PERANCANGAN DAN PENGENDALIAN DIFFERENTIAL DRIVE ROBOT DENGAN MENGAPLIKASIKAN METODE A-STAR

Ghanis Kauchya Nugraha[*)], Iwan Setiawan dan Hadha Afrisal

Program Studi Sarjana Departemen Teknik Elektro, Universitas Diponegoro
Jl. Prof. Sudharto, SH, Kampus UNDIP Tembalang, Semarang 50275, Indonesia

[*)]*E-mail: ghanis@students.undip.ac.id*

Abstrak

Pandemi COVID-19 menyerang seluruh penduduk dunia. Pandemi COVID-19 juga menyebabkan banyaknya kematian tenaga medis di Indonesia dan dunia. Solusi yang dapat diterapkan untuk mengurangi jumlah tenaga medis yang terpapar virus SARS-CoV-2 adalah dengan menerapkan robot yang dapat membantu tenaga medis dalam bekerja. Robot harus bisa bergerak dari tempat satu ke tempat yang lain dan bisa membawa perlengkapan yang diperlukan seperti obat-obatan, alat medis, dll. Robot *differential drive* menggunakan 2 roda yang digerakkan oleh penggerak terpisah sebagai penggerak utama. Mobile robot membutuhkan *path planning* yang optimal dalam melakukan navigasi dari suatu koordinat ke koordinat lain. Dalam tugas akhir ini penulis merancang sistem navigasi robot tipe *differential drive* dengan menerapkan algoritma A*. Pada semua pengujian, robot dapat menemukan rute dari titik awal menuju ke titik target yang aman untuk dilalui robot dan robot dapat bernavigasi melewati rute yang sudah direncanakan tersebut hingga selesai.

Kata kunci: navigasi, path planning, A star, mobile robot

Abstract

The COVID-19 pandemic has affected the entire population of the world. The COVID-19 pandemic has also caused many medical personnel deaths in Indonesia and the world. The solution that can be applied to reduce the number of medical personnel exposed to the SARS-CoV-2 virus is to implement robots that can assist medical personnel in their work. Robots must be able to move from one place to another and be able to carry the necessary equipment such as medicines, medical devices, etc. The differential drive robot uses 2 wheels driven by a separate drive as the main mover. Mobile robots require optimal path planning in navigating from one coordinate to another. In this final project, the author designs a differential drive type robot navigation system by applying the A algorithm. In all tests, the robot can find a route from the starting point to the target point that is safe for the robot to go through and the robot can navigate through the planned route until it is finished.*

Keywords: navigation, path planning, A star, mobile robot

1. Pendahuluan

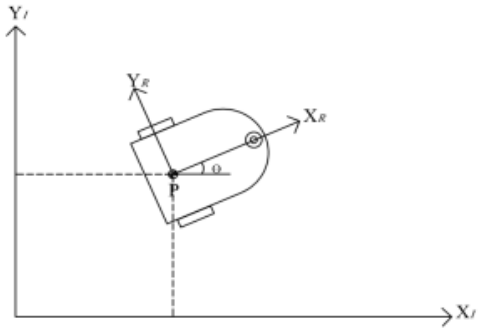
Dewasa ini pandemi COVID-19 menyerang seluruh penduduk dunia. Pandemi COVID-19 menyebabkan banyaknya kematian tenaga medis di Indonesia. Pekerjaan yang intens pada beberapa tenaga medis dapat memicu terjadinya gangguan emosional, stres, insomnia, dan kelelahan emosional [1]. Solusi yang dapat diterapkan untuk mengurangi tenaga medis yang terpapar virus SARS-CoV-2 adalah dengan menerapkan robot yang dapat membantu tenaga medis dalam bekerja. Robot dapat digunakan untuk menggantikan pekerjaan tenaga medis seperti membawa alat-alat medis, obat-obatan, dan barang-barang lain yang perlu diantarkan ke ruangan pasien. Untuk membuat robot yang bisa membantu tenaga medis maka robot harus bisa bergerak dari tempat satu ke tempat yang lain dan bisa membawa perlengkapan yang diperlukan tersebut. Mobile robot memiliki beberapa tipe, salah satunya adalah tipe *differential drive*. Mobile robot tipe ini menggunakan 2 roda yang digerakkan oleh penggerak terpisah sebagai penggerak utama [2]. Mobile robot membutuhkan *path planning* dalam melakukan navigasi dari suatu koordinat ke koordinat lain [3]. *Path planning* merupakan bagian dari sistem navigasi robot yang bertujuan untuk memilih dan mengidentifikasi rute yang cocok untuk dilalui robot di dalam area ruang kerja [4]. Contoh dari metode yang bisa digunakan sebagai *path planning* adalah metode Dijkstra dan A* (A-Star). Dalam penelitian yang sudah dilakukan oleh Prasetyo et al [5], mereka membandingkan efisiensi antara algoritma A* dan Dijkstra apabila diterapkan untuk mencari rute terdekat. Disimpulkan bahwa algoritma A* lebih baik, efisien, dan stabil dibandingkan dengan algoritma Dijkstra. Algoritma A* mirip seperti algoritma *Best First Search* (BFS) tetapi telah dimodifikasi menggunakan fungsi heuristis [6]. Berdasarkan permasalahan di atas, dalam tugas akhir ini

penulis akan merancang sistem navigasi robot tipe *differential drive* dengan menerapkan algoritma A*.

