PENERAPAN METODE MONTECARLO UNTUK GERAK PENGONTROLAN ROBOT BERBASIS RANDOM WALKS

Muhammad Ridwan Effendi
Universitas Mohammad Husni Thamrin
jundi79@gmail.com

Abstract

The development of science and technology encourages the development of robots. Robots are widely used in various fields of human life to facilitate human life. Robot making has developed both in terms of methods and algorithms used in robots. The use of the montecarlo method and the random walks algorithm are widely implemented in robot designs at present. The basic concept of the Montecarlo method in solving differential equations is the probability of a random walk. Based on the approach in the random step process, the Montecarlo method is known for two types of approach that are quite popular, namely the fixed random walk type and the floating random walk type. For this research, the researcher intends to apply the monte carlo method to control the motion of a randon walk based robot. In this research method, the research stages will be described including needs analysis, application of the monte carlo method, making applications to control robots using Arduino.The results of this research, testing using the Montecarlo method, show that the response of the robot is not too slow, the new robot reacts not too far beyond the existing speed change limit. From the above tests, it can be analyzed that the application of the Montecarlo method is more effective and efficient in measuring motion in the robot.

***Keywords** : Montecarlo Method, Robot Control, Randon Walks*

1. PENDAHULUAN

Kemajuan pada bidang ilmu pengetahuan dan teknologi saat ini mengalami kemajuan sangat pesat, salah satu diantaranya ialah kemajuan dalam teknologi robotik. Saat ini hampir semua industri manufaktur menggunakan robot karena biaya perjam untuk mengoperasikan robot jauh lebih murah dibandingkan menggunakan manusia. Robot pada awalnya digunakan untuk melakukan fungsi spesifik, misalnya pengecoran, penyolderan, namun saat ini sudah banyak robot yang melakukan banyak fungsi (Widodo Budiharto, 2009 : p2).

Pada robotik *image processing* dapat digunakan sebagai pengolahan data, sehingga robot dapat mengidentifikasikan keadaan yang nantinya dapat mengatur setiap pergerakan pada robot sesuai dengan tujuan yang diinginkan. Untuk pengaturan pergerakan ini, *image processing* dapat digabungkan dengan *random walks* yang merupakan bagian dari metoda *montecarlo*. Konsep dasar dari metoda *montecarlo* dalam menyelesaikan persamaan diferensial adalah keboleh jadian langkah acak (*random walk*). Berdasarkan pendekatan dalam proses langkah acak, maka di dalam metoda *montecarlo* dikenal dua tipe pendekatan yang cukup populer, yaitu tipe *fixed random walk* dan *floating random walk*. Untuk penelitian ini, peneliti bermaksud untuk melakukan penerapan metode monte carlo untuk mengontrol gerak robot berbasis randon walk. Dalam metode penelitian ini akan

diuraikan tahapan penelitian meliputi analisa kebutuhan, penerapan metode montecarlo, membuat aplikasi untuk mengontrol robot menggunakan arduino.

Hasil dari penelitian ini pengujian dengan menggunakan metoda *monte-carlo*, memperlihatkan bahwa respon dari robot tidak terlalu lambat, robot baru bereaksi tidak terlalu jauh melewati jarak batas perubahan kecepatan yang ada. Dari pengujian diatas dapat dianalisa bahwa penerapan metoda *montecarlo* lebih efektif dan efisien dalam mengukur gerak pada robot.

2. Kajian Literatur

2.1 Penelitian Terkait

Penelitian yang dilakukan oleh R. Febriani dan Suprijadi (2010) yang berjudul "**Aplikasi Metode *Random Walks* Untuk Kontrol Gerak Robot Berbasis Citra"**. Penelitian ini mambahas tentang pengontrolan gerak robot berbasis citra memanfaatkan metode *montecarlo* dengan algoritma *random Walks*. Metode ini akan memanfa atkan input dari titik tengah objek yang tercapture, dan estimasi terhadap pergerakan robot ke step selanjutnya dilakukan melalui bilangan random (acak). Grafik yang diperoleh dari simulasi ini dapat dimanfaatkan untuk mengetahui besarnya error dari pergerakan robot dalam penentuan titik tengah dari citra. Dengan memanfaatkan metode ini dapat dilakukan suatu pe ngont rolan ge rakan robot untuk mencapai titik tertentu dengan lebih efisien karena telah dilengkapi dengan suatu rule yang harus dipatuhi.

2.2 Landasan Teori

2.2.1 Metoda *Monte Carlo*

Metoda *monte carlo* ditemukan oleh *Stanislaw Ulam* ahli matematika pada tahun 1946, bekerja untuk ***J**ohn Von Neumann* di proyek *United State's Manhattan* selama perang dunia II. Dalam metoda *monte carlo* ini mendefinisikan nilai yang mungkin dengan distribusi peluang untuk setiap variabel tidak tentu. Tipe distribusi yang dipilih didasarkan pada kondisi di sekeliling variabel.

