%
    'SampleTime',Ts,...%
    'TargetSmoothFactor',1e-3,...%
    'DiscountFactor',0.9,...%
    'MiniBatchSize',64,...%
    'ExperienceBufferLength',1e6,...%
    'ResetExperienceBufferBeforeTraining',false,...%
    'SaveExperienceBufferWithAgent',true);

agentOpts.NoiseOptions.Variance = 0.3;
agentOpts.NoiseOptions.VarianceDecayRate = 1e-5;

agent = rlDDPGAgent(actor,critic,agentOpts);

maxepisodes = 20000;
maxsteps = ceil(Tf/Ts);
```

Parameter Agent options	Setting
<i>Sample Time</i>	1 s
<i>Target Smooth Factor</i>	1×10^{-3}
<i>Discount Factor</i>	0.9
<i>Mini Batch Size</i>	64
<i>Experience Buffer Length</i>	1×10^6
<i>Reset Experience Buffer Before Training</i>	<i>false</i>
<i>Save Experience Buffer With Agent</i>	<i>true</i>
<i>OUP Variance</i>	0.3
<i>OUP Variance Decay Rate</i>	1×10^{-5}

Tabel 3. 3 Konfigurasi Parameter *Agent Options*

```
%DDPG Training Options
trainOpts = rlTrainingOptions(...
    'MaxEpisodes',maxepisodes,...
    'MaxStepsPerEpisode',maxsteps,...
    'ScoreAveragingWindowLength',64,...
    'Verbose',false,...
    'Plots','training-progress',...
    'StopTrainingCriteria','AverageReward',...
    'StopTrainingValue',4800);
```

Parameter Training options	Setting
<i>Maximum Episode</i>	20000
<i>Maximum Step Per Episode</i>	200
<i>Score Averaging Window Length</i>	20
<i>Verbose</i>	<i>false</i>
<i>Plots</i>	<i>training-progress</i>
<i>Stop Training Criteria</i>	<i>average reward</i>
<i>Stop Training Value</i>	4800

Tabel 3. 4 Konfigurasi Parameter *Training Options*

BAB 4

HASIL DAN PEMBAHASAN

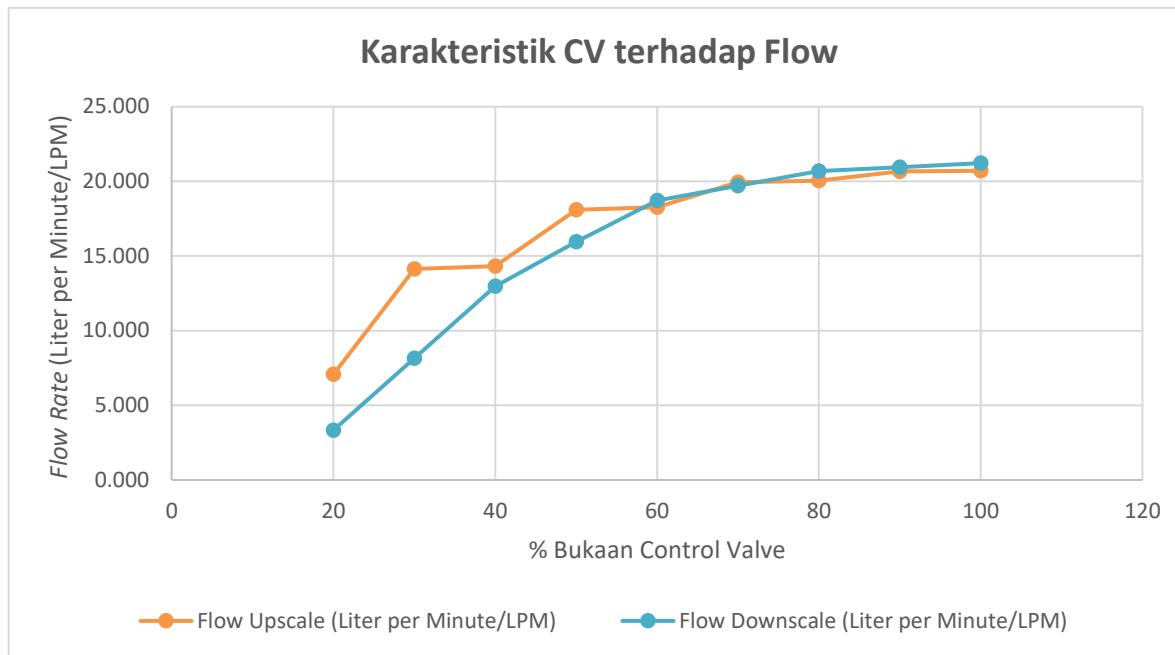
4.1. Karakteristik *Control Valve*

Control valve digunakan untuk mengatur debit air yang masuk pada tangki penampungan dengan mengatur *flow in* yang melewati *valve*-nya. Karakteristik *control valve* perlu diketahui untuk mendapatkan gambaran bagaimana proses pengendalian ketinggian dapat dilaksanakan. Hal tersebut dilakukan membentuk sistem *open loop* untuk mencari relasi antara *flow rate* melalui *flow meter* dan % bukaan *control valve*. Percobaan dilakukan dengan memvariasikan % bukaan control dengan jangkauan 20%-100% dimana *flow rate* bertindak sebagai variabel terikat dan bukaan *control valve* bertindak sebagai variabel bebas. Percobaan dilakukan dengan 2 cara, yaitu membuka % bukaan *control valve* secara berurutan dari kecil hingga besar (*upscale*) dan besar hingga kecil (*down scale*) yang bisa dilihat pada Table 4.1.

Dari grafik pada gambar 4.1, didapatkan kesimpulan bahwa *control valve* memiliki karakteristik non-linear yang dibuktikan dengan pengukuran besaran *flow rate* terhadap % bukaan *control valve*. Pengukuran tersebut juga dapat diinterpretasikan melalui perbedaan *trend* pada pengukuran *upscale* maupun *downscale* yang diakibatkan oleh perbedaan nilai yang di ukur melalui *Manipulated Variable* (*control valve*).

Bukaan CV(%)	Flow Upscale (Liter per Minute/LPM)	Flow Downscale (Liter per Minute/LPM)
20	7,071	3,338
30	14,121	8,159
40	14,332	12,970
50	18,105	15,953
60	18,261	18,728
70	19,929	19,696
80	20,056	20,687
90	20,659	20,956
100	20,705	21,220

Tabel 4. 1 Korelasi antara bukaan *control valve* dengan *flow rate* secara *upscale* dan *downscale*



Gambar 4. 2 Grafik hubungan antara *flow rate* dan % bukaan *control valve*

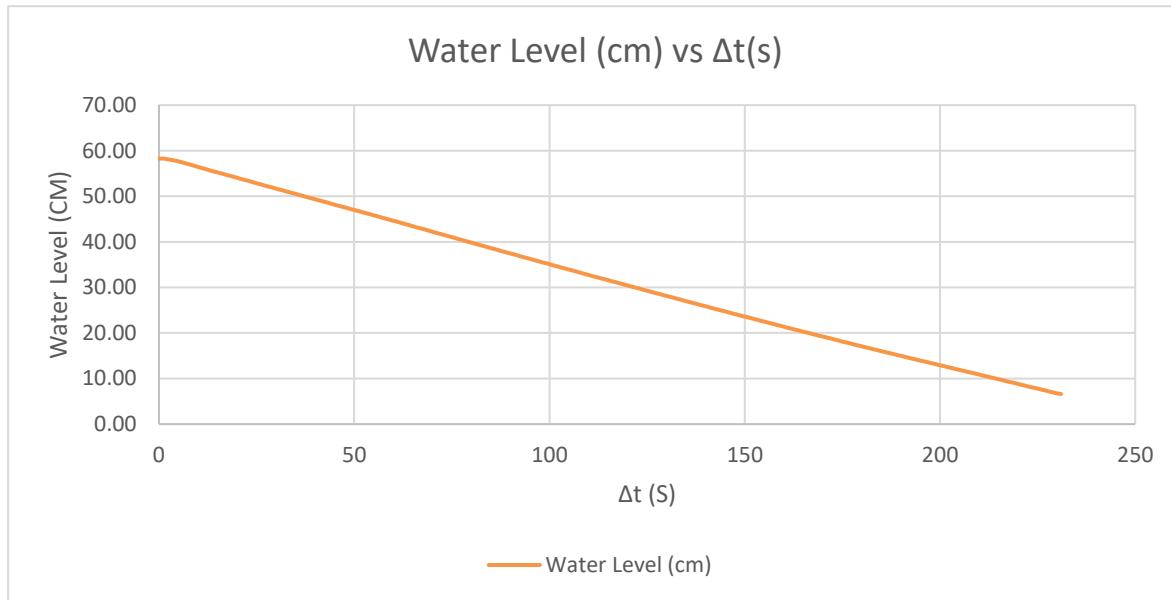
4.2. Pengukuran konstanta *flow out* (q_o)

Pada rancang bangun sistem kendali ini, peneliti menerapkan *water tank system* yang disebut dengan *liquid-level process with constant outlet* (Steven & R.Coughanowr, 2009). Pada dasarnya, sistem ini memiliki kategori spesifik yaitu debit air keluar/*flow out* yang konstan. Hal ini diakibatkan oleh penerapan aktuator tambahan (pompa) yang berguna dalam pentransmisian aliran air yang akan dikembalikan ke tangki utama. Pengukuran debit air yang keluar dilakukan dengan mengosongkan tangki campuran dari *level* maksimal hingga *level* minimal tanpa adanya aliran air yang masuk/*flow in* kedalam tangki. Tujuan utama dari percobaan ini adalah untuk mengukur waktu yang dibutuhkan oleh pompa *flow out* dalam mengosongkan air pada tangki. Hal ini sesuai dengan persamaan debit air

$$Q = \frac{V}{t} \quad (4.1)$$

Dimana Q adalah debit air dalam m^3/s , V adalah *volume* tangki dalam m^3 , sedangkan t adalah waktu dalam sekon yang dibutuhkan untuk aliran air yang keluar

Awalnya, tangki diisi hingga mencapai level maksimumnya dengan kondisi *flow out* yang ditutup. Ketika ketinggian air telah mencapai target yang diinginkan, maka proses pengosongan air dimulai dengan *flow in* ditutup yang disertai dengan akuisisi data menggunakan matlab ke *workspace*. Data ketinggian terhadap waktu kemudian divisualisasikan dalam bentuk grafik pada gambar 4.2.



