IJEIS (Indonesian Journal of Electronics and Instrumenations Systems)
Vol.13, No.1, April 2023, pp. 1~12
ISSN (print): 2088-3714, ISSN (online): 2460-7681
DOI: 10.22146/ijeis.73865

Implementasi Sistem Kendali Keseimbangan Statis Pada Robot *Quadruped* Menggunakan *Reinforcement Learning*

Hidayat Eko Saputro*[1], Nur Achmad Sulistyo Putro[2], Sri Hartati[3], Ilona Usuman[4]
[1]Program Studi Elektronika dan Instrumentasi, FMIPA UGM, Yogyakarta, Indonesia
[2,3,4]Departemen Ilmu Komputer dan Elektronika, FMIPA UGM, Yogyakarta, Indonesia
e-mail: ***[1]hidayat.e.s@mail.ugm.ac.id**, [2]nur.achmad.s.p@ugm.ac.id, [3]shartati@ugm.ac.id, [4]ilona@ugm.ac.id

Abstrak

Hal dasar yang perlu diperhatikan saat membuat robot quadruped adalah masalah keseimbangan. Faktor tersebut sangat menentukan keberhasilan robot quadruped dalam melakukan gerakan seperti menstabilkan tubuh pada sebuah bidang miring, gerakan berjalan dan lain-lain. Metode kendali umpan balik konvensional dengan melakukan pemodelan matematis dapat dimanfaatkan untuk melakukan penyeimbangan robot, namun metode ini masih memiliki kelemahan. Penerapan metode kendali umpan balik konvensional sering menghasilkan pengontrol yang tidak akurat, sehingga harus disetel secara manual untuk penerapannya. Pada penelitian ini digunakan metode reinforcement learning dengan menggunakan algoritma Q-Learning. Penggunaan metode reinforcement learning dipilih karena tidak diperlukan perhitungan matematis untuk mengontrol keseimbangan robot quadruped. Proses pembelajaran sistem untuk melatih kemampuan agent dilakukan dengan menggunakan simulator Gazebo. Hasil pembelajaran menunjukkan bahwa sistem dapat berjalan dengan baik yang dibuktikan dengan nilai sum rewards per episode yang semakin tinggi.

Kata kunci— *Quadruped, Reinforcement Learning, Q-Learning, Rewards, Gazebo*

Abstract

The basic thing to consider when building a quadruped robot is the issue of balance. These factors greatly determine the success of the quadruped robot in carrying out movements such as stabilizing the body on an inclined plane, walking movements and others. Conventional feedback control methods by performing mathematical modeling can be used to balance the robot. However, this method still has weaknesses. The application of conventional feedback control methods often results in an inaccurate controller, so it must be manually tuned for its application. In this study, reinforcement learning methods were used using Q-Learning algorithms. The use of reinforcement learning methods was chosen because no mathematical calculations are needed to control the balance of quadruped robots. The process of learning the system to train the agent's abilities is carried out using a Gazebo simulator. The learning results show that the system could run well as evidenced by the higher value of sum rewards per episode.

Keywords— *Quadruped, Reinforcement Learning, Q-Learning, Rewards, Gazebo*

1. PENDAHULUAN

Robot merupakan perangkat otomatis yang dirancang untuk mampu bergerak sendiri sesuai dengan perintah dan mampu menyelesaikan suatu pekerjaan yang diberikan. Berdasarkan alat geraknya robot diklasifikasikan menjadi dua jenis yaitu robot beroda dan robot berkaki. Kekurangan robot beroda adalah gerakannya terbatas pada medan yang rata. Robot berkaki merupakan alternatif yang dapat dipilih karena fleksibilitasnya untuk bergerak pada medan yang tidak rata. Dalam pembuatan robot berkaki khususnya robot *quadruped*, hal mendasar yang

Received March 31th,2022; Revised April 12th, 2023; Accepted April 29th, 2023

perlu diperhatikan adalah masalah keseimbangan. Keseimbangan menentukan keberhasilan robot *quadruped* dalam melakukan gerakan seperti gerakan berjalan dan menstabilkan tubuhnya pada bidang miring. Oleh karena itu diperlukan sistem kendali yang dapat mengendalikan keseimbangan robot *quadruped*. Pada umumnya suatu sistem kendali membutuhkan model matematis untuk dapat mengendalikan suatu robot. Namun pemodelan yang melibatkan kontak antara robot dan *environment* sulit didekati dengan menggunakan metode kendali umpan balik konvensional [1], sehingga dibutuhkan pengembangan sistem kendali cerdas yang tidak membutuhkan model matematis.

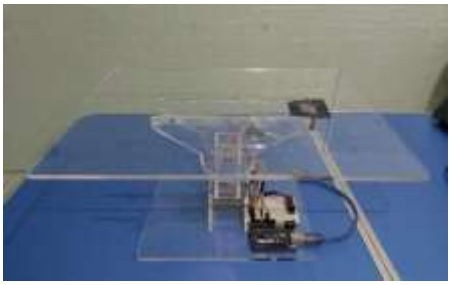
Selama ini penelitian mengenai keseimbangan robot *quadruped* masih terbatas dengan penggunaan metode konvensional seperti pemanfaatan kendali dengan pengontrol proporsional pada algoritma keseimbangan [2], kendali keseimbangan dengan pengontrol PID pada bidang miring [3], kendali koreksi tubuh robot *quadruped* menggunakan pengontrol PID [4], dan penggunaan logika fuzzy pada kendali keseimbangan robot *quadruped* dalam bidang miring [5]. Sementara itu penggunaan sistem kendali keseimbangan cerdas telah diterapkan dalam beberapa penelitian seperti sistem kendali keseimbangan pada robot *wheel-legged* dengan berbasis *learning* [6], sistem kendali keseimbangan pada robot beroda dua menggunakan *reinforcement learning* [7] serta menggunakan algoritma *Q-learning* [8], sistem kendali keseimbangan pada robot *ball-balancing* menggunakan *reinforcement learning* [9], dan sistem kendali keseimbangan pada robot *hexapod* dengan menggunakan *reinforcement learning* [10]. Berdasarkan beberapa penelitian tersebut, masih sedikit peneliti yang membahas mengenai sistem kendali keseimbangan cerdas pada robot *quadruped.* Oleh karena itu penelitian mengenai sistem kendali keseimbangan statis pada robot *quadruped* dengan menggunakan metode *reinforcement learning* merupakan salah satu hal yang menarik untuk diteliti.

