"Sistem Navigasi Berbasis *Maze Mapping* pada Robot Beroda Pemadam Api"

PENELITI / TIM PENELITI

Dr. Erwani Merry Sartika S.T., M.T. (220148)

Muliady S.T., M.T. (220147)

Nelson M.P S.T. (0922049)

Junaidi Sucipto (1022051)

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS KRISTEN MARANATHA

BANDUNG

2014

BAB I

PENDAHULUAN

I.1 Latar Belakang

Perkembangan robotika dewasa ini semakin pesat dengan banyaknya perusahaan industri dan pabrik-pabrik yang menggunakan robot sebagai mesin produksinya. Selain sektor industri, robot juga digunakan di sektor yang dapat membahayakan manusia. Seperti robot penjinak bom dan robot pemadam api.

Teknologi robot berkembang semakin pesat sejak ditemukannya semikonduktor. Semikonduktor ini memungkinkan penciptaan robot dalam ukuran yang lebih kecil, kecepatan kerja yang lebih tinggi dan akurasi yang lebih handal. *Maze* adalah jenis teka-teki yang terdiri dari berbagai rute percabangan kompleks yang harus dicari solusinya. Rute yang berbentuk *island* membuat *maze* menjadi semakin sulit untuk dipecahkan. 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 jaringan jalan yang terbentuk atas lorong-lorong dengan dinding tanpa atap. Robot *maze* ialah robot yang berjalan mencari target pada suatu labirin.

Pada perlombaan robot saat ini kasus-kasus yang diberikan juga semakin kompleks, terutama pada perlombaan robot yang menggunakan maze seperti Kontes Robot Pemadam Api 2013. *Maze* yang digunakan terdapat bentuk *island*, mempunyai banyak rintangan, variable door, serta kombinasi home dan api, sehingga robot harus mempunyai kemampuan untuk memilih navigasi yang sesuai untuk setiap kondisi yang ada untuk menyelesaikan misi.

Permasalahan yang muncul akibat penggunaan *maze* adalah menemukan suatu metoda penyelesaian agar menghasilkan jalur yang dikehendaki. *Maze mapping* merupakan algoritma yang digunakan untuk memecahkan *maze*, yakni mencari dan memilih jalur dari *maze*. *Maze* mapping umumnya digunakan pada robot *wall follower*. Algoritma ini merupakan algoritma dasar, untuk proses pemilihan acuan navigasi robot yaitu berjalan mengikuti dinding kiri atau dinding kanan dan pemilihan jalur yang dikehendaki.

Pada penelitian ini semua kondisi setiap ruangan terlebih dahulu dimasukkan ke program robot, sehingga robot dapat mengetahui keberadaannya. Semua pergerakan robot telah ditentukan sesuai dengan *mapping* tersebut. Program *mapping* tersebut dibuat berdasarkan arena Kontes Robot Pemadam Api 2013. Penelitian ini diharapkan dapat menjadi penelitian yang nantinya dapat diterapkan pada kondisi nyata. dalam suatu rumah.

I.2 Identifikasi Masalah

Maze mapping umumnya digunakan pada robot *wall follower*. Namun kelemahan penggunaan algoritma *wall follower* adalah robot tidak mampu menemukan *finish* apabila terdapat lintasan tidak berbentuk *loop (island)*. Untuk itu diperlukan sistem navigasi yang dapat mengatasi berbagai lintasan, khususnya di arena Kontes Robot Pemadam Api 2013.[1] Sistem navigasi berbasis maze mapping pada robot beroda pemadam api yang mengandalkan sensor saja tidak dapat memberikan keadaan lengkap sekitar robot. Dengan berbagai ketentuan dan arena yang diberikan pada Kontes Robot Pemadam Api 2013, diperlukan sistem navigasi yang tentunya mampu memperhitungkan berbagai keadaan sekitar robot untuk membedakan keadaan yang mempunyai persepsi yang sama tetapi tetap berbeda secara signifikan. Maka permasalahan yang dibahas dalam penelitian ini adalah "Bagaimana untuk mendapatkan sistem navigasi berbasis *maze mapping* pada Kontes Robot beroda Pemadam Api 2013?". Jumlah titik api pada penelitian ini hanya satu sesuai dengan ketentuan pada Kontes Robot Pemadam Api 2013.

I.3 Tujuan Penelitian

Tujuan pembahasan penelitian ini adalah mendapatkan sistem navigasi berbasis *maze mapping* pada Kontes Robot beroda Pemadam Api 2013.

I.4 Manfaat Penelitian

Penelitian ini diharapkan dapat menjadi penelitian yang nantinya dapat diterapkan pada kondisi nyata. dalam suatu rumah.

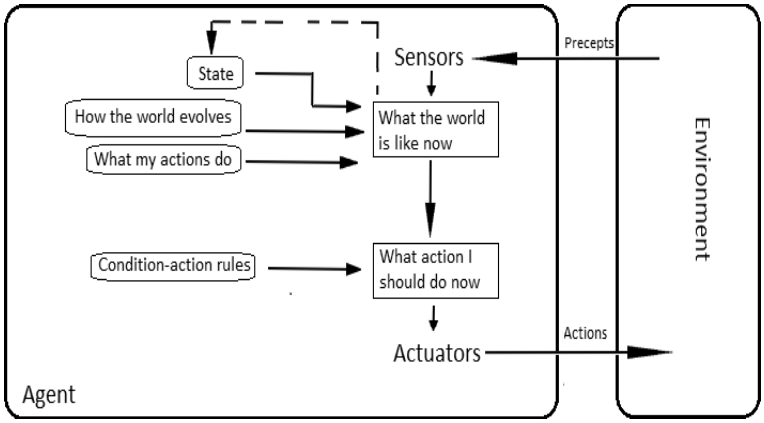
I.5 Kerangka Pemikiran dan Hipotesis

Pada perlombaan robot saat ini diberikan kondisi-kondisi yang semakin kompleks, terutama pada perlombaan robot seperti Kontes Robot Pemadam Api 2013. Maze yang digunakan mempunyai banyak rintangan, variable door, serta kombinasi home dan api yang rumit. Sehingga robot harus mempunyai kemampuan untuk memilih navigasi yang sesuai untuk setiap kondisi yang ada untuk menyelesaikan misi. *Maze mapping* merupakan algoritma yang digunakan untuk memecahkan *maze*, yakni mencari dan memilih jalur dari *maze*. Rute yang berbentuk *island* membuat *maze* menjadi semakin sulit untuk dipecahkan. *Maze* mapping umumnya digunakan pada robot *wall follower*. Algoritma ini merupakan algoritma dasar, untuk proses pemilihan acuan navigasi robot yaitu berjalan mengikuti dinding kiri atau dinding kanan dan pemilihan jalur yang dikehendaki. Pada penelitian ini semua kondisi setiap ruangan terlebih dahulu dimasukkan ke program robot, sehingga robot dapat mengetahui keberadaannya. Semua pergerakan robot telah ditentukan sesuai dengan *mapping* tersebut.

Metoda yang digunakan untuk membangun algoritma *maze mapping* adalah *Reflex Agent with state.* Metoda ini merupakan salah satu metoda dari *Intelligent Agent. Intelligent Agent* secara otonom mengamati lingkungannya melalui sensor dan memerintahkan aktuator agar bertindak untuk mencapai tujuan. Selain itu *Intelligent agent* juga dapat belajar dengan menggunakan pengetahuannya untuk mencapai tujuan.[a] Pada metoda *Reflex Agent with state, state* ditambahkan sehingga dapat menangani lingkungan yang sebagiannya dapat diamati internal memori yang tergantung pada persepsi sebelumnya sehingga mencerminkan beberapa aspek yang tidak teramati oleh *agent* saat ini.[4]

Gambar 1. adalah blok model dari *Reflex Agent With State*. Pada model ini terdiri dari *Agent, Sensor, Actuators, dan environment.* Sensor memberikan

informasi masukan dari *environment* kepada *agent,* sehingga *agent* dapat mengetahui kondisi lingkungannya saat ini, lalu *agent* memeriksa kondisi perubahan dari lingkungannnya. Melalui *condition-action rules* yang telah ditentukan sebelumnya, maka agent dapat membuat keputusan yang harus dilakukan selanjutnya. *Condition-action rules* dibentuk dengan mempertimbangkan action yang dipilih sehingga dapat menyelesaikan misi. Yang dimaksud *agent* dalam hal ini adalah Robot.



Gambar 1 Model *Reflex Agent With State*

Sistem navigasi akan dirancang dengan mengenali target dan ruangan yang tersedia. Selain itu sistem navigasi maze mapping dengan metoda *Reflex Agent with state* dirancang untuk mengambil keputusan pada posisi persimpangan dan keputusan saat memasuki pintu ruangan. Selain itu pengontrol PID akan digunakan untuk mengontrol jalannya robot pemadam api agar dapat berjalan mengikuti dinding (*wall follower*), berjalan saat di persimpangan, dan saat memasuki pintu ruangan.

I.6 Batasan Masalah

Batasan masalah dalam penelitian ini, yaitu:

1. Pokok pembahasan terletak pada keberhasilan sistem navigasi *maze mapping*.

2. Robot bergerak dengan metoda *wall follower*
3. Cara pembuatan robot tidak dibahas secara mendalam. Melainkan hanya bagaimana robot dapat berjalan dengan baik saat robot telah selesai dibuat.
4. Lantai arena terdiri dari 3 jenis warna saja, yaitu abu-abu, hitam, dan putih.
5. Arena yang digunakan merupakan arena Kontes Robot Pemadam Api Indonesia. Serta penempatan rintangan, posisi *Home* dan Api lilin (target) disesuaikan dengan peraturan Kontes Robot Pemadam Api Indonesia (KRPAI) 2013.

I.7 Spesifikasi Alat yang Digunakan

Alat dan bahan yang digunakan dalam penelitian ini, yaitu:

1. Atmega128A
2. UV-Tron (Sensor Api)
3. Sensor *ultrasonic* SRF-05
4. Motor DC
5. Motor *Brushless*
6. Motor Servo
7. Driver motor DC
8. Sensor Warna (Photodioda dan *Infra red* LED)
9. LCD 16x2
10. Kipas
11. Batere lippo 4 *cell* dan 2 *cell*
12. Regulator 12V

I.8 Sistematika Penulisan

Laporan penelitian ini disusun dengan sistematika penulisan sebagai berikut:

1. Bab I Pendahuluan

Pada bab ini berisi latar belakang masalah, rumusan masalah, tujuan, batasan, masalah, spesifikasi alat yang digunakan, dan sistematika penulisan.

2. Bab II Landasan Teori

Pada bab ini berisi teori-teori penunjang, yaitu Teori Kecerdasan buatan, *Maze mapping*, Sensor jarak ultrasonik, sensor api (UV-Tron), sensor warna, pengontrol mikro.

3. Bab III Perancangan dan Realisasi

Pada bab ini dijelaskan tentang perancangan robot beroda pemadam api, perancangan sistem robot beroda pemadam api menggunakan navigasi maze mapping, jenis-jenis sensor dan rangkaian yang dipakai dan algoritma pemrograman robot beroda pemadam api.

4. Bab IV Data Pengamatan dan Analisis Data

Pada bab ini dijelaskan tentang proses pengambilan data pengamatan untuk pemetaan maze pada *check point* api dan *check point home*, pengujian kemampuan robot beroda pemadam api, dan analisisnya.

5. Bab V

Pada bab ini berisi kesimpulan dan saran-saran yang perlu dilakukan untuk perbaikan di masa mendatang.

BAB II

TINJAUAN PUSTAKA

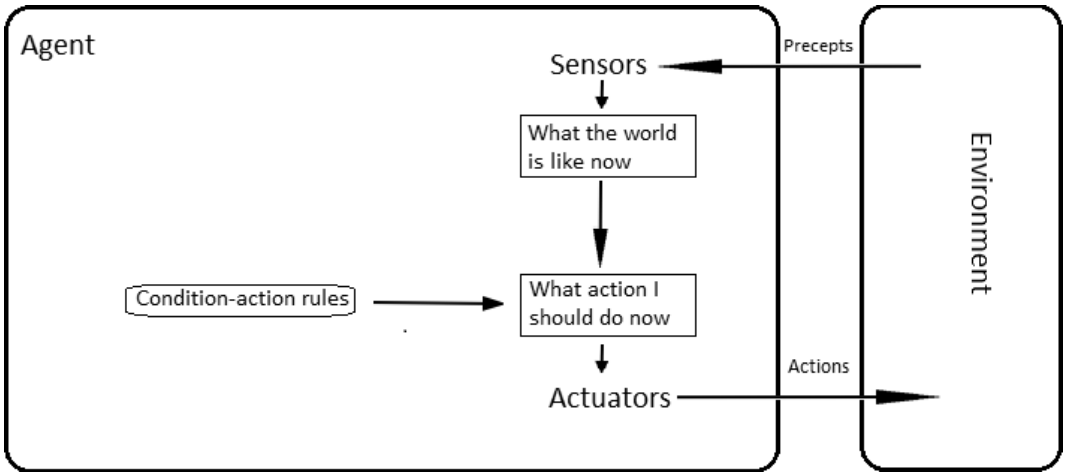
Pada bab ini berisi tinjauan pustaka, yaitu teori *Intelligent Agent, Maze mapping*, sensor jarak ultrasonik, sensor api (UV-Tron), sensor warna, dan pengontrol mikro.

II.1 Kecerdasan Buatan

Intelligent Agent adalah sebuah entitas otonom yang mengamati melalui sensor dan bertindak atas lingkungan menggunakan aktuator dan mengarahkan kegiatan/tindakan ke arah pencapaian tujuan. *Intelligent agent* juga dapat belajar atau menggunakan pengetahuan untuk mencapai tujuan. [5]

II.1.1 *Simple Reflex Agent*

Simple Reflex Agent hanya bertindak atas dasar persepsi saat ini, dan mengabaikan keadaan-keadaan sebelumnya. Fungsi *agent* didasarkan pada *condition-action rule*. *Agent* ini hanya akan berhasil jika lingkungan dari *agent* sepenuhnya diamati.