Gambar 4. 3 Grafik Water Level (cm) terhadap Δt (s)

Sebelum melakukan penghitungan laju aliran pada *outlet*, peneliti melakukan penghitungan *volume* dari tangki penampungan yang berbentuk tabung dengan luas alas dari tangki 0.07065 m^2 dengan diameter 30 cm. Dari informasi tersebut, maka kita dapat menyimbulkan bahwa volume dari tangki berbentuk tabung adalah 0.0364 m^3 . Setelah itu, peneliti melakukan simulasi hingga tangki air kosong. Sehingga didapatkanlah jenjang waktu Δt ketika *level max* hingga *level min* yang bernilai 231 sekon.

$$Q = \frac{0.0364 \text{ m}^3}{231 \text{ s}} = 1,57 \times 10^{-4} \text{ m}^3/\text{s} \quad (4.2)$$

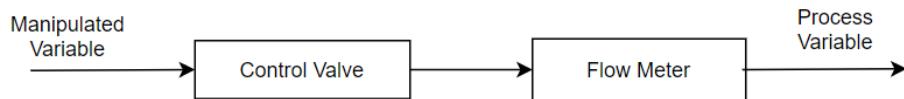
Nilai tersebut dapat dikonversi dalam bentuk standar *flow rate* yang umumnya digunakan, yaitu 9.42 Liter/Minute (**L/m**). 

4.3. Identifikasi Plant

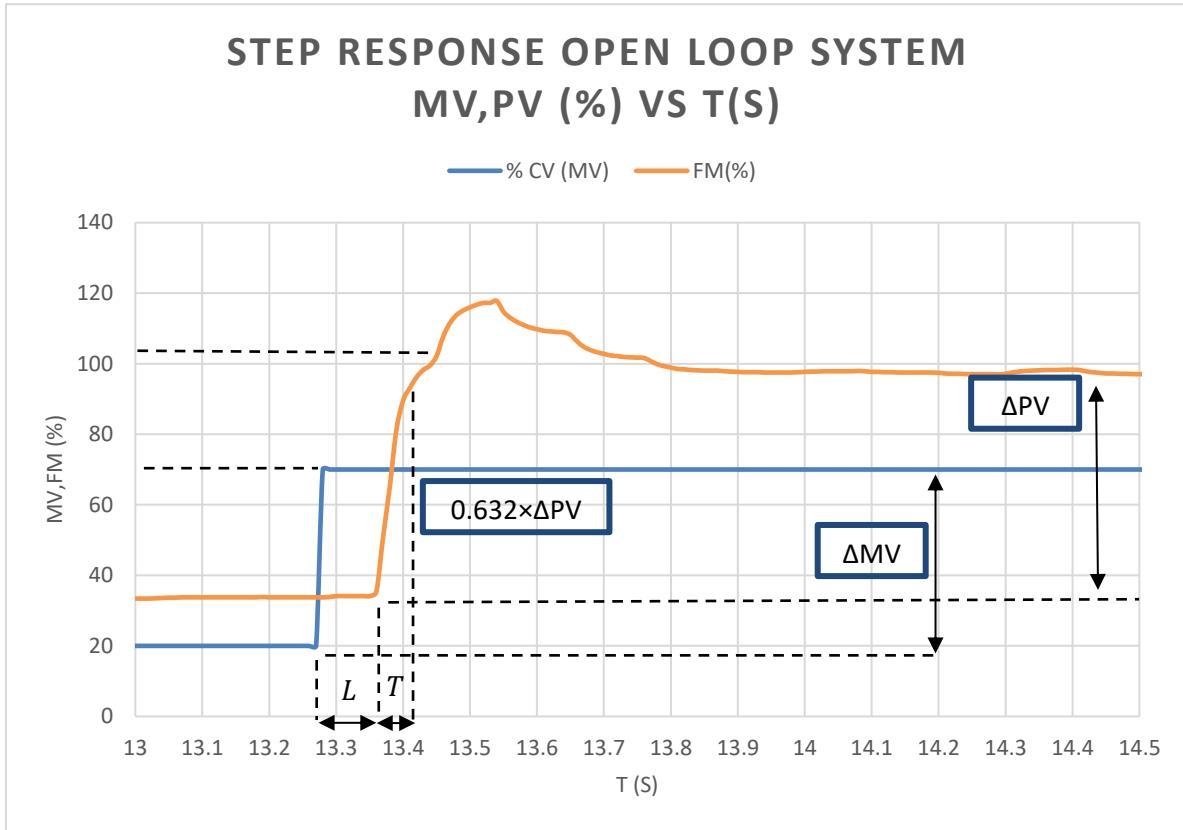
Identifikasi Plant dilakukan dengan mencari fungsi transfer yang akan diaplikasikan pada blok *water tank system*. Fungsi transfer digunakan untuk membentuk *environment* agar dapat menjalankan *training* pada RL. Fungsi transfer menrepresentasikan sistem yang terdapat pada *real plant*, sehingga identifikasi harus dibuat semirip mungkin. Relasi yang dibuat pada fungsi transfer merupakan hubungan antara *flow in* dan % bukaan *control valve*. Fungsi transfer mengikuti prinsip pemodelan *First-Order Plus Time Delay* (FOPTD).

$$G(s) = \frac{k}{(1 + Ts)} e^{-Ls} \quad (4.3)$$

Dimana k merupakan *gain* dari sistem, T merupakan *time constant*, dan L adalah *time delay*. Percobaan dilakukan dengan membuat fungsi transfer menggunakan sistem *open loop* dimana % bukaan *control valve* merupakan *manipulated variable* (MV) sedangkan nilai *flow rate* sebagai *process variable* (PV). Pengambilan data dimulai pada % bukaan control valve sebesar 20% **hingga** 70% menggunakan blok pada Simulink pada gambar 3.10. 



Gambar 4. 4 Diagram *Open-Loop*



Gambar 4. 5 Step Response Open Loop % CV, % FM terhadap waktu (s)

Berdasarkan percobaan sistem *open loop*, maka didapatkan hasil dari variabel-variabel yang dibutuhkan untuk menyusun fungsi transfer dari *plant*.

Identifikasi Plant	
$L(s)$	0.09
PV Max (%)	21.0828
PV Min (%)	7.3978
ΔPV	13.6850
ΔMV	50
k	0.2737
$T(s)$	0.1

Tabel 4. 2 Variabel Fungsi Transfer Plant

- *Delay Time /L(s)*

Time delay merupakan waktu yang dibutuhkan sistem untuk berubah pada nilai input ke nilai output. Pada percobaan *open loop* ini, waktu yang dibutuhkan untuk sistem berubah, yaitu disaat waktu 13.28s hingga 13.37s sehingga didapatkan selisih dari nilai tersebut adalah 0.09s.