Metoda *monte carlo* sebagaimana yang dipahami saat ini, melingkupi *sampling statistik* yang digunakan untuk memperkirakan solusi permasalahan kuantitatif. Metoda *monte carlo* merupakan teknik stokastik, dapat diaplikasikan kedalam berbagai bidang, mulai dari ekonomi sampai fisika, tentu saja cara aplikasinya berbeda dari satu bidang ke bidang lainnya, banyak himpunan bagian dari metoda *monte carlo* meskipun dalam satu bidang yang sama. Hal yang menyamakan semua itu adalah bahwa percobaan *monte carlo* membangkitkan bilangan acak untuk memeriksa permasalahan. Simulasi *monte carlo* adalah proses menurunkan secara acak nilai variabel tidak pasti secara berulang-ulang untuk mensimulasikan model.

Metoda ini didasarkan pada perhitungan yang sederhana dan dapat diadaptasi dengan komputer. Keuntungan atas fasilitas ujicoba (pengulangan) yang sangat cepat pada komputer sangat membantu dalam aplikasi metoda *monte carlo* ini. Didalam operasional *monte carlo* melibatkan pemilihan secara acak terhadap keluaran masing-masing secara berulang sehingga diperoleh solusi dengan pendekatan tertentu.

Konsep dasar dari metoda *monte car*lo dalam menyelesaikan persamaan diferensial adalah kejadian langkah acak (*random walk*). Berdasarkan pendekatan dalam proses langkah acak, maka di dalam metoda *monte carlo* dikenal dua tipe pendekatan yang cukup populer, yaitu tipe *fixed random walk* dan *floating random walk*. Tipe *floating random walk*

adalah model *monte carlo* yang mengizinkan jumlah *walker* selalu berubah dalam simulasi, cara floating random walk bisa kacau karena dalam simulasi bisa timbul sedikit *walker* (kebanyakan terbunuh dalam proses) dan banyak *walker* (timbul *walker* baru dalam proses) sehingga tipe *floating random walk* spesifik untuk satu aplikasi sedangkan tipe *fixed random walk* adalah model *monte carlo* yang menggunakan jumlah *walker* yang konstan jadi *walker* ini bertahan hidup sampai akhir simulasi sehingga untuk beberapa aplikasi hal ini lebih baik dari tipe *floating random walk* (Jonathan Pengelly, 2002)

2.2.2 *Random Walks*

Algoritma *Random Walk* adalah algoritma pencarian lintasan terpendek dengan menggunakan metode acak, berjalan secara acak untuk mendapatkan hasil yang optimum dengan menghasilkan percobaan lebih dari satu kali namun memiliki batas percobaan, *random* (acak) dan *walk* (jalan) memiliki cara yang *fleksibel* untuk menentukan arah untuk sampai ke titik *vertex* yang diinginkan. Karena *Random walk fleksibel* maka hasil yang didapat untuk mengetahui lintasan terpendek akan semakin mudah karena memiliki bahan perbandingan lebih dari satu.(patrick siarry, 2016). Algoritma *Random Walk* adalah proses stokastik yang dibentuk oleh penjumlahan berturut-turut independen, identik variabel acak terdistribusi adalah salah satu topik yang paling dasar dan dipelajari dengan baik probabilitas teori. (patrick siarry 2016).

3. Metode Penelitian

3.1 Analisa Kebutuhan Sistem Kerja Robot

Analisa kebutuhan dilakukan untuk menganalisa kebutuhan dalam pembuatan program aplikasi pada robot. Setelah proses analisa kebutuhan dilakukan dengan benar maka kebutuhan sistem dapat diketahui dengan tepat. Pada analisa kebutuhan dilakukan pendekatan dengan menggunakan robot, dengan penerapan metoda *monte carlo* untuk pengontrolan *random walk* pada robot. Untuk membuat robot tersebut agar sesuai dengan target, maka konsep yang digunakan adalah *monte carlo*. Untuk mendukung robot tersebut digunakan bahasa pemrograman Arduino IDE C sebagai *software* pembangun aplikasi robot.

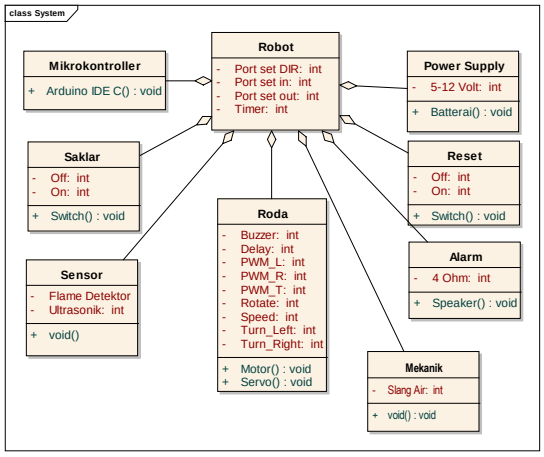
3.2 Perancangan *Sistem Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah metoda pemodelan dengan analisis semantic dan notasi yang menggabungkan beberapa jenis pemodelan yang telah ada sebelumnya. Dalam pemodelannya, Sistem *Unified Modeling Language* (UML) menggunakan beberapa jenis diagram yang mempunyai fungsi masing-masing dalam mendiskripsikan suatu sistem/aplikasi. Beberapa diagram yang akan dibahas untuk aplikasi robot antara lain:

a. Diagram *Class*
b. Diagram *State Chart*
c. Diagram *Activity*

3.2.1 Perancangan Menggunakan Diagram *Class*

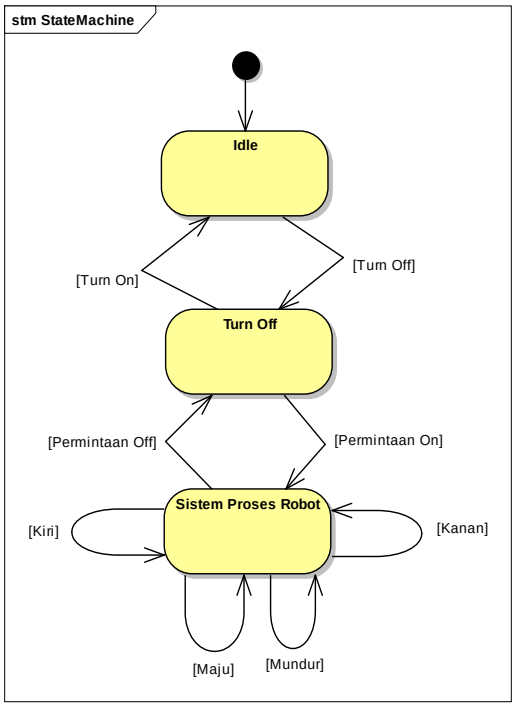
Perancangan sistem pada penelitian ini menggunakan Diagram *Class*, di sini dijabarkan secara grafis dengan menggambarkan proses pada robot. Diagram *Class* dirancang untuk menggambarkan bagaimana cara berinteraksi dengan sistem yang dibuat. Kegunaan dari Diagram *Class* untuk menunjukan antara robot, dengan proses-proses fungsionalnya dari proses kerja robot, Dari masing-masing karakteristik kerja robot, kemudian dijelaskan ke dalam subkarakteristik pada Sistem *Unified Modeling Language* (UML).



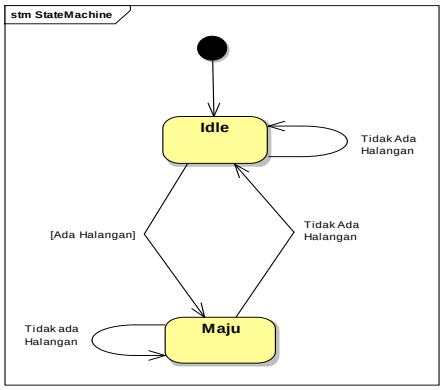
Gambar 3.1 Diagram *Class* Sistem Kerja Robot

3.2.2 Perancangan Diagram *State Chart*

Diagram *state chart* menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya), pada umumnya *statechart* diagram menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *state chart* diagram).

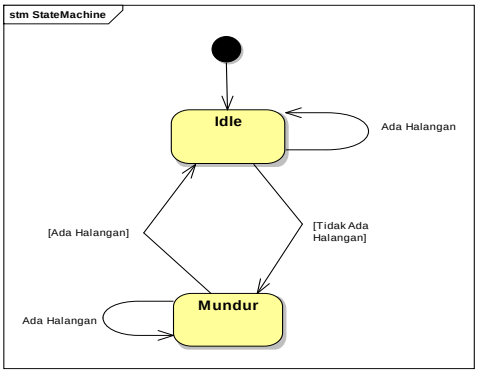


Gambar 3.2 Diagram *State Chart* Sistem Kerja Robot

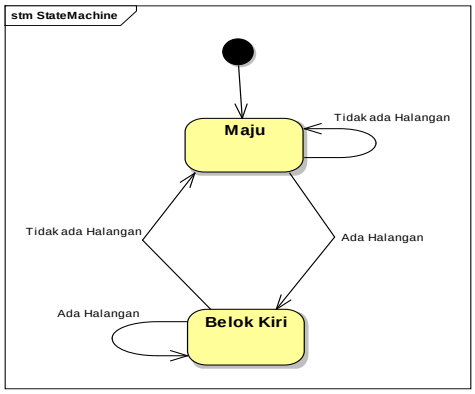


Gambar 3.3 *State Chart* Robot Bergerak Maju

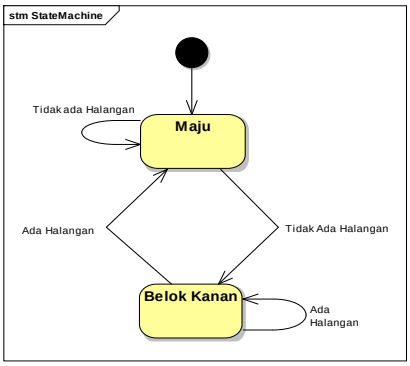
.