Pada penelitian ini dibahas sistem kendali kendali keseimbangan statis robot *quadruped* menggunakan *reinforcement learning.* Algoritma yang digunakan pada proses pembelajaran robot *quadruped* menggunakan *reinforcement learning* ini adalah algoritma *Q-Learning.* Proses pembelajaran untuk membuat robot seimbang pada algoritma ini akan dilakukan pada simulator Gazebo. Proses pembelajaran ini didekati secara *end-to-end*, dimana mencoba membuat robot untuk belajar dengan sendiri supaya dapat seimbang.

2. METODE PENELITIAN

2.1 Perancangan Environment

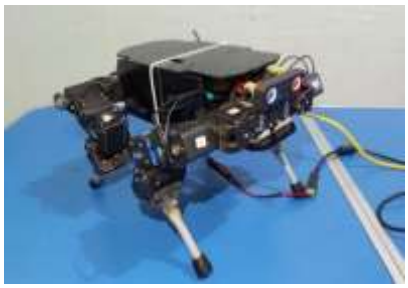
Environment merupakan tempat dimana robot *quadruped* berintaraksi dan melakukan pembelajaran. Pengimplementasian desain bidang *environment* dirancang menggunakan perangkat lunak Solidworks. Pada pembuatannya, digunakan kerangka akrilik sebagai rangka badan bidang *environment*. Bidang lantai yang berfungsi sebagai tempat robot *quadruped* memiliki panjang 40 cm dan lebar 30 cm. Setiap *joint* pada bidang *environment* ini dihubungkan dengan *bearing* sehingga dapat meminimalisir gesekan dari putaran rotasinya serta membuat putarannya berlangsung secara halus. Terdapat sensor IMU yang digunakan untuk mengukur kemiringan bidang *environment* dan *module* OLED *Display* 128x32 Pixel yang digunakan untuk menampilkan hasil bacaan sensor IMU. Kedua *module* tersebut diproses menggunakan mikrokontroller Arduino Uno. Implementasi model bidang *environment* ditunjukkan pada Gambar 1.



Gambar 1 Implementasi model bidang *environment*

2.2 Perancanagan Mekanik

Rangka mekanik robot *quadruped* menggunakan kerangka *frame quadruped* sebagai *link* untuk menyambungkan antar sendi yang dicetak dengan bahan PLA, rangka akrilik sebagai badan robot, serta pipa berbahan dasar *stainless steel* pada kaki. Rangka mekanik robot *quadruped* terdiri dari satu badan dan empat buah kaki. Kaki pada *robot quadruped* memiliki panjang lengan *coxa* 50 mm, panjang lengan *femur* 67 mm, dan panjang lengan *tibia* 120 mm. Rangka-rangka kaki tersebut dihubungkan dengan aktuator motor servo Dynamixel AX-12A sehingga posisi sudutnya dapat diatur. Pada bagian tengah badan robot *quadruped* terdapat sensor IMU yang digunakan untuk mengukur sudut kemiringan robot. Implementasi mekanik robot *quadruped* ditunjukkan pada Gambar 2.

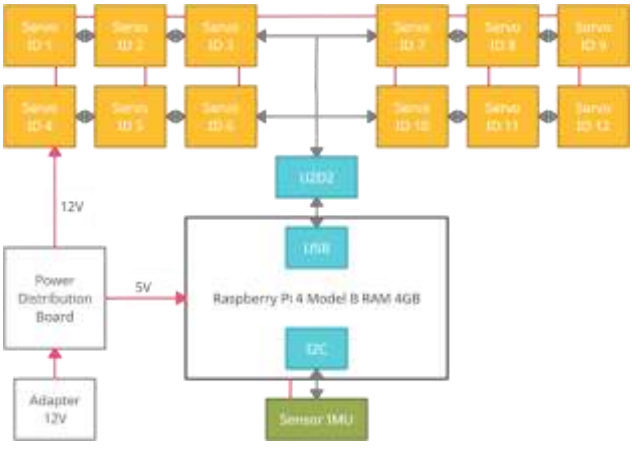


Gambar 2 Implementasi mekanik robot quadruped

Kaki robot yang berjumlah empat dirancang dengan orientasi kaki yang membentuk huruf "X" dengan tujuan untuk memusatkan titik CoM agar berada tepat di tengah badan robot, sehingga untuk pergerakan robot menjadi lebih stabil. Konfigurasi pada keempat kaki robot memiliki panjang lengan *coxa* 50 mm, panjang lengan *femur* 67 mm, dan panjang lengan *tibia* 120 mm dan diberi nomor mulai dari 0 sampai 3 dengan keterangan kaki 0 adalah kaki robot yang berada di sebelah kiri bagian depan, kaki 1 adalah kaki sebelah kanan bagian depan, kaki 2 adalah kaki sebelah kiri bagian belakang, dan kaki 3 adalah kaki sebelah kanan bagian belakang.

