IMPLEMENTASI METODE *MAZE* DAN PID PADA ROBOT *VACUUM CLEANER AUTOMATIC*

Dian Tresnawan[1)], Meidi [2)]

Program Studi Teknik Elektro, Universitas Internasional Batam
Email: dtresnawan@yahoo.com[1)], dtresnaJP@yahoo.co.id[2)]

ABSTRAK

Robot *Vacuum Cleaner Automatic* merupakan peralatan elektronik yang berfungsi sebagai pembersih lantai dari debu yang dapat bekerja mandiri dan dapat kembali di posisi awal/penyimpanan. Agar robot dapat berkerja dengan baik, maka diperlukan pemetaan ruangan yang dapat diatur sesuai kebutuhan. Oleh karena itu diterapkan metode *Maze* dan PID. Metode *Maze* adalah salah satu metode yang mempelajari pergerakan robot, dimana jalur perjalanan yang akan dilalui oleh robot *vacuum cleaner* ditentukan dengan kolom dan baris yang dapat ditentukan sesuai dengan kebutuhan. Sedangkan PID adalah metode pengontrol kestabilan jalan robot**.** Cara kerja dari robot *vacuum cleaner* ini diawali dengan melakukan pemetaan daerah yang akan dibersihkan, pada penelitian ini daerah yang akan dibersihkan 200cm x 200cm dan dibagi ke dalam 8 kolom dan 8 baris, setiap kolom dibagi lagi menjadi 8 indeks, begitu juga dengan barisnya dibagi menjadi 8 indeks, dan dimasukkan ke dalam program utama pergerakan robot. Selanjutnya robot akan bergerak mengikuti pemetaan yang telah diprogramkan dan kembali ke posisi awal setelah membersihkan / melalui seluruh baris dan kolom, dengan menggunakan jalan tercepat. Berdasarkan hasil percobaan dan analisa yang telah dilakukan disimpulkan bahwa robot dapat berjalan sesuai *rule* pada ruangan dengan tingkat keberhasilan sebesar 100% dan menghasilkan pergerakan robot yang baik dengan nilai *error steady state* 6,66%.

Kata kunci:*Vacuum Cleaner,* Robot *Vacuum Cleaner,* Metode *Maze*, Metode PID.

ABSTRACT

The proposed Automatic Vacuum Cleaner is an electronics equipment that has function as cleaning dirt and dust from the floor (a Vacuum Cleaner) that can work by itself and after completed its job, the Vacuum Cleaner will come back where its started. To do the task completely, Maze and PID Method was used in this study, Maze method is a method that learn about one of some robot's movement, To cleans certain floor or surfaces, the robot need mapping method, so we use Maze method to do it, and PID (Proportional-Integral-derivative) method is used to control robot movement also, where by using this method is expected robot's movement to be stable against the a wall with a certain distance. In this study was prepared a floor/surface with dimension 200cm x 200cm to be cleaned, by using Maze method, the above dimension to be divided into 8 columns and 8 rows, each column and row have 8 index. Automatic Robot Vacuum Cleaner will start to cleans all the columns and rows, then back to started point. The result of experiments and Analyzes can be concluded, that the Automatic Robot Vacuum Cleaner can cleans all the surface and back to started point with percentage of success is 100% and percentage of a steady state error is 6,66 % (was controlled by using PID method).

Keywords: The Automatic Robot Vacuum Cleaner, Method Maze,PID method, columns, rows

1. PENDAHULUAN

Membersihkan lantai dari debu adalah salah satu pekerjaan yang wajib dilakukan oleh kebanyakan ibu rumah tangga. Kegiatan tersebut sangat menyita waktu terutama bagi ibu rumah tangga yang berkarier. Oleh karena itu beberapa perusahaan elektronik memberikan solusi berupa alat elektronik pembersih debu atau *vacuum cleaner*. Pada saat ini pasar telah banyak menawarkan *vacuum cleaner* manual, jadi harus ada orang yang menggerakkan peralatan tersebut. Sementara itu masih ada aktivitas lainnya yang harus diselesaikan seperti; berbelanja, mengantar anak ke sekolah, mengerjakan proyek, belajar dan lain-lain.

Berdasarkan dari hal tersebut diatas, maka peneliti merancang dan membuat robot *vacuum cleaner automatic* yang dapat bergerak secara mandiri saat membersihkan lantai rumah dari debu. Robot *vacuum cleaner automatic* yang telah diproduksi industri merupakan *vacuum cleaner* yang menggunakan metode *wall follower* dan *random*. Hal ini memiliki kelemahan yaitu *vacuum cleaner* tersebut tidak dapat mengenali daerah mana yang telah dibersihkan, sehingga dapat menyebabkan pemborosan waktu dan energi [4].

Untuk memperbaiki kinerja dari robot *vacuum cleaner automatic* yang sudah ada, maka penelitian ini merancang dan membuat prototipe robot *vacuum cleaner automatic* dengan menggunakan metode *Maze* dan sensor jarak sebagai nilai masukan. Nilai dari sensor jarak didefinisikan sebagai nilai jarak *robot* terhadap dinding dengan ukuran ruangan maksimal 5 x 5 meter, dengan nilai jarak tersebut akan membentuk suatu pola yang digunakan sebagai *rule* yang harus dilalui oleh robot tersebut, dengan demikian pada saat robot *vacuum cleaner* berjalan, robot *vacuum cleaner* akan melakukan *record* dan memetakan lokasi yang telah dilewati, sehingga robot tidak akan melewati daerah yang telah dilewatinya [2]. Agar *vacuum cleaner automatic* ini dapat melakukan pergerakan dan berjalan sesuai penjelasan diatas, maka dibutuhkannya motor DC sebagai penggerak roda.

