BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Pada beberapa penelitian terdahulu telah dilakukan penelitian terhadap topik yang sama yaitu mengenai robot *maze* dan robot untuk perlombaan KRPAI. Dari beberapa penelitian terdahulu ini dapat dijadikan sebagai referensi sekaligus pembanding dalam melakukan penulisan dan penelitian tugas akhir ini.

Qomaruddin, Alasiry, dan Tamami (2017) melakukan penelitian dengan judul "*Routing Algorithm in Legged Robot with Dynamic Programming and Monte Carlo Localization*". Pada penelitian ini berfokus membahas algoritma *routing* pada *mobile* robot berkaki enam (*hexapod*) untuk misi KRPAI (Kontes Robot Pemadam Api Indonesia). Robot *hexapod* pada KRPAI memiliki misi berjalan di dalam *maze* untuk mencari sebuah titik api dan kemudian memadamkannya. Algoritma *Routing* pada penelitian ini mengusulkan optimasi baru untuk navigasi robot *hexapod* KRPAI, dimana algoritma tidak hanya dirancang untuk menentukan rute jalur terpendek yang efisien berbasiskan metode *routing* (*mapping),* melainkan juga merancang algoritma agar sistem robot mampu melakukan estimasi perbaikan (koreksi) terhadap posisi dirinya berada ketika robot mendapat *error* ketika melakukan *mapping* di dalam *maze*. Metode yang digunakan untuk *mapping* adalah *monte carlo localization,* kemudian untuk melakukan koreksi saat terjadi error ketika melakukan *mapping* digunakan metode *dynamic programming*. Kedua metode ini digunakan dengan tujuan untuk memperkuat navigasi robot dalam misi mencari titik api dan kemudian robot dapat kembali ke posisi awal (*home*) setelah memadamkan api. Hasil yang didapatkan pada peneltian ini yaitu algoritma yang dirancang telah sukses dan berhasil dimplentasikan pada 8 kali percobaan pada *maze* aturan KRPAI 2015.

Hidayatullah, Handayani, dan Fuady (2017) telah melakukan penelitian dengan judul *"Performance Analysis of A* Algorithm to Determine Shortest Path of Fire Fighting Robot"*. Algoritma A**(A Star)* adalah salah satu algoritma pencarian tercepat atau rute terpendek yang menggabungkan antara metode

uniform Cost search dengan algoritma *greedy best first search.* Pada penelitian ini melakukan analisis perbandingan kinerja dari penerapan algoritma A* dengan algoritma *dijkstra* pada misi robot KRPAI (Kontes Robot Pemadam Api Indonesia) dalam menentukan rute (jalur) terpendek. Kriteria yang digunakan pada penelitian ini mengacu pada *time complexity* dan *space complexity*. *Time complexity* adalah waktu yang dibutuhkan oleh algoritma untuk memproses perintah, sedangkan *space complexity* adalah ruang memori yang digunakan oleh algoritma untuk memproses perintah. Dalam implementasi algoritma A* pada robot KRPAI di dalam *map* KRPAI 2016, di dapatkan hasil performa yang lebih unggul untuk menentukan jalur terpendek dibandingkan dengan algoritma *dijkstra.* Pada algoritma A* diperoleh rata-rata eksekusi waktu sebesar 4270.72 ms dengan konsumsi memori sebesar 16768 kB sedangkan pada algoritma *dijkstra* diproleh rata-rata eksekusi waktu sebesar 6757 ms dengan konsumsi memori sebesar 40992 kB

Mishra (2008) melakukan penelitian dengan judul *"Maze Solving Algorithm for Micromouse*". Pada penelitian ini membahas salah satu metode kecerdasan yaitu berupa algoritma pengambil keputusan pada robot *micromouse* untuk memecahkan labirin (*maze solving*) dengan waktu yang cepat dan singkat. Algoritma yang dirancang berbasiskan algoritma *simple wall following* (pengikut dinding dasar) yang kemudian dikembangkan pada algoritma *flood fill* dan algoritma *dijkstra.* Pada hasil akhir, dari perbandingan terhadap ke tiga algoritma yang telah dirancang tersebut, diperoleh waktu tempuh robot tercepat dengan menggunakan algoritma *flood fill*, dengan waktu sebesar 266 detik.

Sitepu (2013) melakukan penelitian dengan judul "*Sistem Navigasi Maze Mapping pada Robot Beroda Pemadam Api".* Pada penelitian ini membahas suatu metode untuk memcahkan labirin (*maze*) dengan teknik *maze mapping* agar menghasilkan jalur yang dikehendaki dan mampu memecahkan jalur *maze* pada arena Kontes Robot Pemadam Api Indonesia beroda tahun 2013. Metode yang digunakan adalah navigasi *wall follower* dengan cara memilih metode *left wall follower* atau *right wall follower* di arena apabila robot telah melewati posisi *check point* yang telah ditentukan sebagai indikator *mapping* navigasi robot. Seluruh pergerakan serta navigasi robot telah ditentukan berdasarkan *mapping* posisi *check*

point tersebut di dalam program robot. Hasil penelitian yang didapatkan yakni algoritma *maze mapping* yang dirancang berhasil direalisasikan. Dari data pengamatan robot dapat menyelesaikan misi dengan waktu tempuh rata-rata 49,8 detik. Dari 70 percobaan terhadap 14 jenis konfigurasi lapangan robot berhasil menyelesaikan misi dengan persentase keberhasilan sebesar 62,8%.

Nugraha, Hendriawan, dan Akbar (2010) melakukan penelitian tentang "*Penerapan Algoritma Maze Mapping Untuk Menyelesaikan Maze Pada Line Tracer".* Pada penelitian ini, permasalahan dalam memecahkan (mencari jalan keluar) *maze* pada robot line *maze* diselesaikan menggunakan algoritma *maze mapping*. Pada algoritma yang diterapkan ini memiliki dua buah *mode*, yakni mode *search* dan mode *return*. Pada mode *search*, robot melakukan perjalanan dari posisi *start* menuju *finish* dengan aturan bahwa robot akan mengutamakan belok kiri bila menjumpai persimpangan. Kode-kode unik dibangkitkan setiap robot berjumpa dengan persimpangan. Pada mode *return*, robot sudah berjalan dari *start* menuju *finish* dengan jalur tependeknya. Jalur terpendek diperoleh dari kode-kode unik yang telah dikonversi. Hasil implementasi algoritma *maze mapping* yang telah dirancang sebagai sistem pencarian jalan keluar optimal dengan memiliki tingkat keberhasilan 90 %.