2. Metode

2.1. Model Kinematika Robot

Pada robot model *differential drive*, roda 1 dan 2 dapat bergerak dengan kecepatan yang berbeda [5]. Model kinematika dari *differential drive mobile robot* mendeskripsikan hubungan antara kecepatan roda dengan kecepatan robot [7]. Perhitungan kinematika robot dilakukan oleh mikrokontroler ESP32. Board tipe ini menggunakan modul ESP-WROOM-32 sebagai mikroprosesornya dan dilengkapi dengan fitur WiFi dan Bluetooth Low Energi (BLE) [8].



Gambar 1. Representasi posisi robot pada sistem koordinat kartesius

Posisi robot dalam sistem koordinat kartesius kerangka acuan global adalah P[x y θ]. Kecepatan linear dan kecepatan sudut robot dapat dilambangkan dengan v dan ω. Maka dapat kita tuliskan hubungan antara sistem kerangka robot terhadap global adalah sebagai berikut:

$$\dot{x} = v.\cos\theta \quad (1)$$

$$\dot{y} = v.\cos\theta \quad (2)$$

$$\dot{\theta} = \omega \quad (3)$$

Hubungan kecepatan tiap roda terhadap kecepatan robot dalam kerangka acuan robot adalah sebagai berikut:

$$v = \frac{\omega_r.R_r + \omega_l.R_l}{2} \quad (4)$$

$$\omega = \frac{\omega_r.R_r - \omega_l.R_l}{D} \quad (5)$$

Parameter R_r dan R_l secara berurutan adalah jari-jari dari roda kanan dan kiri, dan D adalah jarak antara roda kanan dan kiri. Dapat kita tuliskan hubungan antara sistem kerangka global terhadap sistem kerangka robot adalah sebagai berikut:

$$v = \sqrt{\dot{x}^2 + \dot{y}^2} \quad (6)$$

$$\omega = \dot{\theta} \quad (7)$$

Hubungan antara kecepatan robot dalam kerangka acuan robot terhadap kecepatan tiap roda dinyatakan dalam persamaan berikut:

$$\omega_r = \frac{\left(v + \frac{\omega.D}{2}\right)}{R_r} \quad (8)$$

$$\omega_l = \frac{\left(v - \frac{\omega.D}{2}\right)}{R_l} \quad (9)$$

Parameter ω_r, ω_l, R_r, R_l, dan D secara berurutan adalah kecepatan sudut roda kanan, kecepatan sudut roda kiri, jari-jari roda kanan, jari-jari roda kiri, dan jarak antara roda kanan dan kiri. Dengan memvariasikan kecepatan kedua roda, kita dapat memvariasikan lintasan yang diambil robot [9].

2.2. Algoritma A*

A* merupakan algoritma pencarian yang banyak diaplikasikan sebagai metode *path planning* robot. Peter Hart, Nils Nilsson, dan Bertram Raphael dari *Stanford Research Institute* pertama kali menerbitkan algoritma ini pada tahun 1968 [10]. Hingga saat ini telah berkembang banyak variasi dari algoritma A* [11]. Fungsi heuristik memiliki peran yang sangat penting untuk mengontrol pencarian pada algoritma A*, sehingga algoritma ini akan menemukan rute yang optimal [12]. *Cost function* yang digunakan adalah:

$$f(n) = g(n) + h(n) \quad (10)$$

Dimana, f(n) adalah *cost function*, g(n) adalah jarak dari titik awal ke posisi robot saat ini, dan h(n) adalah jarak dari posisi robot saat ini ke titik target. Persamaan jarak *euclidean* digunakan untuk menghitung jarak antara 2 titik:

$$h(n) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (11)$$

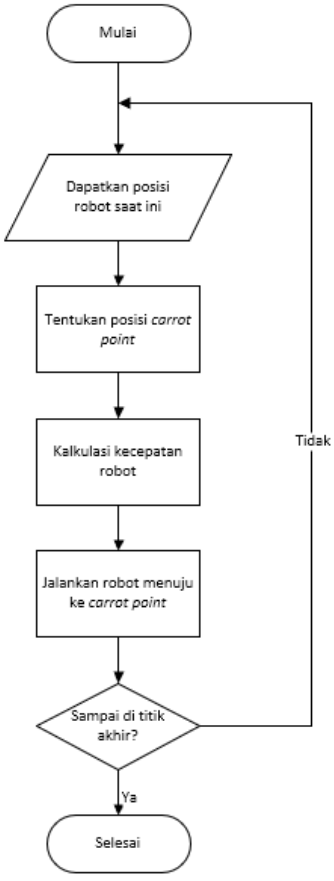
Dengan (x1, y1) adalah koordinat pertama dan (x2, y2) adalah koordinat kedua. *Pseudocode* dari algoritma A* dapat dilihat pada Gambar 2.

```
OPEN        // daftar semua node yang akan dievaluasi
CLOSED      // daftar semua node sudah dievaluasi
current     // node yang sedang dievaluasi
f_cost      // nilai cost function
successor   // node di sekitar current

tambahkan node awal ke OPEN
perulangan:
  current = node di dalam OPEN yang memiliki f_cost terkecil
  hapus current dari OPEN
  masukkan current ke CLOSED

  jika current adalah node target:
    selesai

  untuk setiap successor dari current node:
    jika successor tidak dapat dilalui atau di dalam CLOSED:
      lewati menuju ke successor selanjutnya

    jika jarak ke successor lebih pendek atau di tidak dalam OPEN:
      hitung f_cost dari successor
      set parent dari successor ke current
      jika successor tidak dalam OPEN
        masukkan successor ke OPEN
```