Pengontrolan motor DC yang digunakan pada robot *vacuum cleaner automatic* ini menggunakan metode PID. Kontrol PID merupakan unsur penting dari sebuah sistem kontrol yang tertanam dengan tujuan khusus sesuai kebutuhan pada kontrol sistem. PID kontrol sering dikombinasikan dengan logika, fungsi sekuensial, penyeleksian dan blok fungsi sederhana untuk membangun sistem otomatis yang sangat kompleks [3]. Algoritma PID terdiri dari tiga mode dasar yaitu *Proporsional, Integral* dan *Derivatif* , hasil penggabungan dari semua mode dasar tersebut menjadi parameter *output* [1].

Robot *vacuum cleaner automatic* yang dirancang dan dibuat pada penelitian ini menggunakan metode *mapping maze* untuk melakukan pemetaan ruangan , dan kontrol PID yang digunakan untuk mengontrol pergerakan motor DC agar robot dapat berjalan dengan stabil, dengan menggunakan sensor jarak sebagai masukan yang akan diproses untuk pemetaan dan juga pergerakan robot.

2. TINJAUAN PUSTAKA

A. Robot

Robot merupakan sebuah alat yang dapat melakukan tugas fisik, baik bekerja secara manual maupun otomatis, Istilah robot berasal dari bahasa Cheko *"robota"* yang berarti pekerja yang tidak mengenal lelah atau bosan. Robot dapat digunakan dalam berbagai hal, yang memiliki sistim yang berbeda pada setiap fungsi yang berbeda. Salah satu contoh aplikasi dari robot adalah kemampuan membersihkan debu dari lantai rumah.

Pengertian robot banyak diartikan secara berbeda – beda yang mana setiap sumber yang berbeda memiliki arti yang berbeda pula. Berikut pengertian robot yang berasal dari beberapa sumber :

- Kamus Webster

"Robot is An automatic device that performs function ordinarily ascribed to human beings".

- Kamus Oxford

"Robot is A machine capable of carrying out a complex series of

actions automatically, especially one programmed by a computer".

- Robot Institute of America

"*Robot is A reprogammable multifunctional manipulator designed to move materials, parts, tools or other specialized devices through variable programmed motions for the performance of a variety of tasks".*

- International Standard Organization (ISO 8373)

" Robot is An automatically controlled, reprogrammable, multipurpose, manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications".

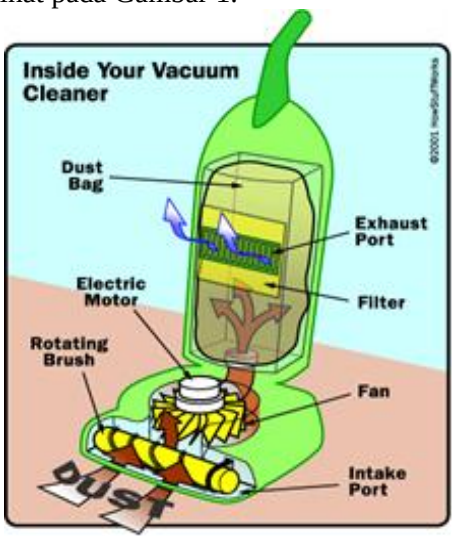
Berdasarkan beberapa definisi diatas, menunjukkan bahwa robot tidak dapat diartikan secara mutlak, tergantung dari sudut pandang dan funsional terhadap robot yang dibuat.

B. *Mobile Robot Kinematics*

Mobile robot kinematics adalah salah satu ilmu yang mempelajari bagaimana sistem mekanisme pergerakan pada robotik. Sesuai dengan namanya *Mobile* maka yang akan dibahas pada sub-bab ini adalah pergerakan roda yang digunakan. Pada penelitian ini, dirancang *vacuum cleaner otomatic* dengan menggunakan 3 roda, dengan alasan bahwa tiga roda sudah cukup efisien dilihat dari struktur lantai yang dilewati. *Mobile* robot 3 roda memiliki 2 roda yang tetap (*fixed*) pada porosnya dan menyatu langsung terhadap as motor serta terdapat 1 (satu) *castor wheel* yang digunakan pada bagian depan. Dengan konfigurasi tersebut, robot dapat memiliki keseimbangan dalam pergerakan robot pada saat berjalan maupun bermanuver.

C. *Vacuum Cleaner*

Vacuum cleaner merupakan suatu perangkat yang bekerja dengan menggunakan pompa udara untuk menciptakan *vacuum parsial* sebagai penghisap debu dan kotoran yang menempel dikarpet atau di lantai. Cara kerja dari *vacuum cleaner* yaitu memanfaatkan perbedaan tekanan udara, dimana udara akan mengalir pada tekanan udara yang lebih tinggi ke tekanan udara yang lebih rendah. Tekanan udara yang terdapat di dalam vacuum cleaner dikurangi oleh kipas, sehingga terjadi *vacuum* (ruang hampa), dengan demikian tekanan atmosfir akan mendorong udara luar ke dalam *vacuum cleaner* sehingga debu akan ikut terhisap melalui penghisap *(intake port)* melewati penyaring (*filter*) dan masuk ke dalam kantong debu (*dust bag*) yang terdapat di dalam *vacuum cleaner*. Penggambaran *vacuum cleaner* dapat dilihat pada Gambar 1.