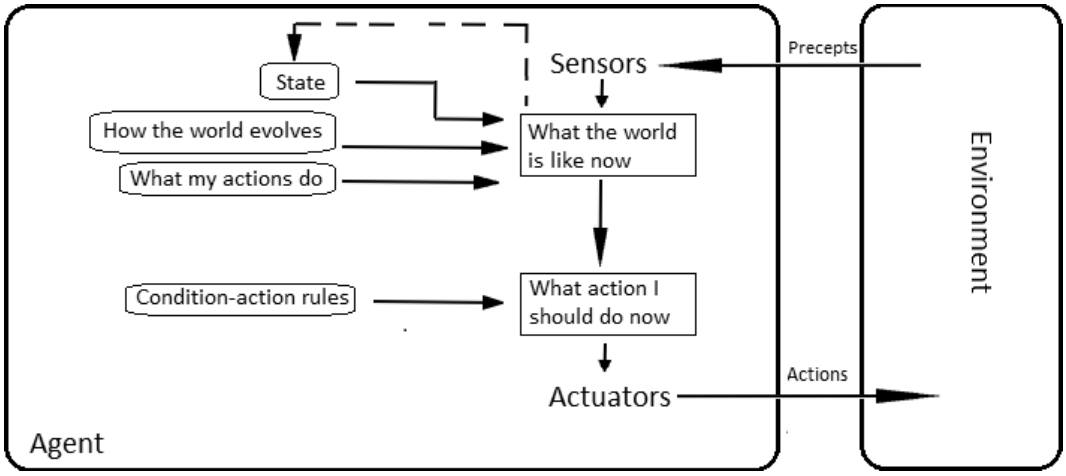
Gambar 2.1 Model *Simple Reflex Agent*

Gambar 2.1 adalah blok model dari *Simple Reflex Agent*. Pada model ini terdiri dari *Agent, Sensor, Actuators, dan environment.* Sensor memberikan informasi masukan dari *environment* kepada *agent*, sehinga *agent* dapat mengetahui kondisi lingkungannya saat ini. Melalui *condition-action rules* maka *agent* dapat membuat keputusan yang harus dilakukan selanjutnya. Tingkat keberhasilan *agent* dalam menyelesaikan misi sangat bergantung dengan *condition-action rules*. Tipe *agent* ini memiliki kekurangan karena tidak mempunyai *state*, sehingga tidak dapat mengetahui perubahan-perubahan kondisi lingkungan *agent*. Keputusan *agent* sepenuhnya bergantung pada *condition-action rules*. [2]

II.1.2 *Simple Reflex Agent With State*

Metoda yang akan digunakan untuk membangun algoritma *maze mapping* adalah *Reflex Agent with state.* Metoda ini merupakan tipe *agent* yang memiliki pengamat dengan respon yang cepat dan memiliki *internal memory*. *Agent* adalah suatu benda yang dapat dipandang sebagai pengamat (*percept*) terhadap lingkungan, melalui sensor-sensor dan bertindak melalui efektor terhadap lingkungan tersebut.

Simple Reflex agent with state dapat menangani lingkungan yang sebagiannya dapat diamati. Keadaan saat ini tersimpan di dalam *agent* dan mempertahankan beberapa jenis struktur yang menggambarkan bagian dari lingkungan yang tidak dapat dilihat. *Agent* harus menjaga internal memori yang tergantung pada persepsi sebelumnya dan dengan demikian mencerminkan setidaknya beberapa aspek yang tidak teramati oleh *agent* saat ini.



Gambar 2.2 Model *Reflex Agent With State*

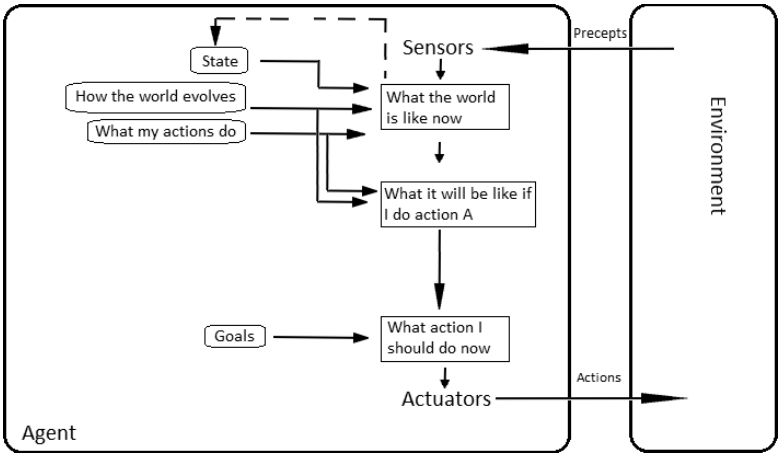
Gambar 2.2 adalah blok model dari *Reflex Agent With State*. Pada model ini terdiri dari *Agent, Sensor, Actuators, dan environment.* Sensor memberikan informasi masukan dari *environment* kepada *agent,* sehinga *agent* dapat mengetahui kondisi lingkungannya saat ini, lalu *agent* memeriksa kondisi perubahan dari lingkungannnya. Melalui *condition-action rules* maka *agent* dapat membuat keputusan yang harus dilakukan selanjutnya.

Refleks Agent With State membangun deskripsi keadaan sekarang dengan menggunakan kedua persepsi lingkungan dan keadaan internal sebelumnya. *Agent* ini meng-*update* keadaan internal secara terus menerus. Namun tipe *agent* ini memiliki kelemahan, yaitu *agent* tidak dapat mengetahui yang akan terjadi jika melakukan suatu tindakan, sehingga ketika melakukan tindakan yang tidak sesuai dengan yang seharusnya maka *agent* tidak dapat memperbaikinya. [2]

II.1.3 *Goal Based Agent*

Goal Based Agent lebih pada mengembangkan kemampuan dari *Simple Reflex. Agent* dengan menggunakan *"goal" information*. *Goal information* menggambarkan situasi yang diinginkan. Hal ini memungkinkan *agent* untuk memilih tindakan yang akan dilakukan di antara beberapa kemungkinan, *agent* akan memilih satu tindakan untuk mencapai *goal state*. *Search and planning* adalah subbidang kecerdasan buatan yang ditujukan untuk menemukan urutan

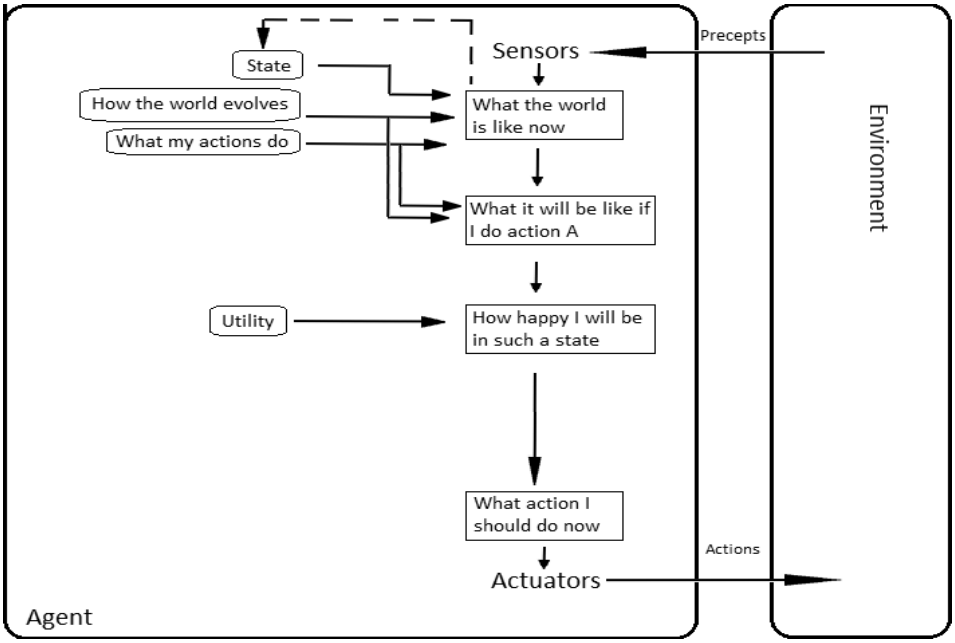
tindakan yang mencapai tujuan *agent*. Pada Gambar 2.3 ditunjukkan blok model *Goal Based Agent.* [2]



Gambar 2.3 Model *Goal Based Agent*

II.1.4 *Utility Based Agent*

Agent berbasis tujuan hanya membedakan antara *goal state* dan *non-goal state*. Pada *Utility Based Agent* ini dimungkinkan untuk menentukan ukuran seberapa diinginkan keadaan tertentu. Langkah ini dapat diperoleh melalui penggunaan fungsi utilitas yang memetakan keadaan untuk ukuran utilitas keadaan. Tolok ukur kinerja dari *agent* yaitu harus memungkinkan perbandingan keadaan lingkungan yang berbeda sesuai dengan seberapa berpengaruh bagi *agent* jika melakukan suatu tindakan. Pada Gambar 2.4 ditunjukkan blok model dari *Utility Based Agent.* [2]

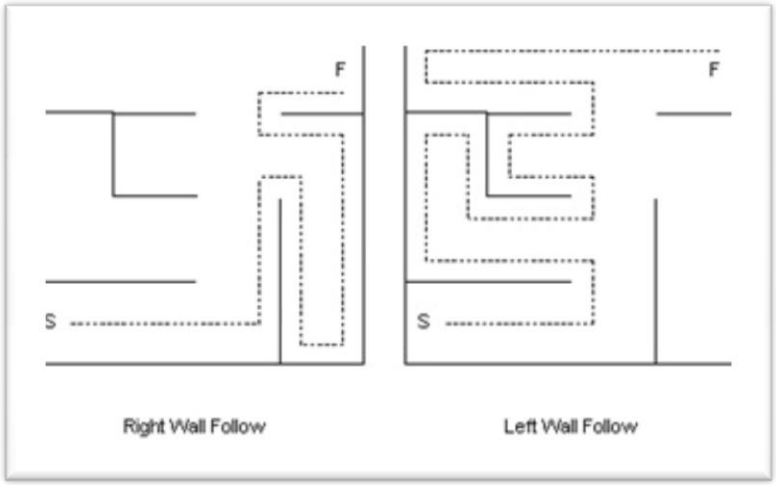


Gambar 2.4 Model *Utility Based Agent*

II.2 *Maze Solving Algorithm*

Ada sejumlah algoritma pemecahan *maze* yang berbeda, yaitu *The Random Mouse*, *wall follower*, dan *Pledge*. Algoritma tersebut dirancang untuk digunakan di dalam *maze* oleh *agent* yang tidak mempunyai pengetahuan sebelumnya dari *maze*.

Wall follower merupakan aturan yang paling terkenal untuk melintasi *maze*. Hal ini juga dikenal dengan aturan tangan kiri (*left-hand rule*) dan aturan tangan kanan (*left-hand rule*). Jika sebuah *maze* seluruhnya saling terhubung maka agen tidak akan pernah tersesat dan pasti menemukan target dengan hanya menggukan salah satu aturan tangan kiri atau aturan tangan kanan. Tetapi jika ada salah satu *maze* yang tidak terhubung dengan *maze* lainnya maka ada kemungkinan agen dapat tersesat dan tidak menemukan target jika hanya menggunakan salah satu aturan saja.[6] Gambar 2.2 menunjukkan perjalanan mengikuti aturan tangan kanan dan aturan tangan kiri.



Gambar 2.5 Rute Aturan Tangan Kanan dan Aturan Tangan Kiri

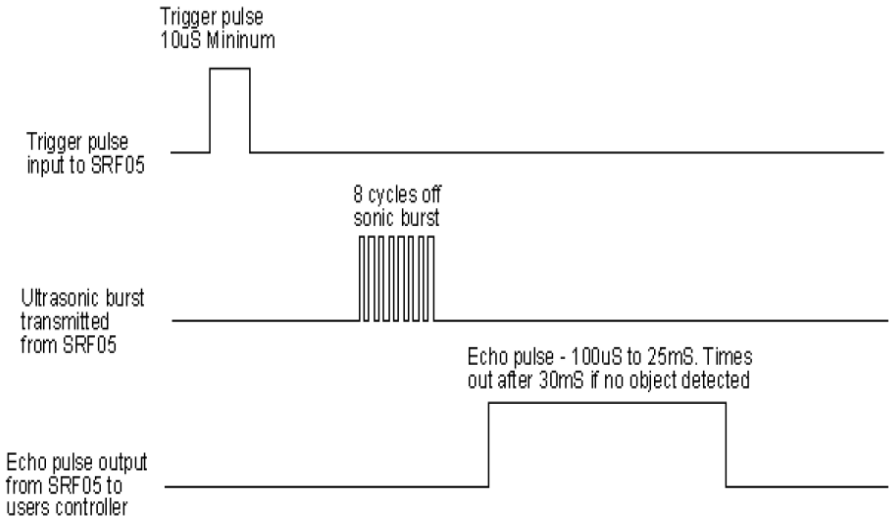
II.3 Sensor dan pengontrol pendukung algoritma maze mapping

Pada penelitian ini digunakan beberapa jenis sensor yaitu sensor jarak ultrasonik, sensor api (UVTron), sensor warna, dan pengontrol mikro sebagai pendukung penerapan algoritma maze mapping pada robot beroda pemadam api.

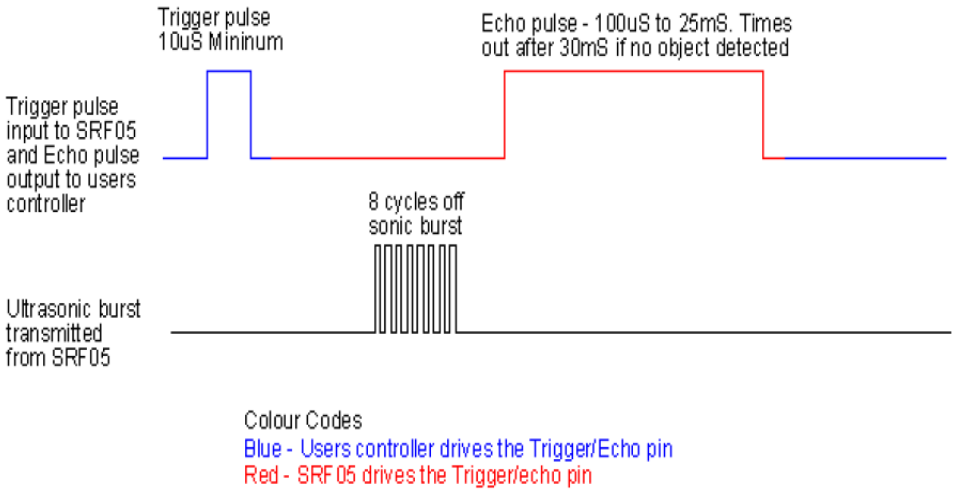
II.3.1 Sensor Jarak Ultrasonik (SRF05)

Sensor jarak ultrasonik dipakai untuk aplikasi-aplikasi yang perlu dilakukan pengukuran jarak. Sensor jarak ultrasonik yang dipakai dalam penelitian ini adalah sensor SRF05. Sensor SRF05 memiliki 2 *mode* untuk penggunaannya.