Gambar 2. *Pseudocode* algoritma A*

2.3. Algoritma Path Tracking

Path tracking merupakan proses yang berkaitan dengan cara menentukan kecepatan dan pengaturan kemudi pada setiap waktu agar robot dapat mengikuti rute tertentu [2]. Tujuan dari algoritma *path tracking* adalah untuk memperkecil kesalahan posisi dari robot terhadap titik referensi pada lintasan dan memandu robot untuk mengikuti lintasan yang sudah direncanakan. Algoritma *path tracking* yang akan digunakan pada tugas akhir ini adalah algoritma FTC (*Follow the Carrot*). Algoritma ini memiliki konsep yang sangat simpel, tentukan titik referensi pada jarak tertentu di depan robot, lalu jalankan robot menuju titik tersebut [13]. Algoritma FTC mencoba mengikuti rencana lintasan seakurat mungkin. Berdasarkan rencana lintasan yang sudah dibuat, algoritma akan menghitung kecepatan untuk sampai ke titik dalam lintasan itu. Diagram alir dari algoritma FTC dapat dilihat pada Gambar 3. Algoritma A* dan FTC berjalan pada kerangka kerja *Robot Operating System* (ROS). ROS adalah sekumpulan *library* perangkat lunak yang dapat digunakan untuk membangun aplikasi robot [14]. ROS bertujuan untuk memudahkan pengembang robot dalam membuat perangkat lunaknya tanpa harus membuat program dari awal serta dapat dikembangkan bersama-sama [15].

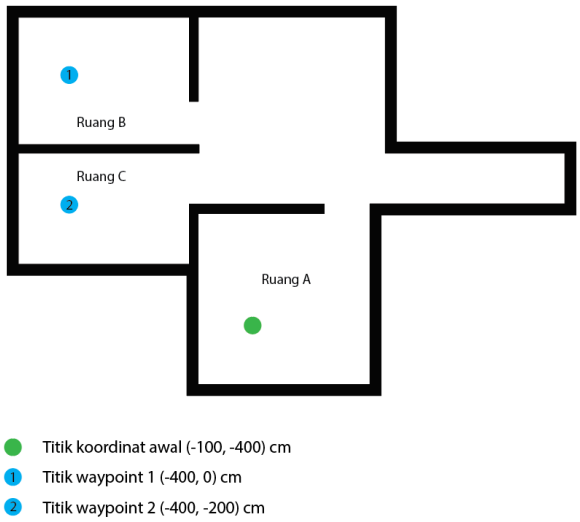


Gambar 3. Diagram alir algoritma FTC

3. Pengujian dan Analisis

3.1. Pengujian Navigasi Otomatis Sederhana

Pengujian navigasi otomatis sederhana dilakukan untuk mengetahui performa robot saat bernavigasi pada kondisi lingkungan yang ideal. Pengujian dilakukan di suatu rumah yang memiliki beberapa ruangan. Robot melakukan navigasi secara otomatis untuk berpindah dari satu ruangan ke ruangan lainnya. Denah rumah yang dipakai pada pengujian ini dapat dilihat pada Gambar 4.



