PERANCANGAN *GRASPING CONTROL* PADA ROBOT TANGAN MENGGUNAKAN *REINFORCEMENT LEARNING*

Muhammad Alamulhuda[1*)], Wahyudi[2] dan Hadha Afrisal[3]

[123]Departemen Teknik Elektro, Fakultas Teknik, Universitas Diponegoro, Semarang, Indonesia

[*)]*E-mail* : m.alamulhuda26@gmail.com

Abstrak

Teknologi robotik telah mengalami kemajuan pesat terutama pada bidang tangan robot yang kini digunakan secara luas di berbagai industri seperti medis, manufaktur, farmasi, dan logistik. Tangan robot hadir dalam berbagai jenis dan model yang dirancang khusus untuk aplikasi tertentu. Tangan robot diharapkan dapat meniru bahkan melampaui fungsi tangan manusia. Semakin kompleks tugas yang dapat dilakukan oleh tangan robot, semakin kompleks pula perancangan sistem kendalinya terutama dalam pemrogramannya. Salah satu fungsi penting dari tangan robot adalah pengendalian gaya cengkeraman. Kompleksitas desain sistem kendali tangan robot disebabkan oleh perbedaan aktuator dan sensor yang digunakan serta spesifikasi setiap komponen. Pemrograman tangan robot dengan banyak variabel memakan waktu yang lama dan rentan terhadap konflik. Tantangan utama adalah menentukan posisi optimal robot saat memiliki cengkeraman yang tepat pada benda kerja. Solusi yang diusulkan adalah sistem pembelajaran kontrol genggaman menggunakan *Reinforcement Learning*. Penggunaan *episode* selama pelatihan mempengaruhi konvergensi atau pendekatan ke nilai optimal untuk menggenggam benda kerja. Peningkatan jumlah *episode* menghasilkan konvergensi yang lebih baik namun waktu pelatihan robot yang lebih lama. Hasil uji validasi menunjukkan tingkat keberhasilan 10 dari 10 percobaan dengan 2 di antaranya tidak stabil.

Kata kunci **:** *robot tangan, robot antropomorfik, pembelajaran penguatan, pengendalian genggaman, State Action State Reward (SARSA)*

Abstract

Advancements in robotics technology, especially in the area of robotic hands, have significantly impacted numerous fields such as medical, manufacturing, pharmaceuticals, and logistics. Various models and types of robotic hands are utilized to perform specific tasks, aiming to imitate and even surpass human hand functions. The complexity of designing control systems for robotic hands arises from the different actuators and sensors employed, as well as the specifications of each component. Programming robotic hands with numerous variables is time-consuming and may result in conflicts. One crucial feature of robotic hands is the ability to control grip force. To address this issue, a grasping control learning system was designed using Reinforcement Learning, implementing dynamic programming and state-action-state-reward (SARSA) with FSR value comparison and linear actuator position. The number of episodes used during training influenced the convergence or approach to the optimal value for gripping the workpiece. The system's results indicated that 300 episodes led to the position (1438, 1578, 1558, 1548) for the index, middle, ring, and little fingers, respectively, being the closest to the optimal gripping value. Validation testing showed a 10 out of 10 success rate, with two of them being unstable.

Keywords : hand robot, anthropomorphic robot, Reinforcement Learning, Grasping Control, State Action State Reward (SARSA)

1. Pendahuluan

Tangan merupakan salah satu bagian tubuh manusia yang memiliki fungsi paling kompleks dan penting, baik sebagai perangkat masukan (indra peraba) maupun keluaran (pekerjaan fisik) bagi manusia. Pada perkembangan robot saat ini, banyak peneliti yang sudah dan sedang berusaha membuat robot yang mereplikasi kinerja tangan manusia, seperti robot lengan dan robot tangan. Beberapa kendala yang ditemui peneliti dalam proses perancangan sistem kendali robot tangan yaitu tingginya kompleksitas perancangan robot tangan [1]. Kompleksitas pada perancangan sistem kendali robot tangan dipengaruhi oleh beberapa hal contohnya seperti banyaknya variabel yang perlu dikendalikan, tingginya dimensi pengendalian, dan pemilihan metode atau sistem pengendalian.

Pemilihan sistem pengendalian yang tepat memiliki peran penting dalam proses merancang sistem kendali robot agar sesuai dengan kemampuan dan kompleksitas yang dikendalikan. Pengontrolan robot tangan dapat dilakukan dengan mengukur resistansi, posisi, sudut, atau percepatan yang nilainya didapat dari sensor yang digunakan yang kemudian dimasukan ke sistem untuk mengatur pergerakan, baik perubahan posisi, perubahan sudut, maupun kecepatan sesuai yang diinginkan.