Gambar 2.6 dan Gambar 2.7 menunjukkan diagram waktu sensor SRF05 untuk masing-masing *mode*. Untuk mulai menghitung jarak dengan menggunakan SRF05, diperlukan pemberian pulsa *trigger* minimal 10μS. Kemudian SRF05 akan memancarkan 8 siklus gelombang ultrasonik (40kHz). Sensor SRF05 mengeluarkan pulsa *output high* pada jalur *echo* (atau jalur trigger pada mode 2) setelah memancarkan gelombang ultrasonik. Setelah gelombang pantul terdeteksi, sensor SRF05 mengeluarkan pulsa *output low* pada jalur *echo*. Lebar pulsa *high* pada jalur *echo* sebanding dengan jarak objek. [9]



Gambar 2.6 Diagram waktu sensor SRF05 *mode* 1

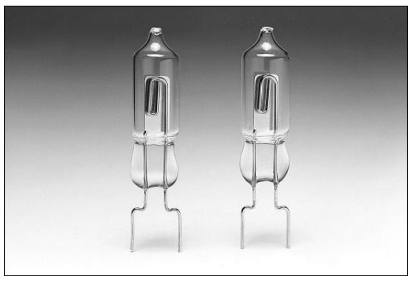


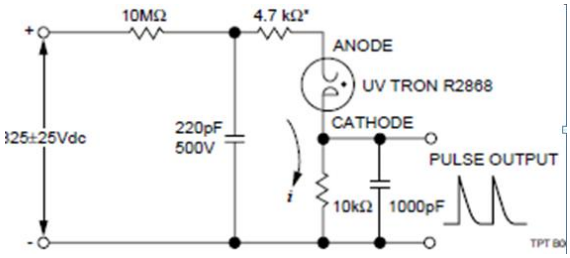
Gambar 2.7 Diagram waktu sensor SRF05 *mode* 2

II.3.2 Sensor UVTron

Sensor UVTron digunakan untuk mendeteksi keberadaan api di dalam ruangan. Sensor ini bekerja dengan cara mendeteksi ada tidaknya panas api. Sensor ini memberikan sinyal aktif apabila mendeteksi adanya api dalam ruangan. Tipe sensor yang dipakai adalah Hamamatsu *Flame Sensor* UVTron R2868 yang

berbentuk bohlam dan dilengkapi dengan rangkaian untuk mengaktifkannya (UVTron *driving circuit*), seperti pada Gambar 2.8a dan Gambar 2.8b.

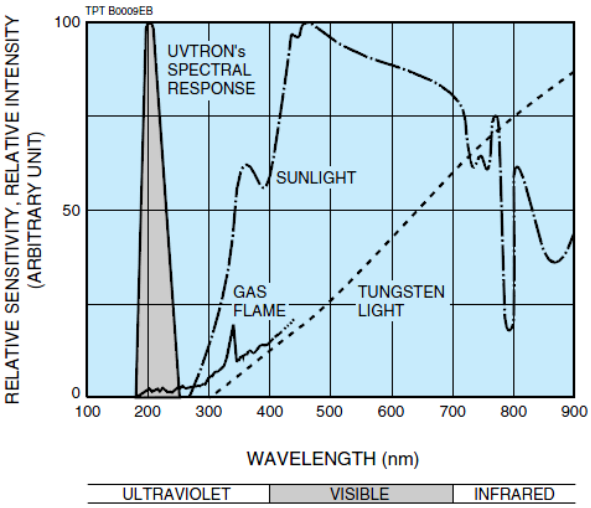




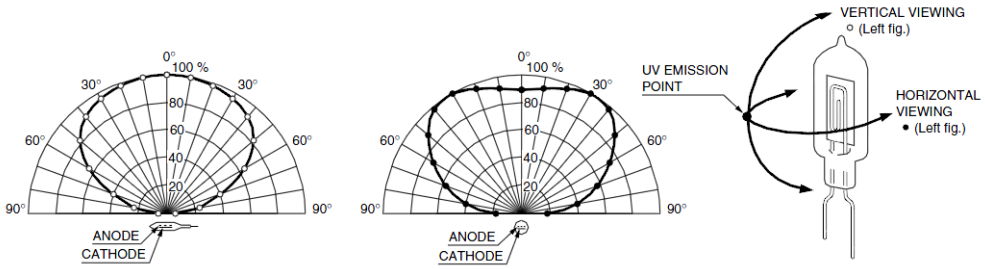
a. **b.**

Gambar 2.8 Sensor Api (a. UVTron R2868 dan b. Rangkaian Pengaktif)

Prinsip kerja Hamamatsu R2868 adalah mendeteksi adanya gelombang ultraviolet yang dihasilkan oleh api, yaitu pada *range* 185-260nm, di area tersebut adalah sinar ultraviolet yang dihasilkan oleh api, ditunjukkan pada Gambar 2.9. Daerah deteksi Hamamatsu R2868, seperti pada Gambar 2.10.



Gambar 2.9 *Spektrum* respon UVTron

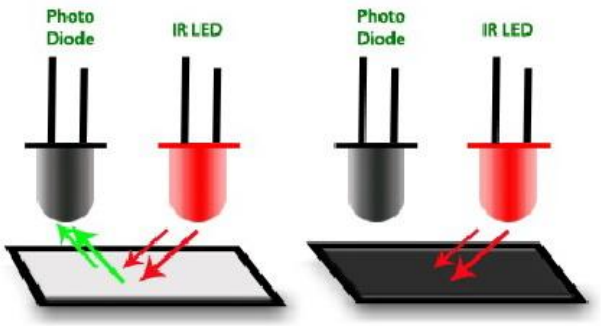


Gambar 2.10 Derajat sensitivitas Hamamtasu R2868

Hamamatsu R2868 memerlukan sebuah modul rangkaian untuk mengaktifkannya. Modul ini bernama UVTron *Driving Circuit* dengan seri C10807. Bekerja pada tegangan 12 volt dengan konsumsi arus sebesar 300 mikro *Ampere* yang akan menghasilkan keluaran digital logika *low* jika terdeteksi api dan logika *high* jika tidak ada api. [4]

II.3.3 Sensor Warna

Sensor warna adalah sensor yang digunakan untuk mendeteksi warna dari lantai robot berada. Sensor warna yang digunakan dalam perancangan robot ini adalah kombinasi antara infrared LED dengan photodioda. Infrared LED akan memantulkan cahaya ke lantai, dan photodioda menerima hasil dari pantulan tersebut. Semakin gelap warna dari lantai maka semakin sedikit cahaya yang diterima oleh photodioda, dan sebaliknya. Ilutrasi pemantulan sensor warna dapat dilihat pada Gambar 2.11.



Gambar 2.11 Ilustrasi Pemantulan Sensor Warna

II.3.4 Pengontrol Mikro

Pengontrol mikro dapat diartikan sebagai pengontrol dalam ukuran mikro. Secara umum pengontrol mikro dapat diartikan sebagai komputer dalam sebuah chip. Berbagai unsur seperti prosesor, memori, *input*, dan *output* terintegrasi dalam satu kemasan yang berukuran kecil. Pengontrol mikro membutuhkan daya yang rendah, murah, dan mudah didapatkan. Pengontrol mikro beroperasi dengan kecepatan detak (*clocking*) *megahertz* atau kurang.

Salah satu pengontrol mikro yang banyak digunakan saat ini yaitu pengontrol mikro AVR. AVR adalah pengontrol mikro RISC (*Reduce Instruction Set Computing*) 8 bit berdasarkan arsitektur Harvard, yang dibuat oleh Atmel pada tahun 1996. AVR mempunyai kepanjangan *Advanced Versatile* RISC atau *Alf and Vegard's RISC processor* yang berasal dari nama penemunya, dua mahasiswa Norwegian Institute of Technology (NTH), yaitu Alf-Egil Bogen dan Vegard Wollan. AVR memiliki keunggulan dibandingkan dengan pengontrol mikro lain, keunggulan utama pengontrol mikro AVR adalah memiliki laju putaran eksekusi program yang lebih cepat karena sebagian besar instruksi dieksekusi dalam satu siklus *clock*, lebih cepat dibandingkan dengan pengontrol mikro MCS51. Selain itu pengontrol mikro AVR memiliki fitur yang lengkap (ADC *Internal*, EEPROM *Internal*, *Timer/Counter*, *Watchdog Timer*, PWM, *Port* I/O, komunikasi serial, komparator, I2C, dll.). **[1]**

Pengontrol mikro ATmega 128 adalah salah satu pengontrol mikro yang dibuat oleh Atmel Corporation, industri yang bergerak di bidang manufaktur semikonduktor. Pengontrol mikro ini termasuk dalam keluarga AVR 8-bit RISC yang dikategorikan dalam kelas ATmegaAVR. Fitur-fitur yang dimiliki oleh ATmega 128 adalah sebagai berikut:

1. Pengontrol mikro AVR 8 bit yang memiliki kemampuan tinggi dengan daya rendah.
2. Arsitektur RISC (*Reduced Instruction Set Computing*) dengan *throughput* mencapai 16 MIPS (*Millions Of Instruction per Second*) pada frekuensi 16MHz.

3. Memiliki kapasitas *flash* memori 128kByte, EEPROM (*Electronically Erasable Programmable Read Only Memory*) 4kByte, dan SRAM (*Static Random Access Memory*) 4kByte.
4. Saluran I/O sebanyak 64 buah.
5. CPU yang terdiri atas 32 buah register.
6. Unit interupsi internal dan eksternal.
7. Port USART (*Universal Synchronous Asynchronous Receiver Transmitter*) untuk komunikasi serial.
8. Fitur *Peripheral*, yaitu :
   a) Tiga buah *Timer/Counter* dengan kemampuan, yaitu :
      - Dua buah *Timer/Counter* 8 bit dengan *prescaler* terpisah dan *mode compare*.
      - Satu buah *Timer/Counter* 16 bit dengan *prescaler* terpisah, *mode compare*, dan *mode capture*.
   b) *Real Time Counter* dengan *oscillator* tersendiri.
   c) Enam kanal PWM.
   d) Delapan kanal 10 bit ADC (*Analog To Digital Converter*).
   e) Byte-*oriented Two-wire Serial Interface*.
   f) Serial USART yang dapat diprogram.
   g) Antarmuka SPI (*Serial Peripheral Interface*).
   h) *Watchdog Timer* dengan *oscillator* internal.
   i) *On-chip Analog Comparator*.

BAB III

METODE PENELITIAN

Bab III menjelaskan metode penelitian berupa perancangan robot beroda pemadam api, perancangan sistem robot beroda pemadam api menggunakan navigasi maze mapping, dan algoritma pemrograman robot beroda pemadam api.

III.1 Perancangan Sistem Robot Beroda

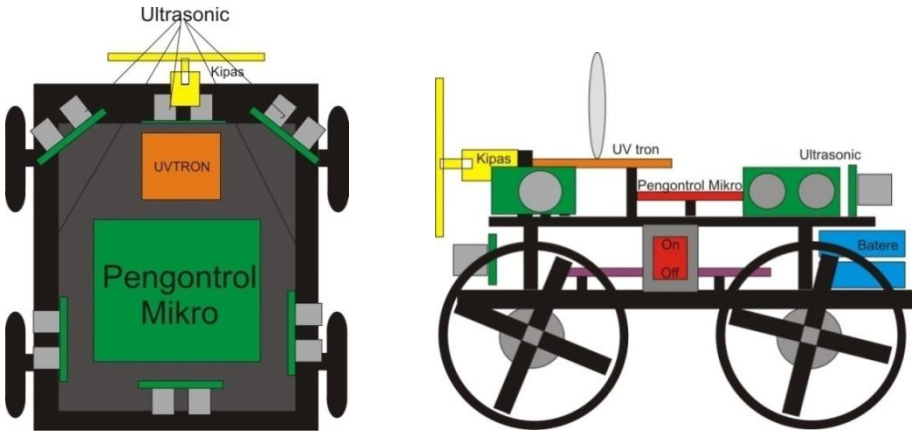
Robot beroda dirancang agar robot dapat melakukan tugas-tugas sebagai berikut: *start* awal dari home, robot dapat bergerak dengan baik untuk menelusuri lorong-lorong dan memiliki kemampuan memutuskan bergerak dengan metoda telusur dinding kiri atau telusur dinding kanan, mendeteksi api, mencari posisi api, memadamkan api, dan kembali ke Home.

III.1.1 Perancangan dan Realisasi Robot Beroda Pemadam Api

Robot yang dibuat berukuran 27.5cm x 18.5cm x 24cm. Robot dibuat dalam dimensi yang cukup kecil. Dimensi robot yang kecil akan memudahkan robot untuk bergerak di dalam *maze* dan menghindari rintangan yang ada. Robot menggunakan ban karet, dimaksudkan agar roda robot tidak mudah selip saat bermanuver di dalam *maze*. Robot dibuat menyerupai sebuah mobil dengan 4 buah roda berdiameter 9 cm dan menggunakan 4 buah motor DC 12 Volt sebagai *actuator*. Penempatan posisi komponen dan sensor penting pada robot untuk mendukung algoritma *maze mapping*. Sensor-sensor yang digunakan adalah sensor jarak ultrasonik, sensor api lilin, dan sensor warna.