Gambar 1. Bagian Dari *Vacuum Cleaner*
Sumber : http://wynwindu.blogspot.de

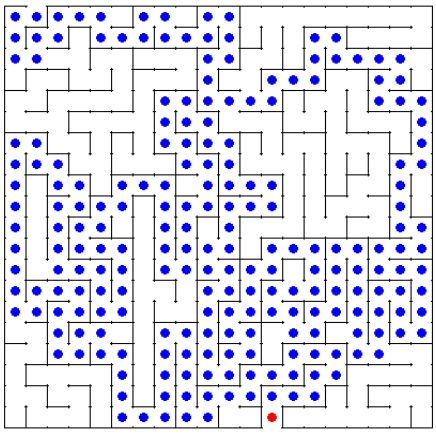
D. Metode *Maze*

Maze adalah suatu rute jalan yang rumit. Pada bidang robotika ada dua jenis *maze* yang umum digunakan, yaitu *wall maze* dan *line maze*. *Wall maze* pada umumnya dikenal dengan istilah labirin, yakni suatu rute jalan yang terbentuk atas lorong-lorong dengan dinding tanpa atap. Permasalahan yang timbul pada *maze* adalah cara untuk mendapatkan jalur terpendek, sehingga dibutuhkan metode untuk menyelesaikannya. *Maze mapping* merupakan algoritma yang digunakan untuk memecahkan *maze*, yakni mencari dan menggambarkan peta dari *maze* [5].

Maze mapping atau lebih umum dengan istilah *path mapping* yang konsep dasar dalam pencariannya mengikuti aturan *wall follower* (pada robot *wall follower*) atau *left/right hand rule* (pada robot *line tracer*). *Path mapping* merupakan mode *maze mapping* yang digunakan pada robot *wall*

follower, biasanya oleh robot tikus. Algoritma ini merupakan *the basic algorithm* [6]. *Path mapping* digunakan untuk berjalan mengikuti dinding kiri atau dinding kanan pada proses memetakan *maze*. Selanjutnya, bila peta yang sudah dibuat tersebut dijalankan, maka robot bisa mengenali bentuk dari ruangan dan dapat mengetahui arah di setiap perjalanan robot tersebut.

Ada 6 algoritma *maze solving* yang berbeda, dimana setiap algoritma akan menyelesaikan *maze*. Akan tetapi setiap algoritma memiliki kelemahan dan kelebihan yang berbeda satu dengan yang lainnya. Selanjutnya ke-6 algoritma *maze solving* akan dijelaskan seperti Gambar 2.



Gambar 2. Ilustrasi *Maze*
Sumber : http://rosettacode.org

a. *Random Mouse*
Random Mouse adalah metode sederhana yang dapat diimplementasikan oleh robot yang tidak cerdas atau bahkan seekor tikus sekalipun. Metode ini, secara sederhana mengikuti jalur yang ada sampai menemukan persimpangan. Pilihan atas persimpangan yang ada diputuskan secara acak. Meskipun secara teoritis, metode tersebut pasti menemukan solusi, ada juga kemungkinan bahwa metode ini tidak mendapatkan solusi apapun. Karena dapat melalui jalur yang sama beberapa kali, metode ini sangatlah lambat.

b. *Wall Follower*
Wall Follower adalah metode yang paling terkenal dalam penyelesaian *maze*. Metode ini juga dikenal dengan aturan tangan kiri *(left-hand rule)* atau aturan tangan kanan *(right-hand rule)*. Jika *maze* secara sederhana dihubungkan, sehingga semua dindingnya terkoneksi satu sama lain, maka dengan menjaga satu tangan dengan kontak terhadap salah satu dinding *maze* akan ditemukan ujung keluar, atau jika tidak ada akan berakhir kembali ke jalan masuk. Hal ini dimungkinkan karena apabila semua dinding tersambung, maka akan terbentuk sebuah *loop* atau lingkaran. Dengan kata lain, ketika mengikuti seluruh dinding, lingkaran tersebut akan menunjukkan jalur dari *start* ke *finish*.

c. *Pledge*
Algoritma ini dirancang untuk menyelesaikan *maze* dengan *start* yang berada di dalam *maze*. Arah menjadi faktor utama dalam penyelesaian *maze* dengan algoritma ini karena pendeteksian jalur yang salah diatasi dengan penghitungan arah. Hal ini memungkinkan penyelesaian sebuah *maze* oleh seorang pemain yang menggunakan kompas dan ingin keluar dari dalam *maze*. Namun tidak dapat digunakan untuk arah sebaliknya, ketika pemain tersebut mulai dari luar *maze* dan ingin mencapai *goal* yang berada di dalam *maze*.

d. *Tremaux*
Algoritma yang ditemukan oleh Charles Pierre Tremaux ini, merupakan metode yang efisien untuk menemukan jalan keluar dari suatu *maze*. Metode ini dilakukan dengan menggambar garis pada lantai jalur. Semua *maze* yang telah terdefinisi sebelumnya pasti bisa diselesaikan oleh algoritma ini. Sebuah jalur yang ada dalam sebuah *maze*, dapat didefinisikan menjadi tiga, yaitu belum dilewati, ditandai satu kali, dan ditandai dua kali. Setiap kali arah dipilih, maka sebuah garis juga digambar di lantai.