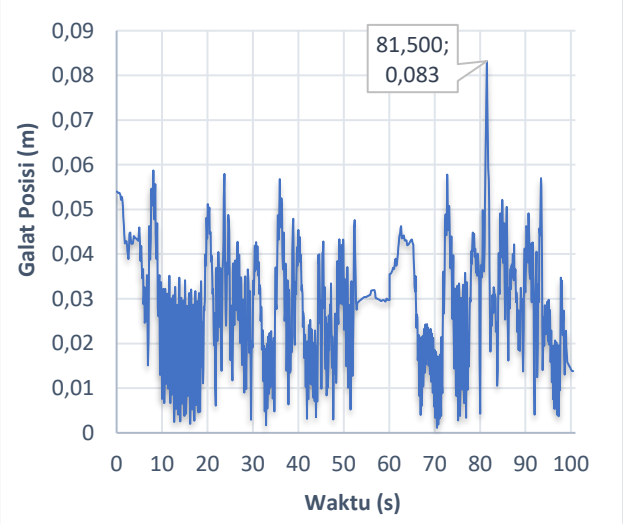
Gambar 4. Denah ruangan rumah

Robot diletakkan pada titik awal di ruangan A, kemudian robot diberikan 2 titik *waypoint* yang masing-masing berada pada ruangan B dan C. Sistem navigasi robot akan merencanakan rute yang dapat membawa robot menuju ke *waypoint* 1 dan 2 secara berurutan tanpa menabrak dinding. Data yang digunakan untuk mengetahui performa navigasi robot adalah waktu yang dibutuhkan program untuk membuat rencana rute, waktu yang dibutuhkan robot dalam melakukan navigasi dari titik awal ke titik target, dan kesalahan rata-rata posisi robot terhadap rute yang sudah dibuat. Parameter *map inflation* pada ROS diset pada nilai 0,3 meter. Parameter *map inflation* merupakan parameter yang berfungsi sebagai pengaman agar robot tidak melakukan navigasi terlalu dekat dengan halangan. Nilai *map inflation* 0,3 meter artinya jarak terdekat robot terhadap halangan adalah 0,3 meter, dihitung dari titik tengah robot. Parameter-parameter lain yang harus diset adalah parameter yang berhubungan dengan *local planner*. Parameter kecepatan linear maksimal robot bernilai 0,18 m/s. Parameter kecepatan sudut maksimal robot bernilai 1,5 rad/s. Jarak *carrot point* maksimal adalah 0,3 m. Toleransi kesalahan maksimum bernilai 0,5 meter untuk linear dan 0,01 rad untuk sudut. Parameter-parameter ini didapat melalui *trial error*.

3.1.1. Hasil Pengujian Posisi Robot

Hasil dari pengujian posisi robot dapat dilihat pada Gambar 5, Gambar 6, Gambar 7, dan Gambar 8. Pada Gambar 5 disajikan data galat posisi terhadap waktu dalam bentuk diagram garis. Galat posisi dicari dengan menghitung nilai *euclidean distance* antara posisi aktual robot dan suatu titik di dalam rute rencana yang terdekat dengan posisi robot aktual. Persamaan *euclidean distance* dapat dilihat pada persamaan 4.1. Nilai galat maksimum adalah 0,083 meter terjadi pada detik 81,5 pada saat robot melakukan gerakan navigasi untuk menghindari dinding dalam navigasi menuju ke *waypoint* 2

$$error = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (12)$$

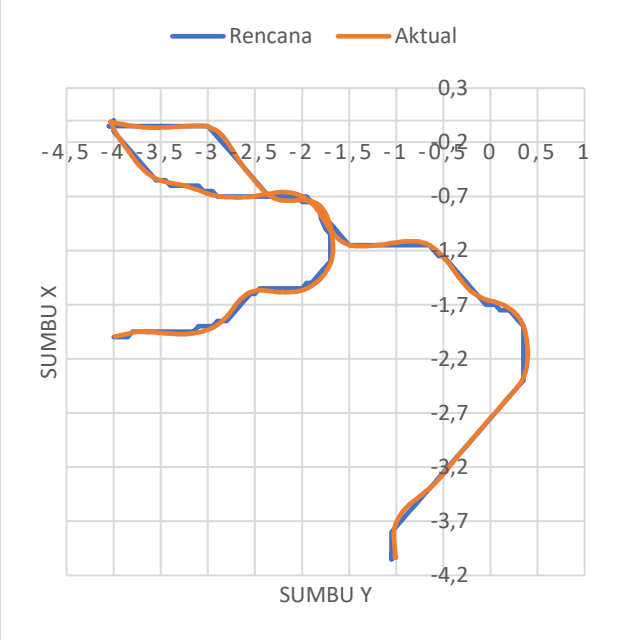


Gambar 5. Grafik galat posisi terhadap waktu

Gambar 6, Gambar 7, dan Gambar 8 memperlihatkan perbandingan antara rute yang benar-benar dilalui robot dengan rute yang sebelumnya sudah direncanakan. Dapat dilihat bahwa antara rute sebenarnya sangat mendekati rute yang sebelumnya sudah direncanakan. Untuk mengetahui galat absolut rata-rata posisi robot dalam bernavigasi dapat dihitung dengan menggunakan *Mean Absolute Error* (MAE) dalam Persamaan 21.

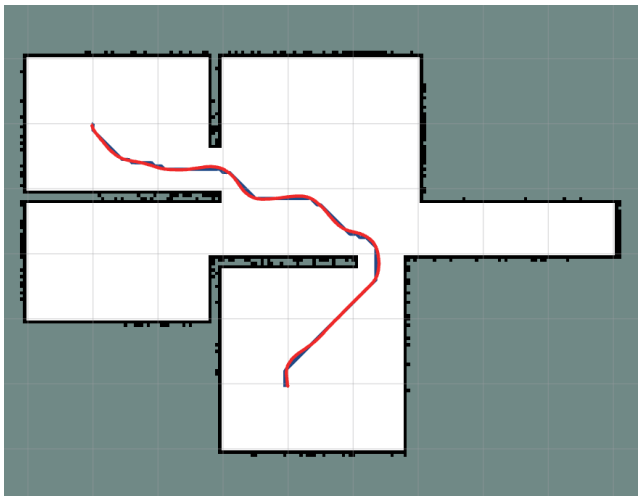
$$MAE = \frac{\sum_{i=1}^{n} \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{n} \quad (13)$$

Hasil dari perhitungan MAE terhadap data galat posisi robot didapatkan nilai rata-rata galat absolut robot dalam bernavigasi sebesar 0,029 meter.