Tabel 2.1 Perbandingan dengan Penelitian Terdahulu

No	Nama Peneliti	Tahun	Judul Penelitian	Objek Penelitian	Metode
1.	Qomaruddin, M.C, Alasiry A.H, dan Tamami, N.	2017	"*Routing Algorithm in Legged Robot with Dynamic Programming and Monte Carlo Localization"*	Merancang metode *mapping* dan koreksi *error mapping* pada robot pemadam api *hexapod* di dalam *maze* arena KRPAI 2015	*Dynamic programmi-ng* dan *monte carlo localization*
2.	Hidayatullah, A.H, Handayani,	2017	*"Performance Analysis of A* Algorithm to*	Analisis menentukan jalur	*A* algorithm*

	A.N, dan Fuady, M.J		*Determine Shortest Path of Fire Fighting Robot"*	terpendek robot pemadam api *hexapod* pada *maze* arena KRPAI 2016	dan *dijkstra algorithm*
3.	Mishra, S	2008	*"Maze Solving Algorithm for Micromouse*"	Melakukan analisis penerapan algoritma *maze solving pada robot wall following micromouse*	*Simple wall following algorithm, flood fill algorithm,* dan *dijkstra algorithm*
4.	Sitepu, N.M	2013	"*Sistem Navigasi Maze Mapping pada Robot Beroda Pemadam Api"*	Merancang dan mengimple-mentasikan sistem navigasi baru untuk menutupi kelemahan navigasi *wall following* pada robot pemadam api beroda di dalam *maze* arena KRPAI beroda 2013	*Simple wall following algorithm dan maze solving (maze mapping) algorithm*
5.	Nugraha,M.I, Hendriawan, A, & Akbar,R	2010	"*Penerapan Algoritma Maze Mapping Untuk Menyelesaikan Maze Pada Line Tracer"*	Merancang algoritma pencarian jalan keluar *(finish)* pada robot *line tracer* di dalam *line maze*	*Simple line following algorithm, maze mapping algorithm,* dan *tuning* kendali PID

6.	Bintang Surya Tryatmojo	2020	*"Implementasi Algoritma Maze Mapping Pada Robot Pemadam Api Hexapod"*	Merancang dan mengimple-mentasikan algoritma baru *(maze mapping)* untuk menutupi kelemahan algoritma *wall following* pada robot pemadam api *hexapod* di dalam *maze* arena KRPAI 2018	*Simple wall following algorithm, tuning* Kendali *Proportional Derivative,* dan *maze mapping algorithm*

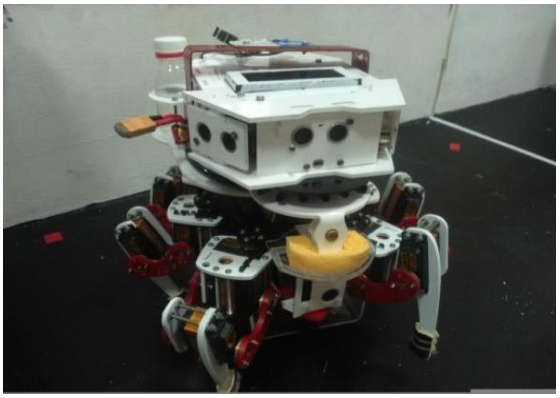
2.2 Dasar Teori

2.2.1 Robot *Hexapod*

Robot *hexapod* adalah salah satu jenis robot berkaki yang berbentuk menyerupai serangga yang bergerak dengan menggunakan enam buah kaki. Robot *Hexapod* dinilai memiliki fleksibilitas yang tinggi karena robot secara statis dapat berjalan dengan stabil dengan menggunakan tiga kaki secara bergantian berdasarkan ritme yang telah diatur dengan menggunakan formula *inverse kinematics*. (Khidir**,** 2014)

Robot *hexapod* tentu juga memiliki kelebihan dan kekurangan. Kelebihan yang dimiliki di antaranya: mampu bermanuver pada medan tanah yang tidak rata, memiliki kebebasan dalam bergerak, dan memiliki kestabilan dalam menompang tubuh robot karenanya robot *hexapod* berjalan dengan jumlah kaki yang sama pada setiap sisinya. Sedangkan kekurangan pada robot *hexapod* yaitu: dalam membangun robot dibutuhkan biaya riset yang cukup tinggi dan pergerakan robot akan sedikit lebih lamban dibandingkan dengan robot jenis beroda, karena pada

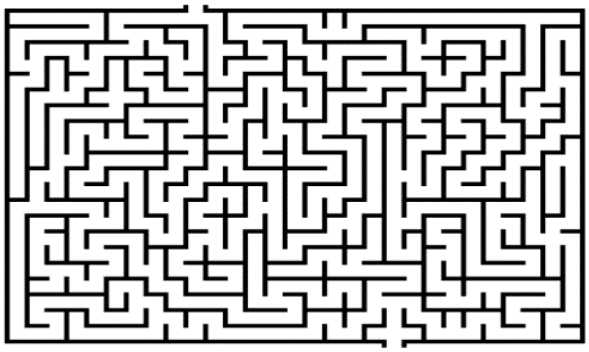
robot jenis ini diperlukan waktu untuk pengaturan koordinasi *gait* dari setiap masing-masing motor penggerak. (Purnama, 2017)



Gambar 2.1 *Prototype* Robot *Hexapod*

2.2.2 Labirin (*Maze)*

Labirin atau *maze* adalah sebuah *puzzle* dalam bentuk percabangan jalan yang kompleks dan memliki banyak jalan buntu. Tujuan di dalam permainan labirin adalah pemain harus mampu menemukan jalan keluar dari sebuah pintu masuk ke satu atau lebih pintu keluar. Kondisi lain pemain dapat memenangkan permainan ini yaitu ketika dia mencapai suatu titik di dalam labirin tersebut. Labirin pada umumnya terbagi menjadi beberapa kategori sesuai jenisnya, yaitu Labirin 2 dimensi, 3 dimensi, bentuk segitiga, *sigma*, dan masih banyak lagi. (Primanio,2007)



Gambar 2.2 Labirin (*Maze*)

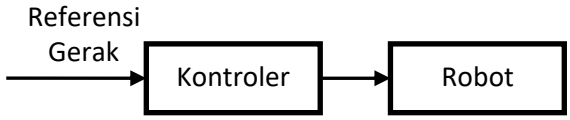
Sumber: (Primanio,2007)

2.2.3 Sistem Kontrol Robotik

Pada dasarnya, sistem kontrol robotik terbagi menjadi dua kelompok, yaitu sistem kontrol terbuka *(open loop)* dan sistem kontrol tertutup *(close loop).*

1. Sistem Kontrol *Open Loop*

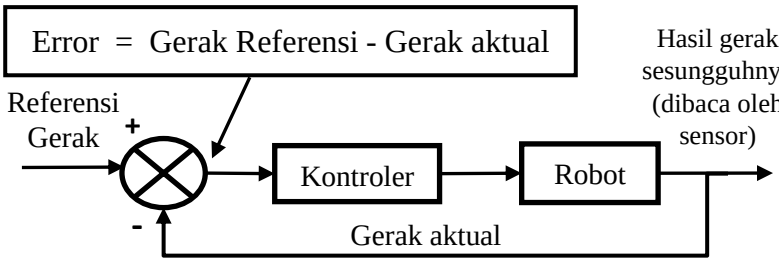
Sistem kontrol *open loop* adalah sistem kontrol yang outputnya tidak diperhitungkan ulang oleh kontroler. Keadaan apakah robot benar-benar akan mencapai terget sesuai *input* referensi yang telah diberikan tidak akan mempengaruhi kinerja dari kontroler. Diagram kontrol *loop* terbuka pada sistem robot dapat dinyatakan dalam diagram gambar 2.3 berikut ini.