Gambar 3.4 *State Chart* Robot Bergerak Mundur

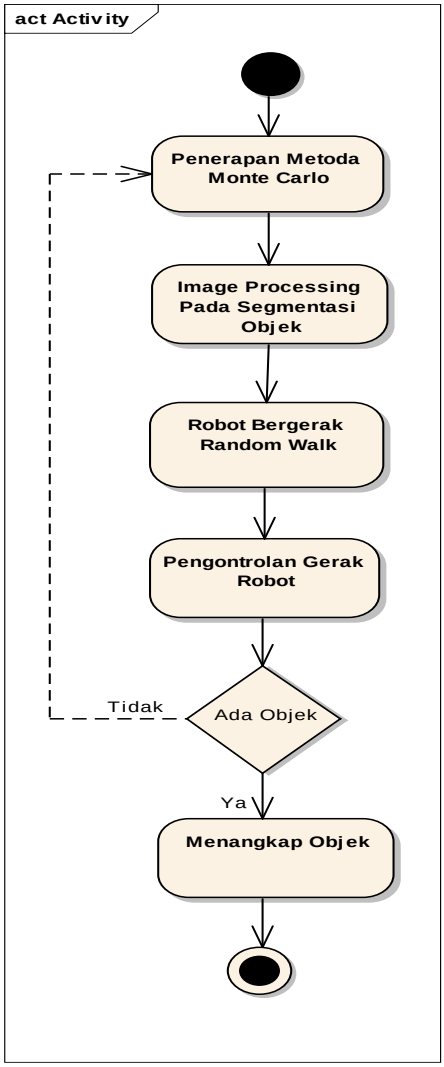


Gambar 3.5 *State Chart* Robot Bergerak Kekiri

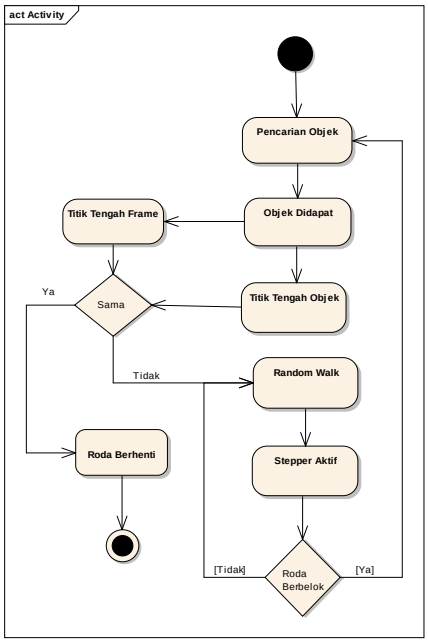


Gambar 3.6 *State Chart* Robot Bergerak Kekanan

3.2.3 Perancangan Diagram *Activity*



Gambar 3.7 Diagram *Activity* Sistem Robot Secara Keseluruhan



Gambar 3.8 Diagram *Activity* Sistem Proses Kerja Robot

3.3 Perancangan Hardware Sistem Kerja Robot

Robot setidaknya harus memiliki *navigator* untuk berjalan, *manipulator* untuk memindahkan suatu objek.

3.4 Prinsip Kerja Mikrokontroler

Pembuatan program pada mikrokontroler merupakan dasar dari pengontrolan kerja pada robot, dimana penerapan dari mikrokontroler adalah sebagai pengendalian suatu sistem berdasarkan informasi *input* yang diterima, lalu diproses oleh mikrokontroler, dan dilakukan aksi pada bagian *output* sesuai dengan program yang telah ditentukan sebelumnya. Mikrokontroler merupakan pengontrol utama perangkat pada robot.

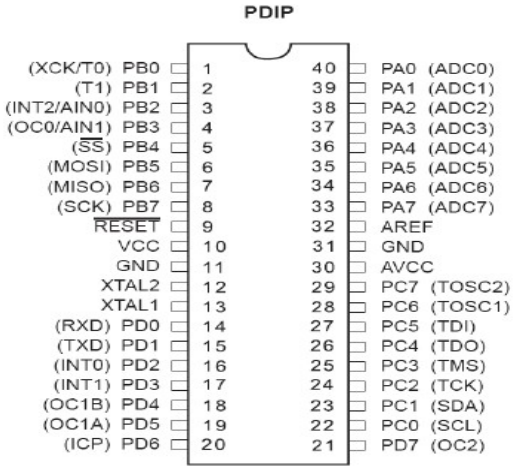
3.4.1 Perancangan Mikrokontroller

Mikrokontroler yang digunakan untuk pembuatan robot adalah dari Arduino Uno. Pada mikrokontroler ini memiliki beberapa fitur yang dapat memudahkan dalam penerapan pemrograman dan fungsi-fungsi tambahannya yang dapat dimanfaatkan untuk pengendalian pada gerak robot.

Mikrokontroler merupakan sistem mikroprosesor lengkap yang terintegrasi dalam sebuah chip IC (*integrated circuit*). Mikroprosesor adalah merupakan suatu komponen rangkaian yang terintegrasi dengan skala besar yang berkemampuan sebagai unit pengolah pusat.

3.4.2 Konfigurasi Pin Arduino Uno

Pada gambar dibawah ini memperlihatkan konfigurasi *Plastic Dual In Package* (PDIP) 40 pin dari mikrokontroler Arduino Uno..



Gambar 3.9 Konfigurasi Pin Arduino Uno

3.5 Prinsip Kerja Sensor

Robot dapat mengenali lingkungan melalui sensor yang dimilikinya. Informasi atau data yang diterima melalui sensor diterima sebagai persepsi, yang digunakan menjadi pengetahuan bagi robot, kemudian disimpan dan dipelajari bagi robot. Dari pengetahuan yang dimiliki robot akan menentukan dan mengambil keputusan tentang apa yang harus dilakukan pada kondisi tertentu sesuai dengan lingkungan yang ada dimana robot melakukan aktifitas. Keputusan tersebut kemudian direalisasikan melalui kontrol terhadap aktuaktor. Aktuaktor menggerakan robot kemudian sensor kembali bekerja dan mengenali lingkungan disekitarnya untuk adaptasi terhadap langkah selanjutnya.