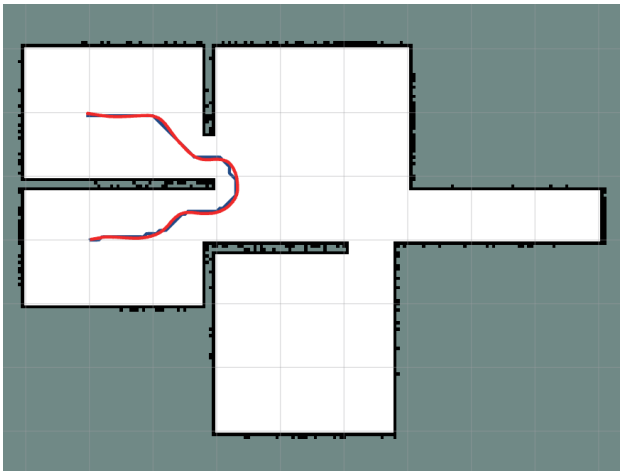


Gambar 6. Grafik perbandingan pergerakan robot aktual dan rencana

Waktu yang dibutuhkan program untuk menghasilkan rencana rute dari titik awal ke waypoint 1 adalah 21,584 ms. Dan waktu yang dibutuhkan program untuk menghasilkan rencana rute dari waypoint 1 ke waypoint 2 adalah 22,156 ms. Waktu ini diukur dari saat koordinat target dikirim sampai dengan program berhasil menemukan rute menuju ke target tersebut. Waktu yang dibutuhkan robot untuk melakukan navigasi dari titik awal ke waypoint 1 adalah 59,979 detik. Dan waktu yang dibutuhkan robot untuk melakukan navigasi dari waypoint 1 ke waypoint 2 adalah 40,078 detik. Panjang rute yang direncanakan program dari titik awal ke waypoint 1 adalah 7,925 meter. Dan panjang rute yang direncanakan program dari waypoint 1 ke waypoint 2 5,663 meter.



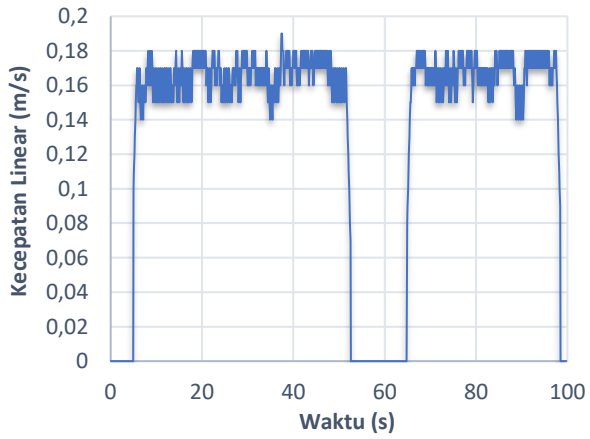
Gambar 7. Rute robot dari titik awal ke waypoint 1



Gambar 8. Rute robot dari waypoint 1 ke waypoint 2

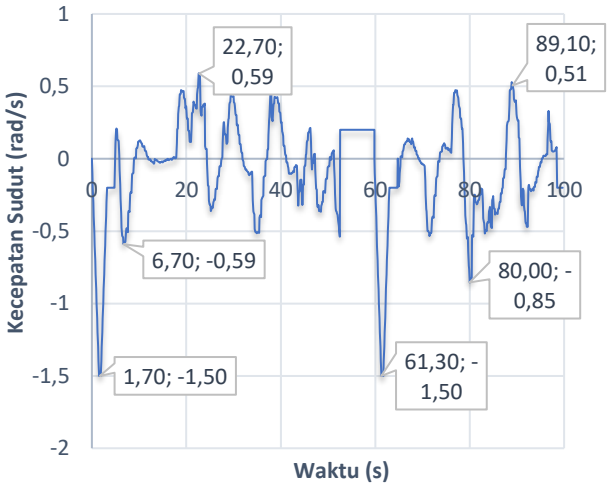
3.1.2. Hasil Pengujian Kecepatan Robot

Data kecepatan linear robot terhadap waktu ditunjukkan pada Gambar 9. Dapat dilihat bahwa kecepatan linear robot pada saat bernavigasi cukup stabil dengan kecepatan rata-rata 0,167 m/s. Kecepatan linear robot baru meningkat pada detik 5,0 karena robot baru memasuki fase navigasi ke dua pada detik tersebut. Pada detik 5,3 hingga detik 51,7 robot bernavigasi dari titik awal menuju waypoint 1 dengan kecepatan cukup stabil pada rentang 0,14 hingga 0,19 m/s. Pada detik 52,6 sampai 64,8 kecepatan robot 0 dikarenakan robot telah mencapai waypoint 1. Robot kemudian melanjutkan navigasi menuju waypoint 2 pada detik 64,9. Pada detik 65,7 sampai 97,8 robot bernavigasi menuju waypoint 2 dengan kecepatan yang cukup stabil pada rentang 0,14 hingga 0,19 m/s. Robot mencapai waypoint 2 pada detik ke 98,5.