Sensor jarak *ultrasonic* yang digunakan berjumlah 7 buah. Sensor jarak *ultrasonic* digunakan sebagai alat navigasi robot dan untuk mengukur jarak robot dengan dinding sebelah depan, kiri, kanan, dan belakang. Pengontrol mikro akan memberikan perintah kepada motor *driver* tergantung pada masukan-masukan dari sensor jarak. Sensor UVTron yang digunakan berjumlah 1 buah. Sensor warna yang digunakan berjumlah 3 buah. Dua sensor warna terletak di bagian depan dan satu sensor warna terletak dibagian belakang kiri robot. Sensor warna depan berfungsi untuk mendeteksi garis putih yang ada di setiap pintu ruangan,

mendeteksi warna lantai dibawah robot, dan mendeteksi lingkaran atau juring lilin. Selain itu, sensor warna depan akan bekerja bersama dengan sensor warna belakang untuk mendeteksi warna putih pada saat robot akan mencari *home.* Desain bentuk dan penempatan sensor-sensor ditunjukkan pada Gambar 3.1.

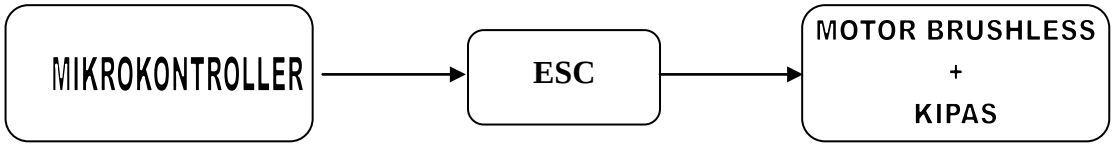


Gambar 3.1 Desain Robot Beroda Pemadam Api dan Penempatan Sensor.

III.1.2 Sistem Gerak dan Pemutar Kipas pada Robot

Robot menerapkan sistem gerak *differential drive*. Sistem ini memungkinkan robot berputar di tempat dengan cara memutar motor dengan arah berlawanan dengan demikian robot tidak memerlukan daerah yang luas untuk bermanuver dan waktu yang lama.

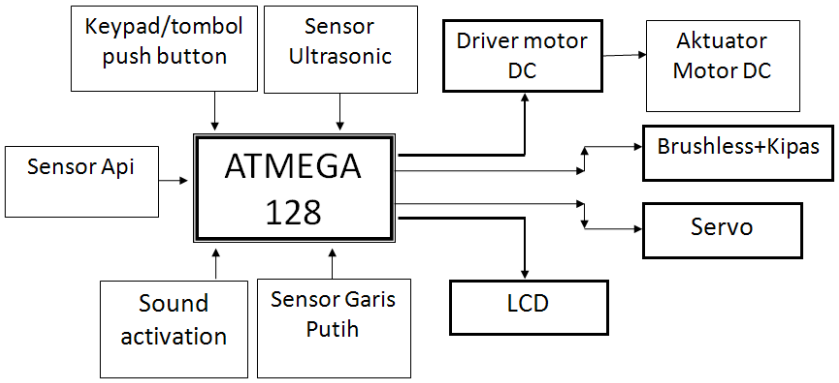
Pada penelitian ini digunakan kipas untuk memadamkan api lilin. Untuk memutar kipas diperlukan rangkaian pemutar kipas yang terdiri mikrokontroler, motor *brushless* dan ESC (*Electronic Speed* Kontrol). ESC digunakan sebagai *driver* motor *brushless*. *Output* pada mikrokontroler digunakan untuk menghasilkan *output* sinyal PWM untuk *input* ESC. Gambar 3.2 memperlihatkan diagram blok pemutar kipas.



Gambar 3.2 Diagram Blok Pemutar Kipas

III.1.3 Pengontrol Robot

Pengontrol mikro ATmega 128 digunakan sebagai pengontrol robot, yang akan mengolah informasi dari sensor-sensor untuk mengatur robot agar dapat menjalankan tugasnya. Pengontrol mikro terhubung dengan sensor-sensor pada robot diantaranya: sensor jarak ultrasonic SRF05, sensor api UV-TRON, sensor warna, dan rangkaian *driver* motor DC VNH3SP30 MD01B. Diagram blok sistem elektronika robot dapat dilihat pada Gambar 3.3.



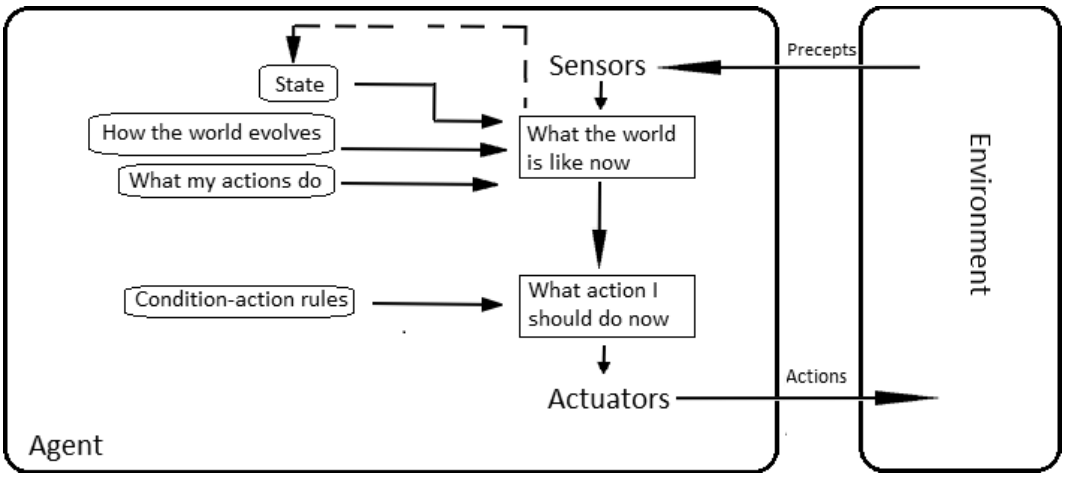
Gambar 3.3 Diagram Blok Sistem Elektronika Robot

Sensor Ultrasonik SRF-05 digunakan untuk mengukur jarak robot dengan objek disekitarnya. Terdapat 7 sensor Ultrasonik yang digunakan, masing-masing sensor memiliki fungsi tersendiri. Sensor depan berfungsi untuk mendeteksi benda yang berada di depan robot, sensor kiri depan digunakan sebagai acuan nilai *error* yang dihasilkan untuk kontroler PID, sensor kiri tengah, kanan depan, dan kanan tengah digunakan saat robot bermanuver di dalam *maze* dengan menggunakan metoda *wall follower*. Sensor jarak ini juga digunakan untuk mengukur jarak depan, kiri, kanan dan belakang dari robot ke dinding maze untuk memetakan dan mengetahui kondisi lingkungan dari robot pada beberapa lokasi yang sudah ditentukan untuk *maze mapping*.

Sensor warna terdiri dari photodioda sebagai penerima cahaya dan infrared LED sebagai pemancar cahaya. Rangkaian sensor warna ini membentuk pembagi tegangan antara resistor dengan photodioda, lalu tegangan *output* dihubungkan dengan port ADC dari pengontrol mikro. Semakin gelap warna lantai maka semakin besar nilai pembacaan ADC, dan sebaliknya, semakin terang warna lantai maka semakin kecil nilai pembacaan ADC.

Dalam menggerakkan sebuah motor DC, diperlukan *driver* sehingga motor DC dapat berputar searah dan berlawanan jarum jam. *Driver* motor yang digunakan adalah VNH3SP30 MD01B. Melalui motor *driver* ini, motor dapat dikontrol dengan cara memberikan *input* berupa bit – bit logika pada *port output* mikrokontroler.

III.2 Sistem Navigasi Robot



Metoda yang akan digunakan untuk membangun algoritma *maze mapping* adalah *Reflex Agent with state.* Metoda ini merupakan tipe *agent* yang memiliki pengamat dengan respon yang cepat dan memiliki *internal memory*. *Agent* adalah suatu benda yang dapat dipandang sebagai pengamat (*percept*) lingkungan, dalam hal ini adalah robot. Sedangkan sensor-sensor yang digunakan dapat mengamati sebagian dari lingkungan *agent* dan bertindak melalui aktuator pada lingkungan tersebut.

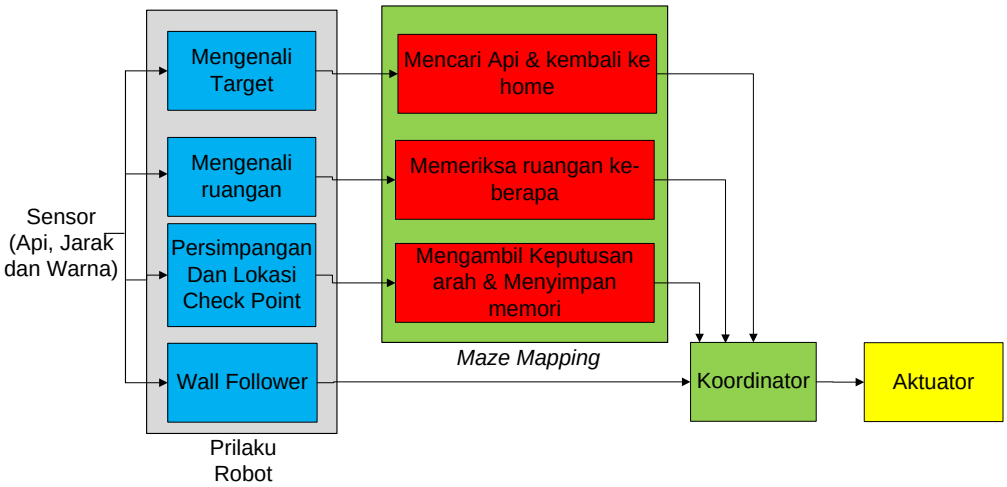
Simple Reflex agent with state dapat menangani lingkungan yang sebagiannya dapat diamati. Keadaan saat ini tersimpan di dalam *agent* dan mempertahankan beberapa jenis struktur yang menggambarkan bagian dari lingkungan yang tidak dapat dilihat. *Agent* harus menjaga internal memori yang tergantung pada persepsi sebelumnya dan dengan demikian mencerminkan setidaknya beberapa aspek yang tidak teramati oleh agent saat ini.

Gambar 2.2 adalah blok model dari *Reflex Agent With State*. Pada model ini terdiri dari *Agent, Sensor, Actuators, dan environment.* Sensor memberikan

informasi masukan dari *environment* kepada *agent*, sehingga *agent* dapat mengetahui bagaimana kondisi lingkungannya saat ini, lalu *agent* memeriksa kondisi perubahan dari lingkungannnya. Melalui *condition-action rules* yang telah ditentukan sebelumnya, maka *agent* dapat membuat keputusan yang harus dilakukan selanjutnya. *Condition-action rules* dibentuk dengan mempertimbangkan action yang dipilih dapat menyelesaikan misi.

Refleks Agent With State membangun deskripsi keadaan lingkungan sekarang dengan cara membandingkan kondisi sebelum dan sesudah *agent* sampai pada suatu titik. *Agent* memperbaharui *internal memory* secara terus menerus dengan menggunakan metode mempertahankan kondisi terakhir dari lingkungan *agent*. Namun tipe *agent* ini memiliki kelemahan, yaitu agent tidak dapat mengetahui apa yang akan terjadi jika melakukan suatu tindakan. Sehingga ketika melakukan tindakan yang tidak sesuai dengan yang seharusnya maka *agent* tidak dapat memperbaikinya.

Pada Gambar 3.4 ditunjukkan Blok sistem navigasi robot beroda pemadam api. Sensor-sensor pada robot berfungsi sebagai masukan yang memberikan informasi keadaan lingkungan robot. Keadaan tersebut adalah ada atau tidaknya api, warna lantai, home, dan jarak robot dengan dinding kiri, kanan, depan, dan belakang. Informasi-informasi tersebut akan dikelompokkan menjadi sebuah kondisi yang menggambarkan (*percept*) lingkungan sekitar (*environment*) robot. Dari kondisi lingkungan tersebut maka robot dapat mengetahui sedang berada di ruang ke berapa dan *check point* yang mana yang sedang dilintasi. *Check point* merupakan daerah ataupun lokasi yang unik dari arena yang telah dipilih terlebih dahulu yang digunakan sebagai tempat berpindah metoda telusur.



Gambar 3.4 Diagram Blok Sistem Navigasi Robot Beroda Pemadam Api.

Koordinator berfungsi sebagai pengambil keputusan berdasarkan informasi-informasi yang telah diperoleh sebelumnya dan membandingkan dengan *state* sebelumnya (kondisi lingkungan sebelumnya) untuk menentukan gerakan robot (*action*) selanjutnya sesuai dengan yang telah dipetakan. Koordinator yang akan menentukan setiap keputusan perpindahan metoda berjalan robot, keputusan memadamkan api, dan keputusan berhenti pada saat sudah kembali ke *home*. Kordinator akan memberikan perintah yang harus dilakukan oleh aktuator sesuai dengan *condition-action rules*. Koordinator juga menyimpan *state-state* yang sudah dilewati untuk dibandingkan dengan *state* terakhir.

Pada realisasi robot beroda pemadam api ini model *reflex agent with state* dapat dideskripsikan sebagai berikut :

1. *Environment*

   Environment merupakan lingkungan sekitar dari *agent*. Pada penelitian ini, *environment* merupakan arena kontes robot pemadam api 2013. *Environment* ini terdiri dari lantai arena (berwarna hitam, putih, dan abu-abu), dinding, api lilin, *home*, furniture penghalang, dan *uneven floor* (berfungsi sebagai “polisi tidur”). Pada penelitian ini robot akan diujikan pada 14 jenis *environment* yang berbeda. Setiap jenis *environment*

memiliki konfigurasi letak home, api lilin, dan rintangan-rintangan yang berbeda-beda.

2. Sensor

   Sensor merupakan pemberi informasi dari *environment*. Sensor bekerja mengambil informasi yang dibutuhkan untuk mendeskrisikan kondisi dan perubahan dari *enviroment*. Pada realisasi robot ini sensor-sensor yang digunakan adalah sensor SRF-05 untuk mengukur jarak, sensor IR LED photodioda untuk membedakan warna lantai, dan sensor UV-tron untuk mendeteksi api.