- Gain/k

Gain adalah perubahan pada output *PV* yang diakibatkan oleh perubahan input *MV*. Pada kasus ini, ΔPV bernilai 13.6850 dan ΔMV bernilai 50 dan dilakukan perbandingan. Hasilnya, didapatkanlah nilai *k* sebesar 0.2737.

- *Time constant/T(s)*

Time Constant adalah waktu yang dibutuhkan untuk *Process Variable* (*PV*) merespon perubahan dari *Manipulated Variable* (*MV*). Secara spesifik, dapat direpresentasikan dengan waktu yang dibutuhkan *PV* untuk mencapai 63.2% dari nilai akhir. Sehingga, didapatkanlah nilai 0.1s dari multiplikasi antara 0.632 dan ΔPV sebesar 13.6850.

Sehingga didapatkan persamaan fungsi transfer dari *plant* dengan variabel-variabel pada tabel 4.2.

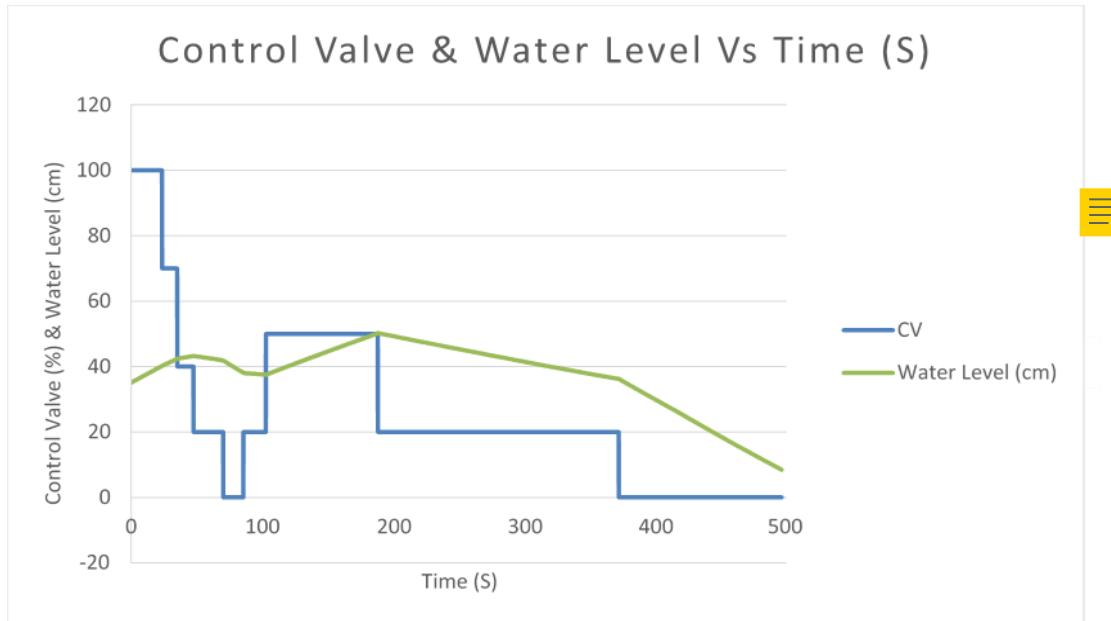
$$G(s) = \frac{0.2737}{(1 + 0.1s)} e^{-0.09s} \quad (4.4)$$

Fungsi transfer ini kemudian diaplikasikan pada blok *water tank system* pada gambar 3.27.

4.4. Simulasi *Open Loop*

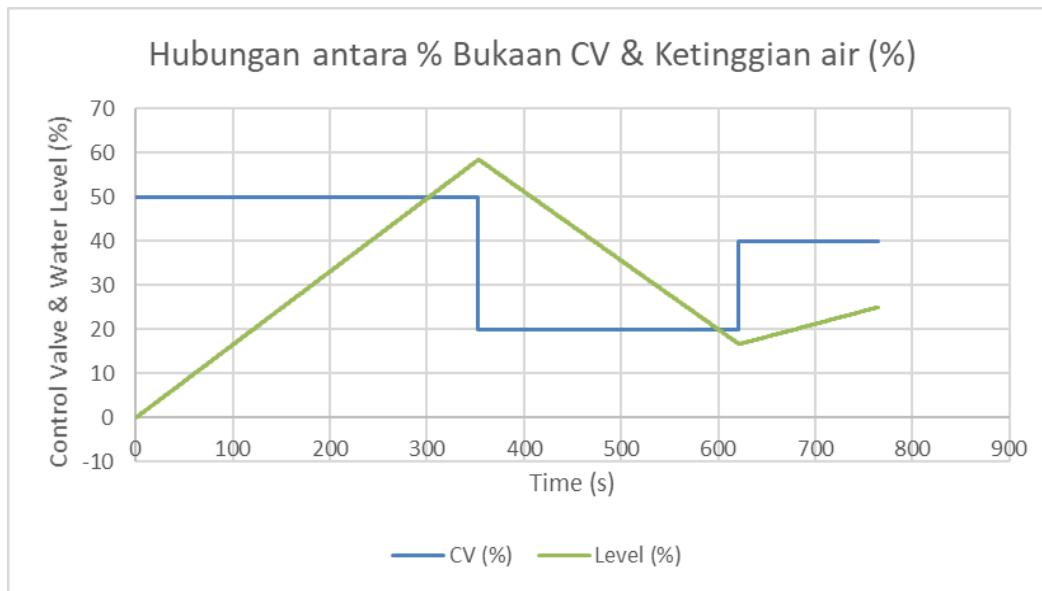
4.4.1. Simulasi *Real Plant*

Hubungan antara % bukaan *control valve* dan ketinggian air yang dihasilkan dapat dilihat melalui gambar 4.5. Hal ini dilakukan untuk memastikan output yang diberikan oleh *control valve* pada % bukaan *control valve* sesuai dengan simulasi yang telah dilakukan secara matematis seperti pada gambar 4.6.



Gambar 4. 6 Grafik Hubungan antara % bukaan *control valve* dengan ketinggian air (cm)

4.4.2. Simulasi MATLAB

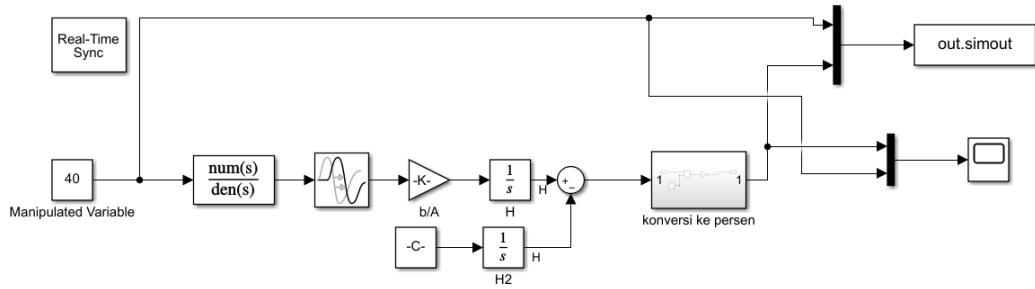


Gambar 4. 7 Grafik Hubungan antara % Bukaan CV & Ketinggian air (%)

Simulasi *open loop* ini dilakukan untuk memastikan persamaan fungsi transfer sudah sesuai dengan kejadian secara *real* atau tidak. Dengan melihat gambar grafik 4.6,

peneliti dapat melihat relevansi antara % bukaan *control valve* dan ketinggian air. Ketika % *control valve* diubah dari 50 ke 20, ketinggian air berkurang secara terus menerus hingga bukaan diubah. Pada gambar 4.5 dan 4.6, dapat dilihat persamaan *trend* ketika bukaan *control valve* diubah dari 50% menjadi 20%.

Peneliti menggunakan blok Real time sync untuk mengatur *time sampling* pada sistem, sehingga simulasi dapat berjalan selayaknya dalam keadaan *real*.