2.3 Perancanagan Elektronik

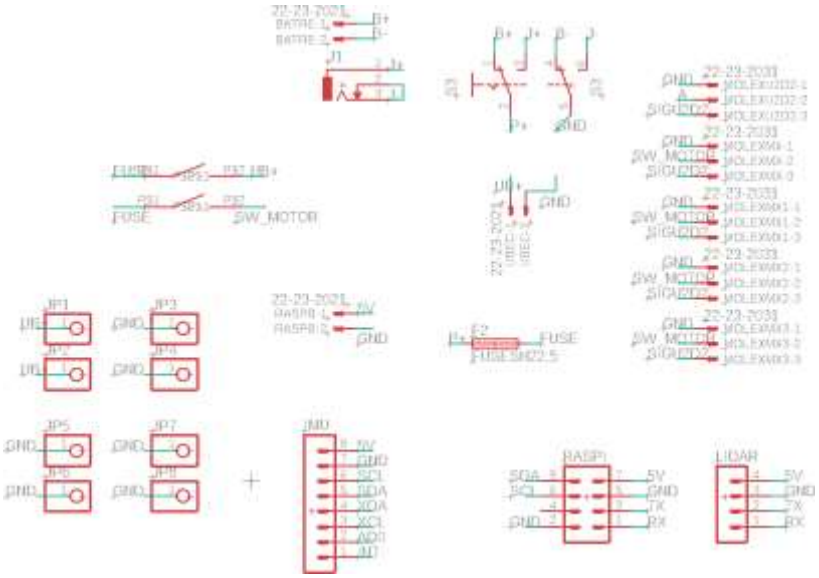
Kebutuhan sistem elektronik dalam penelitian robot *quadruped* ini dibuat menurut arsitektur sistem yang akan digunakan. Arsitektur sistem elektronik robot *quadruped* yang dirancang pada penelitian ini ditunjukkan pada Gambar 3.



Gambar 3 Arsitektur sistem elektronik robot *quadruped*

Kontroler yang digunakan dalam penelitian ini adalah Raspberry Pi Model B RAM 4 GB. Komunikasi antar perangkat dengan kontroler yang digunakan dilakukan menggunakan 2 jenis komunikasi, yaitu USB dan I2C. Terdapat 3 perangkat luar yang diakses yaitu servo, U2D2, dan sensor IMU. Servo Dynamixel AX-12A menggunakan komunikasi *half-duplex* sebagai masukan dalam mengontrol ke dua belas servo. Oleh karena itu, digunakan U2D2 untuk dapat mengubah

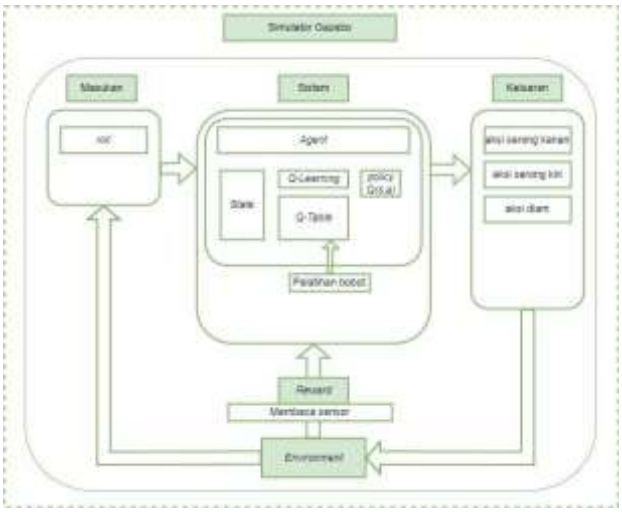
bentuk komunikasi menjadi *half-duplex.* Selanjutnya komunikasi kontroler dengan U2D2 dilakukan menggunakan USB. Sementara itu, sensor IMU diakses melalui komunikasi serial I2C. Rangkaian skematik elektronik yang dirancang dalam penelitian ini ditunjukkan pada Gambar 4.



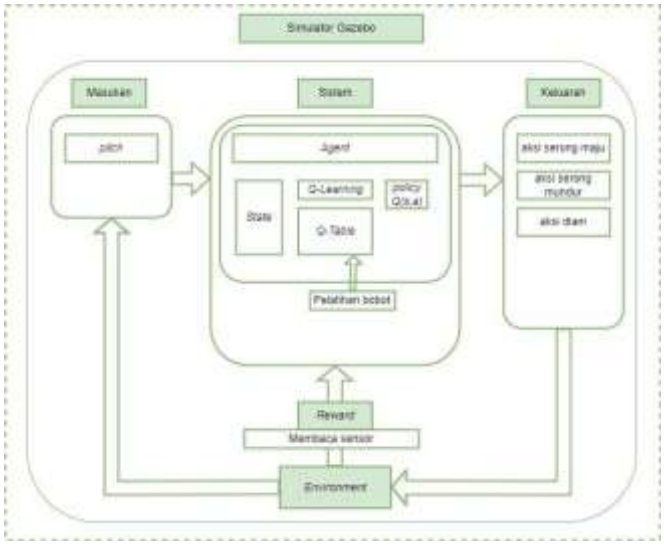
Gambar 4 Rangkaian skematik elektronik

2.4 Perancangan Sistem

Penelitian ini dilakukan dengan melakukan simulasi robot quadruped untuk menyeimbangkan posisinya pada permukaan dengan kemiringan yang berbeda-beda menggunakan metode reinforcement learning dengan algoritma Q-Learning. Simulator yang akan digunakan yaitu Gazebo dengan pengembangan program menggunakan bahasa pemrograman Python. Terdapat dua algoritma Q-Learning dalam penelitian yang dilakukan pada simulator Gazebo ini, yaitu algoritma untuk gerak roll dan gerak pitch. Pada masing- masing algoritma terdapat lima bagian utama, yaitu antara lain bagian masukan, keluaran, sistem, environment, dan reward. Masukan sistem untuk algoritma gerak roll adalah sudut roll sedangkan masukan sistem untuk algoritma gerak pitch adalah sudut pitch. Keluaran sistem pada algoritma gerak roll terdapat tiga yaitu aksi serong kiri, aksi serong kanan, dan aksi diam, sedangkan keluaran sistem pada algoritma gerak pitch yaitu aksi serong maju, aksi serong mundur, dan aksi diam. Adapun desain rancangan sistem penelitian untuk gerak roll dan gerak pitch terlihat seperti pada Gambar 5 dan