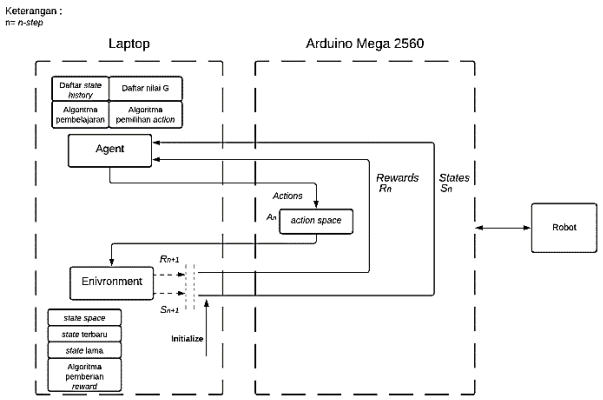
Salah satu fungsi dari tangan yang menjadi tantangan di dunia robotika bagi para peneliti yaitu fungsi menggenggam atau *grasping*. *Grasping control*, terutama pada robot tangan, memiliki kompleksitas yang tinggi dikarenakan variabel yang digunakan dan dibutuhkan yang banyak, juga fleksibilitas dan tingginya varian dari hasil pengendalian. Terdapat beberapa hal yang menjadi acuan dalam *grasping control* contohnya adalah bentuk benda yang akan digenggam, jenis benda, tekstur permukaan benda, dan massa benda.

Penelitian mengenai perancangan sistem kendali robot tangan menggunakan *reinforcement learning* berbasis *vision* yang pernah dilakukan [2], pembahasan tentang perancangan *blind grasping control* menggunakan *reinforcement learning* [3], didapatkan bahwa penggunaan *reinforcement learning* dalam perancangan meningkatkan efisiensi dibandingkan dengan penggunaan metode *programming* secara konvensional [4].

Pada makalah ini dilakukan perancangan sistem pembelajaran *grasping control* dengan menggunakan *reinforcement learning* untuk menemukan kombinasi posisi *linear actuator* paling optimal sehingga dapat memegang benda dengan kekuatan yang optimal/tepat.

2. Metode

Dalam perancangannya, sistem pembelajaran (*learning*) untuk mengatur pengendalian penggenggaman (*grasping control*) pada robot tangan dengan *reinforcement learning* memanfaatkan komponen-komponen masukan dan aktuator yang terdapat pada robot tangan, diantaranya sensor FSR [5] dan nilai *feedback* dari *linear actuator* Actuonix PQ-12-P [6] sebagai masukan, dan motor servo [7] serta *linear actuator* sebagai keluaran. *Agent* dan *Environment* perlu diidentifikasi menggunakan *Markov Decision Process* [8] sebagai dasar perumusan masalah dalam menentukan *state*, *action*, dan *reward*. *Reinforcement Learning* pada dasarnya merupakan *trial and error* yang dilakukan secara otomatis dengan tujuan untuk mengumpulkan *reward* sebanyak-banyaknya melalui perubahan *state* yang terjadi dikarenakan *action* yang dilakukan. *Agent* dan *Environment* memiliki elemen-elemen dan fungsi-fungsi yang digunakan pada sistem pembelajaran *grasping control*. Pada *Agent* terdapat daftar *state history*, daftar nilai G atau *Goals Value*, algoritma pembelajaran (fungsi *learning*), algoritma pemilihan *action*. Pada *Environment* terdapat *state space*, *state* terbaru, *state* lama, algoritma pemberian *reward*. Elemen-elemen dan fungsi-fungsi yang digunakan pada sistem dapat dilihat pada Gambar 1 yang merupakan diagram blok dari sistem kendali robot tangan yang digunakan pada perancangan sistem *Reinforcement Learning* untuk sistem pengendalian penggenggaman (*grasping control*).



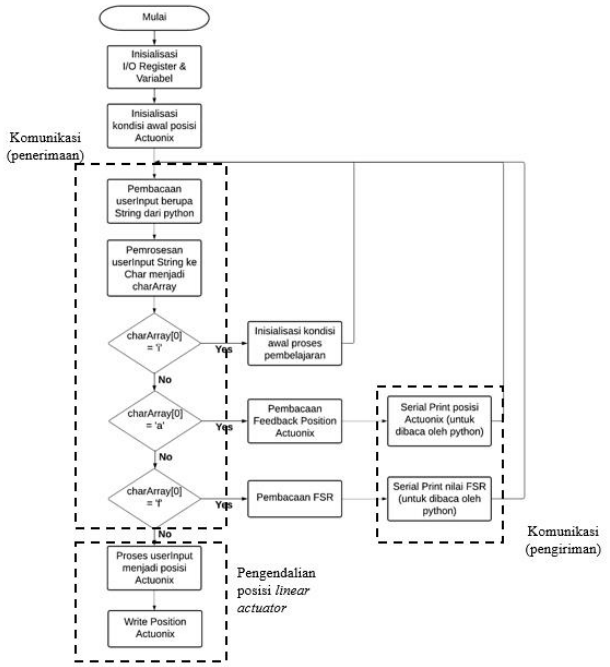
Gambar 1. Diagram blok sistem pembelajaran untuk *grasping control*