Simpangan awal, arah akan dipilih secara acak. Ketika melalui sebuah simpang yang belum pernah dilalui sebelumnya, dipilih arah secara acak dan tandai jalur tersebut. Ketika menemukan simpang yang bertanda (sudah pernah dilewati) dan tanda yang ditemukan hanya satu, maka kembalilah ke arah sebelumnya dan tandai jalur untuk kedua kalinya. Selain itu, jika menemukan

simpang bertanda lebih dari satu, pilih arah dengan tanda yang paling sedikit. Ketika akhirnya solusi dicapai, maka jalur bertanda satu akan menjadi penunjuk jalur pulang ke *start*. Ketika tidak ada jalan keluar, maka algoritma ini akan membawa kembali ke *start* ketika semua jalur sudah ditandai dua kali. Setiap jalur akan dilalui tepat dua kali dengan arah yang berbeda. Jalur yang dihasilkan disebut juga *bidirectional double*.

e. *Dead-End Filling*

Algoritma ini menyelesaikan *maze* yang sudah diketahui jalurnya secara keseluruhan. Metode ini dapat digunakan untuk soal *maze* di kertas atau program komputer, tetapi tidak berguna untuk pemain yang berada di dalam *maze* yang tidak diketahui polanya. Metode ini dilakukan dengan menemukan seluruh jalan buntu dalam *maze* dan kemudian mengisi jalur dari setiap jalan buntu sampai simpang pertama ditemukan. Algoritma ini tidak dapat tiba-tiba memotong jalan dari *start* ke *finish* karena setiap langkah dari proses mempertahankan topologi dari *maze*. Pada *maze* sempurna, setelah algoritma ini selesai dijalankan, akan tersisa jalur langsung dari *start* ke *finish*. Sedangkan, pada *maze* yang memiliki beberapa *loop*, salah satu jalur yang mungkin dilalui akan tersisa.

f. *Shortest Path*

Ketika sebuah *maze* memiliki banyak solusi, mungkin saja yang diinginkan adalah jalur terpendek dari *start* ke *finish*. Algoritma ini menemukan jalur tercepat dengan mengimplementasikan *Breadth First search*. Hal ini dilakukan dengan menggunakan antrian untuk mengunjungi setiap sel dengan nilai yang berubah dari *start* ke *finish*. Setiap sel yang ditemui harus menyimpan nilai jarak dari *start* atau mengubah nilai karena sel-sel didekatnya yang bertambah. Ketika lokasi *finish* ditemukan, jalur pulang ke *start* sudah dapat terlihat sebagai jalur terpendek.

Dari keseluruhan algoritma tersebut, algoritma *Random Mouse*, *Wall Follower*, *Pledge* dan *Trémaux* dirancang untuk digunakan tanpa pengetahuan tentang *maze*. Sementara itu, *Dead-End Filling* dan *Shortest Path* dirancang untuk menyelesaikan *maze* yang secara keseluruhan sudah diketahui.

E. PID *Controller*

PID *Controller* (*Proposional Integral Derivatif*) merupakan suatu *generic controller* yang banyak digunakan pada bidang robotika. PID *Controller* berfungsi untuk memperbaiki kesalahan antara nilai *prosses* dan nilai *setpoint* yang ditentukan dan melakukan pembenaran hingga mendapatkan kesalahan yang seminimal mungkin [7].

Gambar 3 menunjukkan blok diagram PID, dimana PID *Controller* terdiri dari 3 bagian utama, yaitu *Proportional Controller* , *Integral Controller* , dan *Derivatif Controller*. *Proportional* yang menentukan nilai reaksi terhadap kesalahan saat ini. *Derivatif* menentukan nilai perubahan kesalahan yang terjadi dari kesalahan saat ini terhadap kesalahan sebelumnya. *Integral* menentukan hasil penjumlahan nilai kesalahan yang terjadi. Hasil yang diperoleh dengan proses PID, maka dapat diperoleh dengan persamaan 1.

$$u(t) = MV(t) = K_P * e(t) + K_I \int_0^t e(\tau)\,d\tau + K_D \frac{d}{dt} e(t) \quad (1)$$

dimana :

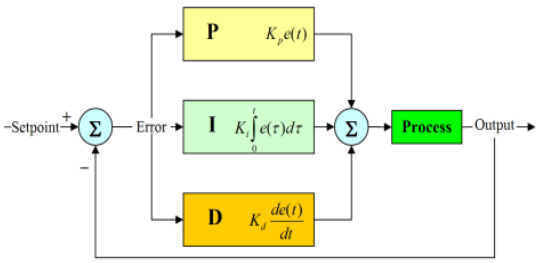
K_p merupakan konstanta *proportional.*

K_I merupakan konstanta *integral.*

K_D merupakan konstanta *derivatif.*

e_n merupakan besarnya nilai *error* saat ini

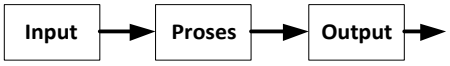
R_n merupakan besarnya nilai *output* saat ini setelah perhitungan



Gambar 3 . Blok Diagram PID
Sumber : http://radhesh.wordpress.com

3. METODE PENELITIAN

Gambar 4 menunjukkan konfigurasi sistem yang meliputi *input*, proses dan *ouput.*