3. Actuator

   Actuator merupakan peralatan mekanis untuk menggerakkan suatu sistem. Pada realisasi robot ini *actuator* yang digunakan ada beberapa, diantaranya adalah motor DC untuk menggerakkan roda robot, motor servo dan motor *brushless* untuk menggerakkan kipas.

4. Percept

   Percept merupakan semua kondisi dari *environment.* Pada penelitian ini yang menjadi *percept* merupakan kondisi-kondisi dari arena, seperti adanya persimpangan, ada tidaknya api lilin, adanya *home*, adanya rintangan, daerah *check point* api, dan *check point home*.

5. State

   State merupakan kondisi-kondisi mengenai lingkungan yang terlebih dahulu telah ditetapkan. *State* ini dapat berubah ataupun diperbaharui sesuai dengan perubahan kondisi lingkungan robot yang diketahui *agent* melalui sensor. Pada perancangan ini yang menjadi *state* merupakan hal-hal seperti : sudah atau belum memadamkan api, keluar masuk ruangan, dan jumlah garis yang dilewati.

6. *How the world evolves*

   Ini merupakan bagian dimana *agent* memeriksa perubahan dari lingkungannya. Misalnya saat melintasi *check point*, mendeteksi adanya api, melintasi persimpangan, mendeteksi adanya home, mendeteksi adanya garis, mendeteksi adanya halangan, dan mendeteksi perubahan warna lantai.

7. *What my action do*

   Ini merupakan bagian dimana *agent* akan menentukan *action* yang akan dilakukan sesuai dengan kondisi-kondisi perubahan lingkungan dari *agent*.

8. *Condition-action rules*

   Condition-action rules merupakan aturan pada suatu kondisi yang menyebabkan aksi itu dapat terjadi. Misalkan : jika melintasi *check point* maka robot harus berpindah metoda berjalan.

9. *Action*

   Action merupakan tindakan yang dilakukan *agent*. Untuk robot beroda pemadam api ini yang menjadi *action*-nya merupakan semua pergerakan dari robot, mulai dari menelusuri dinding, bermanuver, menemukan api, memadamkan api, dan berhenti di *home*.

III.3 Perancangan Posisi *Check Point* Pada Arena Kontes

Pada algoritma maze mapping digunakan 2 jenis check point, yaitu *check point* api dan *check point home*. *Check point* api digunakan untuk berpindah metoda berjalan dalam mencari api, sedangkan *check point home* digunakan untuk kembali ke *home*. Posisi untuk *check point* pencarian api terlebih dahulu ditentukan sesuai dengan kebutuhan untuk pemecahan masalah pada bentuk *maze* yang banyak. Titik *check point* ini ditentukan dengan pertimbangan :

Check point 1. daerah *check point* yang dipilih mempunyai parameter yang unik dibandingkan daerah lain.

Check point 2. Parameter yang digunakan antara lain adalah jarak robot terhadap dinding kiri, kanan, depan, dan belakang.

Check point 3. Daerah *check point* api yang dipilih merupakan daerah yang terdekat dari pintu setiap ruangan dengan catatan robot berjalan menggunakan metoda berjalan telusur dinding kiri.

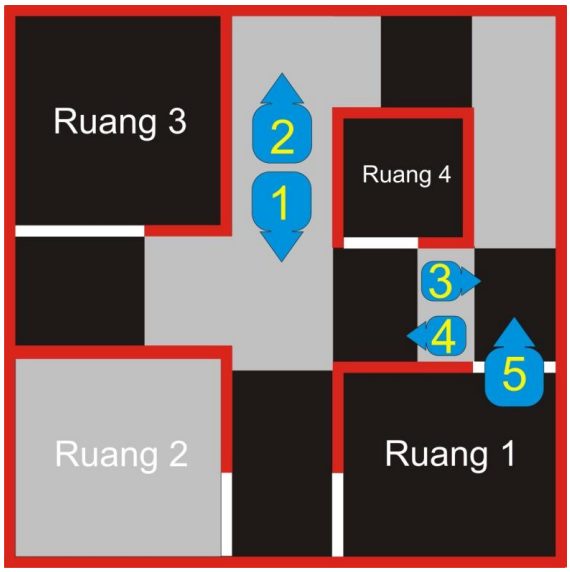
Check point 4. Daerah *check point home* yang dipilih merupakan daerah yang terdekat setelah ruangan api dengan catatan robot berjalan menggunakan metoda berjalan telusur dinding kanan.

Check point 5. Penentuan titik *check point* menggunakan metoda *trial & error* hingga mendapatkan daerah yang cocok dipakai untuk menyelesaikan setiap misi.

Warna lantai dan *counter* internal pada robot juga menjadi pertimbangan. Dengan kondisi yang unik tersebut maka dapat ditentukan pemilihan metoda berjalan dari robot pada posisi posisi dan keadaan tertentu.

III.3.1 Posisi *Check Point* Pencarian Api

Pada Gambar 3.15 ditunjukkan posisi-posisi penentuan *check point* pencarian api. Titik-titik tersebut digunakan untuk perpindahan metoda berjalan robot, sehingga robot dapat menemukan api.

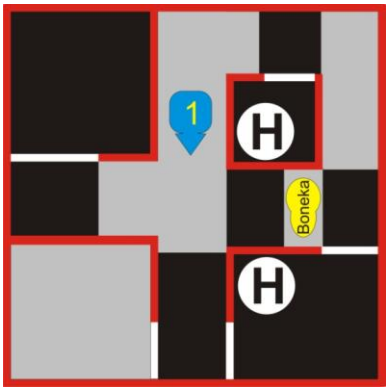


Gambar 3.5 Posisi-posisi *check point* pencarian api

Berikut penggunaan *check point* api tersebut :

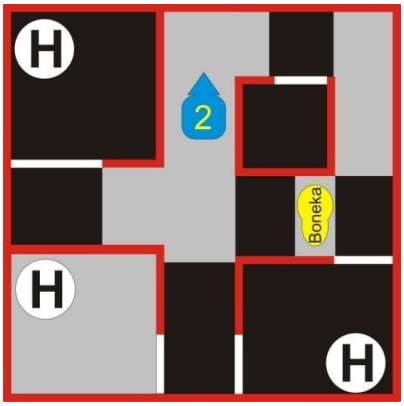
1) *Check point* api 1, 2, 3, dan 4 digunakan untuk berpindah dari metoda *wall* kiri ke *wall* kanan, karena pada kondisi ini api berada pada loop yang berbeda dengan loop home.

- *Check point* api 1 digunakan jika posisi *home* di ruang 4 atau ruang 1. Posisi home untuk *check point* 1 ditunjukkan pada Gambar 3.16.



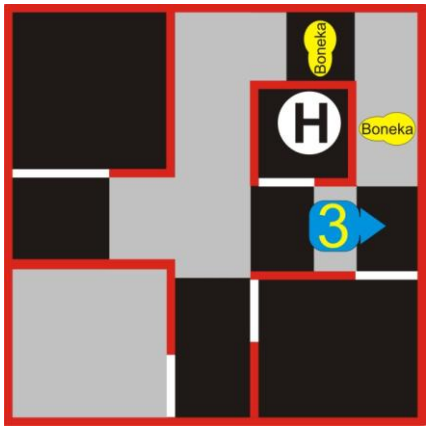
Gambar 3.6 Posisi *home* untuk *check point* 1

- *Check point* api 2 digunakan jika posisi *home* di ruang 1 atau ruang 2 atau ruang 3. Posisi *home* untuk *check point* 2 ditunjukkan pada Gambar 3.17.



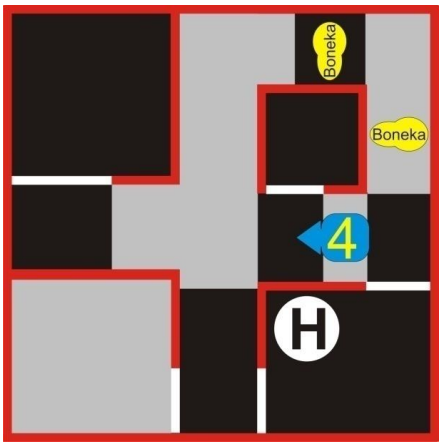
Gambar 3.7 Posisi *home* untuk check point 2

- *Check point* api 3 digunakan jika posisi *home* di ruang 4. Posisi *home* untuk *check point* 3 ditunjukkan pada Gambar 3.8.



Gambar 3.8 Posisi *home* untuk check point 3

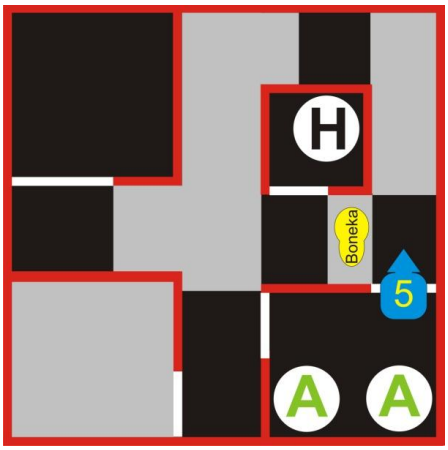
- *Check point* api 4 digunakan jika posisi *home* di ruang 1. Posisi *home* untuk *check point* 4 ditunjukkan pada Gambar 3.9.



Gambar 3.9 Posisi *home* untuk check point 4

2) *Check point* api 5 digunakan jika posisi *home* di ruang 4 dan api berada di ruang 1, karena pada kondisi ini api berada pada loop yang berbeda dengan loop home. Posisi *home* dan api untuk *check point* 5 ditunjukkan pada Gambar 3.10. Pada kondisi ini fungsi dari sensor api digunakan untuk mengecek posisi api berada di depan robot atau tidak. Jika posisi api tidak di depan robot maka *check point* 5

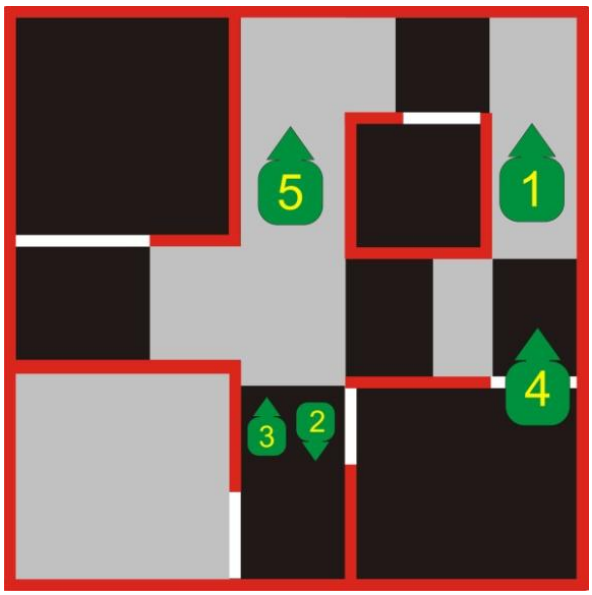
digunakan untuk berpindah metoda berjalan dari *wall* kiri ke *wall* kanan.



Gambar 3.10 Posisi *home* untuk check point 5

III.3.2 Posisi *Check Point Home*

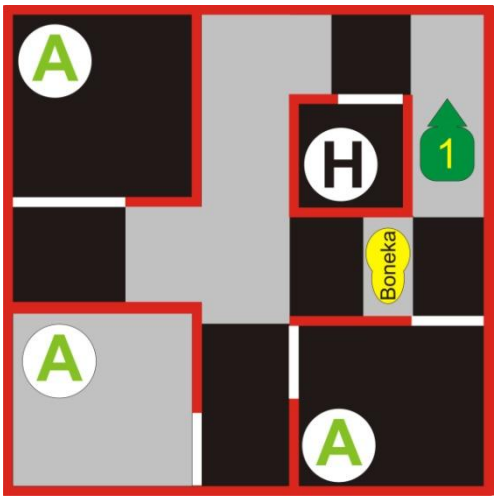
Pada Gambar 3.21 ditunjukkan posisi-posisi penentuan *check point* pencarian *home*. Titik-titik tersebut digunakan untuk perpindahan metoda berjalan robot untuk berpindah dari metoda *wall* kiri ke *wall* kanan, sehingga robot dapat menemukan *home*.