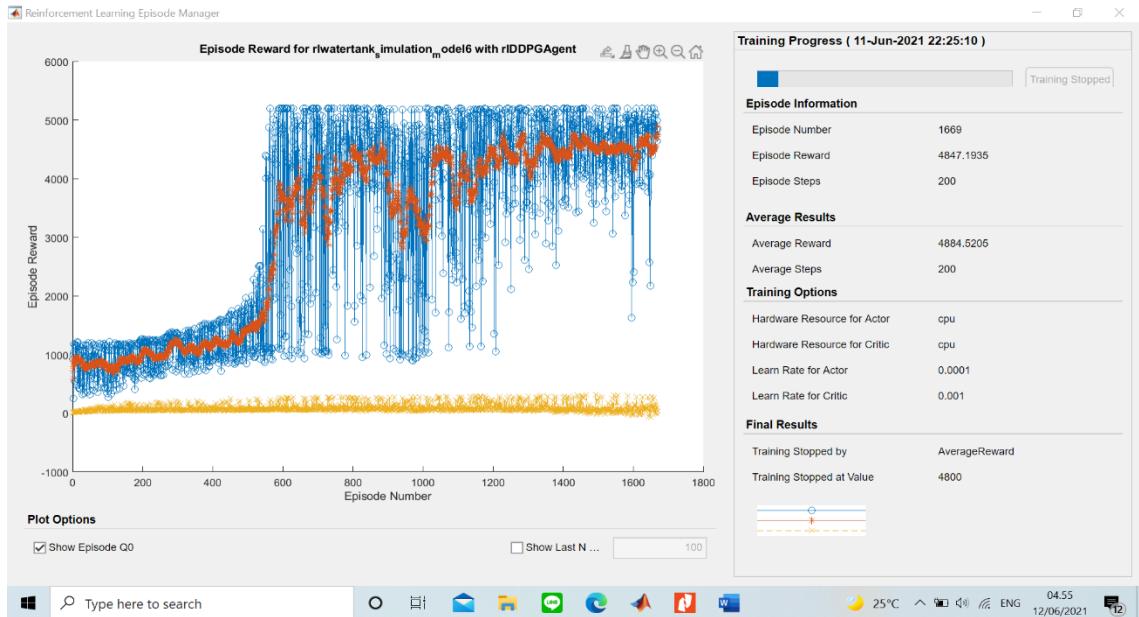
Gambar 4. 8 Blok sistem simulasi dengan % bukaan *control valve* sebagai input dan ketinggian Air sebagai output

4.5. Training Algoritma DDPG

4.5.1. Proses *Training*

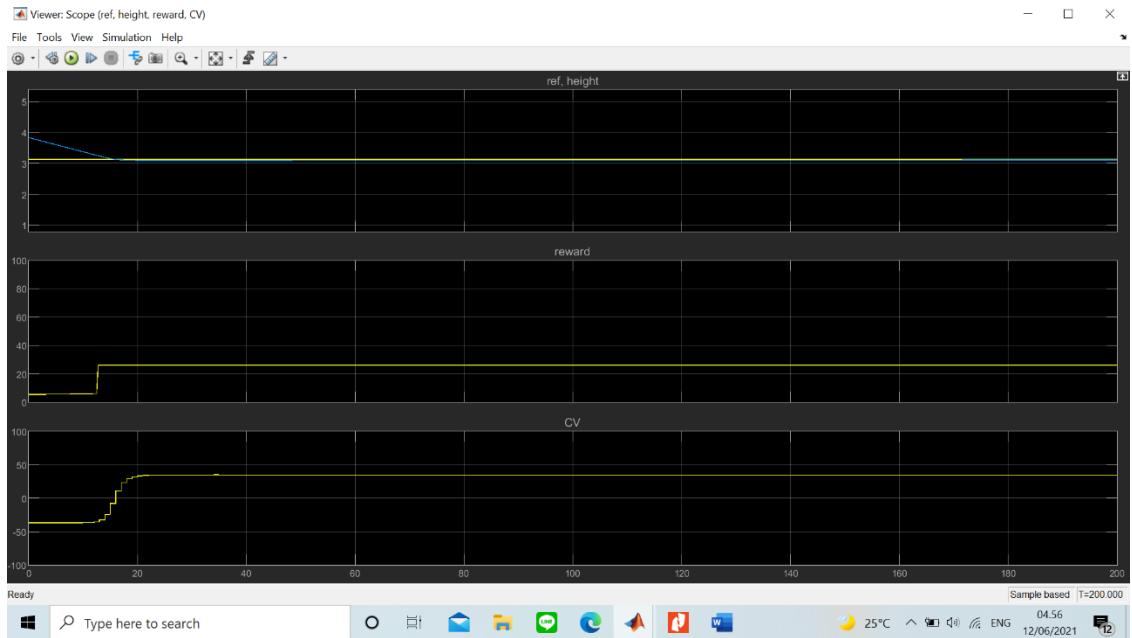
Proses *training* yang dilakukan peneliti terdiri atas beberapa tahapan dengan beberapa percobaan *trial and error*. Proses ini melibatkan variasi *reward* yang telah diujicoba dengan berbagai *training*. Dan secara khusus, *agent* DDPG diberikan tugas untuk melakukan kontrol bukaan *control valve* yang diketahui memiliki *action space* yang bersifat kontinyu (Mathworks, 2020). Proses *training* secara spesifik dilakukan melalui pengamatan *average reward* serta *trend stabilitas process variable* pada *policy* dengan mengubah sistem *reward* di dalam blok *calculate reward* pada gambar 3.22.

Pada tahap awal proses *training*, peneliti melakukan *training* dengan target *average reward* maksimum reward sebesar 4800. Dan dapat dilihat melalui hasil *training* dibawah ini.



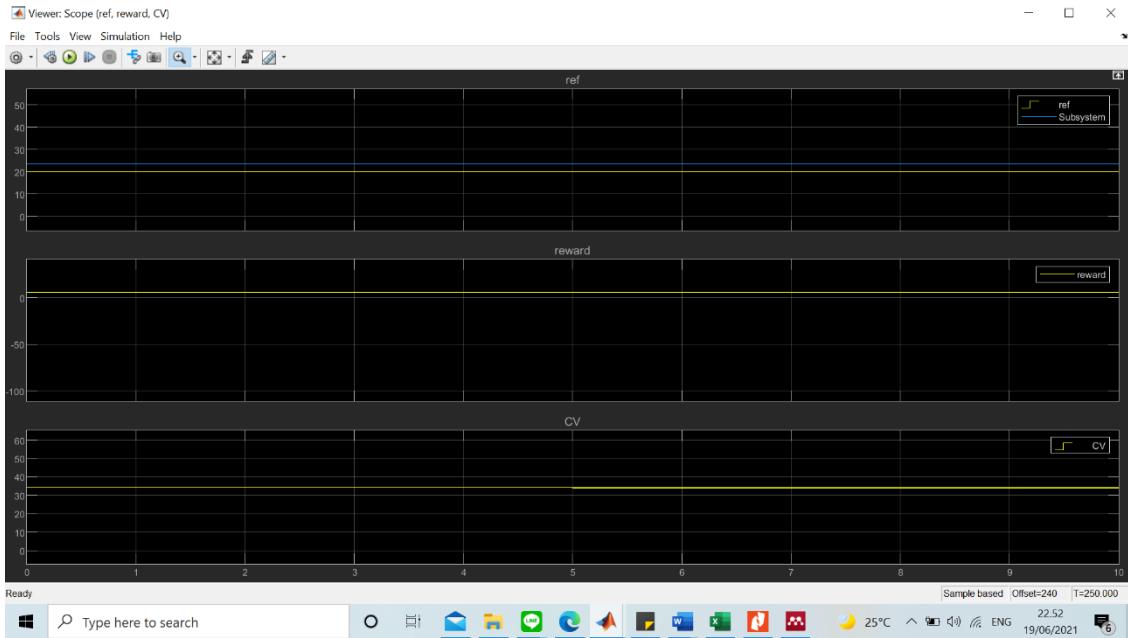
Gambar 4. 10 RL Episode manager dengan *average reward maximum* sebesar 4800

Agent berhasil mencapai target *average reward* yang diinginkan dengan *training* yang cenderung bergerak naik. Jika dilihat dengan seksama, performa *training agent* di setiap kenaikan *episode* mengalami perbaikan yang cukup signifikan. Namun, pada *episode* ke 900, *agent* mengalami penurunan *reward* akibat proses eksplorasi yang



Gambar 4. 9 Display proses training oleh agent dengan *average reward maximum* sebesar 4800

dilakukan dan ditandai dengan banyaknya kesalahan yang dilakukan hingga *episode* 1100. Hasil dari *training* ini dapat dilihat pada Gambar 4.9 yang menunjukkan 3 buah parameter, yaitu Set Point dan *process variable* pada *display* 1, *reward* yang didapatkan tiap step pada *display* 2, dan % bukaan *control valve* pada *display* 3. Pada *display* 1, dapat dilihat hasil yang cukup memuaskan dengan *error* antara PV (garis berwarna biru) dan *Set Point* (garis berwarna kuning) kurang dari 0.5%. Hal ini juga dapat diindikasikan melalui jumlah *reward* yang didapatkan step, dimana *reward* bernilai lebih dari 20. Namun, % bukaan *control valve* belum dapat dimaksimalkan hingga 100%.