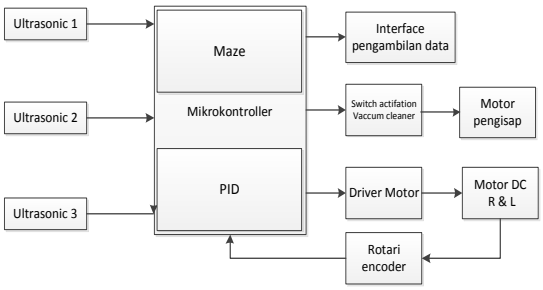


Gambar 4. Konfigurasi Sistem

Input merupakan parameter masukan yang diterima yang terdiri dari 3 buah sensor *ultrasonic* yang berfungsi sebagai sensor jarak, masukan dari 3 sensor jarak akan diproses oleh bagian proses yang menggunakan mikrokontroler.

Mikrokontroler menerima data masukan dari sensor jarak. Data yang berasal dari sensor jarak adalah jarak antara sensor terhadap halangan, dimana nilai jarak tersebut digunakan dalam pemetaan dan juga sebagai data masukan untuk pergerakan robot *vacuum cleaner automatic* tersebut.

Robot *vacuum cleaner automatic* ini bergerak menggunakan motor DC yang dilengkapi dengan *rotary encoder* sebagai masukan dalam pengukuran jarak tempuh robot. Semua data yang diterima dari masukan dan telah diolah oleh mikrokontroler, dapat ditampilkan melalui LCD 2x16 dan serial komunikasi yang dapat menghubungkan antara robot terhadap laptop. Blok diagram sistem robot dapat dilihat pada Gambar 5.



Gambar 5. Blok Diagram Sistem Robot

A. Perancangan Perangkat Keras

Perancangan perangkat keras terdiri beberapa bagian, diantaranya adalah :

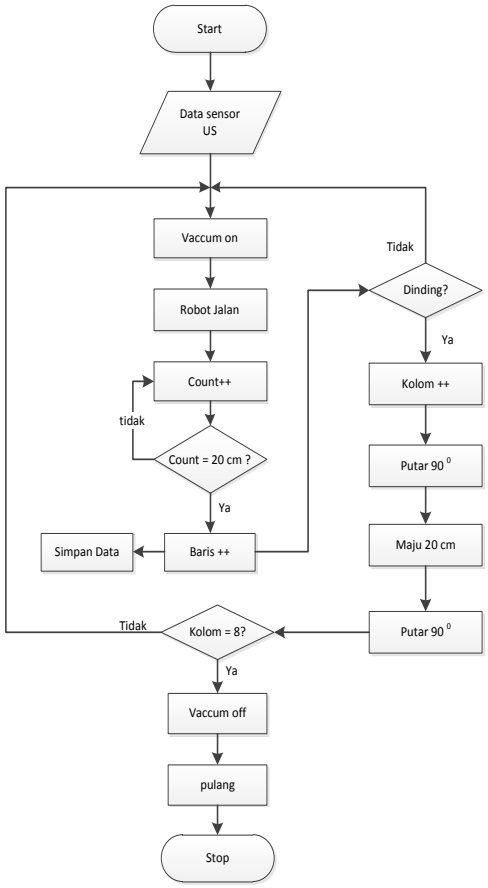
1. Rangkaian catu daya
2. Rangkaian Mikrokontroler
3. Rangkaian *Liquid Crystal Display (LCD)* 16x2 karakter
4. Rangkaian Serial RS232
5. Rangkaian Driver Motor H-*Bridge*
6. Rangkaian Penguat *Rotary Encoder*
7. Rangkaian *Switching Motor Vaccum*
8. Rangkaian Sensor Ultrasonik Terhadap Mikrokontroler

Perancangan perangkat keras ditunjukkan pada Gambar 6.

Gambar 6. Perancangan *Hardware*

B. *Flowchart*

Gambar 6 menunjukkan hasil perancangan perangkat kerasnya, sedangkan *flowchart* akan menjelaskan alur dari sistim kerja robot *vacuum cleaner* yang dirancang. Dimulai dari robot *vacuum cleaner* diaktifkan hingga robot *vacuum cleaner* kembali ke posisi awal setelah menyelesaikan tugasnya. Lebih jelas dapat dilihat pada Gambar 7.



Gambar 7. *Flowchart* Sistem

Robot *vacuum cleaner* mulai diaktifkan, robot akan menerima masukan dari sensor jarak. Selanjutnya robot akan berjalan lurus (apabila tidak ada halangan) dan melakukan penghisapan debu serta mememori jalan yang telah dilaluinya, selanjutnya robot akan melakukan pemetakan ruangan dengan menambahkan baris (dalam penelitian ini digunakan ukuran pemetakan 8 kolom dan 8 baris, setiap kolom dibagi lagi menjadi 8 indeks, begitu juga dengan barisnya dibagi menjadi 8 indeks), sehingga setelah 8 baris maka robot akan bergerak dan berputar 90˚ menambah kolom dan bergerak 20 cm berputar 90˚ lagi selanjutnya akan menghitung baris lagi begitu seterusnya sampai di kolom ke-8 dan baris ke-8 robot akan otomatis mencari jalan tercepat untuk kembali ke posisi awal/penyimpanan.