Gambar 3.11 Posisi-posisi *check point* pencarian *Home*

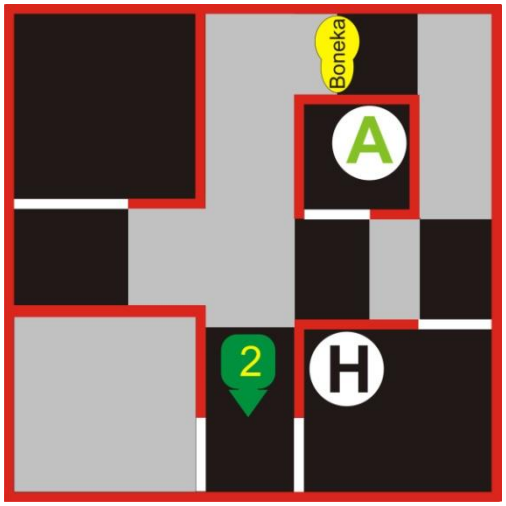
Berikut penggunaan *check point home* tersebut :

- *Check point home* 1 digunakan jika *home* berada di ruang 4 dan api berada di ruang 1 atau 2 atau 3. Posisi *home* dan api untuk *check point home* 1 ditunjukkan pada Gambar 3.12



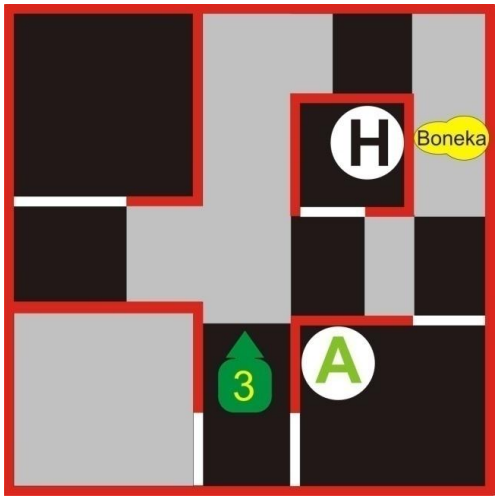
Gambar 3.12 Posisi *home* dan api untuk *check point Home* 1

- *Check point home* 2 digunakan jika *home* berada di ruang 1 dan api berada di ruang 4. Posisi *home* dan api untuk *check point home* 1 ditunjukkan pada Gambar 3.13



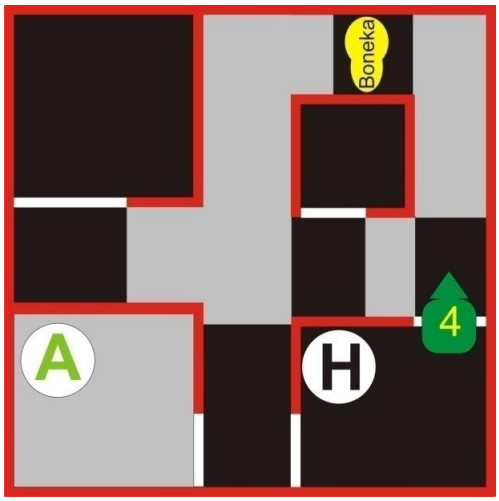
Gambar 3.13 Posisi *home* dan api untuk *check point Home* 2

- *Check point home* 3 digunakan jika *home* berada di ruang 4 dan api berada di ruang 1. Posisi *home* dan api untuk *check point home* 3 ditunjukkan pada Gambar 3.14



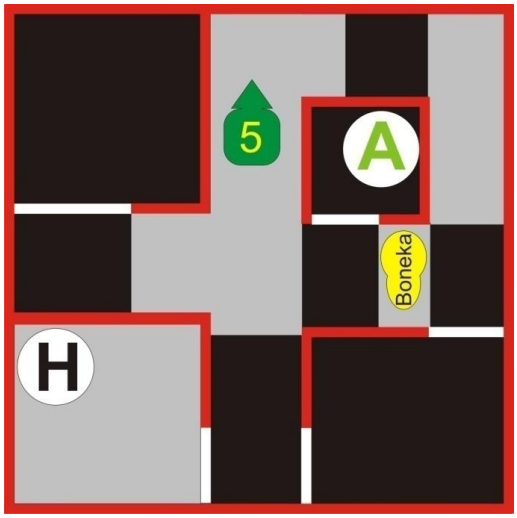
Gambar 3.14 Posisi *home* dan api untuk *check point Home* 3

- *Check point home* 4 digunakan jika *home* berada di ruang 1 dan api berada di ruang 2. Posisi *home* dan api untuk *check point home* 4 ditunjukkan pada Gambar 3.15



Gambar 3.15 Posisi *home* dan api untuk *check point Home* 4

- *Check point home* 5 digunakan jika *home* berada di ruang 2 dan api berada di ruang 4. Posisi *home* dan api untuk *check point home* 5 ditunjukkan pada Gambar 3.16



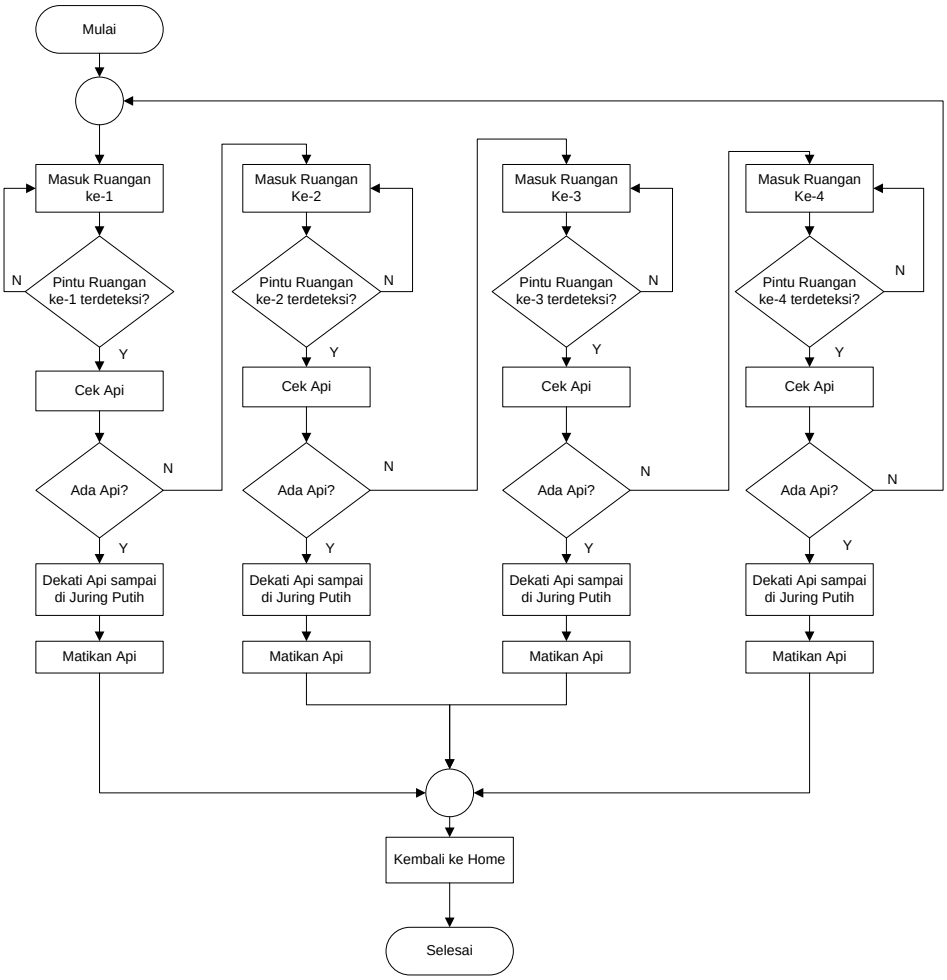
Gambar 3.16 Posisi *home* dan api untuk *check point Home* 5

III.4 Algoritma Pemrograman Pada Robot

Algoritma yang digunakan dalam pengontrolan robot ini berupa diagram alir. Diagram alir dibuat berdasarkan proses pengontrolan robot menggunakan pengontrol mikro ATmega128A.

III.4.1 *Flowchart* Dasar Robot Beroda Pemadam Api

Gambar 3.17 adalah *flowchart*/diagram alir dari program robot *wall follower*. Pada *flowchart* ini robot berjalan dengan satu metoda berjalan saja, metoda telusur kiri atau metoda telusur kanan. Pada Kontes Robot Pemadam Api 2013 ini target (api lilin dan *home*) diberi alas berupa juring lingkaran yang berwarna putih.

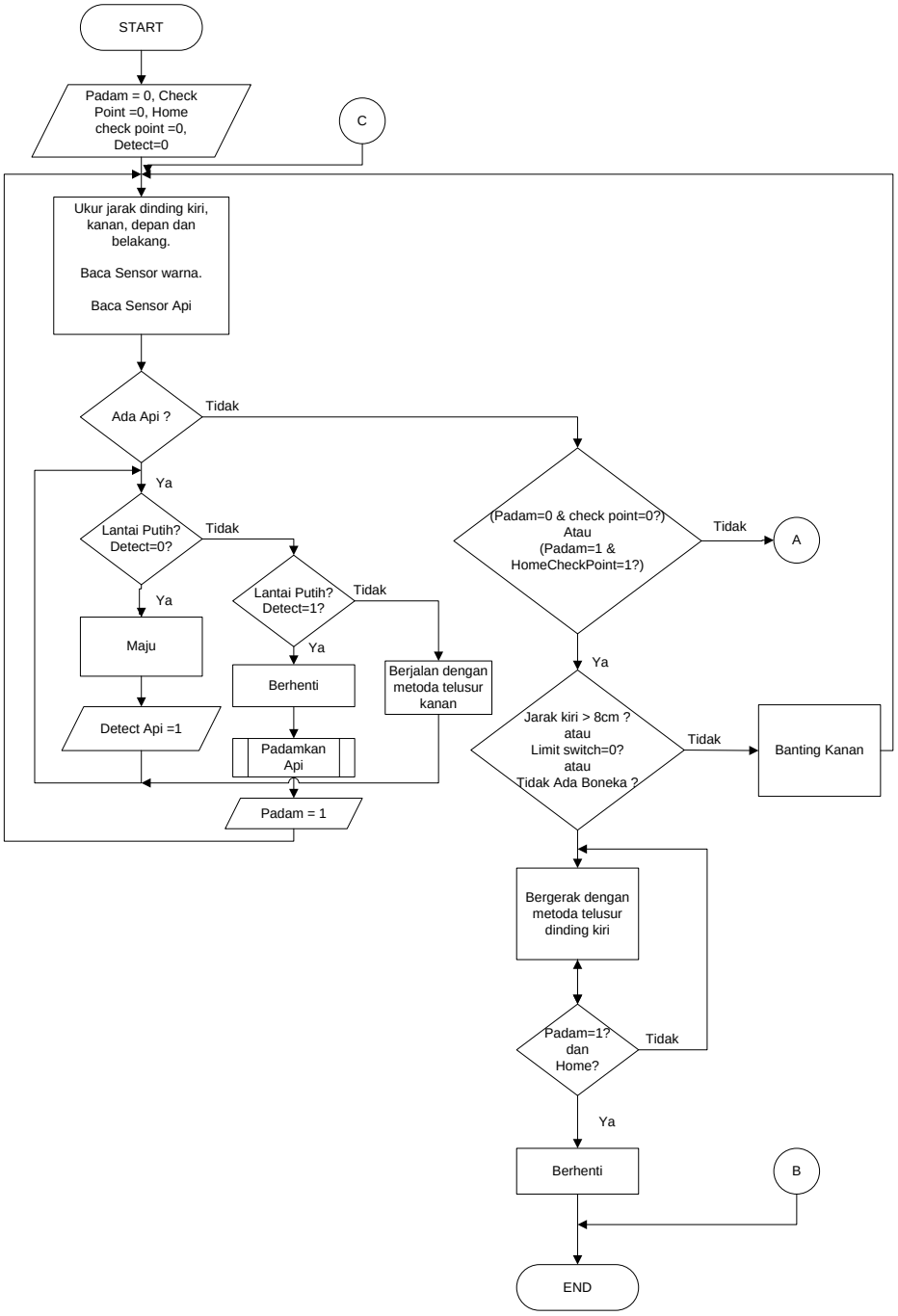


Gambar 3.17 *Flowchart* Dasar Robot Beroda Pemadam Api

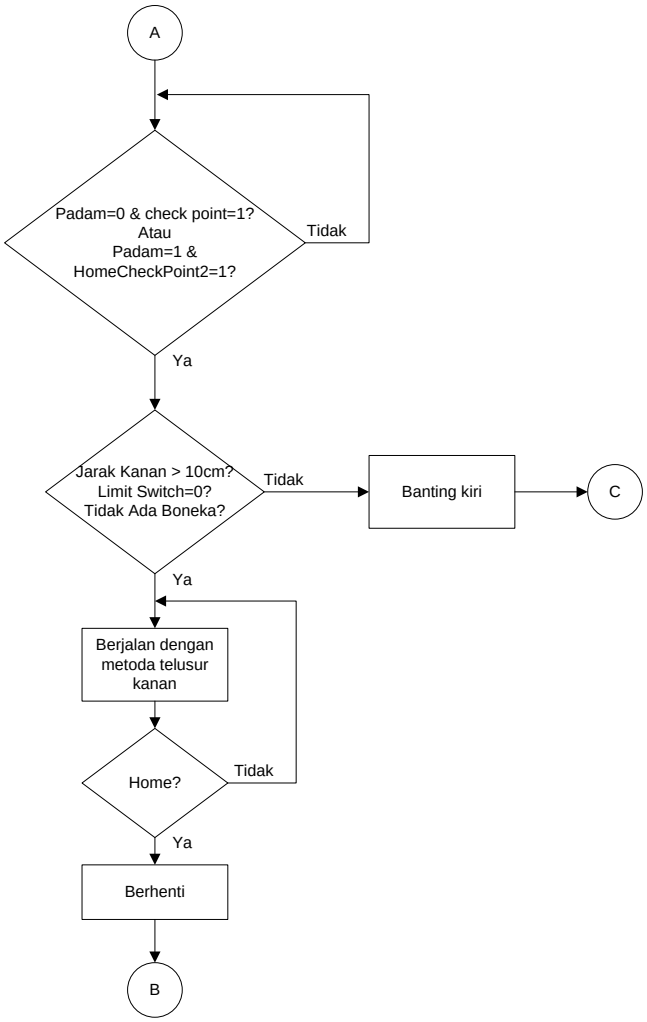
Robot akan berjalan menelusuri dinding kiri atau dinding kanan (salah satu) dan masuk ke ruangan pertama, kemudian robot mencari dan mendeteksi ada atau tidaknya api, jika robot menemukan adanya api maka robot memadamkan api, apabila robot tidak menemukan adanya api maka robot melanjutkan perjalanan menelusuri dinding untuk selanjutnya memasuki ruangan kedua dan seterusnya hingga menemukan api. Jika robot sudah memadamkan api maka robot akan mencari *home* dengan berjalan menggunakan metoda berjalan yang sama dengan sebelumnya.

III.4.2 *Flowchart* Robot Beroda Pemadam Api dengan Navigasi *Maze Mapping*

Pada Gambar 3.18a dan 3.18b ditunjukkan *flowchart* robot beroda pemadam api dengan navigasi *maze mapping*.



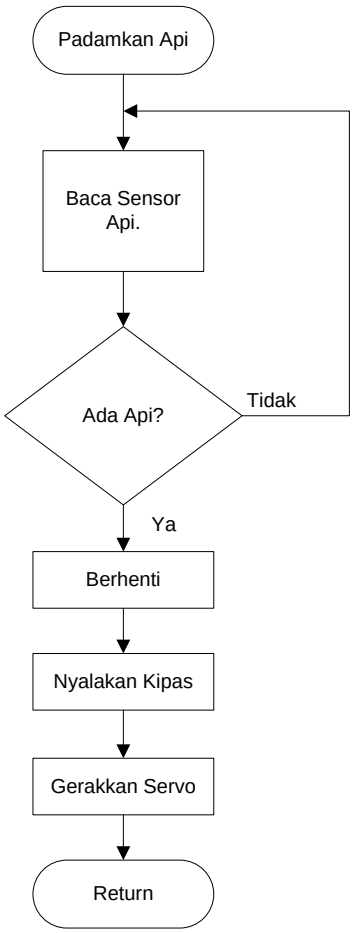
Gambar 3.18 (a) *Flowchart* robot beroda pemadam api dengan navigasi *maze mapping*.