Gambar 4.11 Display proses training oleh *agent* dengan *average reward maximum* sebesar 4500

Pada Gambar 4.11, dapat dilihat perbedaan hasil *training* pada *training* sebelumnya dimana error yang dihasilkan oleh PV terhadap setpoint cukup tinggi yang berkisar antara 1 hingga 2 %. Pada *Display* 2 juga ditunjukkan *reward* yang dihasilkan oleh *agent* tiap step-nya dimana bernilai kurang dari 5 poin. Bukaan *control valve* yang dihasilkan juga lebih rendah dibandingkan pada gambar 4.10 dimana berkisar $\pm 35\%$ untuk bukaan maksimum. Hal ini memperlambat proses pengendalian ketinggian air dikarenakan *flow* air yang dihasilkan oleh bukaan sebesar itu tergolong kecil. Maka dari

itu, peneliti melakukan perubahan nilai *average reward* dari 4500 (gambar 4.11) menjadi 4800 (gambar 4.9). Dengan penggantian *average reward*, %bukaan control valve juga menjadi lebih besar dengan nilai 72,3% dari yang awalnya hanya berkisar antara $\pm 35\%$.

4.5.2. Optimalisasi *training*

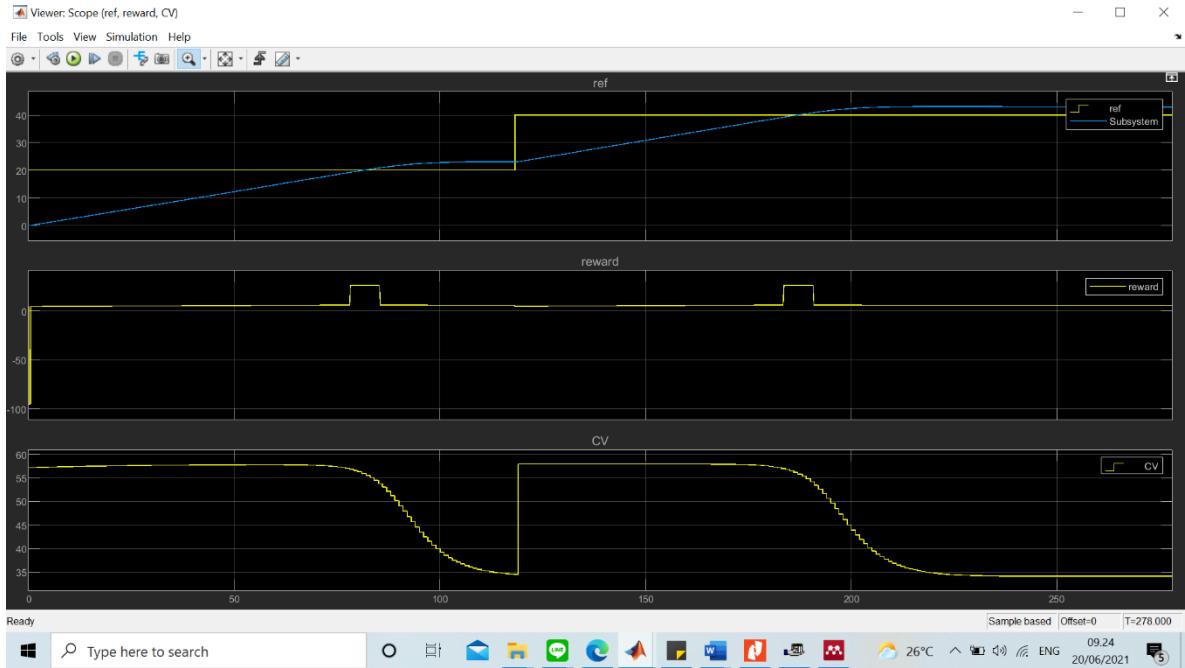
Peneliti menyadari bahwa terdapat *error* yang cukup besar pada uji coba simulasi dengan *agent* yang telah di training dengan sistem strategi *training* yang telah dijelaskan pada Subbab 3.3.2.6. Awalnya, peneliti melakukan startegi *training* dengan menerapkan *syntax* berikut

```
function in = localResetFcn_model2(in)

% randomize reference signal
blk = sprintf('rlwatertank_simulation_model6/Desired \nWater Level');
h = 1+randi(4);rand
while h <= 0 || h >= 100
h = 1+randi(4) + rand;
end
in = setBlockParameter(in,blk,'Value',num2str(h));

% randomize initial height
h = 1+randi(4) + rand;
while h <= 0 || h >= 100
h = 1+randi(4) + rand;
end
blk = 'rlwatertank_simulation_model6/Water-Tank System/H';
in = setBlockParameter(in,blk,'InitialCondition',num2str(h));
end
```

Sehingga didapatkan karakteristik respons PV terhadap *Set point* seperti berikut



Gambar 4. 12 Respon ketika menggunakan strategi *training* pada proses awal *training*

Dapat dilihat bahwa *error* yang dihasilkan melalui strategi *training* diatas cukup besar dan bisa mencapai 3% dari *set point*. Dengan permasalahan ini, peneliti kemudian mengganti susunan strategi *training* dengan memperpendek jangkauan randi menjadi 2 dimana memberikan nilai acak dari 1 hingga yang kemudian akan ditambah dengan rand serta angka bernilai 2. Sehingga, variasi angka *set point* pada saat *training* dapat bervariasi dengan lebih kecil.