Gambar 5 Rancangan sistem pembelajaran gerak *roll*

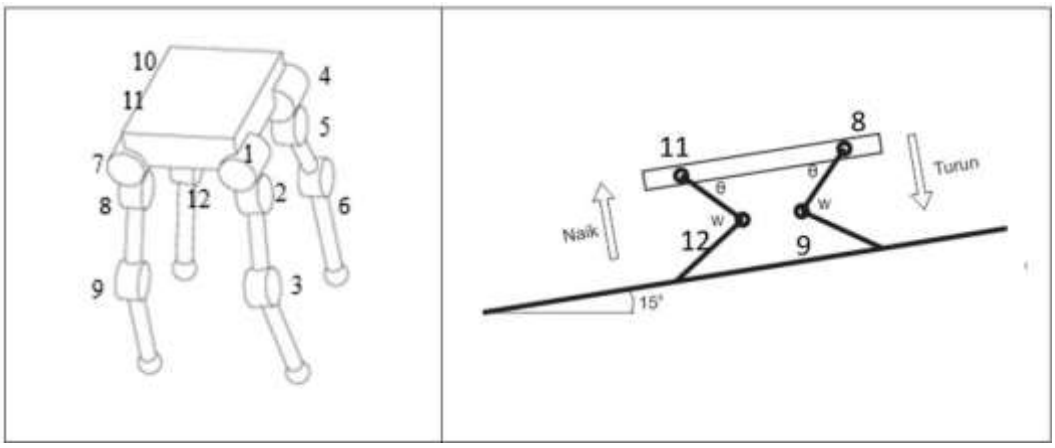


Gambar 6 Rancangan sistem pembelajaran gerak *pitch*

Dalam rancangan sistem pembelajaran gerak *roll* dan *pitch*, nilai *reward* dihitung sebagai bentuk umpan balik dari aksi yang dipilih. Semakin banyak aksi yang dilakukan secara akurat, maka semakin tinggi pula nilai *reward* yang akan diperoleh. Setelah memperoleh *reward* tahapan selanjutnya yaitu nilai *reward* akan diteruskan ke sistem, dimana model *Q-Learning* ini akan memproses semua masukan dan menghasilkan keluaran dalam bentuk distribusi probabilitas aksi. Aksi yang paling menguntungkan dilakukan oleh *quadruped* akan menghasilkan nilai probabilitas yang lebih tinggi dari aksi yang lain dalam satu state. *Environment* merupakan bagian yang menjadi tempat *quadruped* berdiri untuk menyeimbangkan posisinya.

2.4.1 Perancangan State

Rancangan *state* yang ini berisi kondisi mengenai keadaan yang penting bagi *agent* dalam menentukan aksi selanjutnya. *State* pada penelitian ini diperoleh dari sudut *roll* dan *pitch* yang didapat oleh sensor IMU robot pada simulator Gazebo saat melakukan penyeimbangan. Informasi *state* lainnya yang akan digunakan pada penelitian ini yaitu mengenai posisi sudut servo. Jumlah *joint* yang digunakan pada robot quadruped adalah dua belas buah, apabila semua posisi sudut motor servo pada setiap *joint* ditambahkan sebagai informasi *state* maka akan mengakibatkan *Q-table* menjadi sangat besar. Oleh karena itu perlu dilakukan beberapa kombinasi pada gerakan joint yang geraknya searah atau berlawanan. Visualisasi kombinasi gerak joint ditunjukkan pada Gambar 7.



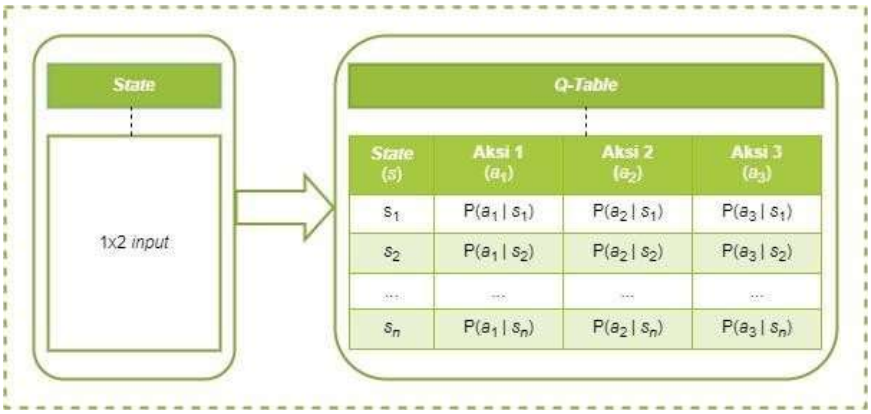
Gambar 7 Visualisasi kombinasi gerak joint

2.4.2 Preprocessing

Informasi sudut *roll* dan *pitch* yang diperoleh dari sensor IMU yang berada pada robot *quadruped* saat melakukan penyeimbangan akan diolah terlebih dahulu untuk menjadi informasi *state* yang akan digunakan. Proses pengolahan sudut yang diperoleh pada saat penyeimbangan *quadruped* ini dikenal dengan istilah *preprocessing*. Pada tahap ini nilai sudut yang diperoleh *quadruped* pada saat penyeimbangan perlu dilakukan proses diskritisasi. Proses diskritisasi dilakukan dengan memetakan data sudut hasil filter ke dalam himpunan yang lebih kecil dengan membulatkan nilainya. Hal ini dilakukan untuk memudahkan proses komputasi yang terjadi. Proses diskritisasi juga dilakukan pada informasi posisi sudut servo. Pada sudut *roll* dan *pitch* nilainya dipetakan kedalam enam puluh data, sedangkan pada posisi sudut servo nilainya dipetakan kedalam 188 data.

2.4.3 Perancangan Model Q-Table

Pada penelitian ini *Q-table* berfungsi untuk memetakan *state* dengan aksi. Jumlah baris pada *Q-table* disesuaikan dengan banyak kombinasi dari *state*, sedangkan kolomnya berjumlah banyaknya aksi ditambah satu. Masukan dari *Q-table* ini merupakan *state* hasil penggabungan dari informasi sudut *pitch/rol*l dengan posisi sudut servo. Skema rancangan model *Q-table* yang akan dilakukan pada penelitian ini akan ditunjukkan seperti pada Gambar 8 dimana inputnya adalah matriks baris 1x2 yang dihasilkan oleh robot *quadruped* pada saat melakukan penyeimbangan dan telah dilakukan *preprocessing*. Setiap *state* dalam *Q-table* memetakan tiga buah aksi yang berisi *Q-value* yang merupakan probabilitas dari aksi yang akan dipilih. Keluaran dari *Q-table* ini adalah pemilihan aksi yang memiliki probabilitas terbesar dalam suatu *state*.