Sistem pembelajaran menggunakan dua prosesor yaitu laptop dan Arduino Mega 2560, di mana pada laptop menggunakan Python sebagai dasar programnya yang berfungsi untuk pengolahan data. Pada laptop terdapat dua elemen utama yaitu *Agent* dan *Environment*. Beberapa fungsi utama pada *Agent*, yaitu daftar *state history* yang digunakan untuk menyimpan riwayat *state* yang sudah dilalui, daftar nilai *G* (*Goals Value*) sebagai elemen paling penting yang digunakan saat pemilihan *action*, algoritme pemilihan *action* untuk pemrosesan data agar dapat menentukan *action* selanjutnya, algoritme pembelajaran yang befungsi agar robot dapat belajar sehingga dapat mengambil keputusan yang terbaik. Fungsi-fungsi dan elemen-elemen utama pada *Environment*, yaitu *state space* yang merupakan jangkauan kemungkinan *state* yang tersedia, *state* terbaru digunakan untuk menyimpan data *state* saat ini, *state* lama digunakan untuk menyimpan data *state* pada *step* sebelumnya, dan algoritme pemberian *reward*. Elemen-elemen yang terdapat pada Arduino yaitu pengambilan data FSR dan nilai posisi *linear actuator* dari robot tangan serta pemberian data yang sudah diproses untuk mengeksekusi gerakan selanjutnya dari robot tangan. Seluruh sistem menggunakan komunikasi dua arah secara serial.

2.1 Perancangan Perangkat Lunak pada Arduino Mega

Perancangan ini menggunakan Arduino Mega 2560 [9] sebagai mikrokontrolernya, terdiri dari beberapa fungsi utama pada sistem pembelajaran, yaitu pengambilan data yang akan digunakan dalam pemrosesan data yang akan dilakukan di laptop menggunakan Python dan penggerak

linear actuator pada robot dengan data yang didapat dari pemrosesan pada laptop. Diagram alir Arduino sistem *grasping control* dengan *reinforcement learning* dapat dilihat pada Gambar 2.



Gambar 2. Diagram alir sistem *grasping control* dengan *reinforcement learning* pada Arduino

Tahap awal sistem pembelajaran dimulai dari inisialisasi awal posisi *linear actuator* pada kombinasi posisi (1028, 1028, 1028, 1028) dalam *microseconds* servo.

Arduino melakukan pembacaan *Serial Monitor* sebagai metode komunikasi secara serial untuk menerima *command* atau perintah dari program Python pada laptop. *Command* atau perintah yang diberikan memiliki beberapa variasi dan berguna untuk men-*trigger* fungsi dalam proses pembelajaran. Terdapat 3 variasi perintah yang digunakan untuk pengambilan (pembacaan dan pengiriman) data dan pengeksekusian untuk menggerakkan aktuator pada robot.

Pengendalian posisi *linear actuator* terdapat dua bagian, pertama pada saat inisialisasi untuk mengatur posisi *linear actuator* pada posisi awal pengujian, yaitu jari satu per satu menutup sampai mencapai titik sentuh di mana nilai FSR sudah membaca adanya benda yang disentuh dan kedua pada saat mendapatkan nilai yang sudah diproses oleh Python pada laptop untuk mengubah posisi dan kekuatan genggaman pada saat pengujian.

Nilai posisi *linear actuator*, yang digunakan sebagai *state* pada sistem pembelajaran, diambil menggunakan salah satu fungsi yang digunakan untuk motor servo yaitu servo.`readMicroseconds()`.

Sistem pembelajaran membutuhkan nilai FSR yang akan digunakan sebagai *reward* yang akan diproses pada laptop menggunakan Python. Pengambilan data FSR tersebut dilakukan melalui mikrokontroler Arduino Mega 2560 yang nantinya akan dikirimkan ke Python pada laptop. Data FSR diambil menggunakan salah satu fungsi dari Arduino, yaitu `AnalogRead()`.

Proses penge-print-an data yang sudah diambil merupakan bagian dari komunikasi bagian pengiriman.

Pengendalian kedua yaitu pada saat Arduino tidak membaca command apapun yang dikirimkan oleh Python.

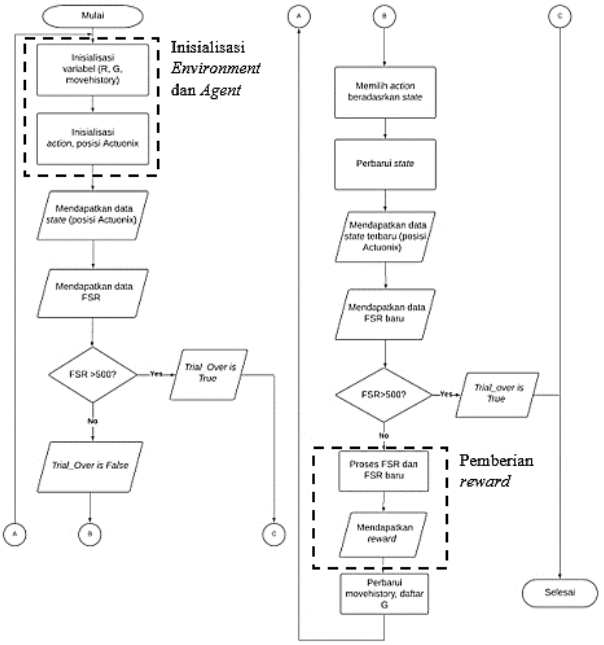
Saat tidak teridentifikasinya perintah dari Python pada laptop, mikrokontroler menganggap data yang dikirimkan merupakan data dari posisi *linear actuator* terbaru, sehingga data mentah yang didapatkan dapat diproses sehingga dapat digunakan untuk dikirimkan ke robot untuk menggerakkan tiap jari sesuai dengan posisi yang dikirimkan dari Python.