```

function in = localResetFcn_modeldm(in)
% Randomize reference signal
blk = sprintf('rlwatertank_simulation_model6/Desired \nWater Level');
h = randi(2)+ rand +2;
while h <= 0 || h >= 6
    h = randi(2)+ rand +2;
end
in = setBlockParameter(in,blk,'Value',num2str(h));

% Randomize initial height
h = h = randi(2)+ rand +2;
while h <= 0 || h >= 6

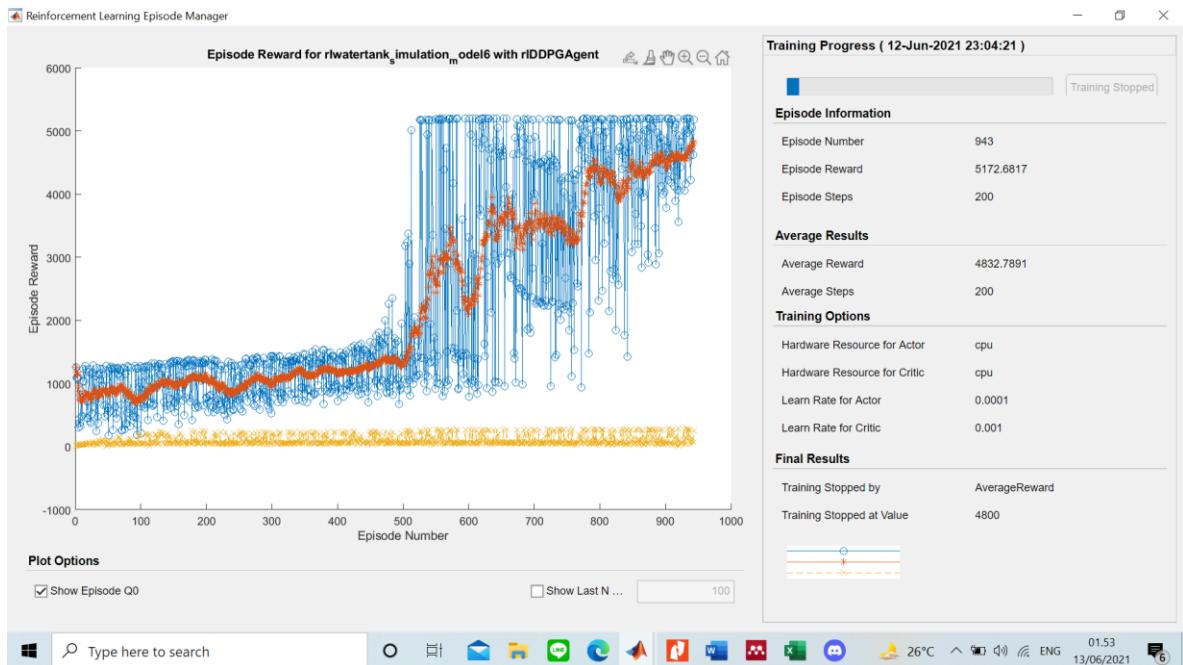
```

```

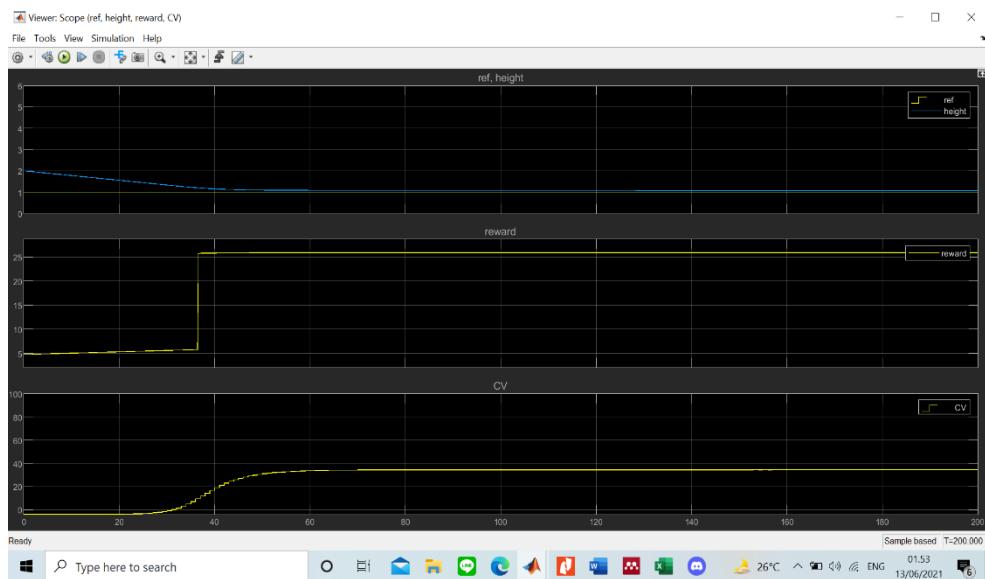
h = randi(2)+ rand +2;
end

blk = 'rlwatertank_simulation_model6/Water-Tank System/H';
in = setBlockParameter(in,blk,'InitialCondition',num2str(h));
end

```



Gambar 4. 13 Proses training ketika strategi training diganti dengan varian baru



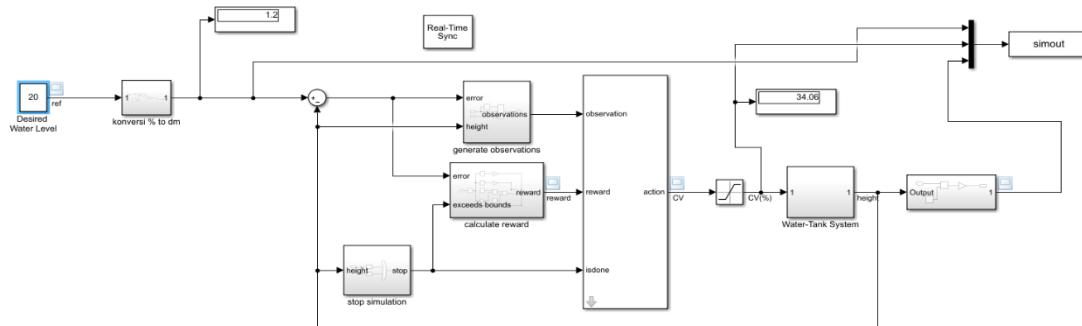
Gambar 4. 14 Respon ketika mengganti strategi training dengan varian baru

Hasilnya, respon PV terhadap *set point* menjadi semakin baik yang ditandai dengan berkurangnya *error* pada sistem dan nilai reward yang maksimal di setiap *step*-nya.

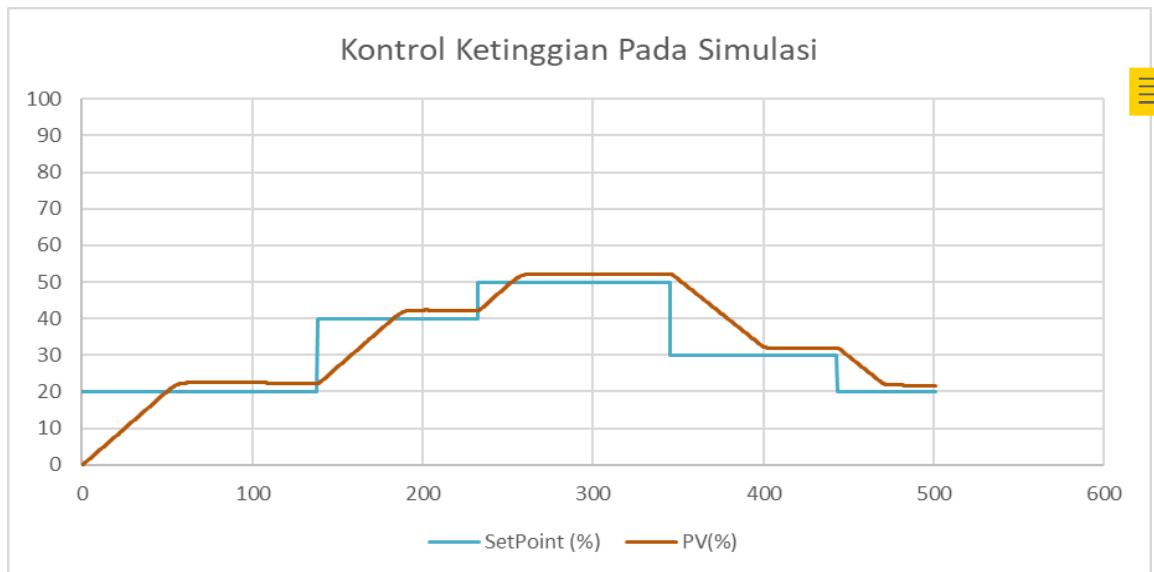
4.6. Implementasi pada *Plant*

4.6.1. Kontrol Ketinggian Pada Simulasi

Setelah tahap *training agent* selesai, maka perlu dilakukan tahap uji coba untuk mengetahui respon yang dihasilkan oleh *agent* ketika *process variable* diberikan perintah untuk meraih ketinggian tertentu ketika *set point* diinisiasi. Dalam tahap uji coba ini, diperlukan desain simulasi untuk melakukan pengambilan data, meliputi % bukaan *control valve/MV*, *Set Point*, dan PV secara *real time*. Skala input *set point* dan *output* ketinggian dibuat dalam format persentase 0-100% untuk memudahkan pengamatan terkait pengendalian ketinggian air.



Gambar 4.15 Desain *Simulink* Untuk Uji Coba Simulasi



Gambar 4. 16 Gambar Grafik Proses Pengendalian Ketinggian Air Berbasis Simulasi

Uji coba ini dilakukan dengan memberikan *input step up* dan *step down* dengan urutan 20-40%, 40-50%, 50-30%, dan 30-20%. Dari gambar grafik diatas, didapatkan parameter-parameter performa dari proses pengendalian yang telah dilakukan.