Gambar 8 Skema rancangan model *Q-table*

2.4.4 Perancangan Pembelajaran

Percobaan pembelajaran pada penelitian ini akan dilakukan pada simulator Gazebo yang merupakan *platform open-source* yang dibuat dengan tujuan untuk mempersempit kesenjangan antara simulasi dan kenyataan. Pembelajaran dibuat dengan menggunakan teknik *experience* untuk melatih *Q-table*. Dimana dengan teknik tersebut robot *quadruped* akan mencoba berbagai macam gerakan pada berbagai kondisi sehingga memiliki banyak *experience* yang digunakan sebagai dataset pembelajaran (*training*). Pembelajaran dilakukan dengan jumlah minimal sepuluh ribu episode dimana setiap episode terdiri dari tiga ratus langkah. Setiap langkah dalam episode pembelajaran, sistem akan menghitung kesalahan model dan memperbarui nilai koefisien model. Kemudian untuk kebijakan tingkah laku selama fase pembelajaran digunakan *ε-greedy* untuk melakukan eksplorasi. Arsitektur model *Q-table* yang sama digunakan dengan menggunakan algoritma pembelajaran dan pengaturan *hyperparameter* seperti pada Tabel 1. Dalam pembelajaran pada penelitian ini akan dilakukan evaluasi terhadap nilai *reward* yang diperoleh. Adapun daftar *hyperparameter* dan nilai yang digunakan pada penelitian ini tertulis seperti pada Tabel 1.

Tabel 1 Daftar *hyperparameter* dan nilainya

Hyperparameter	Nilai	Deskripsi
Discount factor	0.95	*Discount factor* akan digunakan dalam memperbarui *Q-learning.*
Initial exploration	1	Nilai awal dari ε dalam eksplorasi ε-*greedy.*
Final exploration	0.1	Nilai akhir dari ε dalam eksplorasi ε-*greedy.*
Start exploration decay	4000	Episode dimulainya eksplorasi ε-*greedy.*
Learning rate	0.1	*Learning rate* digunakan oleh *Q-learning.*

2.4.5 Perancangan Algoritma

Algoritma yang akan digunakan untuk penelitian ini guna melatih *Q-table* yaitu menggunakan algoritma *Q-learning*. Pada algoritma ini, *agent* akan memilih dan mengeksekusi aksi sesuai dengan kebijakan ε-*greedy* yang didasarkan pada nilai *Q*. Pada tahap awal pembelajaran dilakukan inisialisasi *Q-table Q(s, a)* dengan menggunakan bobot acak. Pembelajaran robot *quadruped* untuk menyeimbangkan posisinya pada bidang yang berubah kemiringannya akan dilakukan dalam M episode. Setiap awal episode akan dilakukan inisialisasi posisi *environment* dengan sudut acak. Pada tahap selanjutnya akan dilakukan inisialisasi *state* s1 dan dilakukan *preprocessing* $\phi 1 = \phi(s1)$ untuk mempermudah proses komputasi.

Setiap satu episode akan dilakukan perulangan sebanyak T langkah. Pada perulangan ini, *agent* dapat memilih aksi acak atau memilih aksi $argmaxaQ(\phi(s1),a)$ dari *Q-table* sesuai dengan kebijakan ε-*greedy*. Setelah mengeksekusi aksi, akan didapatkan *reward* r_t dan *next state* s_{t+1}. Pada tahap selanjutnya akan dilakukan *preprocessing* pada *next state*. Apabila langkah yang dilakukan oleh *agent* belum mencapai episode terminal, maka nilai *action-value function* pada s_t dan a_t akan diperbarui. Namun apabila langkah yang dilakukan oleh *agent* sudah mencapai episode terminal, maka hasil *reward* yang didapat pada episode tersebut akan diakumulasi dan proses tersebut akan diulangi lagi sampai episode berakhir. Adapun algoritma dari metode *Q-learning* dapat dilihat seperti pada Gambar 9.

Algoritma 1 : *Q-Learning*

Inisialisasi *Q-table* $Q(s,a)$ dengan bobot acak
For episode=1, M do
  Inisialisasi posisi *environment* secara acak
  Inisialisasi *state* s_1 dan *preprocessing* $\phi_1 = \phi(s_1)$
  For t=1, T do
    Dengan probabilitas ε memilih aksi acak (a_t)
    jika tidak, memilih $a_t = argmax_a Q(\phi(s_1), a)$
    Eksekusi aksi a_t dan observasi *reward* r_t dan *next state* s_{t+1}
    Melakukan *preprocessing* $\phi_{t+1} = \phi(s_{t+1})$
    Jika episode terminal, menghitung episode *reward*
    Jika tidak, memperbarui *action-value function* pada s_t dan a_t
    Memperbarui *next state* menjadi *current state* ($s_t = s_{t+1}$)
  End For
End For

Gambar 9 Algoritma *Q-Learning*

2.4.6 Perancangan Aksi

Pada penelitian ini dirancang jenis aksi yang akan dilakukan oleh robot *quadruped* dalam melakukan penyeimbangannya. Aksi merupakan suatu bentuk interaksi yang terjadi antara *agent*

dan *environment*. Aksi-aksi yang diberikan ini akan didasarkan pada posisi sudut robot *quadruped* dalam sumbu x dan y. Terdapat dua jenis aksi yang dirancang pada penelitian ini, yaitu aksi pada gerak *pitch* dan aksi pada gerak *roll*. Jenis aksi yang diberikan pada gerak *pitch* adalah aksi diam, aksi serong maju, dan aksi serong mundur. Sedangkan jenis aksi yang diberikan pada gerak *roll* adalah aksi diam, aksi serong kanan, dan aksi serong kiri. Adapun daftar jenis aksi yang digunakan pada penelitian ini tertulis seperti pada Tabel 2.