Gambar 2.3 Diagram Sistem Kontrol *Open Loop*

2. Sistem Kontrol *Close Loop*

Sistem kontrol *close loop* merupakan sistem kontrol yang outputnya masih diperhitungan dan dibandingkan kembali terhadap *input* referensi yang diberikan.



Gambar 2.4 Diagram Sistem Kontrol *Close Loop*

Berdasarkan gambar 2.4 dapat diketahui bahwa jika hasil gerak aktual telah sama dengan referensi maka *input* kontroler akan sama dengan nol, artinya adalah bahwaa pada kontroler tidak lagi memberikan sinyal aktuasi kepada robot karena terget akhir perintah telah diperoleh. Jika didapatkan makin kecil *error* terhitung, maka semakin kecil pula sinyal pengemudi kontroler terhadap robot, sampai akhirnya mencapai kondisi tenang (*steady state*). (Pitowarno,2006)

2.2.4 Kontrol PID

Kontrol PID merupakan sistem kendali klasik yang terdiri atas 3 elemen yaitu *Proportional, Integral,* dan *Derivative.* Kontrol PID bekerja dengan memanfaatkan mekanisme umpan balik atau *feed back* pada sebuah sistem kendali. Kontrol PID secara kontinyu akan menghitung nilai *error* atau nilai kesalahan sebagai beda antara nilai *set point* yang diinginkan dengan nilai variabel terukur *(output),* supaya didapatkan nilai *error* yang seminimal mungkin.

1. Aksi Kontrol *Proportional*

Aksi kontrol *Proportional* bertindak sebagai *gain* atau penguat yang mampu mengubah *output* dari sebuah sistem secara proporsional tanpa memberikan efek dinamik pada kinerja pengendali tersebut. Persamaan respon dari aksi *proportional* dapat dinyatakan dengan persamaan:

$$u_{(t)} = K_p e(t) \quad \text{..... (2.1)}$$

Pengaturan aksi Kontrol *proportional* ini mampu digunakan untuk memperbaiki respon transien dari sebuah sistem, khususnya *rise time*, dan di samping itu juga mampu memperbaiki *settling time* dari sebuah sistem.

2. Aksi Kontrol *Integral*

Aksi kontrol *integral* adalah aksi kontrol yang dapat digunakan untuk memperbaiki respon tunak *(steady state)* dari sebuah sistem, sehingga aksi kontrol ini mampu memperkecil *error* dari sistem yang terlihat tampak *overshoot*.

$$u_{(t)} = K_i \int_0^t e(t)\, dt \quad \text{..... (2.2)}$$

Dengan melakukan pengaturan yang tepat pada konstanta *integral*, nilai *error steady state* dapat diperkecil dalam waktu yang lebih cepat,sehingga nilai *output* akan lebih cepat mengikuti *set point*.

3. Aksi Kontrol *Derivative*

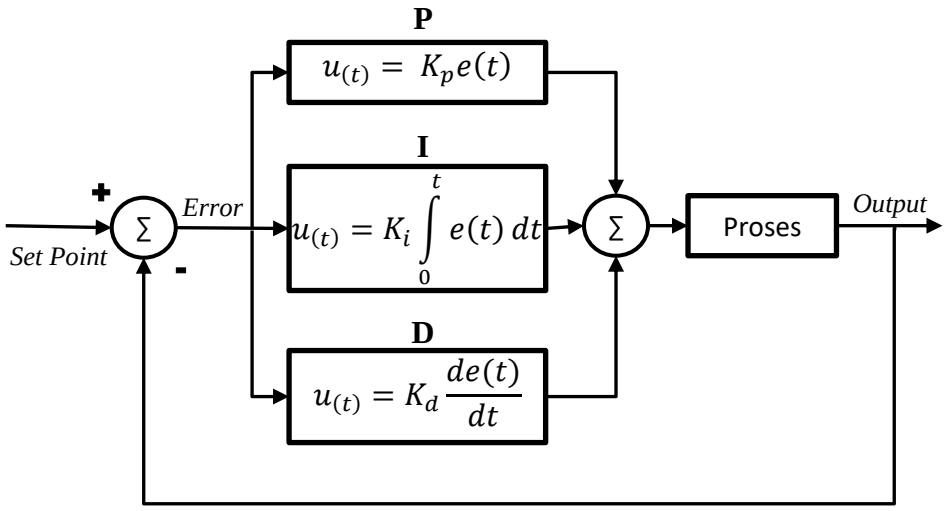
Aksi kontrol *derivative* adalah aksi kontrol yang dapat digunakan untuk memperbaiki respon transien dari sebuah sistem.

$$u_{(t)} = K_d \frac{de(t)}{dt}$$ ..... **(2.3)**

Selain mampu memperbaiki respon transien, aksi kontroler *derivative* juga mampu mengurangi nilai *overshoot* yang timbul akibat penggunaan aksi kontrol *integral*.

4. Gabungan Aksi Kontrol PID

Setelah mengetahui kelebihan dan kekurangan antara masing-masing aksi kontrol P, I, dan D, maka ketiganya dapat dikombinasikan sebagai sebuah kontroler PID untuk mendapatkan respon yang baik, kombinasi tersebut dapat dilihat seperti gambar 2.5 pada blok diagram berikut.



Gambar 2.5 Blok Diagram Gabungan Kontrol P,I, dan D

Berdasarkan blok diagram pada gambar 2.5, maka gabungan dari kontroler P,I, dan D dapat dituliskan seperti pada persamaan 2.4 berikut ini.

$$u_{(t)} = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt}$$ ..... **(2.4)**

5. Kontrol PID Digital

Kontrol PID digital adalah bentuk lain dari kontrol PID analog yang diprogram atau dijalankan menggunakan sebuah komputer atau mikrokontroler. Persamaan kontroler PID digital diperoleh dari hasil konversi turunan persamaan

matematis kontroler PID analog, sehingga persamaan yang diperoleh berupa bentuk diskrit seperti persamaan 2.5 dibawah ini

$$u_{(k)} = K_p e_k + K_i T \sum_0^K e_k + \frac{1}{T} K_d (e_k - e_{k-1}) \quad \textbf{(2.5)}$$

Berdasarkan persamaan diskrit pada persamaan 2.5 jika diterjemahkan ke dalam bahasa pemrograma maka diperoleh persamaan 2.6 berikut ini.