3.6. Pengembangan Gerak Robot

3.6.1 Driver Motor DC

Motor DC adalah sebuah elekrik motor yang memakai tegangan DC dengan mengkonversikan besaran listrik menjadi besaran mekanik. Pada umumnya motor DC sering dipakai dan lebih cocok digunakan pada aplikasi–aplikasi elektronika misalkan pada robot mobil. Motor DC ini memiliki dua terminal elektrik. Dengan memberikan beda tegangan pada kedua terminal tersebut maka motor akan dapat berputar pada satu arah dan apabila polaritas dari tegangan tersebut dibalik, maka arah putaran motor akan terbalik pula.

3.6.2 Belok Kanan

Robot melakukan belok kanan disesuaikan dengan halangan yang diberikan, yaitu halangan yang berbentuk busur 45 derajat dan 90 derajat.

3.6.3 Belok Kiri

Robot melakukan belok kiri, disesuaikan dengan halangan yang diberikan, yaitu halangan yang berbentuk busur 45 derajat dan 90 derajat

4. Hasil Penelitian dan Pembahasan

4.1 Hasil

Hasil dari perancangan perangkat lunak adalah bagian yang sangat penting pada pembuatan robot. Program mikrokontroler dirancang untuk melakukan proses algoritma dan mengontrol gerakan *random walk* pada robot.

Pemrograman dilakukan dengan menggunakan bahasa arduino IDE C yang ditanamkan pada mikrokontroler. Berikut adalah hasil implementasi sistem.

4.1.1 Spesifikasi Perangkat Lunak dan Perangkat Keras

Spesifikasi sistem dibagi menjadi tiga yaitu:

a. Perangkat keras berupa komputer
b. Instalasi sistem
c. Perangkat robot

a. Perangkat Keras Berupa Komputer

Spesifikasi perangkat keras (*hardware)* di bawah ini dapat menjadi acuan untuk implementasi aplikasi program yang akan ditanamkan (*download*) pada robot.

- *Processor intel 5* atau lebih,
- RAM dengan kapasitas 4 GB atau lebih,
- *Harddisk* dengan kapasitas 1024 GB atau lebih,
- *CD-Rom*
- *Mouse, Keyboard*
- *Monitor*.

b. Instalasi Sistem

Untuk menjalankan aplikasi program arduino IDE C yang akan ditanamkan pada mikrokontroler pada robot, perlu melakukan langkah-langkah sebagai berikut:

1. Sistem operasi yang digunakan adalah *microsoft windows 10*.
2. Sistem Arduino IDE C

Sistem Arduino IDE C merupakan perangkat lunak (*software*) yang dapat digunakan dalam hal untuk merancang program aplikasi yang akan ditanamkan (*download*) pada mikrokontroler yang ada pada robot.

c. Perangkat Robot

Pada robot yang akan dibuat, yang perlu ditentukan dalam pembuatan robot ini antara lain:

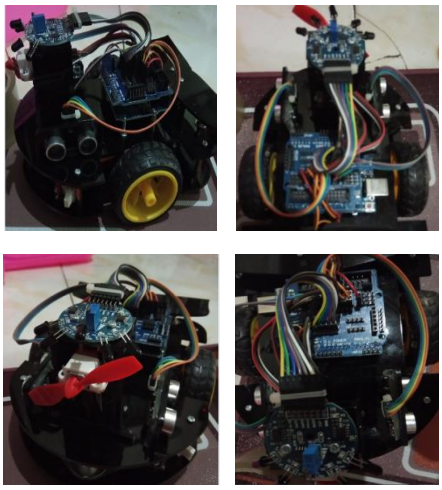
1. Dimensi
2. Komponen Material
3. Sistem Kerja Robot
4. Sensor
5. Metode Pengontrolan
6. Mekanisme

4.1.2 Implementasi Perangkat Keras Robot

Pada sistem robot yang dibangun secara garis besarnya terdiri dari tiga bagian *hardware* yaitu motor DC, motor *stepper*, dan rangkaian mikrokontroler. Motor DC berfungsi untuk menggerakkan roda agar robot dapat bergerak maju dan mundur, sedangkan motor *stepper* berfungsi untuk pergerakan robot ke arah kiri dan kanan. Bagian ketiga adalah mikrokontroler digunakan untuk menerima data dari komputer dan memberkan instruksi yang mengatur pergerakan kedua motor sesuai *input* dari komputer.