Gambar 3.18 (b) *Flowchart* robot beroda pemadam api dengan navigasi *maze mapping*.

Pada *flowchart* tersebut robot terlebih dahulu memeriksa ada tidaknya api, jika robot tidak mendeteksi api maka robot akan berjalan dengan metoda telusur dinding kiri. Pada saat berjalan, jika robot melintasi salah satu lokasi *check point* api maka koordinator akan mengganti metoda berjalan robot menjadi metoda telusur dinding kanan. Ketika robot mendeteksi dan menemukan target (api lilin) maka robot akan memadamkan api, dan setelah memadamkan api maka robot akan melanjutkan perjalanan dengan metoda telusur dinding kanan. Namun bila selama perjalanan robot melintasi salah satu lokasi *check point home*, maka koordinator akan mengganti metoda berjalan robot menjadi metoda telusur dinding kiri sehingga robot dapat menemukan *home*. Jika robot mendeteksi adanya *home* maka robot akan berhenti.

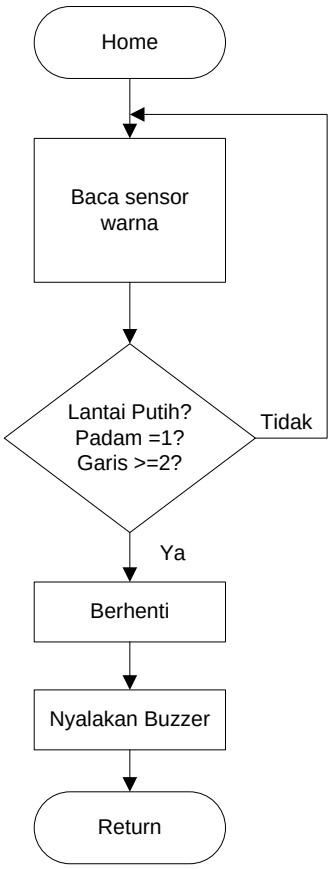
III.4.3 *Flowchart* Memadamkan Api



Gambar 3.19 *Flowchart* Memadamkan Api

Gambar 3.19 ditunjukkan *flowchart* memadamkan api. Pada saat menemukan target (api lilin) maka robot akan berhenti diatas juring lingkaran berwarna putih, dan mulai menyalakan kipas dan menggerakkan servo untuk memadamkan lilin hingga api lilin padam.

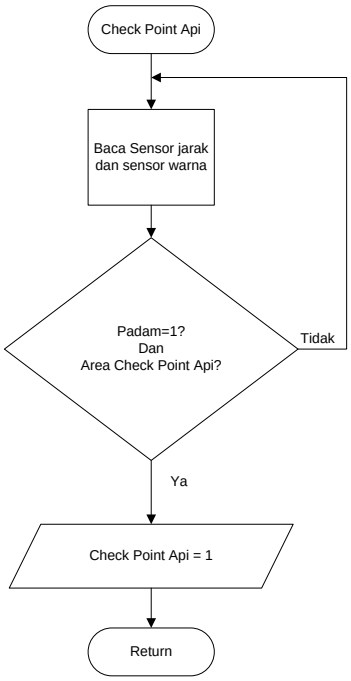
III.4.4 *Flowchart* Mendeteksi *Home*



Gambar 3.20 *Flowchart* Mendeteksi *Home*

Gambar 3.20 menunjukkan *Flowchart* mendeteksi *home*. Pada saat robot telah selesai memadamkan api maka robot akan mencari *home*, *home* ditandai dengan juring lingkaran berwarna putih. Maka untuk mendeteksi *home* robot akan memeriksa kondisi state sebelumnya, yaitu robot sudah memadamkan api (ditandai dengan variable padam=1), robot sudah memasuki ruangan setelah ruangan yg ada api (ditandai dengan variable garis>=2), dan mendeteksi warna lantai yang putih. Jika kondisi-kondisi diatas terpenuhi maka robot akan berhenti dan menyalakan *buzzer* sebagai penanda sudah menyelesaikan misi.

III.4.5 *Flowchart* Mendeteksi Lokasi *Check Point* Api



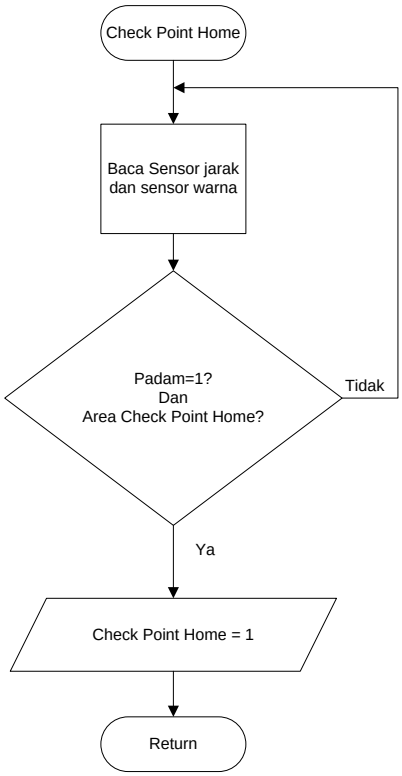
Gambar 3.21 *Flowchart* Mendeteksi Lokasi *Check Point* Api

Gambar 3.21 ditunjukkan *flowchart* mendeteksi lokasi *check point* api. Untuk mendeteksi lokasi *check point* api robot akan memeriksa kondisi *state* sebelumnya, yaitu robot belum memadamkan api (ditandai dengan variable padam=0), warna lantai *check point* api, dan lokasi *check point api* yang dideteksi robot dengan cara mengukur jarak robot dengan dinding kiri, kanan, depan dan belakang pada lokasi-lokasi tertentu yang terlebih dahulu telah ditentukan. Jika kondisi-kondisi diatas terpenuhi maka nilai variable *check point* akan menjadi 1.

III.4.6 Perancangan *Flowchart* Mendeteksi Lokasi *Check Point Home*

Gambar 3.22 ditunjukkan *flowchart* mendeteksi lokasi *check point home*. Untuk mendeteksi lokasi *check point* robot akan memeriksa kondisi state sebelumnya, yaitu robot sudah memadamkan api (ditandai dengan variable padam=1), warna lantai abu-abu, dan lokasi *check point home* yang dideteksi

robot dengan cara mengukur jarak robot dengan dinding kiri, kanan, depan dan belakang pada lokasi-lokasi tertentu yang terlebih dahulu telah ditentukan. Jika kondisi-kondisi diatas terpenuhi maka nilai variable *check point home* akan menjadi 1.



Gambar 3.22 *Flowchart* Mendeteksi Lokasi *Check Point Home*

III.5 Pemetaan *Maze* Pada *Check Point* Api

Untuk pemetaan *maze* pada *check point* api maka dilakukan pengukuran terhadap 2 parameter, yaitu parameter jarak dinding dan parameter warna lantai. Pengukuran parameter jarak dinding dilakukan antara robot terhadap dinding kiri, kanan, depan, dan belakang. Pengukuran parameter warna lantai dilakukan terhadap lantai setiap *check point* api. Nilai-nilai parameter tersebut dipakai untuk mewakili setiap *check point* api.

III.5.1 Pengukuran Jarak Robot terhadap Dinding pada setiap *check point* api

Pengukuran jarak dilakukan menggunakan sensor ultrasonik SRF05 terhadap dinding depan, belakang, kiri, dan kanan dari robot. Pengambilan data dilakukan sebanyak 5 kali percobaan pada setiap *check point* api dengan nilai yang berbeda, hal ini dilakukan agar data yang diperoleh dapat mewakili nilai pembacaan jarak pada saat robot bergerak. Untuk rentang nilai jarak robot terhadap dinding diberi tolerasi +/- 5 cm, karena area pantul dari sensor selalu berubah-ubah seiring pergerakan robot yang terus bergerak.

- **Posisi *Check Point* api 1**

Untuk pengambilan data pada *check point* api 1, tabel percobaan ditunjukkan pada Tabel 3.1.

Tabel 3.1. Tabel Pengukuran Jarak Pada *Check Point* 1

No	Jarak Depan (cm)	Jarak Belakang (cm)	Jarak Kiri (cm)	Jarak Kanan (cm)
1	79	41	13	11
2	87	46	11	12
3	78	32	4	18
4	64	35	6	18
5	75	50	13	11
Range	59 – 92	27 – 55	0 – 18	0 – 23

Berdasarkan Tabel 3.1 dapat disimpulkan bahwa range nilai pengukuran jarak antara robot dengan dinding depan berkisar antara 59 cm sampai 92 cm, jarak dengan dinding belakang berkisar antara 27 cm sampai 55 cm, jarak dengan dinding kiri berkisar antara 0 cm sampai 18 cm, sedangkan jarak dengan dinding kanan berkisar antara 0 cm sampai 23 cm.

- **Posisi *Check Point* api 2**

Untuk pengambilan data pada *check point* api 2 posisi robot ditunjukkan pada pada Tabel 3.2. Berdasarkan Tabel 3.2 dapat disimpulkan bahwa range nilai pengukuran jarak antara robot dengan dinding depan berkisar antara 28 cm sampai 59 cm, jarak dengan dinding belakang berkisar antara 48 cm sampai 83 cm, jarak dengan dinding kiri dan dinding kanan berkisar antara 0 cm sampai 21 cm.

Tabel 3.2. Tabel Pengukuran Jarak Pada *Check Point* 2

No	Jarak Depan (cm)	Jarak Belakang (cm)	Jarak Kiri (cm)	Jarak Kanan (cm)
1	44	78	11	12
2	54	65	7	16
3	33	62	11	13
4	41	72	16	7
5	43	52	13	10
Range	28 – 59	48 – 83	0 – 21	0 – 21

- **Posisi *Check Point* api 3**

Untuk pengambilan data pada *check point* api 3 posisi robot ditunjukkan pada pada Tabel 3.3.

Tabel 3.3 Tabel Pengukuran Jarak Pada *Check Point* 3

No	Jarak Depan (cm)	Jarak Belakang (cm)	Jarak Kiri (cm)	Jarak Kanan (cm)
1	36	33	9	10
2	37	76	5	16
3	37	74	7	15
4	38	44	12	11
5	38	31	11	8
Range	31 – 43	26 – 81	0 – 17	0 – 21

Berdasarkan Tabel 3.3 dapat disimpulkan bahwa range nilai pengukuran jarak antara robot dengan dinding depan berkisar antara 31 cm sampai 43 cm, jarak dengan dinding belakang berkisar antara 26 cm sampai 81 cm, jarak dengan dinding kiri berkisar antara 0 cm sampai 17 cm, sedangkan jarak dengan dinding kanan berkisar antara 0 cm sampai 21 cm.

- **Posisi *Check Point* api 4**

Untuk pengambilan data pada *check point* api 4 posisi robot ditunjukkan pada pada Tabel 3.4. Berdasarkan Tabel 3.4 dapat disimpulkan bahwa range nilai pengukuran jarak antara robot dengan dinding depan berkisar antara 60 cm sampai 83 cm, jarak dengan dinding belakang berkisar antara 29 cm sampai 44 cm, jarak dengan dinding kiri berkisar antara 0 cm sampai 19 cm, sedangkan jarak dengan dinding kanan berkisar antara 0 cm sampai 22 cm.

Tabel 3.4 Tabel Pengukuran Jarak Pada *Check Point* 4

No	Jarak Depan (cm)	Jarak Belakang (cm)	Jarak Kiri (cm)	Jarak Kanan (cm)
1	66	39	4	17
2	65	37	7	16
3	76	36	13	11
4	75	38	14	9
5	78	34	8	15
Range	60 – 83	29 – 44	0 – 19	0 – 22

- **Posisi *Check Point* api 5**

Untuk pengambilan data pada *check point* api 5 posisi robot ditunjukkan pada pada Tabel 3.5.

Tabel 3.5 Tabel Pengukuran Jarak Pada *Check Point* api 5

No	Jarak Depan (cm)	Jarak Belakang (cm)	Jarak Kiri (cm)	Jarak Kanan (cm)
1	74	53	16	7
2	75	65	16	7
3	74	60	15	9
4	87	48	17	7
5	74	65	13	8
Range	69 – 92	48 – 70	0 – 22	0 – 14

Berdasarkan Tabel 3.5 dapat disimpulkan bahwa range nilai pengukuran jarak antara robot dengan dinding depan berkisar antara 69 cm sampai 92 cm, jarak dengan dinding belakang berkisar antara 43 cm sampai 70 cm, jarak dengan dinding kiri berkisar antara 0 cm sampai 22 cm, sedangkan jarak dengan dinding kanan berkisar antara 0 cm sampai 14 cm.

III.5.2 Pembacaan Sensor Warna terhadap Warna Lantai pada setiap *check point* api.

Sensor warna yang digunakan adalah photodioda sebagai penerima dan LED infra red sebagai pemancar. Jarak sensor depan dari lantai adalah 2 cm dan jarak sensor belakang dari lantai adalah 3 cm. Percobaan yang dilakukan adalah pengukuran terhadap warna lantai pada saat robot berada di setiap *check point* api. Pengukuran dilakukan pada beberapa posisi untuk masing-masing warna sehingga diketahui range nilai yang dibaca sensor pada saat bergerak di daerah check point

api yang telah ditentukan. Penentuan *range* nilai pembacaan sensor warna terhadap lantai diberi nilai toleransi +/- 50, hal ini dilakukan untuk menghindari kesalahan pembacaan warna lantai pada saat robot bergerak. Berikut adalah nilai pembacaan sensor warna pada setiap *check point* api.

- Pada *check point* api 1 warna lantai adalah warna abu-abu. Data pengamatan pembacaan sensor warna di *check point* 1 dapat dilihat pada Tabel 3.6.