Gambar 9. Grafik kecepatan linear robot terhadap waktu

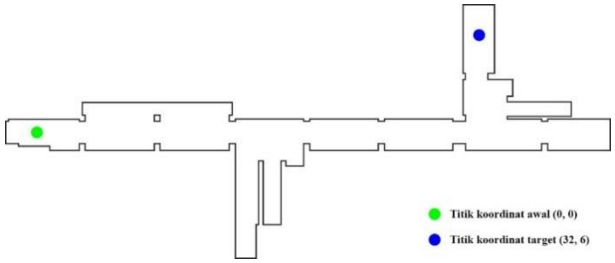
Data kecepatan sudut robot dapat dilihat pada Gambar 10. Dapat dilihat bahwa kecepatan sudut robot bernilai -1,5 rad/s pada detik 1,9. Hal ini dikarenakan robot pada detik 1,9 dalam fase navigasi kesatu sehingga robot hanya bergerak rotasi untuk menyesuaikan arah robot menghadap ke rute yang sudah dibuat. Pada saat robot bergerak dari titik awal menuju waypoint 1, kecepatan sudut robot bervariasi antara -0,57 dan 0,584 rad/s. Variasi ini terjadi karena arah hadap robot berubah secara dinamis menyesuaikan rute yang sudah dibuat. Pada detik 52,7 hingga 59,7 robot dalam fase navigasi ketiga dimana posisi robot sudah mencapai waypoint 1 dan robot hanya berputar pada tempatnya untuk menyesuaikan arah hadap saja. Pada detik 61,7 kecepatan robot bernilai -1,5 rad/s karena robot berada dalam fase navigasi pertama untuk mempersiapkan navigasi menuju ke waypoint 2. Pada saat robot bernavigasi menuju waypoint 2, kecepatan sudutnya bervariasi antara -0,854 rad/s sampai dengan 0,518 rad/s. Robot mencapai waypoint 2 pada detik 98,9.



Gambar 10. Grafik kecepatan sudut robot terhadap waktu

3.2. Pengujian Navigasi Otomatis Skala Besar

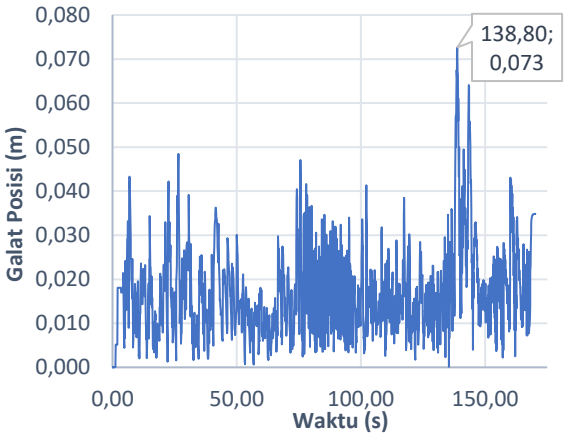
Pengujian navigasi otomatis skala luas bertujuan untuk mengetahui performa dari sistem navigasi otomatis robot saat diterapkan pada kondisi lingkungan sebenarnya. Pengujian dilakukan di Gedung B Teknik Elektro Universitas Diponegoro. Robot diberikan satu titik target tertentu pada ruangan dan robot melakukan navigasi secara otomatis dari titik awal ke titik target yang ditentukan. Data yang digunakan untuk mengetahui performa navigasi robot sama dengan data saat pengujian navigasi sederhana yaitu waktu yang dibutuhkan program untuk membuat rencana rute, waktu yang dibutuhkan robot dalam melakukan navigasi dari titik awal ke titik target, dan kesalahan rata-rata posisi robot terhadap rute yang sudah dibuat. Titik awal robot berada pada koordinat (0, 0). Sedangkan titik target robot diset pada koordinat (32, 6) meter. Parameter-parameter yang digunakan pada ROS masih sama seperti pada pengujian navigasi otomatis sederhana kecuali untuk parameter kecepatan linear maksimal robot bernilai 0,25 m/s.