Pada robot mobil didesain untuk malakukan perjalanan mengikuti rute (*maze*) yang sudah ditentukan. Ada tiga macam pergerakan pada robot, antara lain:

a. *Tracking* adalah gerakan pada robot untuk mengikuti trayektori.
b. *Docking* adalah gerakan pada robot untuk menuju pada suatu posisi dan orientasi tertentu. Jadi selama melakukan tracking robot akan menangani permasalahan sistem kendali trayektori, sedangkan saat melakukan docking, robot akan menangani permasalahan sistem kendali posisi.
c. *Obstancle avoidance* adalah gerakan pada robot untuk menghindari suatu halangan yang muncul sewaktu-waktu. Halangan tersebut dapat muncul sewaktu-waktu, maka robot akan melakukan gerakan *docking* maupun *tracking.*
d. Perancangan *hardware* yang dibangun adalah seperti gambar berikut:



Gambar 4.0 Rancangan hardware robot dan rangkaian mikrokontroler

4.1.3 Implementasi Perangkat Lunak

Implementasi perangkat lunak (*software)* dilakukan apabila robot yang telah dibuat sudah siap untuk diujikan. Perangkat lunak (*Software)* dibuat pada komputer dengan menggunakan bahasa pemrograman arduino IDE C, setelah perangkat lunak selesai dibuat maka dapat ditanamkan (*download)* pada mikrokontroler yang ada pada robot, sehingga robot dapat berfungsi sesuai dengan yang diharapkan. Pada pembuatan program perangkat lunak ini memanfaatkan suatu teknik *monte carlo* berbasis *random walk.*

4.2 Verifikasi Kinerja Sistem Robot

4.2.1 Pengujian Sistem Keseluruhan dan Analisis Data

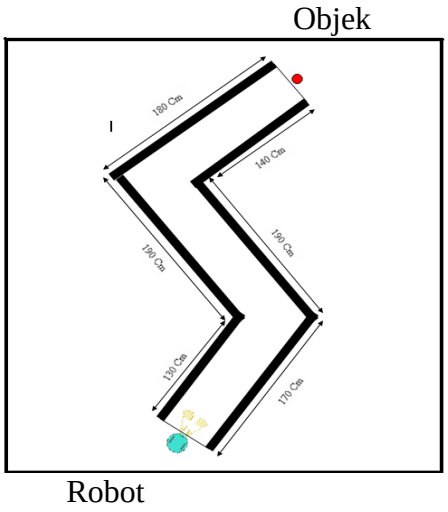
Pada masalah ini akan dijelaskan cara pengujian yang diterapkan, serta data yang didapat dari hasil pengujian, analisis terhadap masalah yang dihadapi pada saat dilakukan pengujian, serta analisis kesalahan (*error*) yang terjadi pada saat pengujian tiap modul dan sistem secara keseluruhan. Titik tengah objek inilah yang selanjutnya akan dibandingkan dengan titik tengah yang sebenarnya yaitu titik tengah dari *frame* yang digunakan. Proses r*andom walks* sendiri akan dimulai pada saat posisi titik tengah objek tidak berada pada posisi tengah *frame*. Pada posisi ini maka robot akan mengalami pergerakan ke arah kiri atau arah kanan sesuai dengan *random walks* yang digunakan. Berdasarkan hasil pengujian sistem secara keseluruhan, maka metoda *monte carlo* yang telah terintegrasi dengan teknik-teknik *random walks* dapat dimanfaatkan untuk pengontrolan gerak robot.

4.2.2 Pengujian dan Analisis Sistem Terintegrasi

Pada pengukuran sistem terintegrasi dibagi menjadi 2 bagian, yaitu pengukuran tanpa menggunakan metoda *monte carlo* dan pengukuran dengan menggunakan metoda *monte carlo*, pada pengukuran gerak pada robot ini dengan menggunakan halangan diam berupa tembok yang dipasang disebelah kiri dan sebelah kanan pada lokasi lintasan.

Untuk data yang dianalisa adalah pada pengukuran gerak robot mobil berupa jarak saat robot bergerak dimulai dari titik awal dan selesai pada titik akhir dengan lokasi (maze). Untuk pengukuran variabel kecepatan robot dan keakuratan waktu tempuh pada robot dilakukan dengan cara menghadapkan robot secara tegak lurus, kemudian robot dilepas dengan kecepatan yang diatur, dengan cara merubah variabel kecepatan pada setiap kali pengukurannya. Pada pengukuran pergerakan robot ini hanya mengukur sebanyak 10 kali pengukuran. Adapun pengukuran dari algoritma yang ada pada jarak posisi awal pada robot terhadap dinding tidak boleh terlalu jauh (maksimal 30 cm), Hal ini diakibatkan oleh gerakan dari robot yang cenderung berusaha mendekati dinding. Sudut awal posisi robot sebelum *proses* pergerakan dilakukan, keadaan robot harus tegak lurus dari dinding.

Dibawah ini adalah denah lokasi (*maze*) untuk pengukuran gerak pada robot, dengan mengukur variabel jarak tempuh serta waktu tempuh yang dilakukan pada robot.



Gambar 4.1 Denah Lokasi Pengujian Gerak Robot

Diatas adalah peta (*maze*) perjalanan awal yang dibuat dengan tanpa memperhitungkan adanya halangan yang muncul ditengah jalan. Bila tidak ada halangan, maka robot hanya akan berada pada kondisi *tracking* dan *docking*. Bila ditengah jalan ada halangan, maka robot akan berada pada kondisi *obstacle avoidance*, untuk menghindari adanya halangan. *Tracking* suatu gerakan dengan tujuan mengikuti *trayektori*. *Docking* suatu gerakan untuk menuju pada suatu posisi dan orientasi tertentu. Selama melakukan *tracking* robot akan menangani permasalahan sistem kendali *trayektori*, sedangkan saat melakukan *docking*, robot akan menangani permasalahan sistem kendali posisi. *Obstancle avoidance* suatu gerakan untuk menghindari halangan yang muncul sewaktu-waktu. Jika halangan muncul sewaktu-waktu maka robot dapat melakukan gerakan *docking* maupun *tracking*.