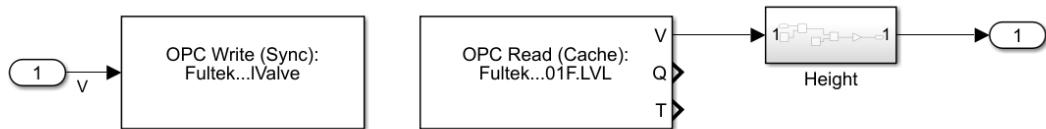
Set Point	Rise Time (s)	Settling Time (s)	Overshoot (%)	Steady-state error (%)	Rata-rata PV (%)
40	44.6	61.7	2.3	2.2	42.2
50	20.1	31.7	2.2	2.1	52.1
30	46	74.2	0	1.9	31.9
20	22.2	41.2	0	1.7	21.7

Tabel 4. 3 Performa Kendali Pada Simulasi

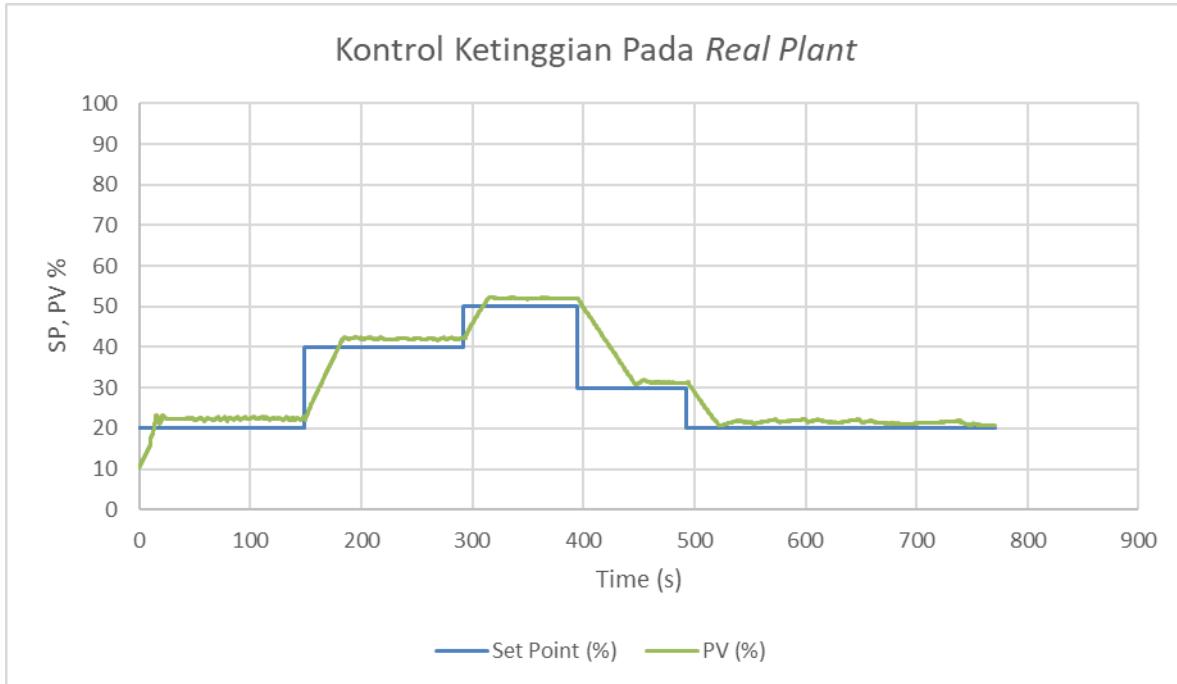
Dalam tahap uji coba simulasi, *error* yang dihasilkan oleh *process variable* dapat dikatakan stabil karena berkisar kurang dari 5% dari *set point* (Tay et al., 1997).

4.6.2. Kontrol Ketinggian Pada Real Plant

Pada tahap uji coba *real plant*, desain Simulink didesain kembali untuk diterapkan pada rancang bangun sebenarnya. Hal tersebut dilakukan dengan memodifikasi *blok water tank system* menjadi format komunikasi OPC server dimana input dikoneksikan dengan blok OPC write yang akan mengirimkan sinyal % bukaan *control valve* dan output dikoneksikan dengan pembacaan OPC Read yang terhubung dengan *level transmitter* pada tangki.



Gambar 4. 17 Blok Water Tank System Pada Uji Coba *Real Plant*



Gambar 4. 18 Gambar Grafik Proses Pengendalian Ketinggian Air Berbasis *Real Plant*

Uji coba ini dilakukan dengan memberikan *input step up* dan *step down* dengan urutan 20-40%, 40-50%, 50-30%, dan 30-20%. Dari gambar grafik diatas, didapatkan parameter-parameter performa dari proses pengendalian yang telah dilakukan.

Set Point (%)	Rise Time (s)	Settling Time (s)	Overshoot (%)	Steady-state error (%)	PV Rata-rata (%)
40	27.7	55.1	2.4	2.1	42.1
50	15.1	24.6	2.3	2	52
30	39.5	56.1	0	1.3	31.3
20	20.9	26.8	0	1.5	21.5

Tabel 4. 4 Performa kendali pada *real plant*

Dalam tahap uji coba *real plant*, *error* yang dihasilkan oleh *process variable* dapat dikatakan stabil karena berkisar kurang dari 5% dari *set point* (Tay et al., 1997).

BAB 5

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan penelitian yang telah dilakukan serta dipaparkan pada bab-bab sebelumnya, kesimpulan yang dapat diambil, yaitu:

1. Telah berhasil merancang sistem kendali ketinggian air pada *plant* dengan menenrapkan algoritma *Deep Deterministic Policy Gradient* sebagai *controller*. Hal tersebut terlihat dari setiap perpindahan *set point*, nilai *process variabel* akan mengikuti. Contohnya pada *set point* 30% dengan PV rata-rata bernilai 31,3%.
2. Pengujian Algoritma RL *Deep Deterministic Policy Gradient* sebagai *controller* ketinggian air secara teoritis atau simulasi, didapatkan *rise time* sebesar 20,1 – 44,6 sekon, *settling time* sebesar 31,7 – 61,7 sekon, *Overshoot* sebesar 0 – 2,3 %, dan *steady-state error* sebesar 1,7 – 2,7 %.
3. Pengujian Algoritma *Deep Deterministic Policy Gradient* sebagai *controller* ketinggian air pada *real plant*, didapatkan *rise time* sebesar 15,1 – 39,5 sekon, *settling time* sebesar 24,6 – 56,1 sekon, *Overshoot* sebesar 0 – 2,4 %, dan *steady-state error* sebesar 1,3 – 2,1 %.
4. Hasil penelitian menunjukkan performa *agent* yang cukup baik dengan *overshoot* yang kecil hingga bernilai 0% pada beberapa *set point*, serta *steady-state error* yang dibawah 2,5 %.

5.2. Saran

Dalam penelitian rancang bangun sisstem kendali ketinggian air yang telah peneliti lakukan, ada beberapa hal yang menurut peneliti dapat menjadi pertimbangan untuk dilakukan pada penelitian selanjutnya. Berikut beberapa hal yang ingin penulis sarankan:

1. Penerapan Laptop atau PC dengan spesifikasi *Dual Graphic Cards* sebagai unit pemograman sehingga memungkinkan pelaksanaan *training agent* dengan lebih cepat serta parallel.

2. Menerapkan sistem *Multiple-Input Multiple-Output* (MIMO) dengan menambah input masukan berupa sensor dan menambah variabel output yang lain.
3. Menggunakan sistem pemograman *Reinforcement Learning* yang berbeda, seperti *phyton*.

DAFTAR PUSTAKA

- Abhishek, N., & Biswas, M. (2012). Reinforcement learning with Open AI, TensorFlow and Keras Using Python. In *Learning* (Vol. 3, Issue 9). Apress.
<http://incompleteideas.net/sutton/book/the-book.html%5Cnhttps://www.dropbox.com/s/f4tnuhipchpkgoj/book2012.pdf>
- Ahmad, B. (2019). *Rancang bangun sistem debit air berbasis PLC dengan menerpakan NN Controller.*
- Beysolow II, T. (2019). Applied Reinforcement Learning with Python. In *Applied Reinforcement Learning with Python*. Apress. <https://doi.org/10.1007/978-1-4842-5127-0>
- Bolton, W. (2017). *Programmable Logic Controllers* (5th ed., Vol. 53, Issue 9). Elsevier.
<http://www.elsevier.com/locate/scp>
- Chen, C.-T. (1993). *Analog & Digital Control Systems Design Transfer-function, State-space, & Algebraic Methods*. Saunders College Publishing.
- Cheung, T. F., & Luyben, W. L. (1979). Liquid-Level Control in Single Tanks and Cascades of Tanks with Proportional-Only and Proportional-Integral Feedback Controllers. *Industrial and Engineering Chemistry Fundamentals*, 18(1), 15–21.
<https://doi.org/10.1021/i160069a004>
- Dabney, J., & Harman, T. L. (1998). Mastering SIMULINK 2. In *MATLAB curriculum series*.
- Dale E. Seborg, Edgar, T. F., Mellichamp, D. A., & III, F. J. D. (2011). *Process Dynamics and Control* (A. Spicehandler (ed.); 3rd ed.). John Wiley & Sons, Inc.
- Dorf, R. C., & Bishop, R. H. (2008). *Modern Control System* (11th ed.). Pearson Education International.