Gambar 11. Posisi titik awal dan titik target pada peta

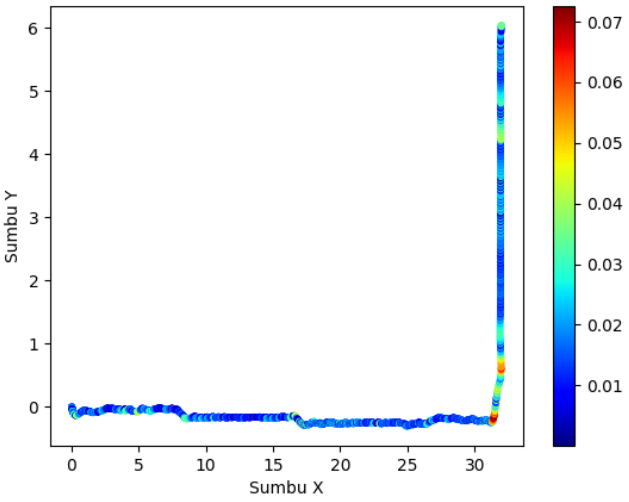
3.2.1. Hasil Pengujian Posisi Robot

Hasil dari pengujian posisi robot dapat dilihat pada Gambar 12, Gambar 13, dan Gambar 14. Pada Gambar 12 disajikan data galat posisi terhadap waktu dalam bentuk diagram garis. Galat posisi dihitung dengan menghitung nilai *euclidean distance* antara posisi aktual robot dan suatu titik di dalam rute rencana yang terdekat dengan posisi robot aktual. Nilai galat maksimum adalah 0,073 meter pada detik 138,8 karena robot melakukan gerakan berputar untuk menyesuaikan arah robot sesuai dengan rute yang sudah direncanakan. Namun dapat dilihat bahwa sistem navigasi robot bekerja dengan cepat untuk mengoreksi galat tersebut sehingga grafik turun secara drastis.

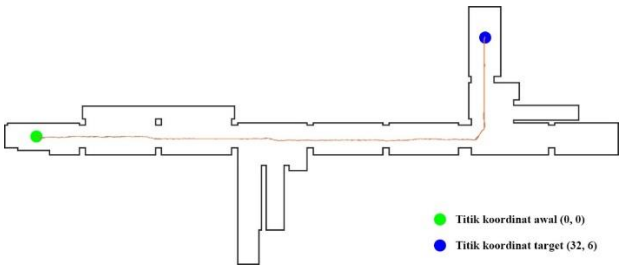


Gambar 12. Grafik kesalahan posisi terhadap waktu

Gambar 13 dan Gambar 14 memperlihatkan perbandingan antara rute yang benar-benar dilalui robot dengan rute yang sebelumnya sudah direncanakan. Dapat dilihat bahwa antara rute sebenarnya sangat mendekati rute yang sebelumnya sudah direncanakan. Hasil dari perhitungan MAE terhadap data galat posisi robot didapatkan nilai galat absolut rata-rata robot dalam bernavigasi sebesar 0,017 meter atau sebesar 1,7 cm. Waktu yang dibutuhkan program untuk menghasilkan rencana rute adalah 422,99 ms. Waktu ini diukur dari saat koordinat target dikirim sampai dengan program berhasil menemukan rute menuju ke target tersebut. Waktu yang dibutuhkan robot untuk melakukan navigasi dari titik awal ke titik target adalah 170,16 detik.



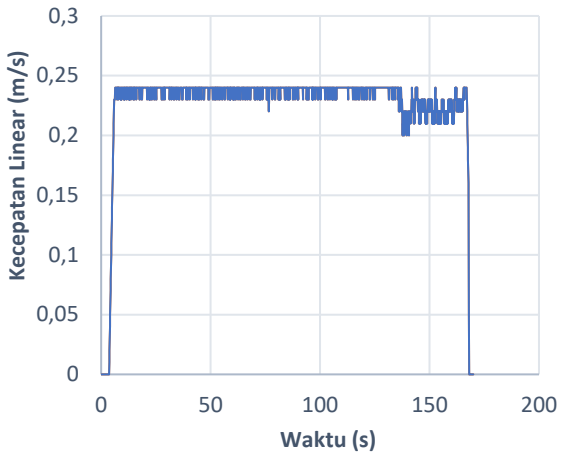
Gambar 13. Grafik pergerakan robot aktual dan galat posisi



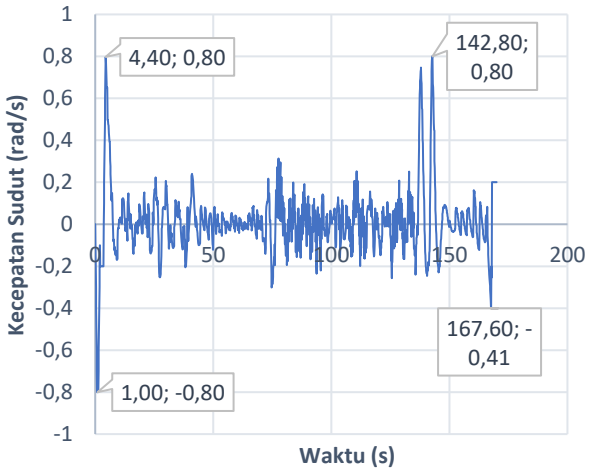
Gambar 14. Pergerakan Robot Aktual pada Peta