Tabel 2 Daftar jenis aksi

Tipe	Aksi 1	Aksi 2	Aksi 3
Gerak *pitch*	Aksi diam	Aksi serong maju	Aksi serong mundur
Gerak *roll*	Aksi diam	Aksi serong kanan	Aksi serong kiri

2.4.7 Perancangan Reward

Reward merupakan salah satu elemen penting yang ada dan diperlukan pada metode *reinforcement learning*. *Reward* yang dibuat akan dijadikan sebagai parameter atau indikator dalam menentukan aksi yang baik atau tepat dan aksi yang kurang tepat bagi *agent*. Terdapat dua jenis *reward* yang dirancang pada penelitian ini, yaitu *reward* gerak *pitch* dan *reward* gerak *roll*. Pada *reward* gerak *pitch*, nilai *reward* sebesar 10 poin akan diberikan kepada *agent* apabila posisi sudut *pitch* yang dibaca oleh sensor IMU berada dalam rentang nilai $-1 \leq \phi \leq 1$. Hal yang sama juga diterapkan pada *reward* gerak *roll*, nilai *reward* sebesar 10 poin akan diberikan kepada *agent* apabila posisi sudut *roll* yang dibaca oleh sensor IMU berada dalam rentang nilai $-1 \leq \theta \leq 1$. Perancangan *reward* gerak *pitch* dapat dituliskan dalam Persamaan (1) sedangkan perancangan *reward* gerak *roll* dapat dituliskan dalam Persamaan (2).

$$r_{t\ pitch} = \begin{cases} 10, & if\ (-1 \leq \phi \leq 1) \\ 0, & otherwise \end{cases} \quad (1)$$

$$r_{t\ roll} = \begin{cases} 10, & if\ (-1 \leq \theta \leq 1) \\ 0, & otherwise \end{cases} \quad (2)$$

dimana:

ϕ : sudut *pitch*
θ : sudut *roll*
$r_{t\ pitch}$: *reward* gerak *pitch*
$r_{t\ roll}$: *reward* gerak *roll*

3. HASIL DAN PEMBAHASAN

Pada bagian ini dijelaskan hasil implementasi sistem kendali keseimbangan statis robot *quadruped* pada dunia nyata dengan menggunakan penerapan metode *reinforcement learning*. Implementasi ini merupakan penerapan sistem kendali yang sebelumnya telah dilakukan pada simulator Gazebo. Implementasi ini dilakukan untuk mengetahui sistem kendali yang sebelumnya dibuat dengan menggunakan *reinforcement learning* dapat diterapkan pada robot secara langsung atau tidak. Oleh karena itu dilakukan beberapa pengujian pada perangkat keras robot yaitu pengujian sistem kendali keseimbangan statis pada sumbu *roll*, sumbu *pitch*, dan pada sumbu *roll* dan *pitch*.

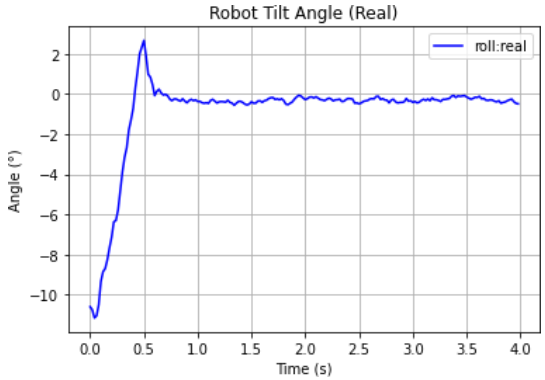
3.1 Hasil Pengujian Sistem pada Sumbu Roll

Hasil pengujian sistem kendali keseimbangan statis ini dilakukan berdasarkan proses pembelajaran pada sumbu *roll* dengan penerapan algoritma *Q-learning*. Pengujian ini dilakukan dengan menempatkan robot *quadruped* diatas bidang *environment* yang diberi kemiringan pada sumbu x. Kemudian robot *quadruped* akan mencoba untuk menyeimbangkan posisinya. Aksi

yang dapat dilakukan oleh robot *quadruped* adalah aksi gerak *roll*, yaitu aksi diam, aksi serong kanan, dan aksi serong kiri. Posisi awal robot dan bidang *environment* ketika akan melakukan penyeimbangan dengan sistem kendali keseimbangan statis untuk sumbu *roll* pada dunia nyata ditunjukkan pada Gambar 10a dan posisi akhir robot setelah melakukan penyeimbangan ditunjukkan pada Gambar 10b.

Gambar 10. a. Posisi awal robot dan bidang *environment* ketika akan melakukan penyeimbangan sumbu *roll*. b. Robot selesai melakukan penyeimbangan untuk sumbu *roll*

Kedua gambar tersebut menunjukkan bahwa sistem kendali keseimbangan statis menggunakan reinforcement learning dapat diimplementasikan pada perangkat keras robot quadruped untuk menyeimbangkan posisinya pada sumbu x atau roll. Pada awalnya bidang environment diberi kemiringan pada sumbu x terlebih dahulu. Grafik hasil pengujian sistem kendali keseimbangan statis untuk sumbu roll ditunjukkan pada Gambar 11.



Gambar 11. Grafik hasil pengujian sistem kendali keseimbangan statis untuk sumbu *roll*

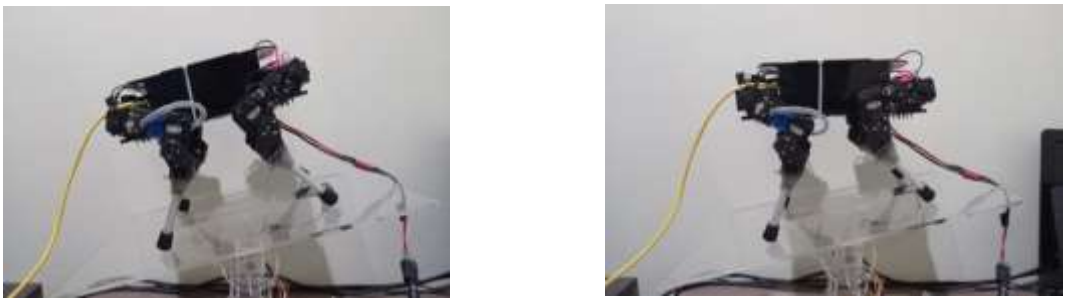
Pada pengujian sistem kendali keseimbangan statis sumbu *roll* yang ditunjukkan oleh Gambar 11, *risetime* sistem ketika terjadi gangguan sebesar -10,59° pada sumbu *roll* adalah 0,40 detik. Kemudian waktu *settling time* setelah terjadi gangguan adalah 0,7 detik. Nilai *sum reward* yang didapat dari penjumlahan *reward* pada episode pengujian ini adalah 2750 poin. Pada pengujian ini terdapat *overshoot* sebesar 2,67° yang teramati pada respon yang diberikan oleh sistem. Hasil dari beberapa kali pengujian sistem kendali keseimbangan statis untuk sumbu *roll* disajikan pada Tabel 3.