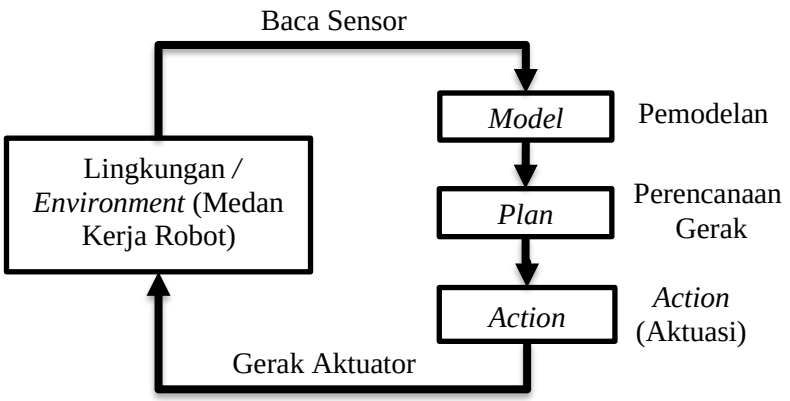
$$u = K_p \, x \, error + K_i \, x \, (error + last\_error) \, x \, T_s \, + \frac{K_d}{T_s} \, x \, (error - last\_error). \quad \textbf{(2.6)}$$

Dimana:

- Kp adalah konstanta *proportional*
- Ki adalah konstanta *integral*
- Kd adalah konstanta *derivative*
- *error* adalah nilai kesalahan *(error)*
- *last_error* adalah nilai kesalahan *(error)* sebelumnya
- Ts adalah *sampling time* atau waktu cuplik

2.2.5 Kontrol Gerak Robot Berbasis *Model Plan Action (MPA)*

Kontrol gerak robot berbasis *Model Plan Action (MPA)* merupakan salah satu bagian dari *high level control* robot. Prinsip kerja utama dari metode ini yaitu berdasarkan pembacaan data-data dari sensor yang merupakan informasi tentang lingkungan *(environment mapping*) yang akan memandu robot untuk mencapai targetnya. Metode kontrol *model plan action* diilustrasi seperti pada blok diagram gambar 2.6 berikut.



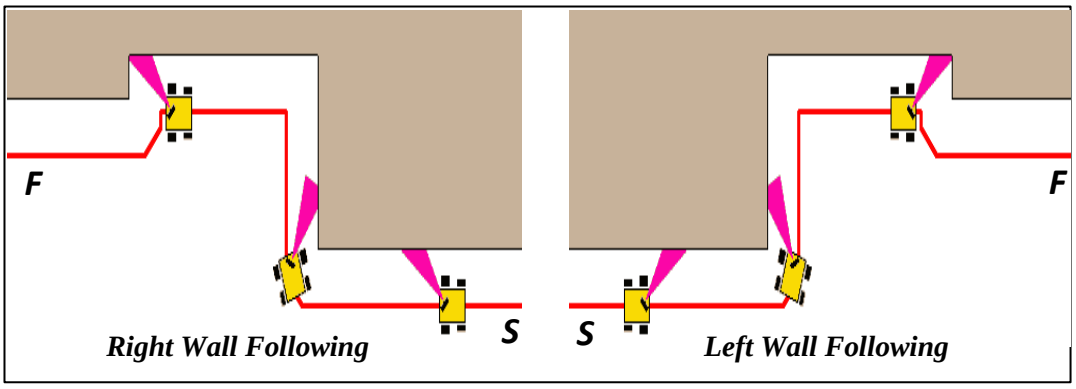
Gambar 2.6 Kontrol Gerak Robot Berbasis *Model, Plan,* dan *Action*

Pertama, robot melakukan pembacaan data-data sensor, berdasarkan data-data sensor ini kontroler melakukan pemodelan medan kerja robot (*the world*). Jika model telah dikenal atau diidentifikasi secara pasti, maka kontroler dapat melakukan perencanaan jelajah. Jika rencana jelajah telah matang atau telah terpenuhi, makan robot melakukan aksi bergerak menuju target. Sesampai pada posisi target, robot melakukan pembacaan sensor lagi. Dari sini langkah pemodelan dilakukan lagi. Demikian seterusnya sehingga setiap kali robot telah mencapai target yang baru dirinya akan menjalankan prosedur yang berulang hingga target terakhir yang diinginkan tercapai.(Pitowarno, 2006)

2.2.6 Wall Following Algorithm

Wall Following algorithm adalah salah satu algoritma untuk menyediakan orientasi dan gerak kepada robot dalam bernavigasi dengan metode menyusuri kontur dinding. Prinsip kerja dari sistem navigasi ini adalah dengan mengatur jarak robot dengan dinding supaya tetap konstan. Apabila terjadi perubahan jarak robot terhadap dinding, maka robot akan bergerak untuk menyesuaikan jarak kembali sesuai *set point* yang telah ditentukan. Proses ini dilakukan dengan teknik *scanning* terhadap dinding secara berulang-ulang.

Algoritma *Wall following* juga merupakan aturan yang paling terkenal untuk melintasi labirin atau *maze*. Hal ini mengacu pada prinsip metode aturan tangan kiri (*left hand rule*) dan aturan tangan kanan (*right hand rule*). Dengan metode *left hand rule (left wall following)* robot akan cenderung menjadikan sisi dinding kiri sebagai acuan bernavigasi, sedangkan pada metode *right hand rule (right wall following)* robot akan cenderung menjadikan sisi dinding kanan sebagai acuan bernavigasi, untuk ilustrasi lebih detail dapat dilihat pada gambar 2.7.



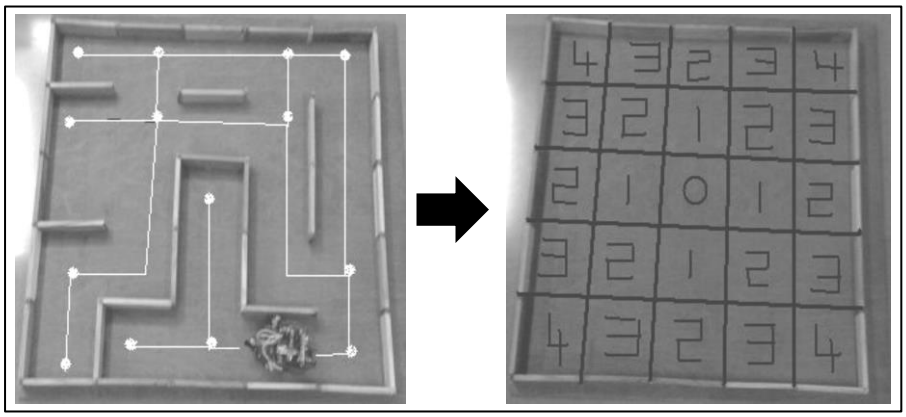
Gambar 2.7 Ilustrasi Navigasi *Wall Following*

Sumber : (Purnama, 2017)

2.2.7 Maze Mapping Algorithm

Maze mapping Algorithm atau *maze solving algorithm* adalah algoritma yang bekerja dengan cara memetakan atau menggambarkan beberapa kondisi lingkungan di dalam sebuah labirin (*maze*), dimana kemudian kondisi lingkungan yang telah dipetakan tersebut digunakan untuk menemukan jalan keluar atau jalur terpendek dari sebuah labirin (*maze*). (Mishra, 2008)

Algoritma *maze mapping* pada umumnya juga dikenal dengan istilah metode *path planning*, yakni salah satu metode yang biasa diimplementasikan pada robot *micromouse* untuk mencari jalan keluar dan jalur terpendek pada sebuah *path* di dalam labirin (*maze)*. Dalam pencarian jalan keluar menggunakan algoritma *maze mapping*, robot *micro mouse* memiliki konsep mengikuti aturan *wall following* dimana beropsikan untuk berjalan mengikuti dinding kiri atau dinding kanan pada proses memetakan *maze*. Indikator hasil pemetaaan digambarkan dalam bentuk memorisasi berupa kode-kode unik. Kode-kode unik di dalam memori ini akan dibangkitkan apabila robot telah berhasil memetakan beberapa parameter lingkungan yang dicari secara rekrusif. Kemudian hasil kode-kode unik yang telah yang telah dibangkitkan tersebut digunakan untuk memandu gerakan robot dalam mencapai jalan keluar atau titik tujuan akhir. Gambar 2.8 adalah ilustrasi memorisasi *maze mapping* pada robot *wall maze micromouse.*