C. Perancangan *Software*

Perancangan *software* merupakan suatu perancangan program yang digunakan untuk mengontrol robot dan juga membuat suatu aplikasi. Adapun perancangan *software* ialah sebagai berikut :

a. Program AVR
   1. Program LCD
   2. Program Komunikasi Serial
   3. Program Pembacaan Sensor Ultrasonik
   4. Program *Rotary Encoder*
   5. Program *Switching Motor Vaccum*
   6. Program Kontrol PID
   7. Program *Maze*

b. Program VB.NET 2008
   1. Program penerimaan data
   2. Program penyimpanan data
   3. Tampilan

4. HASIL DAN PEMBAHASAN

A. Pengujian PID

Pengujian PID ini merupakan suatu pengujian terhadap respon pergerakan robot dalam mencapai nilai *set point* yang ditentukan, yang ditunjukkan pada Gambar 8. Pada pengujian ini maka kita dapat mengetahui besarnya *error steady state* yang dihasilkan dan dapat menentukan nilai konstanta yang baik dari hasil percobaan. Adapun alat yang dibutuhkan dalam pengujian ini adalah sebagai berikut :

1. Sensor *Ultrasonic*
2. Robot *vacuum cleaner automatic*
3. Laptop
4. *Serial Port*

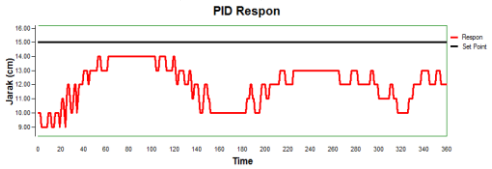
Setelah peralatan diatas disediakan, maka kita dapat melakukan pengujian PID yang dilakukan seperti langkah – langkah berikut ini.

1. Gunakan sensor *ultrasonic* sebagai masukan pada pengujian
2. Robot *vacuum cleaner automatic* telah terdapat program kontrol PID
3. Atur konfigurasi nilai K_P, K_I dan K_D pada robot *vacuum cleaner*
4. Hubungkan robot dengan laptop dengan koneksi serial *port* untuk menampilkan data dan melihat respon robot pada aplikasi yang dibuat pada VB.net 2008
5. Lakukan percobaan sebanyak 9 kali dengan konfigurasi nilai K_P , K_I dan K_D yang berbeda.

Gambar 8. Prosedur Pengujian PID

Setelah dilakukan nya pengujian PID, maka hasil dari pengujian dapat dilihat pada Gambar 9 sampai 17.

1. Nilai $K_P = 5$, $K_I = 0$, dan $K_D = 0$

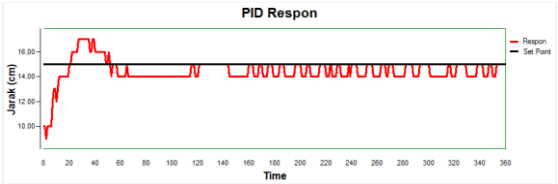


Gambar 9. Respon $K_P = 5$, $K_I = 0$, $K_D = 0$

2. Nilai $K_P = 15$, $K_I = 0$, dan $K_D = 0$

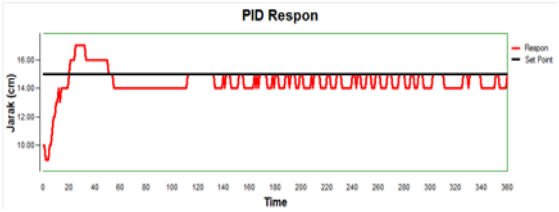
Gambar 10. Respon $K_P = 15$, $K_I = 0$, $K_D = 0$

3. Nilai $K_P = 25$, $K_I = 0$, dan $K_D = 0$



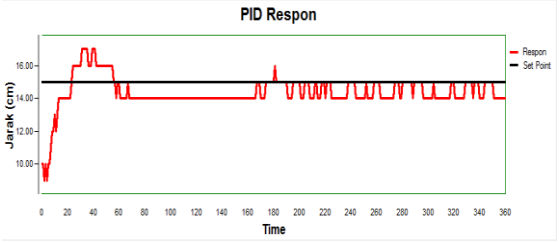
Gambar 11. Respon $K_P = 25$, $K_I = 0$, $K_D = 0$

4. Nilai $K_P = 25$, $K_I = 0$, dan $K_D = 5$



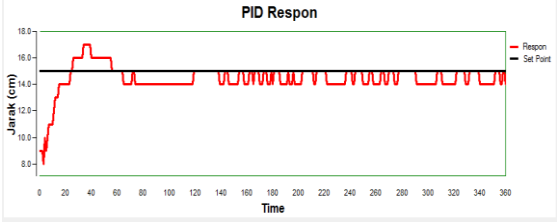
Gambar 12. Respon K_P = 25 , K_I = 0, K_D = 5

5. Nilai K_P = 25, K_I= 0, dan K_D = 10



Gambar 13. Respon K_P = 25 , K_I = 0, K_D = 10

6. Nilai K_P = 25, K_I= 0, dan K_D = 20

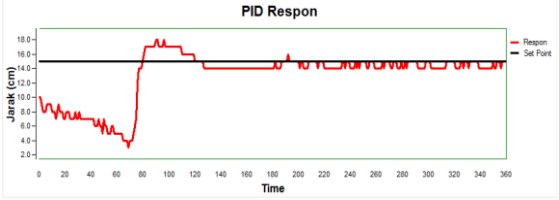


Gambar 14. Respon K_P = 25 , K_I = 0, K_D = 20

7. Nilai K_P = 25, K_I= 0,03 , dan K_D = 5

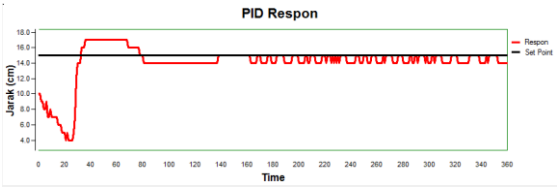
Gambar 15. Respon K_P = 25 , K_I = 0,03 , K_D = 5