Fan, L., Zhang, J., He, Y., Liu, Y., Hu, T., & Zhang, H. (2021). Optimal Scheduling of Microgrid Based on Deep Deterministic Policy Gradient and Transfer Learning. *Energies*, 14(3), 584. <https://doi.org/10.3390/en14030584>

Guillaume Matheron, Nicolas Perrin, O. S. (2020). *THE PROBLEM WITH DDPG: UNDERSTANDING FAILURES IN DETERMINISTIC ENVIRONMENTS WITH SPARSE REWARDS.*

Jian, W., & Wenjian, C. (2001). DEVELOPMENT OF AN ADAPTIVE NEURO-FUZZY METHOD FOR SUPPLY AIR PRESSURE CONTROL IN HVAC SYSTE. *IEEE*, 3806–3809. <https://doi.org/10.1109/ICSMC.2000.886603>

Khaled Kamel, & Kamel, E. (2013). *Programmable Logic Controllers: Industrial Control*. McGraw-Hill Professional Publishing.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*.

Madhu Sanjeevi. (2018, June 29). *Model Free Reinforcement learning algorithms (Monte Carlo, SARSA, Q-learning) / by Madhu Sanjeevi (Mady) / Deep Math Machine learning.ai / Medium*. <https://medium.com/deep-math-machine-learning-ai/ch-12-1-model-free-reinforcement-learning-algorithms-monte-carlo-sarsa-q-learning-65267cb8d1b4>

Mathworks. (2020). *Matlab User ' s Guide R 2020 a*. MathWorks, Inc.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *ArXiv*. <http://arxiv.org/abs/1312.5602>

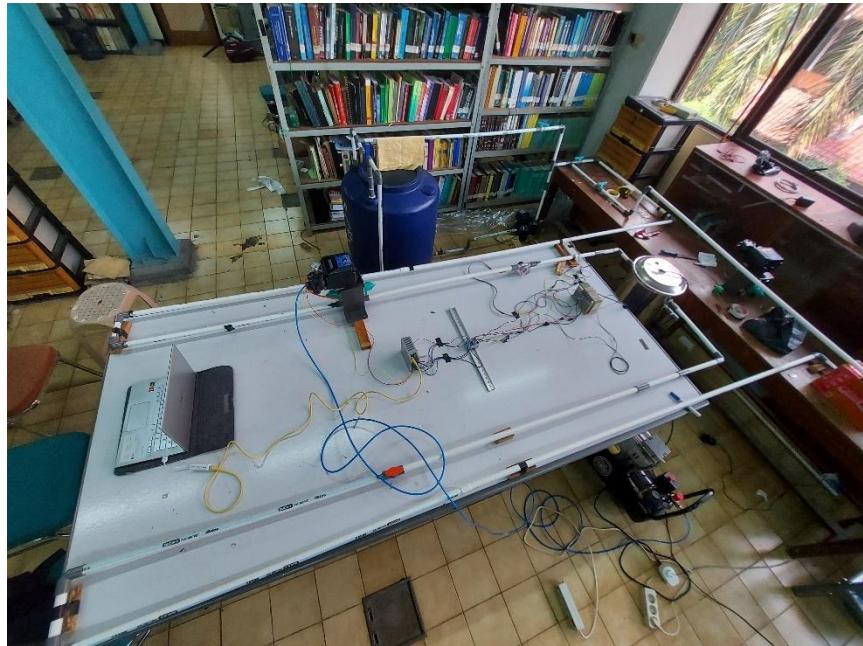
Nian, R., Liu, J., & Huang, B. (2020). A review On reinforcement learning: Introduction and applications in industrial process control. *Computers and Chemical Engineering*,

- 139, 106886. <https://doi.org/10.1016/j.compchemeng.2020.106886>
- Nise, N. S. (1990). CONTROL SYSTEMS ENGINEERING. In *The Knowledge Engineering Review* (Vol. 5, Issue 3). <https://doi.org/10.1017/S0269888900005403>
- Powell, W. B., Shewhart, W. A., & Wilks, S. S. (2011). Approximate Dynamic Programming. In *Markov Decision Processes in Artificial Intelligence: MDPs, beyond MDPs and applications* (2nd ed.). Wiley-Interscience.
<https://doi.org/10.1002/9781118557426.ch3>
- Richard Evans, & Jim Gao. (2016, July 20). *DeepMind AI Reduces Google Data Centre Cooling Bill by 40% / DeepMind*. <https://deepmind.com/blog/article/deepmind-ai-reduces-google-data-centre-cooling-bill-40>
- Saito, S., Wenzhuo, Y., & Shanmugamani, R. (2018). *Python Reinforcement Learning Projects*.
- Shaked Zychlinski. (2019, February 24). *The Complete Reinforcement Learning Dictionary*. Towards Data Science. <https://towardsdatascience.com/the-complete-reinforcement-learning-dictionary-e16230b7d24e>
- Siraskar, R. (2021). Reinforcement Learning for Control of Valves. *Elsevier*, 17(1 PART 1). <https://doi.org/10.3182/20080706-5-KR-1001.1520>
- Steven, E. L., & R.Coughanowr, D. (2009). Process systems analysis and control third edition. In *McGraw-Hill*. <http://www.slideshare.net/accelerate786/process-systems-analysis-and-control-third-edition>
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: an introduction* (2nd ed.). MIT Press.
- Sutton, R. S., Barto, A. G., & Book, A. B. (2018). *Reinforcement Learning: An Introduction*.
- Tay, T. T., Mareels, I. M. Y., & Moore, J. B. (1997). *High Performance Control*.

Wang, Z., & Hong, T. (2020). Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269(April), 115036.
<https://doi.org/10.1016/j.apenergy.2020.115036>

Yang, G., Zhang, F., Gong, C., & Zhang, S. (2019). Application of a Deep Deterministic Policy Gradient Algorithm for Energy-Aimed Timetable Rescheduling Problem. *MDPI*, 12, 1–19.

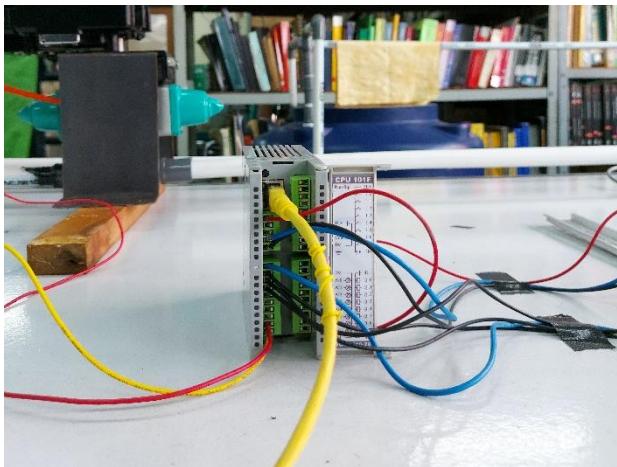
LAMPIRAN



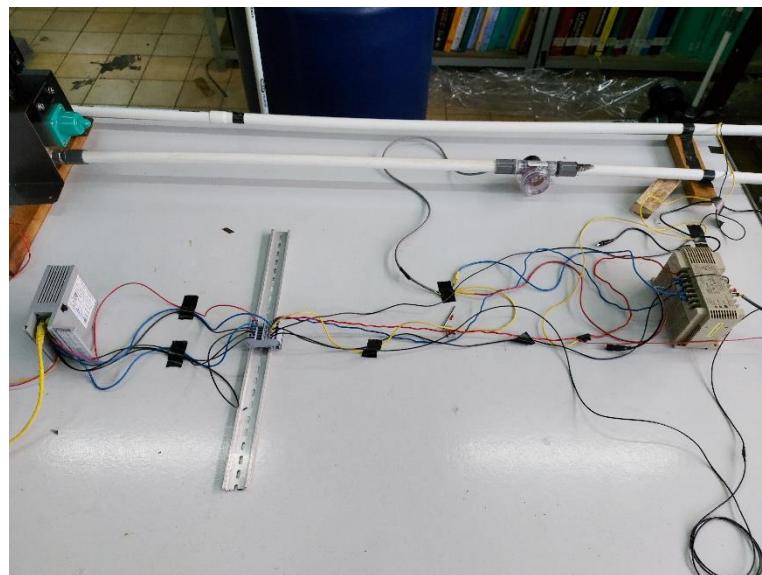
Lampiran 1. Plant Pengendali Ketinggian Level Air (tampak samping)



Lampiran 2. Plant Pengendali Ketinggian Level Air (tampak depan)



Lampiran 3. Konfigurasi PLC menggunakan koneksi ethernet



Lampiran 4. wiring system rancang bangun pengendali ketinggian air