3.2.2. Hasil Pengujian Kecepatan Robot

Data kecepatan linear robot terhadap waktu ditunjukkan pada Gambar 15. Dapat dilihat bahwa kecepatan linear robot pada saat bernavigasi cukup stabil dengan kecepatan rata-rata 0,23 m/s. Kecepatan linear robot baru meningkat pada detik 3,7 karena robot baru memasuki fase navigasi ke dua pada detik tersebut. Pada detik 3,7 hingga detik 135,6 robot bernavigasi dengan kecepatan cukup stabil pada rentang 0,22 hingga 0,24. Namun pada detik 135,6 hingga terakhir terjadi penurunan kecepatan pada robot. Hal ini terjadi karena robot melakukan manuver untuk berbelok 90 derajat ke arah sumbu y positif. Data kecepatan sudut robot dapat dilihat pada Gambar 16. Dapat dilihat bahwa kecepatan sudut robot cukup tinggi pada detik 0 sampai 3,6. Hal ini dikarenakan robot pada detik 0 sampai 3,6 pada fase navigasi kesatu sehingga robot hanya bergerak rotasi untuk menyesuaikan arah robot menghadap ke rute yang sudah dibuat. Pada detik 138,0 dan 142,8 terjadi perubahan kecepatan sudut yang cukup drastis. Pada detik tersebut robot melakukan rotasi untuk mengubah arah pergerakan robot sebesar 90 derajat. Pada detik 168,0 hingga 170,0 merupakan fase ketiga navigasi dimana robot hanya melakukan gerakan rotasi untuk menyesuaikan arah hadap robot sesuai dengan target.



Gambar 15. Grafik kecepatan linear robot terhadap waktu.



Gambar 16. Grafik kecepatan sudut robot terhadap waktu.

4. Kesimpulan

Pada pengujian navigasi sederhana di dalam rumah, sistem navigasi robot dapat menemukan rute yang aman untuk bernavigasi dari titik awal ke waypoint 1 dan 2 dengan menggunakan algoritma A* dalam waktu yang cepat yaitu bernilai 43,74 ms. Robot dapat bernavigasi mengikuti rute yang sudah dibuat dari titik awal menuju ke waypoint 1 dan 2 dengan stabil dan presisi, dibuktikan dengan nilai galat posisi maksimal bernilai 8,3 cm dan galat posisi absolut rata-rata bernilai 2,9 cm. Pada pengujian navigasi otomatis sekala luas, sistem navigasi robot dapat menemukan rute yang aman untuk bernavigasi dari titik awal menuju ke titik akhir dengan menggunakan algoritma A* dalam waktu yang cepat yaitu bernilai 422,99 ms. Robot dapat bernavigasi mengikuti rute yang sudah dibuat dari titik awal menuju ke titik akhir dengan stabil dan presisi, dibuktikan dengan nilai galat posisi maksimal bernilai 7,3 cm dan galat posisi absolut rata-rata bernilai 1,7 cm.

Referensi

[1]. S. K.Brooks *et al.*, The psychological impact of quarantine and how to reduce it: rapid review of the evidence. *Lancet*. 2020; 95(10227): 912–920.

[2]. Chung Y, Park C, Harashima F. A position control differential drive wheeled mobile robot. *IEEE Trans. Ind. Electron.* 2001; 48(4): 853–863.

[3]. Guruji AK, Agarwal H, Parsediya DK. Time-efficient A* Algorithm for Robot Path Planning. *Procedia Technol. 2016*; 23(April): 144–149.

[4]. Sariff N, Buniyamin N. *An overview of autonomous mobile robot path planning algorithms*. SCOReD 2006 - Proc. 2006 4th Student Conf. Res. Dev. Towards Enhancing Res. Excell. Reg. 2006: 183–188.

[5]. Prasetyo AC, Arnandi MP, Hudnanto HS, Setiaji B, Perbandingan Algoritma Astar dan Dijkistra Dalam Menentukan Rute Terdekat. *Sisfotenika*. 2019; 9(1): 36.

[6]. Siahaan APU. Implementation of A-Star Algorithm in Determining the Shortest Path on Graph. *IJIRMF Journal*. 2018; 4(10): 374–378, 2018.

[7]. Sandeep KM, Majumdar J. Kinematics, Localization and Control of Differential Drive Mobile Robot. *Global Journals Inc*. 2014; 14(1): 1-8.

[8]. Maier A, Sharp A, Vagapov Y. *Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things*. 2017 Internet Technologies and Applications. Wrexham nited Kingdom. 2017; 143-148.

[9]. Gómez-Estern F. Computational principles of mobile robotics. *Automatica*. 2002; 38(10): 1.

[10]. Hart PE, Nilsson NJ, and Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*. 1968; 4(2): 100–107.

[11]. Liu X, Gong D. *A comparative study of A-star algorithms for search and rescue in perfect maze*. 2011 Int. Conf. Electr. Inf. Control Eng. Wuhan China. 2011; 1: 24–27.

[12]. Taufiq PJ, Wibowo AT, Septiana G. Implementation and Analysis of A * Algorithm for Multiple Goal Pathfinding Used at NPC (Non-Playable Charactrer) Movement. BEng Thesis. Bandung: Telkom University; 2015.

[13]. Lundgren M. Path Tracking for a Miniature Robot. Master Thesis, Sweden: Umea Univ; 2003.

[14]. Quigley M *et al. ROS: an open-source Robot Operating System*. ICRA workshop on open source software. 2009; 3(3.2): 5.

[15]. Jalil A. Robot Operating System (Ros) Dan Gazebo Sebagai Media Pembelajaran Robot Interaktif. *Ilk. J. Ilm.* 2018; 10(3): 284–289.