8. Nilai K_P = 25, K_I= 0,5 , dan K_D = 5



Gambar 16. Respon K_P = 25 , K_I = 0,5 , K_D = 5

9. Nilai K_P = 25, K_I= 1 , dan K_D = 5



Gambar 17. Respon K_P = 25 , K_I = 01 , K_D = 5

Berdasarkan dari grafik respon PID (Gambar 9 sampai Gambar 17) dapat dilihat nilai *error steady state* yang bervariasi, tergantung dari setiap konfigurasi nilai K_P, K_I dan K_D. Dari beberapa konfigurasi nilai K_P, K_I dan K_D didapatkan bahwa nilai yang terbaik dapat dilihat pada saat nilai konfigurasi K_P = 25, K_I= 0.03 dan K_D = 5.

Dapat diketahui dari hasil percobaan, respon PID yang baik, merupakan respon PID yang memiliki nilai *error steady state* yang paling kecil, maksud dari nilai *error steady state* yang kecil yaitu mendekati nilai *set point* yang ditentukan dan mencapai nilai *set point* tercepat. Dari hasil penelitian yang telah dilakukan respon PID pada nilai K_P = 25, K_I= 0.03 dan K_D = 5 memiliki nilai *error steady state* sebesar 16, nilai tersebut didefinisikan sebagai nilai terkecil dikarenakan nilai tersebut mendekati nilai *set point*-nya, dimana perhitungan dari *Error* dirumuskan pada persamaan 2.

Error = [*error steady state- set point*]/*set point* *100% (2)

Dengan menggunakan persamaan 2 maka nilai *error* pada saat nilai K_P = 25, K_I= 0.03 dan K_D = 5 adalah 6,66% . Selanjutnya perhitungannya dengan menggunakan persamaan 2 adalah sebagai berikut :

Error = ([16 – 15]) / 15 * 100%
= 6,66 %

Sedangkan nilai *error* terbesar terjadi pada konfigurasi nilai K_P=25, K_I=0.5, K_D=5 dimana nilai *error steady state*-nya adalah 4, dengan menggunakan persamaan 2 di atas didapatkan :

Error = ([4 – 15]) / 15 * 100%
= 73,33 %

Respon dengan nilai *error steady state* 6,66% merupakan respon yang tercepat dari percobaan yang telah dilakukan pada penelitian ini, saat mencapai *set point* 15cm pada waktu ke 0.6 detik.

Berdasarkan percobaan yang telah dilakukan didapatkan bahwa semakin besar nilai K_P yang ditentukan akan didapatkan nilai K_P terbaik, dan respon yang dihasilkan semakin baik. Nilai K_D dapat mempercepat respon untuk mencapai *set point* hingga

stabil. Nilai K_I dapat mengurangi nilai *error steady state* pada *set point* seperti yang ditunjukkan dari hasil pengujian, namun apabila salah dalam menentukan nilai K_I maka akan membuat respon menjadi tidak baik.

B. Pengujian Maze

Pengujian *maze* adalah pengujian jalannya robot, seperti ditunjukkan pada Gambar 19. Pentingnya pengujian *maze* ini, agar mengetahui agar dapat dketahui apakah robot sudah berjalan sesuai dengan *rule* yang telah ditentukan atau sebaliknya. *Rule* yang dimaksud adalah jalan atau rute yang akan dilalui oleh robot pada baris dan kolom yang dibentuk dari suatu ruangan, sehingga pada saat *robot* berada pada baris dan kolom terakhir (titik akhir), maka robot akan mengambil jalan pintas untuk kembali pada titik awal *robot* bekerja. Adapun alat yang digunakan pada pengujian ini adalah sebagai berikut :

1. Robot *vacuum cleaner automatic*
2. Laptop
3. Serial *Port*

Gambar 18. Prototipe Robot *Vacuum Cleaner Automatic*

Setelah menyiapkan peralatan yang dibutuhkan, maka kita dapat melakukan pengujian *maze* pada penelitian ini. Area yang akan dibersihkan adalah ruangan 200 cm x 200 cm dan menggunakan 8 baris dan 8 kolom dengan delapan indeks di setiap kolom dan barisnya seperti yang ditunjukkan pada Tabel 1. Langkah selanjutnya dapat dilakukan sesuai prosedur berikut ini :

1. Hubungkan laptop pada robot dengan komunikasi serial untuk pengambilan data *maze*.
2. Letakkan robot pada posisi awal yang telah ditentukan.
3. Robot akan berjalan dan memberi nilai baris dan kolom setiap 20 cm
4. Setelah robot melalui seluruh ruangan dan balik ke titik awal hingga, selanjutnya menyimpan semua data yang dihasilkan seperti yang ditunjukkan pada Tabel 1.