2.2 Perancangan Perangkat Lunak pada Laptop menggunakan Python

Salah satu elemen dalam perancangan sistem *Reinforcement Learning* yaitu memodelkan *Environment* dan *Agent*-nya. *Reinforcement Learning* menggunakan *Markov Decision Process* sebagai fondasi atau dasar dalam memodelkan permasalahan untuk sistem yang akan dibuat. Kegiatan memodelkan masalah ini juga dapat dikatakan sebagai tahap perencanaan sebelum merancang dan membuat sistem *Reinforcement Learning*. Diagram alir yang digunakan perangkat lunak pada laptop menggunkan python dapat dilihat pada Gambar 3.

Penginisialisasian *Environment* memiliki beberapa elemen dalam *reinforcement learning* yang dideklarasi. Elemen yang dideklarasi pada penginsialisasian *environment* adalah *state*, *list* sebagai wadah dari FSR yang nantinya akan diproses menjadi *reward*. *Action* yang akan digunakan oleh sistem pembelajaran juga ikut dideklarasi. Sistem pembelajaran yang dirancang menggunakan *Action* yang terbagi menjadi tiga, yaitu {0 (mempertahankan posisi akhir), 10 (menambah nilai posisi linear actuator), -10 (mengurangi nilai posisi *linear actuator*)}. Dalam penentuannya, diputuskan untuk melakukan aksi atau perubahan posisi yang sama pada semua jari untuk mengurangi kompleksitas dari sistem. Pendeklarasian variabel *action* dapat dilihat pada senarai program inisialisasi *environment* baris kedua sebagai berikut.

```
Actions = [(0,0,0,0), (10,10,10,10), (-10,-10,-10,-10)]
```



Gambar 3. Diagram alir *grasping control* dengan *reinforcement learning* pada robot tangan pada laptop

Angka yang dideklarasikan pada variabel *action* berturut-turut menunjukkan perubahan posisi dari Actuonix pada jari telunjuk, tengah, manis, dan kelingking.

Proses penentuan nilai yang akan digunakan sebagai *state* dan pendeklarasian pada inisialisasi *environment* diikuti oleh algoritme dan program yang dibutuhkan sebagai *tools* untuk pengambilan dan pemrosesan data *state* dalam sistem pembelajaran. Pemilihan dan pemrosesan *state* dilakukan sepenuhnya melalui pemrograman. Berikut merupakan beberapa baris pada pemrograman python yang digunakan untuk melakukan pemilihan *state*.

Pengambilan nilai *feedback* Actuonix dari Arduino menggunakan fungsi getActxdata() yang dapat dilihat pada bagian "Mendapatkan data Actuonix" di Gambar 3.

Pembaharuan nilai *State* menggunakan beberapa fungsi, berikut merupakan fungis utama yang digunakan dalam pemrosesan pembaharuan nilai atau data dari *State*

```
def update_state(self, action, s):
        #assign the new state to old state
        self.oldactx = self.newactx
        #update the new state given choosen
action
        self.newactx = tuple(sum(x) for x in
zip(self.newactx,Actions[action]))
        #add tie step
        self.step += 1
        s = self.newactx
        return s
```

Pengirman nilai posisi *linear actuator* terbaru yang sudah diproses dari python ke Arduino menggunakan fungsi berikut

```
def write_actxnewpos(x):
    x = str(x)
    ser.write(bytes(x, 'ascii'))
    time.sleep(1)
```

Nilai *Force Sensitive Resistor* (FSR) digunakan sebagai dasar dari pemberian Reward pada sistem pembelajaran.

FSR yang digunakan sebagai dasar dari pemberian *reward* digunakan juga untuk fungsi lainnya, yaitu sebagai pemberhenti *episode* jika nilai melebihi 500 dalam ADC 10-bit. Pemberian batasan tersebut bertujuan agar genggaman dari robot tidak terlalu kuat sehingga mengakibatkan kerusakan pada benda kerja maupun pada aktuator yang digunakan pada robot tangan. Pemberian batasan ini disebut juga dengan *termination state*[11].