Gambar 2.8 Ilustrasi Metode Memorisasi *Maze Mapping Micromouse*

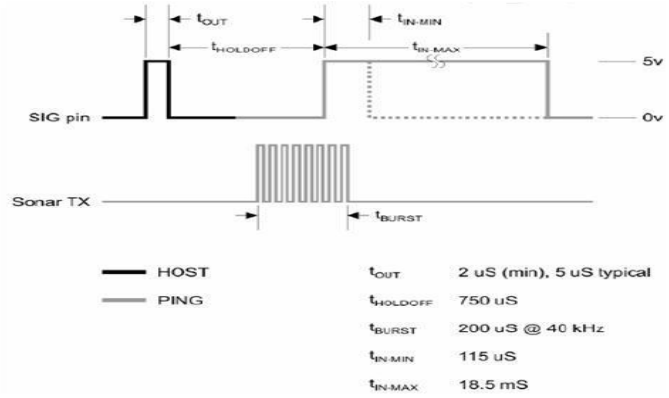
Sumber : (Mishra, 2008)

2.2.8 Sensor *PING Parallax Ultrasonic*

Sensor *PING Parallax* adalah sebuah sensor ultrasonik yang dapat mengukur jarak terhadap objek dengan jangkauan antara 2 cm sampai 300 cm.

Gambar 2.9 Sensor *PING Ultrasonic*

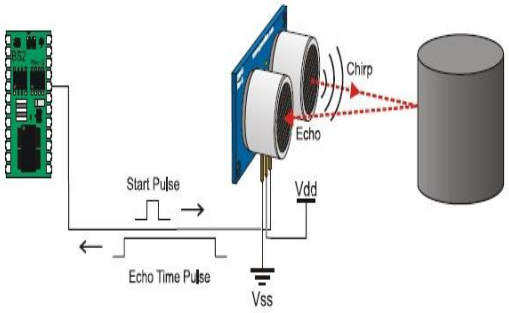
Prinsip kerja dari sensor ini yakni dengan cara memancarkan gelombang ultrasonik sebesar 40KHz selama 200 uS dari *transmitter* sensor terhadap jarak objek yang akan diukur.



Gambar 2.10 *Timing* Akses *Sensor PING Ultrasonic*

Sumber: (*Data Sheet Ping Ultrasonic)*

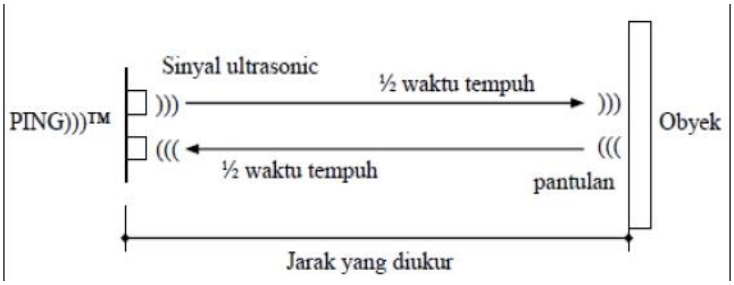
Pemancaran gelombang pada sensor ping ultrasonik hanya dilakukan apabila mendapat pulsa *trigger* (pulsa *high*) dari mikrokontroler selama 5uS. Gelombang yang di pancarkan dari (*trasnmitter*) sensor ultrasonik ini merambat di udara dengan kecepatan sebesar 344.424 *m / s* atau 1 cm setiap 29.034uS.



Gambar 2.11 Ilustrasi Prinsip Kerja Sensor *PING Ultrasonic*

Sumber: (*Data Sheet Ping Ultrasonic***)**

Gambar ilustrasi perhitungan jarak menggunakan sensor ultrasonik dapat diilustrasikan pada gambar 2.12 berikut.



Gambar 2.12 Ilustrasi Perhitungan Jarak Sensor PING *ultrasonic*

Sumber: (Suryatini, Kustija, & Haritman, 2013)

Berdasarkan ilustrasi gambar 2.12 untuk dapat mengukur jarak terhadap objek tertentu dapat digunakan persamaan berikut:

$$x\ (cm) = \frac{Vt}{2} \quad \text{..... (2.7)}$$

Dimana :

x = Jarak yang akan diukur antara sensor ultrasonik dengan objek (cm)

v = Kecepatan perambatan gelombang ultrasonik di udara (344 m/s)

t = Selisih waktu total antara pemancaran dan penerimaan pantulan gelombang (s)

Agar mudah diimplementasikan dalam bahasa pemrograman mikrokontroler dalam mengakses sensor, persamaan (2.7) berdasarkan data sheet sensor ping ultrasonik dapat dikonversi menjadi persamaan (2.8) atau (2.9) berikut:

$$x\ (cm) = \frac{y/29.034\text{uS}}{2} \quad \text{..... (2.8)}$$

Sehingga,

$$x\ (cm) = \frac{y}{58uS} \quad \text{..... (2.9)}$$

Dimana:

x = Jarak (cm)

y = Lebar pulsa yang diterima dari transmiiter menuju objek lalu kembali ke *receiver*

2.2.9 Sensor LED-Photodioda

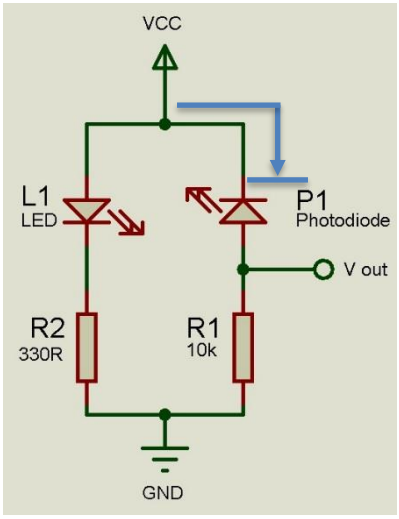
Sensor LED-photodioda merupakan sensor yang digunakan untuk mendeteksi warna dan garis pada sebuah bidang. Sensor photodioda ini tersusun atas lampu LED sebagai pemancar (*transmitter)* dan photodioda itu sendiri sebagai penerima (*receiver*). Seperti ilustrasi pada gambar 2.13 LED akan memantulkan cahaya ke sebuah bidang, kemudian photodioda menerima hasil dari pantulan tersebut. Semakin gelap warna dari sebuah bidang maka semakin sedikit cahaya yang diterima oleh photodioda, begitupun sebaliknya jika semakin terang warna dari sebuah bidang maka semakin banyak cahaya yang diterima photodioda.