Gambar 19 : Prosedur Pengujian *Maze*

Tabel 1 : *Rule Robot*

Baris \ Kolom	1	2	3	4	5	6	7	8
1	16	48	70	76	93	119	143	163
2	21	36	58	79	105	121	147	157
3	24	44	74	84	103	124	134	161
4	19	33	62	91	109	128	149	164
5	22	35	53	82	97	119	145	160
6	18	40	57	73	108	120	141	153
7	20	38	76	91	103	123	147	166
8	19	43	64	87	101	118	139	164

Dengan dilakukannya pengujian ini, maka dapat dilihat hasil dari penentuan *rule* dan jalan balik ke titik awal pada pengujian *maze* dapat dilihat sebagai berikut ini :

1. Data *Rule* jalannya robot
   Berdasarkan Tabel 1 , maka dapat dilakukannya perhitungan sebagai berikut :
   L. Ru = 200 cm x 200 cm
   L. Bot = 160 cm x 160 cm
   LT.Bot = L.Ru – L.Bot
   = 40000 – 25600
   = 14400
   %LT.Bot = LT.Bot / L.Ru * 100
   = 14400 / 40000 * 100
   = 36 %
   dimana :
   L.Ru = Luas ruangan
   L.Bot = Luas daerah yang dilalui robot
   LT.Bot = Luas daerah yang tidak dilalui robot
   %LT.Bot = Persentase Luas daerah yang tidak dilalui robot

Sehingga persentase luas daerah ruangan yang telah dilalui oleh robot dapat dirumuskan seperti persamaan 3.

%L.Bot = 100%-%LT.Bot (3)

Dengan menggunakan persamaan 3 maka persen daerah yang dilalui robot sebesar 64%.

1 kotak = 20 cm x 20 cm
2. Data jalan balik ke titik awal

Berdasarkan Tabel 1 robot tidak melalui ruangan sebesar 36 % dari luas ruangan sehingga ruangan yang dilalui sebesar 64% dari luas ruangan. Daerah yang tidak dilalui dikarenakan *set point* awal = 20 cm, sehingga robot berkerja langsung berada pada jarak dinding 20 cm, hal tersebut dilakukan agar robot tidak menabrak dinding.

Robot melewati *rule* dengan 8 baris dan 8 kolom dengan masukan nilai dari jarak pembacaan *sensor ultrasonic* terhadap dinding dengan menentukan *set point* setiap kolomnya dan data masukan dari *rotary encoder* dilakukan untuk menentukan posisi baris dengan nilai jarak yang dihasilkan pada setiap jarak 20 cm.

Maze yang dilakukan pada saat jalan balik menuju titik awal hanya melalui baris 1 seperti yang ditunjukkan pada Tabel 2, sehingga robot tidak melawati daerah lainnya atau melalui *rule* awal dengan melewati semua baris dan kolom untuk mencapai titik awal robot. Pemilihan jalan balik robot menggunakan sama dengan metode *maze wall follower*, agar mempermudah menemukan titik awal robot berkerja.

Tabel 2. : Jalan Balik Ke Titik Awal
(Sumber : Data primer)

Baris \ Kolom	1	2	3	4	5	6	7	8
1	20	40	60	80	100	120	140	160
2								
3								
4								
5								
6								
7								
8								

5. KESIMPULAN

Dari hasil percobaan dan analisa pada penelitian ini kesimpulan yang bisa diambil adalah sebagai berikut :

1. Robot *vacuum cleaner dapat* bekerja sesuai dengan tujuan dari penelitian yaitu dapat bekerja dengan mandiri dan berhenti dengan sendirinya setelah melaksanakan tugasnya kita sehingga tidak perlu kita yang mengendalikan dan mengawasi robot tersebut pada saat berkerja.
2. Robot *Vacuum cleaner automatic* mengidentifikasikan setiap 20cm x 20cm bidang dengan membuat pola baris dan kolom yang dilewati.
3. Penerapan metode PID pada robot *vacuum cleaner automatic*, dapat membuat robot berjalan lurus dan stabil dengan nilai *error steady state* yang terkecil sebesar 6,66 %.
4. Penerapan metode *maze* dengan penentuan *rule* pada jalannya robot, dapat membuat robot berjalan dengan teratur hingga mencapai titik akhir dengan persentase daerah yang dilalui sebesar 64% dan dapat kembali ke titik awal dengan mengambil jalan singkat dengan melewati baris 1 dengan sistem *wall follower*.

DAFTAR PUSTAKA

[1] Dr. M. J. Willis *(1998, November 17). "Proportional-Integral Derivative Control". University of Newcastle.*

[2] Iqbal, N.M. dan Hendriawan, A. dan Akbar, R. , (2010), "Penerapan Algoritma *Maze Mapping* Untuk Menyelesaikan *Maze* Pada *Line Tracer*". Surabaya:PENS-ITS Sukolilo.

[3] Johan, A.K., (2002), *" Control System Design". University of California.*

[4] R. Ulrich,Iwan and Mondada, Francesco and Nicoud, J.-D. (1997)*," Autonomous vacuum cleaner". Swiss: Federal Institute of Technology.*

[5] Mishra, Swati. (2008), "*Maze Solving Algorithm for Micro Mouse*". IEEE

International Conference on Signal Image Technology and Internet Based Systems .

[6] Wahyu, D.H., Thomas. Wahyu, A.P., (2004), "Analisis dan Desain Sistem Kontrol dengan Matlab". Yogyakarta.

[7] Frederick , Shuwanto, F., Stefen (2009), "*Mobile Robot Navigation Using Depth First Search Algorithma*". Universitas Bina Nusantara.