Proses pemilihan *Action* yang akan dilakukan oleh robot terbagi menjadi dua pilihan, yaitu eksplorasi dan eksploitasi dengan memanfaatkan nilai posisi *linear actuator* sebagai *state* dan FSR sebagai *reward*. Eksplorasi dilakukan dengan kemungkinan awal 30% yang terus berkurang setiap *episode* yang sudah dilalui sampai robot hanya memilih *Action* dengan *Goals value* tertinggi di mana disebut juga dengan eksploitasi, yaitu pemilihan *state* selanjutnya yang memiliki *Goal Value* tertinggi.

Pembaruan nilai *state* menggunakan beberapa fungsi. Fungsi utama yang digunakan dalam pemrosesan pembaruan nilai atau data dari *state* adalah sebagai berikut.

```
def update_state(self, action, s):
        #assign the new state to old state
        self.oldactx = self.newactx
        #update the new state given choosen
action
        self.newactx = tuple(sum(x) for x in
zip(self.newactx,Actions[action]))
        #add tie step
        self.step += 1
        s = self.newactx
        return s
```

Pengiriman nilai posisi *linear actuator* terbaru yang sudah diproses dari python ke Arduino menggunakan fungsi berikut.

```
def write_actxnewpos(x):
    x = str(x)
    ser.write(bytes(x, 'ascii'))
    time.sleep(1)
```

Penentuan *Reward*, biasanya, menggunakan nilai skalar seperti 0, 1, 2, 3, -1, -2, -3. *Reward* didapatkan dari perbandingan antara nilai FSR awal dengan FSR akhir. Sistem bertujuan untuk mengumpulkan nilai *reward* sebanyak-banyaknya.

3. Hasil dan Analisis

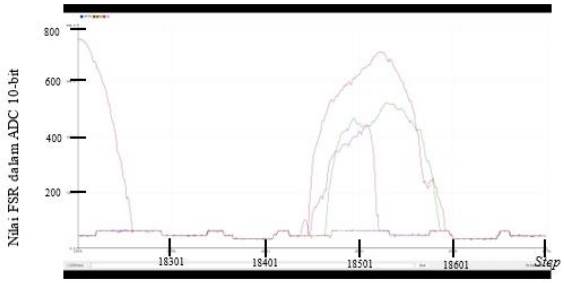
Pengujian dan analisis dilakukan untuk mengetahui kinerja dari penerapan rancangan sistem pembelajaran *Reinforcement Learning* untuk *Grasping Control* pada robot tangan. Pengujian dilakukan dengan menggunakan botol air mineral ukuran 600 ml yang diisi air sehingga memiliki tingkat kekerasan yang lembut. Kondisi robot tangan pada kondisi diam dan benda diletakkan pada bagian dalam tangan robot.

3.1 *Pre-Tuning Policy* Sistem Pembelajaran pada Robot Tangan

Pre-tuning dilakukan sebelum melakukan proses pembelajaran dengan program sederhana pada Arduino dengan menggerakkan jari untuk menyentuh benda kerja dengan tingat kekerasan yang lembut (dalam kasus ini, botol air mineral dengan ukuran 600 ml berisi air yang dilapisi isolasi tipis). Percobaan yang dilakukan meliputi percobaan menggerakkan keseluruhan jari (kecuali jempol) secara bersamaan dan menggerakkan satu per satu jari (kecuali jempol) menyentuh benda kerja. Pada percobaan digunakan batasan nilai FSR sebesar 450 dalam ADC sebagai nilai batas untuk setiap jari kembali ke posisi semula. Nilai 450 ditentukan dari mempertimbangkan nilai 517 dalam ADC 10-bit pada sensor FSR mencapai gaya 1,5 N, untuk itu diambil secara langsung 450 sebagai pembatas (*constraint*) untuk menghindari genggaman yang terlalu kuat dikarenakan *delay* dari respon pembacaan nilai FSR. Percobaan digambarkan pada Gambar 4 dengan hasil respon aktual dari sensor FSR yang dapat dilihat pada Gambar 5.

Gambar 4. *Pre-tuning policy*

Terlihat dari Gambar 5 yang didapat dari *serial plotter* pada Arduino IDE, bahwa nilai FSR bisa mencapai lebih dari 700 ADC dari yang dibatasi pada program sebesar 450. Perbedaan nilai yang dibatasi dan pembacaan aktual akan sangat berpotensi merusak benda kerja saat proses pembelajaran, juga berpotensi merusak aktuator yang digunakan sebagai penggerak tiap jari.