4.2.2.1 Variabel Perhitungan Gerak Dasar Robot Dengan Metoda M*onte Carlo*

Beberapa variabel yang menjadi parameter perhitungan pergerakan dasar robot dengan menggunakan metoda *monte carlo* berbasis *random walk* adalah:

Gerakan robot secara *random walk* pada dua dimensi (2D) adalah dua garis kerja pada robot, dimana robot dapat maju maupun mundur atau berbelok kekiri maupun kekanan, maupun berputar.

Perpindahan gerak kekiri dan kekanan atau berputar akan dipengaruhi oleh probabilitasnya adalah sebagai berikut:

$$f_X(\chi) = \begin{cases} 1/4, & \text{jika } \chi = \pm 1, \\ 0, & \text{lainnya} \end{cases}$$

Gambar 4.2 Rumus gerakan robot kekiri dan kekanan atau berputar

Dimana:

$f_X(\chi)$ = Peluang, fungsi peluang dan variabel

$1/4, \text{ jika } \chi = \pm 1,$ = Pergerakan robot kekiri dan kekanan

0 = Peluang /kejadian yang mustahil

1

Sedangkan gerakan keseluruhan dari robot, dapat dihitung berdasarkan rumus sebagai berikut.

$$x(T) = x(0) + \sum_{k=1}^{T} X_k$$

Gambar 4.3 Rumus gerakan keseluruhan dari gerak robot secara *random walk*

Dimana:

x (T) = Jumlah seluruh langkah

$_{T}$ = Banyaknya langkah

$_{k-1}$ = Index dari T

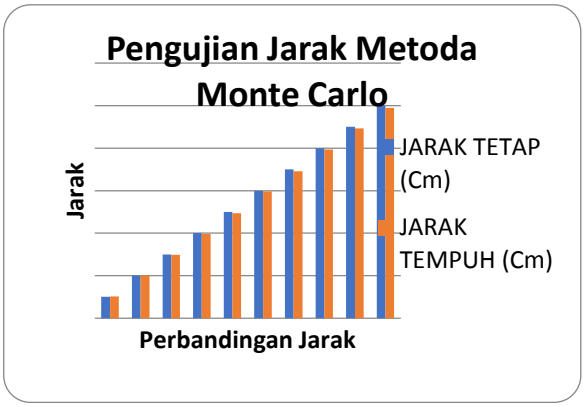
4.2.2.2 Pengujian dengan Halangan Menggunakan Metoda *Monte Carlo*

Pada pengujian robot ini dilakukan dengan menghadapkan robot secara tegak lurus, sama seperti pengujian yang di-lakukan tanpa menggunakan metoda *monte carlo*, kemudian robot dilepas dengan kecepatan maksimum 55 cm/s dan diharapkan berubah kecepatannya pada jarak 500 cm, 450 cm, 400 cm, 350 cm, 300 cm, 250 cm, 200 cm, 150 cm, 100 cm, 50 cm, dengan kecepatan yang terus menurun dari 55 cm/s, 50 cm/s, 45 cm/s, 40 cm/s, 35 cm/s, 30 cm/s, 25 cm/s, 20 cm/s, 15 cm/s, 10 cm/s. Hasil dari angka *monte carlo* ini kemudian ditambahkan pada nilai angka kecepatan (Cm/s). Hasilnya adalah sebagai berikut: Dibawah ini adalah data-data dalam bentuk tabel dari hasil pengujian robot dengan menggunakan metoda *monte carlo*.

Tabel 4.1 Pengujian Menggunakan Metoda *Monte Carlo*

NO	KECEPATAN (Cm/S)	JARAK TETAP (Cm)	JARAK TEMPUH (Cm)	TABRAK DINDING	WAKTU TEMPUH
			$f x(\chi)=½$		(Detik)
1	10	50	51	-	0:06
2	15	100	101	-	0:11
3	20	150	149	1	0:17
4	25	200	198	2	0:23
5	30	250	247	2	0:30
6	35	300	298	2	0:37
7	40	350	345.5	3	0:41
8	45	400	397	4	0:50
9	50	450	446.5	4	0:56
10	55	500	495	5	1:05
RATA-RATA		275	272.8	2.875	0.02333333

Dibawah ini adalah grafik dari hasil pengujian perbandingan jarak ideal dengan jarak terukur dengan menggunakan metoda *monte carlo*.



Gambar 4.4 Grafik Pengujian Jarak Dengan Metoda *Monte Carlo*

Dari data-data grafik diatas terlihat bahwa pengujian dengan menggunakan metoda *monte carlo*, memperlihatkan bahwa respon dari robot tidak terlalu lambat, robot baru bereaksi tidak terlalu jauh melewati jarak batas perubahan kecepatan yang ada. Dari pengujian diatas dapat dianalisa bahwa penerapan metoda *monte carlo* lebih efektif dan efisien dalam mengukur gerak pada robot.