Gambar 2.13 Ilustrasi Mekanisme Kerja Sensor LED-Photodioda

Sumber: (Fatchurrohman, 2014**)**

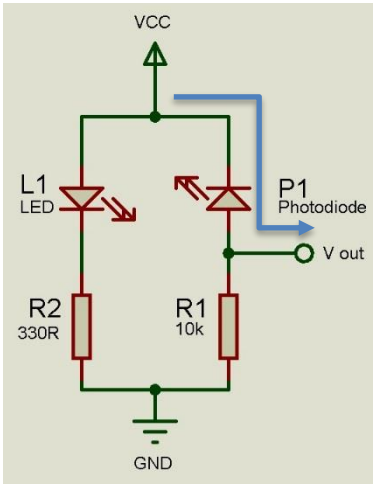
Berdasarkan prinsip kerja rangkaian sensor LED-photodioda pada gambar 2.13 jika sensor diletakkan pada bidang gelap (sedikit pantulan cahaya) maka nilai

resistansi dari photodioda akan besar atau diasumsikan tak terhingga, sehingga pada rangkaian tidak ada arus bocor yang menuju ke komparator.



Gambar 2.14 Prinsip Kerja Rangkaian Sensor LED-Photodioda pada Bidang Gelap

Jika sensor LED-photodioda diletakkan pada bidang terang (banyak pantulan cahaya) maka nilai resistansi dari photodioda akan menjadi kecil, sehingga pakan ada arus bocor yang menuju ke komparator. Gambar 2.15 adalah prinsip prinsip kerja rangkaian sensor LED-photodioda apabila diletakkan pada bidang terang.



Gambar 2.15 Prinsip Kerja Rangkaian Sensor LED-Photodioda pada Bidang Terang

Nilai V out photodioda ataupun nilai resistansi yang dihasilkan oleh sensor photodioda berdasarkan rangkaian pada gambar 2.14 atau gambar 2.15 di atas, dapat dicari dengan persamaan:

$$Vout = \frac{R_{photodioda}}{R_{photodioda} + R1} \; x \; Vin \quad \text{.....(2.10)}$$

Keterangan :

Vin = tegangan masukan pada rangkaian sensor photodioda

Vout = tegangan keluaran pada rangkaian sensor photodioda

$R_{\text{photodioda}}$ = resistansi dari photodioda

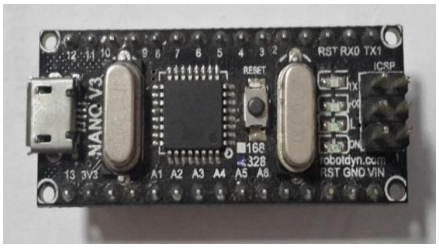
R2 = resistansi resistor pada rangkaian sensor photodioda

Jika menggunakan sebuah mikrokontroler, maka sensor LED-Photodioda ini dapat diakses melalui fitur ADC, dimana nilai *input* ADC mikrokontroler dapat dirumuskan dengan persamaan 2.11 berikut

$$ADC\ in = \frac{V_{INPUT}}{V_{REFERENSI}} \; x \; Resolusi\ ADC \quad \text{.....(2.11)}$$

2.2.10 Robotdyn Nano V3 *Board*

Robotdyn Nano V3 *board* adalah *board* sistem minimal mikrokontroler 8 bit berarsitektur *harvard* dengan menggunakan *chip* Atmega 328P produksi *Atmel AVR*. Robotdyn Nano V3 merupakan salah satu *board* versi lain dari *board* Arduino Nano V3 yang diproduksi dan dikembangkan oleh Robotdyn dengan *port input* USB menggunakan *micro USB*. Fisik dari Robotdyn Nano V3 *board* dapat dilihat pada gambar 2.16.



Gambar 2.16 *Robotdyn Nano V3 Board*

Spesifikasi *Robotdyn Nano V3* dapat diliht pada tabel 2.2 di bawah ini:

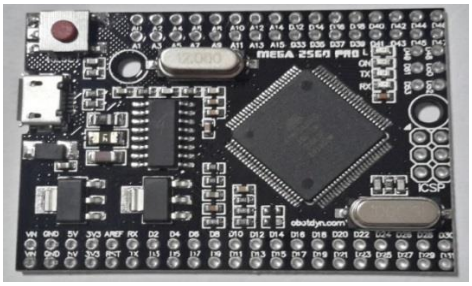
Tabel 2.2 Spesifikasi *Robotdyn Nano V3*

Microcontroller	Atmega328P
USB-TTL converter	CH340
Power Out	5V - 40mA
Power IN, VIN/DC jack	7-12 V
Power Consumtion	5V 220mA
Logic Level	5V
USB	Micro USB
Clock Frequency	16 MHZ
Digital I/O Pin	14 (6 as output PWM)
Analog I/0	6
Memory Size	32kb(0,5kb for bootloader)
Data Ram Type/Size	2kb
Data Rom Type/Size	1kb
Interface Type	ISP
Operating temperature	$-40^0c/+85^0c$

Tabel 2.2 di atas merupakan tabel spesifikasi dari *board* Robotdyn Nano V3. Board sistem minimal mikrokontroler ini memiliki sumber *clock* osilator kristal eksternal sebesar 16 Mhz. *Board ini* Dilengkapi dengan 14 buah pin *input output* digital (6 pin sebagai *output* PWM) dan 6 pin *input output* analog. Tegangan operasi pada Robotdyn Nano V3 adalah 5V. Tegangan *supply* yang direkomendasikan yakni berkisar antara 7 – 12V. Memory pada mikrokontroler ini memiliki kapasitas 32 kb Flash PEROM (*Flash Programmable and Eraseble Read Only Memory), 2kb data SRAM size*, dan *1kb data EEPROM size*. Fitur lain yang dimiliki mikrokontroler ini antara lain seperti *USART, SPI, TWI, I2C, TIMER/COUNTER,* dan *PWM.*

2.2.11 Robotdyn Mega 2560 *Pro Mini Board*

Robotdyn Mega 2560 *pro mini board* adalah *board* sistem minimal mikrokontroler 8 bit berarsiektur *harvard* dengan menggunakan *chip* Atmega 2560 produksi Atmel AVR. Robotdyn Mega 2560 *pro mini board* merupakan salah satu *board* versi lain dari *board* Arduino Mega 2560 yang diproduksi dan dikembangkan oleh Robotdyn dengan ukuran yang lebih kecil serta memiliki port USB versi *micro*.



Gambar 2.17 Robotdyn Mega 2560 *Pro Mini Board*

Spesifikasi *board* Robotdyn Mega 2560 *pro mini* dapat diliht pada **tabel 2.3** di bawah ini:

Tabel 2.3 Spesifikasi *Board* Robotdyn Mega 2560 *Pro Mini*

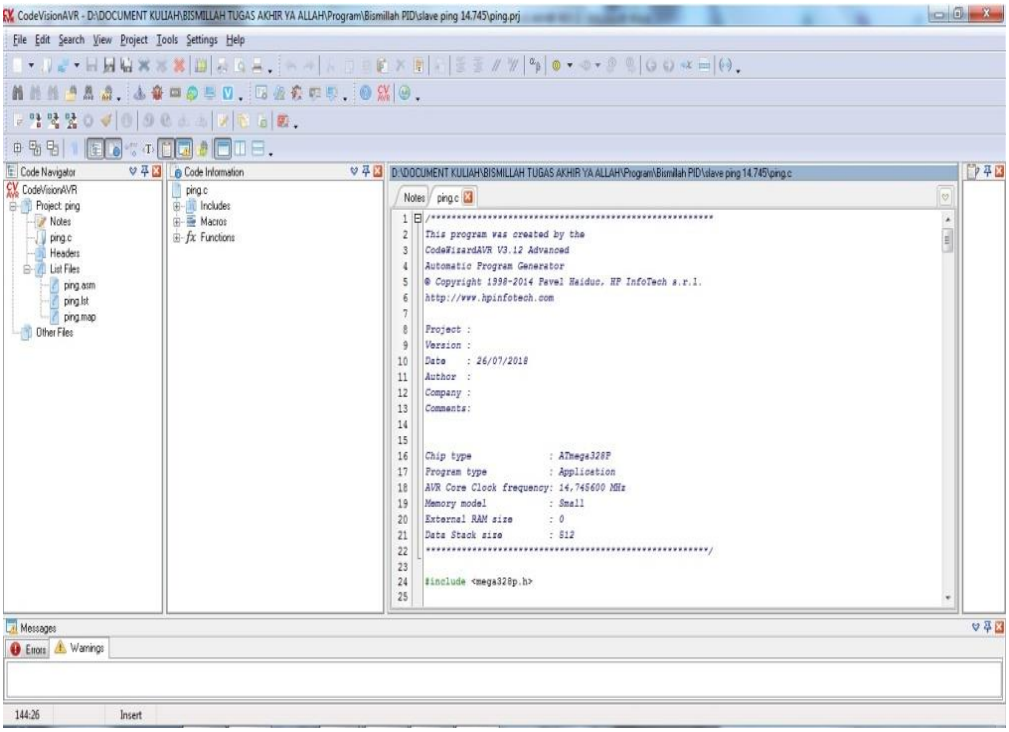
Microcontroller	ATmega2560
USB-TTL converter	CH340
Power Out	5V - 800mA
Power IN, VIN/DC jack	9-12 V
Power Consumtion	5V 220mA
Logic Level	5V
USB	Micro USB
Clock Frequency	16MHZ
Digital I/O Pin	54 (15 output PWM)
Analog I/0	16
Memory Size	256kb
Data Ram Type/Size	8kb
Data Rom Type/Size	4kb
Interface Type	ISP
Operating temperature	$-40^0c/+85^0c$

Tabel 2.3 merupakan tabel spesifikasi dari *board* Robotdyn Mega 2560 *Pro Mini*. *Board* sistem minimal mikrokontroler ini memiliki sumber *clock* osilator kristal eksternal sebesar 16 Mhz. *Board* ini dilengkapi dengan 54 buah pin *input output* digital (15 pin digunakan sebagai *output* PWM), dan 16 pin *input* dan *output* analog. Tegangan operasi pada Robotdyn Mega 2560 *pro mini* adalah 5V. Tegangan *supply* yang direkomendasikan yakni berkisar antara 9 – 12V. *Memory* pada mikrokontroler ini memiliki kapasitas 256kb *Flash PEROM* (*Flash Programmable and Eraseble Read Only Memory*), 8kb *data RAM size*, dan 4kb *data ROM size.* Fitur lain yang dimiliki mikrokontroler ini antara lain seperti *USART, SPI, TWI, I2C, TIMER/COUNTER,* dan *PWM*

2.2.12 *Code Vision AVR (CV AVR)*

CodeVisionAVR adalah sebuah *software* pemrograman bahasa C dengan fitur *C cross-compiler*, *Integrated Development Environment* (IDE) dan *automatic program generator* (pembangkit program otomatis) yang didesain untuk keluarga mikrokontroler AVR. *C cross compiler CV AVR* dapat mengolah dan mengakses semua element ANSI bahasa pemrograman *C* yang didukung oleh arsitektur AVR. *Integrated Development Environment* (IDE) pada *CV AVR* berperan untuk menulis bahasa pemrograman, melakukan kompilasi bahasa pemrograman menjadi bahasa mesin (kode biner) dan melakukan *upload* ke dalam memori mikrokontroler AVR.

CV AVR memuat fitur *automatic program generator* yang memungkinkan menulis beberapa struktur fungsi kode program secara otomatis pada fitur *chip* IC AVR seperti: inisialisasi *input* atau *output*, *external memory access setup*, inisialisasi ADC, dan lain sebagainya. *Software* ini juga telah dilengkapi dengan fasilitas pemrograman *chip* melalui metode *In-System Programming* sehingga dapat secara otomatis mentransfer *file* program ke dalam *chip* mikrokontroler AVR setelah sukses dikompilasi. Gambar 2.8 pada halaman selanjutnya adalah tampilan dari *software CV AVR.*

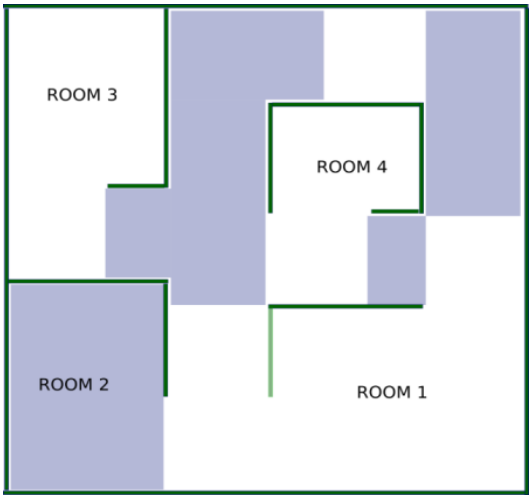


Gambar 2.18 Tampilan *Code Vision AVR*

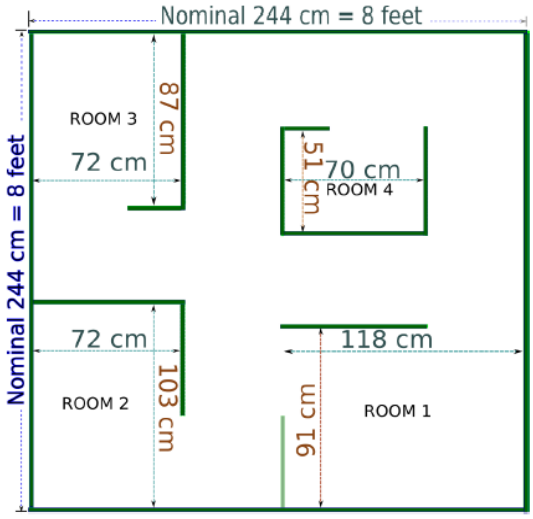
2.2.13 Kontes Robot Pemadam Api Indonesia (KRPAI) 2018

Kontes Robot Pemadam Api Indonesia (KRPAI) adalah sebuah kompetisi robot tingat mahasiswa antar perguruan tinggi se-Indonesia yang diselenggarakan oleh KEMENRISTEK-DIKTI. Robot yang dikompetisikan pada KRPAI ini adalah berupa robot *mobile* jenis berkaki. Aturan KRPAI yang diterapkan di Indonesia ini mengacu pada aturan internasional di *Trinity College International Robot Contest (TCIRC),* Amerika.