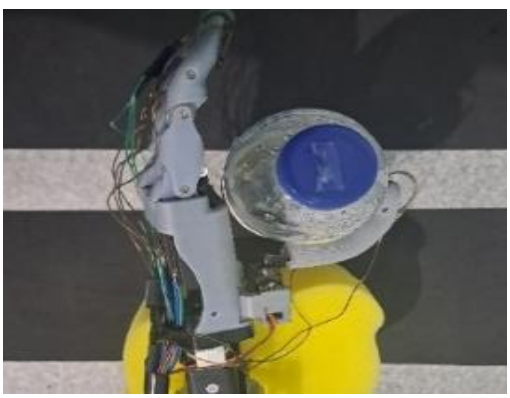


Gambar 5. Tampilan *Serial Plotter* respon nilai sensor FSR saat menyentuh benda kerja

Benda kerja sudah menunjukkan keremukan atau deformasi pada nilai ADC 550. Berdasarkan perbedaan data batasan dan yang terbaca juga pengaruh delay respon pembacaan maka ditentukan *policy* yang digunakan pada sistem pembelajaran yaitu dengan setiap *action* ke-n yang dilakukan pada *state* ke-n, *reward* yang diberikan didapatkan dari perbedaan nilai FSR pada *state* ke-n dikurangi dengan nilai FSR pada *state* ke-n-1, dengan nilai FSR yang melebihi 550 dalam ADC disamadengankan dengan 225, yang merupakan setengah dari nilai maksimum, dalam ADC untuk memberikan *reward negatif* atau biasa disebut sebagai *punishment* yang bertujuan agar melalui sistem pembelajaran robot belajar untuk menghindari nilai FSR yang berlebihan.

3.2 Pengujian Sistem Pembelajaran pada Jari Robot Tangan

Pengujian dilakukan dengan memvariasikan jumlah *episode* yang dilakukan, bertujuan untuk melihat tingkat konvergensi data. Variasi yang digunakan yaitu 100, 150 dan 300 *episode* Data memperlihatkan G *Value* pada sumbu y dan *state* ke-n pada sumbu x. Data dengan G *Value* tertinggi menujukkan nilai untuk posisi paling optimal dari percobaan. G *Value* dihitung menggunakan persamaan perhitungan nilai Q pada Q-*Learning* yang diimplementasikan ke dalam program. Penempatan dan posisi robot tangan serta benda kerja pada saat pengujian dapat dilihat pada Gambar 6.



Gambar 6. Penempatan dan Posisi Robot Tangan dan Benda Kerja saat Pengujian Sistem Pembelajaran

3.2.1 Pengujian Sistem Pembelajaran dengan 100 *Episode*

Pengujian yang dilakukan menggunakan jumlah *episode* 100 menggenggam benda kerja. Data hasil pengujian dapat dilihat pada Tabel 1.

Tabel 1 Hasil pengujian sistem pembelajaran dengan variasi 100 *episode*

No	*Goals Value*
1	-9,925
2	-65,70
3	-26,67
4	-3,96
5	-12,99
6	0,42
7	-7,06
8	0,17
9	0,22
10	-9,85
11	0,20
12	0,27
13	0,13
14	0,35
15	6,22
16	4,83
17	0,84
18	0,69
19	0,32
20	0,79
21	-3,52
22	0,63

Berdasarkan data yang disajikan pada Tabel 1, terdapat 22 *state* atau kombinasi posisi *linear actuator* jari robot tangan yang didapat dengan jumlah *step* sebanyak 112 selama percobaan dengan 100 *episode*. Terlihat dari data tersebut, nilai *Goals* terendah terletak pada posisi ke-2 dengan nilai -65,70 dimana kombinasi posisi *linear actuator* untuk jari telunjuk, tengah, manis dan kelingking berurutan (1438, 1468, 1568, 1508) dan nilai tertinggi pada posisi ke-15 dengan nilai 6,22 pada kombinasi *linear actuator* (1478, 1508, 1608, 1548).

3.2.2 Pengujian Sistem Pembelajaran dengan 150 *Episode*

Pengujian yang dilakukan menggunakan metode dan pengaturan yang sama seperti pengujian sebelumnya dengan perbedaan pada variasi jumlah *episode* pembelajaran yang berupa 150 *episode*. Data dari pengujian dapat dilihat pada Tabel 2.

Tabel 2 Hasil pengujian sistem pembelajaran dengan variasi 150 *episode*

No	*Goals Value*
1	0,809
2	0,14
3	-9,01
4	-2,88
5	0,53
6	0,74
7	0,62
8	0,76
9	0,568
10	0,348
11	0,809
12	0,84
13	0,53
14	0,48
15	0,647
16	0,89
17	0,20
18	0,798
19	0,65
20	0,189
21	0,465
22	0,462
23	0,39
24	0,71

Berdasarkan data hasil pengujian pada Tabel 2, didapatkan 24 *state* atau kombinasi posisi *linear actuator* jari robot tangan dengan jumlah step 164. Terlihat dari data tersebut, nilai G terendah terletak pada kombinasi posisi ke-3 dengan nilai -9,01 dimana kombinasi posisi *linear actuator* untuk jari telunjuk, tengah, manis dan kelingking berurutan merupakan kondisi awal dari robot tangan (1438, 1478, 1578, 1488), terendah kedua setelah itu terletak pada kombinasi posisi ke-4 dengan nilai -2,885 dengan kombinasi *linear actuator* (1428, 1468, 1568, 1478) dan nilai tertinggi pada posisi ke-16 dengan nilai 0,847 pada kombinasi *linear actuator* (1468, 1508, 1608, 1518).