5. Kesimpulan

Dari hasil pengujian sistem serta analisa yang dilakukan pada robot, maka dapat diambil beberapa kesimpulan yaitu:

1. Dalam penerapan Metoda *monte carlo* dapat dimanfaatkan untuk pengontrolan *random walks* pada gerak robot dimana hasilnya lebih efektif.
2. Pada *proses* gerak robot secara *random walk*s dalam mencari suatu objek sebelum diterapkan metoda *monte carlo*, hasilnya terlalu lama dalam waktu mencari suatu objek, serta terlalu jauh jarak tempuhnya, sedangkan objek yang dicari tidak terlalu jauh.
3. Penerapan Metoda *monte carlo* dalam pengontrolan gerak robot secara *random walks* lebih efektif dan efisien dalam waktu tempuh, serta jarak tempuh robot tersebut.

DAFTAR PUSTAKA

1. Fadlisyah, Adzuha Desmi, Iqbal Faridiansyah.2008. Robotika, Reasoning, Planning, Learning. Graha Ilmu.
2. H. Greenspan, G. Dvir, and Y. Rubner. 2002. Context-based Image Modeling. *International Conference on Pattern Recognition*. Quebec City. Canada.
3. Rani Febriani. 2007. Kontrol gerak Robot Menggunakan Metoda *Random Walk*
4. Budiharto, Widodo. Robotika Modern Teori & Implementasi (Edisi Revisi); Yogyakarta: Andi, 2015.
5. Yohandri & Asrizal, 2016. *Elektronika Dasar 1: Komponen, Rangkaian, dan Aplikasi*. Edisi Pertama. Jakarta: Kencana.
6. Dharmawan, H. A., 2017. *Mikrokontroler Konsep Dasar dan Praktis*. Edisi Pertama. Malang: UB Press.
7. Safaat H, Nazruddin. 2015. Aplikasi Berbasis Android. Informatika. Bandung.
8. Clark, Denis and Owings, Michael**.** 2003. *Building Robot Drive Trains, 1st Edition.* USA McGraw-Hill.
9. Endra Pitowarno. 2006. Robotika, Desain, Kontro, dan Kecerdasan Buatan. Andi Yogyakarta.

10. Eko, Agfianto.2005. Belajar Mikrokontroller Teori dan Aplikasi. Gava Media. Yogyakarta.
11. Fadlisyah, Adzuha Desmi, Iqbal Faridiansyah.2008. Robotika, Reasoning, Planning, Learning. Graha Ilmu.
12. Joseph Schmuller. 2004. Sams Teach Yourself UML in 24 Hours, Third Edition.
13. Rani Febriani. 2007. Kontrol gerak Robot Menggunakan Metoda *Random Walk*
14. Russel, Stuart and Norvig, Peter. 2003. *Artificial Intelligence: A Modern Approach.* Prentice Hall.
15. Suparman. 2007. Komputer Masa Depan Pengenalan Artificial Intelligence. CV. Andi Offset. Yogyakarta.
16. Thomas J. Kakiay, 2004, Pengantar Sistem Simulasi, Andi Offset Yogyakarta
17. Widodo Budiharto, Paulus Andi Nalwan.2009. Membuat Sendiri Robot Humanoid. PT.Elex Media Komputindo.
18. Widodo Budiharto.2008. 10 Proyek Robot Spektakuler. PT.Elex Media Komputindo.

Jurnal

1. Setia, S. C. (2017). Smart Tester Berbasis Mikrokontroler ATMega 328P. *JUSIKOM, 2*, 46-47.
2. Pada Robot Boat Pengintai Menggunakan XBEE Series 1 Berbasis Arduiono, Palembang:Politeknik Negeri Sriwijaya, 2015
3. https://www.arduino.cc/en/Main/Products [Accessed 7 Maret 2020].
4. Michael Fowler. 2007. *The One-Dimensional Random Walk*. UVa Physics [Online] [Dikutip: 08 Nopember 2020.]. http://galileo.phys.virginia.edu/classes/152.mf1i.spring02/RandomWalk.html
5. Jonathan Pengelly. 2002. *Monte Carlo Methods*. [Dikutip: 28 Nopember 2020.] http://www.cs.otago.ac.nz/cosc453/student_tutorials/monte_carlo.pdf
6. Siegwart, Roland and Nourbakhsh, Illah**.** *Autonomous Mobile Robot.2004* [Online] [Dikutip: 30 Nopember 2020] http://autonomousmobilerobots.epfl.ch/.
7. S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), San Francisco, CA, 2000. IEEE.
8. Thrun, S. and Fox, D. and Burgard, W. and Dellaert, F. 2000. *Monte Carlo Localization for Mobile Robots*. JOURNAL Artificial Intelligence
9. Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. 2001. *Robust Monte Carlo Localization for Mobile Robots.* Artificial Intelligence Journal.
10. Michael Fowler. 2007. *The One-Dimensional Random Walk*. UVa Physics [Online] [Dikutip: 08 Nopember 2020.]. http://galileo.phys.virginia.edu/classes/152.mf1i.spring02/RandomWalk.html
11. Pyroelectric sensor package. *Acroname Robotics.* [Online] [Dikutip: 13 Nopember 2020.] http://www.acroname.com/robotics/parts/R3-PYRO1.html
12. Jonathan Pengelly. 2002. *Monte Carlo Methods*. [Dikutip: 28 Nopember 2020.] http://www.cs.otago.ac.nz/cosc453/student_tutorials/monte_carlo.pdf