Tabel 3. Hasil pengujian sistem kendali keseimbangan statis untuk sumbu *roll*

No	Sudut *roll* awal (derajat)	Sudut *roll* akhir (derajat)	*Risetime* (detik)	*Overshoot* (derajat)	*Sum Reward*
1	-14,72	1,02	0,56	2,28	1670
2	-10,59	0,28	0,40	2,66	2750
3	-5,83	-0,28	0,22	2,62	1480
4	5,77	-2,01	0,26	4,01	1560
5	10,64	0,17	0,44	4,18	2700
6	13,30	-1,97	0,46	2,14	1150

3.2 Hasil Pengujian Sistem pada Sumbu Pitch

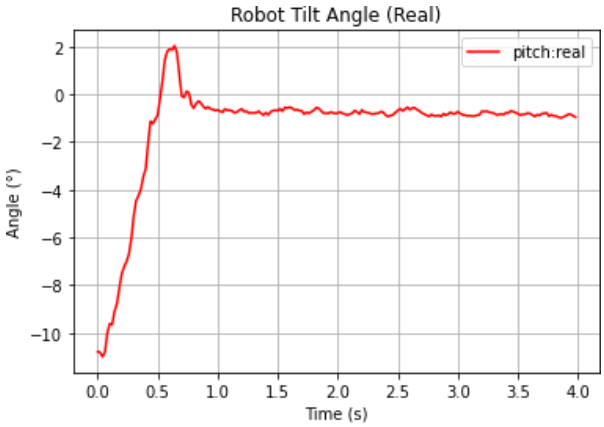
Hasil pengujian sistem kendali keseimbangan statis ini dilakukan berdasarkan proses pembelajaran pada sumbu pitch dengan penerapan algoritma Q-learning. Pengujian ini dilakukan dengan menempatkan robot quadruped diatas bidang environment yang diberi kemiringan pada sumbu y. Kemudian robot quadruped akan mencoba untuk menyeimbangkan posisinya. Aksi yang dapat dilakukan oleh robot quadruped adalah aksi gerak pitch, yaitu aksi diam, aksi serong maju, dan aksi serong mundur. Posisi awal robot dan bidang environment ketika akan melakukan penyeimbangan dengan sistem kendali keseimbangan statis untuk sumbu pitch pada dunia nyata ditunjukkan pada Gambar 12a dan posisi akhir robot setelah melakukan penyeimbangan ditunjukkan pada Gambar 12b.



a b

Gambar 12. a. Posisi awal robot dan bidang environment ketika akan melakukan penyeimbangan sumbu pitch. b. Robot selesai melakukan penyeimbangan untuk sumbu pitch

Kedua gambar tersebut menunjukkan bahwa sistem kendali keseimbangan statis menggunakan *reinforcement learning* dapat diimplementasikan pada perangkat keras robot *quadruped* untuk menyeimbangkan posisinya pada sumbu y atau *pitch*. Seperti pada simulasi, pada awalnya bidang *environment* diberi kemiringan pada sumbu y terlebih dahulu. Grafik hasil pengujian sistem kendali keseimbangan statis untuk sumbu *pitch* pada dunia nyata ditunjukkan pada Gambar 13.



Gambar 13. Grafik hasil pengujian sistem kendali keseimbangan statis untuk sumbu *pitch*

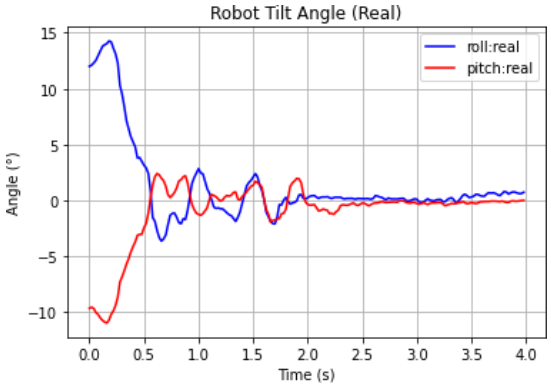
Pada pengujian sistem kendali keseimbangan statis sumbu *pitch* yang ditunjukkan oleh Gambar 13, *risetime* sistem ketika terjadi gangguan sebesar -10,79° pada sumbu *pitch* adalah 0,50 detik. Kemudian waktu *settling time* setelah terjadi gangguan adalah 0,8 detik. Nilai *sum reward* yang didapat pada pengujian di dunia nyata ini adalah 2260 poin. Pada pengujian ini terdapat *overshoot* sebesar 2,04° yang teramati pada respon yang diberikan oleh sistem. Hasil dari beberapa kali pengujian sistem kendali keseimbangan statis untuk sumbu *roll* disajikan pada Tabel 4.