3.2.3 Pengujian Sistem Pembelajaran dengan 300 *Episode*

Pengujian yang dilakukan menggunakan metode dan pengaturan yang sama seperti pengujian-pengujian sebelumnya dengan perbedaan pada variasi jumlah *episode* pembelajaran yang berupa 300 *episode*. Data dari pengujian dapat dilihat pada Tabel 3.

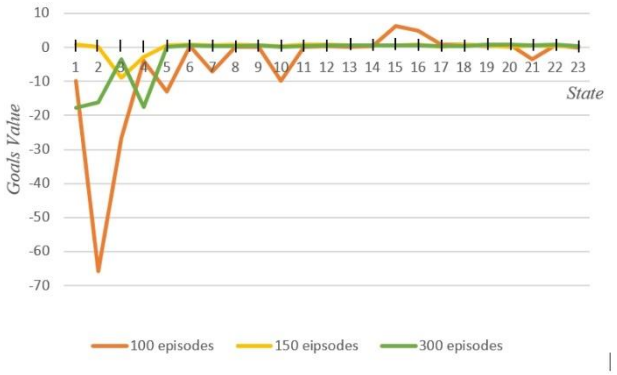
Tabel 3 Hasil pengujian sistem pembelajaran dengan variasi 300 *episode*

No	*Goals Value*
1	-17,75
2	-16,20
3	-3,59
4	-17,47
5	0,204
6	0,596
7	0,287
8	0,397
9	0,68
10	0,22
11	0,446
12	0,522
13	0,523
14	0,588
15	0,669
16	0,647
17	0,385
18	0,396
19	0,748
20	0,847
21	0,635
22	0,8278
23	0,412
24	0,287

Berdasarkan data hasil pengujian pada Tabel 3, didapatkan 23 *state* atau kombinasi posisi *linear actuator* Actuonix dengan jumlah *step* sebanyak 331. Terlihat dari data pada Tabel 3, nilai *G* terendah terletak pada kombinasi posisi ke-1 dengan nilai -17,75 di mana kombinasi posisi *linear actuator* untuk jari telunjuk, tengah, manis, dan kelingking berurutan merupakan kondisi awal dari robot tangan (1028, 1028, 1028, 1028), terendah kedua terletak pada kombinasi ke-4 dengan nilai -17,47 dengan kombinasi posisi (1418, 1418, 1558, 1438) dan nilai tertinggi pada posisi ke-20 dengan nilai 0,847 pada kombinasi *linear actuator* (1438, 1578, 1558, 1548).

3.2.4 Perbandingan Hasil Pembelajaran dengan Variasi *Episode* 100, 150, 300

Pengujian sistem pembelajaran yang dilakukan dengan variasi jumlah *episode* 100, 150, dan 300 mendapatkan hasil nilai *Goals Value* pada *state* ke-n seperti pada representasi di Gambar 7.



Gambar 7. Grafik perbandingan hasil pengujian tiap variasi *episode*

Kombinasi posisi dengan nilai tertinggi dari setiap variasi *episode* pada pengujian yaitu (1478, 1508, 1608, 1548) pada 100 *episode*, (1468, 1508, 1608, 1518) pada 150 *episode*, dan (1438, 1578, 1558, 1548) pada 300 episode. Data yang ditunjukan Gambar 7 menggambarkan bahwa pengujian sistem pembelajaran dengan variasi 300 *episode* menunjukkan konvergensi yang lebih baik dibandingkan dengan variasi 100 dan 150 episode. Dari yang terlihat dari data pada Gambar 7, peningkatan jumlah *episode* mempengaruhi konvergensi antara tiap *state*. Hasil dari pengujian bersesuaian dengan prinsip kerja dari *reinforcement learning* semakin banyak robot belajar, maka semakin tepat penilaian robot terhadap nilai suatu tindakan atau posisi. Konvergensi data hasil pembelajaran merupakan salah satu indikasi dari kualitas sistem pembelajaran menggunakan *reinforcement learning* [10]. Grafik perbandingan hasil pengujian tiap variasi *episode* dapat dilihat pada Gambar 7.

3.3 Analasis Hasil Pengujian Sistem Pembelajaran

Pengujian sistem pembelajaran *grasping control* dengan *reinforcement learning* yang sudah dilakukan dengan variasi 100, 150, dan 300 *episode* memberikan keluaran nilai optimal kombinasi posisi dari *linear actuator* melalui *Goals Value* tertinggi. Selanjutnya dilakukan pengujian validasi sebagai parameter dan validasi penilaian bahwa nilai yang diberikan benar optimal. Dilakukan dengan robot tangan menggenggam benda, kemudian akan diangkat secara manual untuk melihat apakah dapat mengangkat benda dengan kekuatan yang tepat dan stabil.

Proses pengujian dapat dilihat pada Gambar 8.