Gambaran singkat sistem pertandingan KRPAI yaitu sebuah robot berkaki diletakkan pada sebuah arena berupa labirin (*maze*) yang menyerupai rumah dengan 4 buah ruangan dimana pada salah satu ruangan terdapat sebuah sebuah lilin yang mewakili sebuah titik api. Robot harus dapat menemukan keberadaan titik api tersebut lalu kemudian memadamkannya. Bentuk dan dimensi labirin *(maze)* pada arena KRPAI aturan tahun 2018 dapat dilihat pada gambar 2.19 dan gambar 2.20.



Gambar 2.19 Bentuk Labirin *(maze)* Arena KRPAI 2018



Gambar 2.20 Dimensi Labirin *(maze)* Arena KRPAI

Pada KRPAI tahun 2018, aturan yang digunakan menggunakan level 2 pada aturan *TCIRC*, dimana terdapat bonus-bonus yang wajib di ambil yakni berupa:

1. Sound Activation;

2. Non-air Extinguisher;

3. Arbitrary Start;

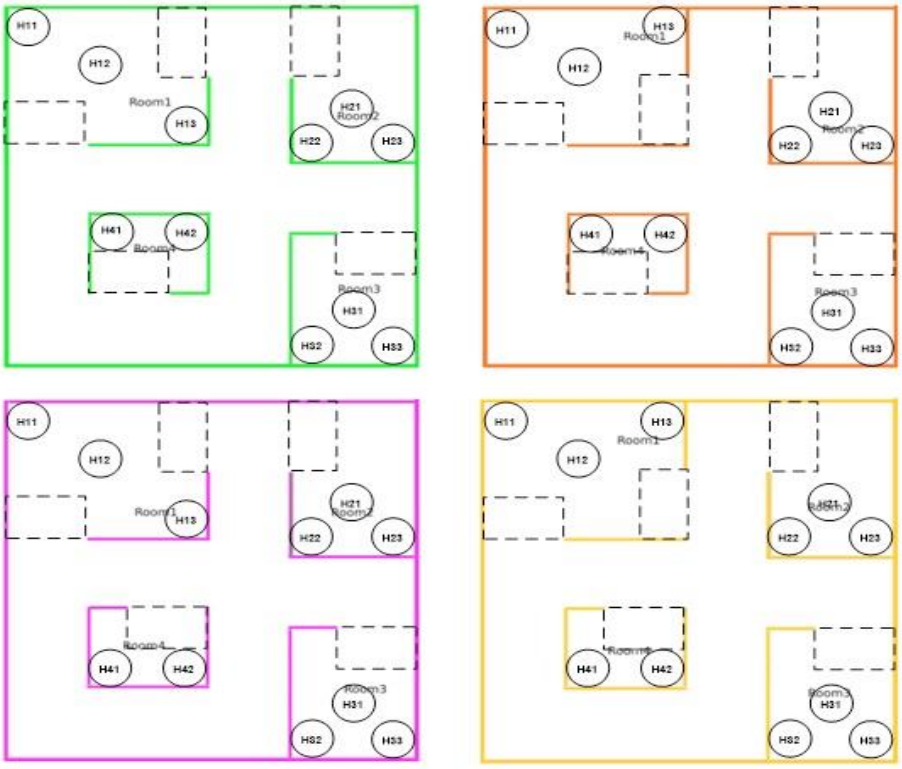
4. Candle Location,

5. Variable Door location,

6. Furniture;

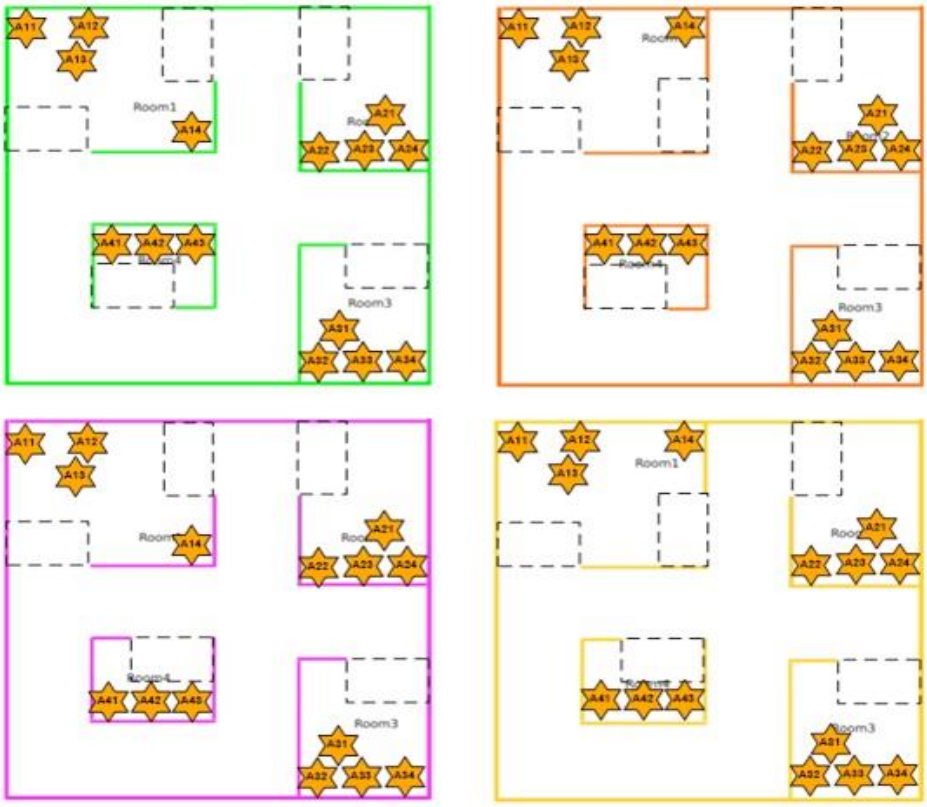
7. Return Trip.

Kandidat posisi *home* (*start)* pada rule KRPAI 2018 terdapat pada salah satu *room* dari 4 buah *room*, dengan bentuk konfigurasi pintu *room* 1 dan pintu *room* 4 yang berbeda-beda. *Start* dilakukan di dalam *room*, karena pada aturan tahun 2018 ini diwajibkan mengambil bonus *arbitrary start*. Sebagai rintangan, posisi robot saat *start* di *home* akan diletakkan di antara *furniture* berupa tabung berwarna kuning dengan dinding arena. Jarak minimal *furniture* terhadap dinding ini adalah sesuai ukuran dimensi maksimal robot yaitu 31 cm. Gambar 2.21 merupakan kandidat posisi *home* yang diberlakukan pada aturan KRPAI 2018.



Gambar 2.21 Kandidat Posisi *Home Rule* KRPAI 2018

Letak Kandidat posisi api juga sama halnya dengan letak kandidat posisi *home*. Posisi api diletakkan pada salah satu *room* dari 4 buah *room* secara acak sesuai undian yang diambil, namun syarat peletakkan titik api ini adalah tidak boleh 1 ruang dengan posisi *start* (*home)*. Sebagai rintangan, letak posisi titik api ini juga akan ditutupi oleh furniture tabung berwarna kuning, yaitu dengan jarak minimal 31 cm terhadap letak posisi titik api tersebut. Gambar 2.22 adalah konfigurasi posisi kandidat titik api pada *rule* KRPAI 2018.



Gambar 2.22 Konfigurasi Undian Titik Api *Rule* KRPAI 2018