Tabel 4. Hasil pengujian sistem kendali keseimbangan statis untuk sumbu *pitch*

No	Sudut *pitch* awal (derajat)	Sudut *pitch* akhir (derajat)	*Risetime* (detik)	*Overshoot* (derajat)	*Sum Reward*
1	-13,83	0,85	0,58	1,70	2170
2	-10,79	-0,54	0,50	2,04	2260
3	-6,17	-0,11	0,28	2,13	2620
4	5,92	-1,14	0,28	2,15	1810
5	10,01	-0,15	0,48	2,62	2250
6	15,34	0,84	0,70	2,29	1420

3.3 Hasil Pengujian Sistem pada Sumbu Roll dan Pitch

Pada bagian ini dijelaskan hasil dari proses pengujian sistem kendali keseimbangan statis robot *quadruped* pada sumbu *roll* dan *pitch* menggunakan penerapan algoritma *Q- learning.* Seperti pada proses pengujian sebelumnya, pengujian ini dilakukan dengan menempatkan robot *quadruped* diatas bidang *environment.* Pada tahap ini, orientasi pada bidang *environment* memiliki kemiringan pada sumbu x dan sumbu y. Misi robot *quadruped* pada pengujian ini adalah untuk menyeimbangkan posisinya terhadap sumbu x dan sumbu y. Aksi yang dapat dilakukan oleh robot *quadruped* adalah aksi gerak *roll*, yaitu aksi diam, aksi serong kanan, dan aksi serong kiri, serta aksi gerak *pitch*, yaitu aksi diam, aksi serong maju, dan aksi serong mundur. Pengguanaan aksi gerak *roll* dan aksi gerak *pitch* secara bersamaan robot pada *quadruped* menyebabkan pergerakan robot dapat bebas kedalam dua arah. Grafik hasil pengujian sistem kendali keseimbangan statis sumbu *roll* dan *pitch* ditunjukkan pada Gambar 14.



Gambar 14. Grafik hasil pengujian sistem kendali keseimbangan statis untuk sumbu *roll* dan *pitch*

Respon sistem saat terjadi gangguan sebesar 11,99° pada sumbu *roll* dan -9,66° pada sumbu *pitch* yang dilakukan pada dunia nyata dapat dilihat pada Gambar 6.18. Nilai *risetime* ketika terjadi gangguan tersebut adalah 0,56 detik baik untuk sumbu *roll* maupun untuk sumbu *pitch. Overshoot* yang terjadi baik pada sumbu *roll* maupun pada sumbu *pitch* memiliki nilai yang lebih besar dari pengujian yang dilakukan pada simulator Gazebo. Nilai *overshoot* yang terjadi setelah terjadi gangguan pada sumbu *roll* adalah 3,64°. Sementara nilai *overshoot* yang terjadi setelah terjadi gangguan pada sumbu *pitch* adalah 2,41°. Waktu *settling time* pada sumbu *roll* dan sumbu *pitch* bernilai sama yaitu 2 detik. Nilai *sum reward* yang didapat dari penjumlahan *reward* dalam episode pengujian ini adalah 2280 poin untuk gerak *roll*, sedangkan nilai *sum reward* yang didapat pada gerak *pitch* adalah 2300 poin dengan *total sum reward* sebesar 4580 poin.

4. KESIMPULAN

Sistem kendali keseimbangan statis pada robot *quadruped* menggunakan *reinforcement learning* dengan penerapan algoritma *Q-Learning* telah diimplementasikan dan masing-masing

komponen robot dapat bekerja sesuai fungsinya. Kemudian setelah dilakukan proses pembelajaran pada simulator Gazebo, model hasil pembelajaran dapat diterapkan pada robot *quadruped* secara langsung. Robot *quadruped* ini dapat menyelesaikan tugasnya yaitu menyeimbangkan posisi tubuhnya pada sumbu *roll* dan *pitch.*

Kerja robot pada saat melakukan penyeimbangan pada kemiringan yang berbeda-beda dapat menjadi rekomendasi untuk kemudian dapat dibuat robot yang dapat difungsikan untuk membantu pekerjaan manusia.

DAFTAR PUSTAKA

[1] T. Johannink *et al.*, "Residual Reinforcement Learning for Robot Control," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029, Dec. 2018, [Online]. Available: http://arxiv.org/abs/1812.03201

[2] R. C. Prayogo, A. Triwiyatno, and Sumardi, "Quadruped Robot with Stabilization Algorithm on Uneven Floor using 6 DOF IMU based Inverse Kinematic," *2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, pp. 39–44, 2018.

[3] Z. Nasution, A. F. I. Suparman, G. A. Prasetyo, A. H. Alasiry, E. H. Binugroho, and A. Darmawan, "Body Balancing Control for EILERO Quadruped Robot while Walking on Slope," *2019 International Electronics Symposium (IES)*, pp. 364–369, 2019.

[4] C. Yu, L. Zhou, H. Qian, and Y. Xu, "Posture Correction of Quadruped Robot for Adaptive Slope Walking," *Proceedings of the 2018 IEEE International Conference on Robotics and Biomimetics, Kuala Lumpur, Malaysia*, pp. 1220–1225, Dec. 2018.

[5] A. T. B. Antok *et al.*, "Quadruped Robot Balance Control for Stair Climbing Based on Fuzzy Logic," in *International Electronics Symposium 2021: Wireless Technologies and Intelligent Systems for Better Human Lives, IES 2021 - Proceedings*, Sep. 2021, pp. 552– 557. doi: 10.1109/IES53407.2021.9594046.

[6] L. Cui *et al.*, "Learning-Based Balance Control of Wheel-Legged Robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7667–7674, Oct. 2021, doi: 10.1109/LRA.2021.3100269.

[7] E. Li *et al.*, "Model Learning for Two-Wheeled Robot Self-Balance Control," *Proceeding of the IEEE International Conference on Robotics and Biomimetics Dali, China, December 2019*, pp. 1582–1587, 2019.

[8] L. Guo, S. A. A. Rizvi, and Z. Lin, "Optimal Control of a Two-Wheeled Self-Balancing Robot by Reinforcement Q-learning," in *IEEE International Conference on Control and Automation, ICCA*, Oct. 2020, vol. 2020-October, pp. 955–960. doi: 10.1109/ICCA51439.2020.9264485.

[9] Y. Zhou, J. Lin, S. Wang, and C. Zhang, "Learning Ball-Balancing Robot through Deep Reinforcement Learning," in *2021 International Conference on Computer, Control and Robotics, ICCCR 2021*, Jan. 2021, pp. 1–8. doi: 10.1109/ICCCR49711.2021.9349369.

[10] B. Qin, Y. Gao, and Y. Bai, "Sim-to-real: Six-legged Robot Control with Deep Reinforcement Learning and Curriculum Learning," in *2019 4th International Conference on Robotics and Automation Engineering, ICRAE 2019*, Nov. 2019, pp. 1–5. doi: 10.1109/ICRAE48301.2019.9043822.