Gambar 8 Pengujian validasi hasil pembelajaran dengan variasi 100, 150, 300 *episode.*

Data hasil percobaan dapat dilihat pada Tabel 4 dengan V berhasil, O berhasil namun tidak stabil, dan X tidak berhasil atau gagal

Tabel 4 Hasil pengujian validasi nilai posisi dari tiap variasi *episode*

No	Variasi *Episode*		
	100	150	300
1	**V**	X	**V**
2	X	**O**	**V**
3	**O**	**V**	**V**
4	**O**	**O**	**V**
5	**V**	**O**	**O**
6	**V**	**V**	**V**
7	**V**	**O**	**V**
8	**V**	**V**	**V**
9	**V**	**V**	**O**
10	X	**V**	**V**

Keterangan:
V : berhasil
O : berhasil namun tidak stabil
X : gagal

Dari pengujian yang dilakukan dapat divalidasi bahwa peningkatan *episode* saat pembelajaran mempengaruhi nilai kombinasi yang didapatkan dengan nilai optimal untuk menggenggam benda tertentu. Pengujian pada data hasil pembelajaran dengan variasi 100 *episode* menunjukkan kinerja yang baik dengan berhasil mengangkat benda sebanyak 8 dengan 2 diantaranya tidak stabil, dan gagal sebanyak 2 kali, namun saat pengujian berlangsung, genggaman menekan benda kerja dengan cukup kencang. Pengujian dengan variasi 150 *episode* menunjukkan hasil yang tidak jauh berbeda, namun jumlah berhasil yang tidak stabil lebih banyak dengan 9 berhasil, 4 diantaranya tidak stabil, dan gagal 1 kali. Pada saat pengujian genggaman yang dilakukan oleh robot tangan jauh lebih lembut dan memilki tenaga yang lebih rendah dibanding pengujian dengan data hasil pembelajaran variasi 100 *episode*. Hasil pada pengujian dengan data yang didapatkan dari hasil pembelajaran variasi 300 *episode* menunjukkan hasil yang lebih baik, dengan jumlah keberhasilan 10 dari 10, 2 diantaranya tidak stabil, dan 0 gagal. Pada pengujian dengan menggunakan data dari hasil pembelajaran dengan variasi 300 *episode* robot tangan menggenggam dengan tenaga atau kekuatan yang lebih optimal dibandingkan saat menggunakan data dari hasil pembelajara dengan variasi 100 dan 150 *episode*.

4. Kesimpulan

Sistem pembelajaran *Grasping Control* menggunakan *Reinforcement Learning* berhasil diterapkan dengan menggunakan *constraint* yang didapat dari proses *pre-tuning* berupa batasan nilai FSR tidak melebihi 550 dalam ADC. Hasil pengujian sistem pembelajaran dengan 300 *episode* menunjukkan grafik data yang lebih konvergen dibandingkan dengan variasi 100 dan 150 *episode.* Kombinasi posisi paling optimal dari ketiga variasi pengujian berada pada hasil dari pembelajaran dengan variasi 300 *episode* ditunjukkan dari hasil pengujian validasi dengan total keberhasilan 10 kali dari 10 kali percobaan, dengan dua diantaranya tidak stabil

REFERENSI

[1] M. Tavakoli, M. R. Homaeinezhad, and M. Moghavvemi. Design and development of an anthropomorphic robot hand. In IEEE International Conference on Control and Automation, pp. 1374–1379, 2007.

[2] Judith P. Florez, Juan D. Velasquez, and Manuel E. Acosta, "*Control System Design for a Robotic Arm Based on Vision and Reinforcement Learning*," 2019 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), pp. 1-6, 2019.

[3] C. Chen, R. Wang, Y. Zhou, Y. Lu, L. Jiang and X. Zhang, "*Blind Grasping Control Based on Reinforcement Learning for Robotic Manipulator*," 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, pp. 509-514, 2018.

[4] R. T. Yenamandra, S. S. Vadlamudi, S. R. Yemula and S. K. Sivapuram, "*Reinforcement learning based robotic grasping control system for object manipulation*," 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, pp. 1-5, 2017.

[5] Interlink Electronics. FSR 402 Datasheet. Diakses pada 1 November 2022 dari https://www.interlinkelectronics.com/datasheets/FSR402.pdf. 2015

[6] Actuonix Motion Devices, Inc. Actuonix PQ12-P Datasheet. Diakses pada 28 November 2022, dari https://www.actuonix.com/download/datasheets/PQ12-User-Guide.pdf. 2021

[7] Tower Pro. MG90S. [Datasheet]. Diambil dari http://www.towerpro.com.tw/product/mg90s-1/. Diakses pada 2022.

[8] Sutton, R. S., & Barto, A. G. *Reinforcement learning: An introduction (2nd ed.)*. MIT Press. 2018

[9] Arduino. Arduino Mega 2560. https://store.arduino.cc/usa/mega-2560-r3. 2021.

[10] Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). *Deep reinforcement learning*: A brief survey. IEEE Signal Processing Magazine, 34(6), 26-38.2017.

[11] Mean Well. LRS-150-12 *Datasheet*. Diakses pada 28 November 2022, dari https://www.meanwell.com/Upload/PDF/LRS-150/LRS-150-SPEC.PDF. 2021.