Tabel 3.6 Nilai pembacaan sensor warna di *check point* api 1

No	Depan Kiri	Depan Kanan	Belakang
1	402	417	639
2	417	369	636
3	396	374	633
4	404	368	644
5	419	355	629
Range	376 – 439	335 – 437	619 – 664

Berdasarkan Tabel 3.6 dapat disimpulkan bahwa range nilai pembacaan sensor warna pada *check point* 1 berkisar antara 346 sampai 469 untuk sensor depan kiri, 305 sampai 467 untuk sensor depan kanan, dan 583 sampai 694 untuk sensor belakang.

- Pada *check point* api 2 warna lantai adalah warna abu-abu. Data pengamatan pembacaan sensor warna di *check point* 2 dapat dilihat pada Tabel 3.7.

Tabel 3.7 Nilai pembacaan sensor warna di *check point* api 2

No	Depan Kiri	Depan Kanan	Belakang
1	432	380	665
2	452	383	654
3	448	376	387
4	450	366	651
5	457	395	666
Range	512 – 477	336 – 415	367 – 686

Berdasarkan Tabel 3.7 dapat disimpulkan bahwa range nilai pembacaan sensor warna pada *check point* 2 berkisar antara 482 sampai 507 untuk sensor depan kiri, 316 sampai 345 untuk sensor depan kanan, dan 337 sampai 716 untuk sensor belakang.

- Pada *check point* api 3 warna lantai adalah warna abu-abu. Data pengamatan pembacaan sensor warna di *check point* 3 dapat dilihat padaTabel 3.8.

Tabel 3.8 Nilai pembacaan sensor warna di *check point* api 3

No	Depan Kiri	Depan Kanan	Belakang
1	421	362	634
2	429	379	640
3	437	357	632
4	428	351	638
5	425	396	642
Range	401 – 457	331 – 416	612 – 662

Berdasarkan Tabel 3.8 dapat disimpulkan bahwa range nilai pembacaan sensor warna pada *check point* 3 berkisar antara 371 sampai 487 untuk sensor depan kiri, 301 sampai 446 untuk sensor depan kanan, dan 582 sampai 692 untuk sensor belakang.

- Pada *check point* api 4 warna lantai adalah warna abu-abu. Data pengamatan pembacaan sensor warna di *check point* 4 dapat dilihat pada Tabel 3.9.

Tabel 3.9 Nilai pembacaan sensor warna di *check point* api 4

No	Depan Kiri	Depan Kanan	Belakang
1	398	378	647
2	402	336	652
3	410	367	645
4	407	350	656
5	416	386	661
Range	378 – 387	316 – 406	615 – 691

Berdasarkan Tabel 3.9 dapat disimpulkan bahwa range nilai pembacaan sensor warna pada *check point* 4 berkisar antara 348 sampai 417 untuk sensor depan kiri, 282 sampai 436 untuk sensor depan kanan, dan 695 sampai 771 untuk sensor belakang.

- Pada *check point* api 5 warna lantai adalah warna hitam. Data pengamatan pembacaan sensor warna di *check point* 5 dapat dilihat pada Tabel 3.10.

Tabel 3.10 Nilai pembacaan sensor warna di *check point* api 5

No	Depan Kiri	Depan Kanan	Belakang
1	502	624	772
2	520	615	784
3	548	623	780
4	520	619	785
5	501	614	788
Range	481 – 568	594 – 644	760 – 808

Berdasarkan Tabel 3.10 dapat disimpulkan bahwa range nilai pembacaan sensor warna pada *check point* 5 berkisar antara 451 sampai 598 untuk sensor depan kiri, 564 sampai 673 untuk sensor depan kanan, dan 735 sampai 834 untuk sensor belakang.

III.6 Pemetaan *Maze* Pada *Check Point Home*

Untuk pemetaan *maze* pada *check point home* maka dilakukan pengukuran terhadap 2 parameter, yaitu parameter jarak dinding dan parameter warna lantai. Pengukuran parameter jarak dinding dilakukan antara robot terhadap dinding kiri, kanan, depan, dan belakang. Pengukuran parameter warna lantai dilakukan terhadap lantai setiap *check point home*. Nilai-nilai parameter tersebut dipakai untuk mewakili setiap *check point home*.

III.6.1 Pengukuran Jarak Robot Terhadap Dinding pada setiap *check point home*

Pengukuran jarak dilakukan menggunakan sensor ultrasonik SRF05 terhadap dinding depan, belakang, kiri, dan kanan dari robot. Pengambilan data dilakukan sebanyak 5 kali percobaan dengan posisi yang berbeda-beda, pengukuran dilakukan saat robot diam (tidak berjalan). Hal ini dilakukan agar data yang diperoleh dapat mewakili nilai pembacaan jarak pada saat robot bergerak. Untuk rentang nilai jarak robot terhadap dinding diberi tolerasi +/- 5 cm, karena area pantul dari sensor selalu berubah-ubah seiring pergerakan robot yang terus bergerak.

- Untuk pengambilan data pada *check point home* 1 posisi robot ditunjukkan pada Tabel 3.11.

Tabel 3.11. Tabel Pengukuran Jarak Pada *Check Point Home* 1

No	Jarak Depan (cm)	Jarak Belakang (cm)	Jarak Kiri (cm)	Jarak Kanan (cm)
1	37	63	11	12
2	38	51	10	13
3	47	54	7	14
4	36	61	9	15
5	36	34	9	14
Range	31 – 52	29 – 68	0 – 16	0 – 20

Berdasarkan Tabel 3.11 dapat disimpulkan bahwa range nilai pengukuran jarak antara robot dengan dinding depan berkisar antara 31 cm sampai 52 cm, jarak dengan dinding belakang berkisar antara 29 cm sampai 68 cm, jarak dengan dinding kiri berkisar antara 0 cm sampai 16 cm, sedangkan jarak dengan dinding kanan berkisar antara 0 cm sampai 20 cm.

- Untuk pengambilan data pada *check point home* 2 posisi robot ditunjukkan pada Tabel 3.12.

Tabel 3.12. Tabel Pengukuran Jarak Pada *Check Point Home* 2

No	Jarak Depan (cm)	Jarak Belakang (cm)	Jarak Kiri (cm)	Jarak Kanan (cm)
1	38	60	10	4
2	50	71	5	11
3	48	48	17	13
4	36	49	14	6
5	37	67	8	7
Range	31 – 55	43 – 76	0 – 22	0 – 18

Berdasarkan Tabel 3.12 dapat disimpulkan bahwa range nilai pengukuran jarak antara robot dengan dinding depan berkisar antara 31 cm sampai 55 cm, jarak dengan dinding belakang berkisar antara 43 cm sampai 76 cm, jarak dengan dinding kiri berkisar antara 0 cm sampai 22 cm, sedangkan jarak dengan dinding kanan berkisar antara 0 cm sampai 18 cm.

- Untuk pengambilan data pada *check point home* 3 posisi robot ditunjukkan pada Tabel 3.13.

Tabel 3.13 Tabel Pengukuran Jarak Pada *Check Point Home* 3

No	Jarak Depan (cm)	Jarak Belakang (cm)	Jarak Kiri (cm)	Jarak Kanan (cm)
1	56	44	6	4
2	98	52	8	12
3	110	48	10	15
4	54	47	7	11
5	105	54	6	16
Range	49 – 110	39 – 57	0 – 15	0 – 21

Berdasarkan Tabel 3.13 dapat disimpulkan bahwa range nilai pengukuran jarak antara robot dengan dinding depan berkisar antara 49 cm sampai 110 cm, jarak dengan dinding belakang berkisar antara 39 cm sampai 57 cm, jarak dengan dinding kiri berkisar antara 0 cm sampai 15 cm, sedangkan jarak dengan dinding kanan berkisar antara 0 cm sampai 21 cm.

- Untuk pengambilan data pada *check point home* 4 posisi robot ditunjukkan pada Tabel 3.14.

Tabel 3.14 tabel pengukuran jarak pada *check point home* 4

No	Jarak Depan (cm)	Jarak Belakang (cm)	Jarak Kiri (cm)	Jarak Kanan (cm)
1	80	62	16	4
2	79	64	13	10
3	76	60	16	11
4	78	72	17	9
5	67	79	18	10
Range	62 – 85	55 – 84	0 – 23	0 – 16

Berdasarkan Tabel 3.14 dapat disimpulkan bahwa range nilai pengukuran jarak antara robot dengan dinding depan berkisar antara 62 cm sampai 85 cm, jarak dengan dinding belakang berkisar antara 55 cm sampai 84 cm, jarak dengan dinding kiri berkisar antara 0 cm sampai 23 cm, sedangkan jarak dengan dinding kanan berkisar antara 0 cm sampai 16 cm.

- Untuk pengambilan data pada *check point home* 5 posisi robot ditunjukkan pada Tabel 3.15.

Tabel 3.15 Tabel Pengukuran Jarak Pada *Check Point Home* 5

No	Jarak Depan (cm)	Jarak Belakang (cm)	Jarak Kiri (cm)	Jarak Kanan (cm)
1	63	54	12	4
2	51	64	11	11
3	54	41	13	13
4	61	55	18	10
5	34	68	11	4
Range	29 – 68	36 – 73	0 – 23	0 – 18

Berdasarkan Tabel 3.15 dapat disimpulkan bahwa range nilai pengukuran jarak antara robot dengan dinding depan berkisar antara 29 cm sampai 68 cm, jarak dengan dinding belakang berkisar antara 36 cm sampai 73 cm, jarak dengan dinding kiri berkisar antara 0 cm sampai 23 cm, sedangkan jarak dengan dinding kanan berkisar antara 0 cm sampai 18 cm.

III.6.2 Pembacaan Sensor Warna Terhadap Warna Lantai pada setiap *check point home*

Sensor warna yang digunakan adalah photodioda sebagai penerima dan LED infra red sebagai pemancar. Jarak sensor depan dari lantai adalah 2 cm dan jarak sensor belakang dari lantai adalah 3 cm. Percobaan yang dilakukan adalah pengukuran terhadap warna lantai pada saat robot berada di setiap *check point* api. Pengukuran dilakukan pada beberapa posisi untuk masing-masing warna sehingga diketahui range nilai yang dibaca sensor pada saat bergerak di daerah check point api yang telah ditentukan. Penentuan *range* nilai pembacaan sensor warna terhadap lantai diberi nilai toleransi +/- 50, hal ini dilakukan untuk menghindari kesalahan pembacaan warna lantai pada saat robot bergerak.

- Pada *check point home* 1 warna lantai adalah warna abu-abu. Data pengamatan pembacaan sensor warna di *check point home* 1 dapat dilihat pada Tabel 3.16.

Tabel 3.16 Nilai pembacaan sensor warna di *check point home* 1

No	Depan Kiri	Depan Kanan	Belakang
1	452	372	648
2	441	382	653
3	451	361	647
4	449	403	654
5	449	392	645
Range	421 – 469	341 – 423	625 – 674

Berdasarkan Tabel 3.16 dapat disimpulkan bahwa range nilai pembacaan sensor warna pada *check point home* 1 berkisar antara 391 sampai 499 untuk sensor depan kiri, 311 sampai 453 untuk sensor depan kanan, dan 595 sampai 704 untuk sensor belakang.

- Pada *check point home* 2 warna lantai adalah warna hitam. Data pengamatan pembacaan sensor warna di *check point home* 2 dapat dilihat pada Tabel 3.17.

Tabel 3.17 Nilai pembacaan sensor warna di *check point home* 2

No	Depan Kiri	Depan Kanan	Belakang
1	558	635	783
2	562	610	777
3	559	649	792
4	581	627	784
5	564	638	785
Range	538 – 601	590 – 669	757 – 812

Berdasarkan Tabel 3.17 dapat disimpulkan bahwa range nilai pembacaan sensor warna pada *check point home* 2 berkisar antara 508 sampai 631 untuk sensor depan kiri, 560 sampai 699 untuk sensor depan kanan, dan 727 sampai 842 untuk sensor belakang.

- Pada *check point home* 3 warna lantai adalah warna hitam. Data pengamatan pembacaan sensor warna di *check point home* 3 dapat dilihat pada Tabel 3.18.

Tabel 3.18 Nilai pembacaan sensor warna di *check point home* 3

No	Depan Kiri	Depan Kanan	Belakang
1	532	625	794
2	527	643	782
3	534	627	785
4	530	627	786
5	532	640	791
Range	507 – 554	605 – 663	702 – 814

Berdasarkan Tabel 3.18 dapat disimpulkan bahwa range nilai pembacaan sensor warna pada *check point home* 3 berkisar antara 477 sampai 584 untuk sensor depan kiri, 575 sampai 693 untuk sensor depan kanan, dan 732 sampai 844 untuk sensor belakang.

- Pada *check point home* 4 warna lantai adalah warna hitam. Data pengamatan pembacaan sensor warna di *check point home* 4 dapat dilihat pada Tabel 3.19.

Tabel 3.19 Nilai pembacaan sensor warna di *check point home* 4

No	Depan Kiri	Depan Kanan	Belakang
1	524	631	779
2	524	646	781
3	522	641	768
4	531	658	782
5	569	642	770
Range	502 – 589	611 – 688	748 – 802

Berdasarkan Tabel 3.19 dapat disimpulkan bahwa range nilai pembacaan sensor warna pada *check point home* 4 berkisar antara 472 sampai 619 untuk sensor depan kiri, 581 sampai 708 untuk sensor depan kanan, dan 718 sampai 832 untuk sensor belakang.

- Pada *check point home* 5 warna lantai adalah warna abu-abu. Data pengamatan pembacaan sensor warna di *check point home* 5 dapat dilihat pada Tabel 3.20.

Tabel 3.20 Nilai pembacaan sensor warna di *check point home* 5

No	Depan Kiri	Depan Kanan	Belakang
1	435	380	631
2	447	359	646
3	441	392	653
4	438	417	653
5	437	382	641
Range	415 – 467	339 – 437	611 – 673

Berdasarkan Tabel 3.20 dapat disimpulkan bahwa range nilai pembacaan sensor warna pada *check point home* 5 berkisar antara 385 sampai 497 untuk sensor depan kiri, 309 sampai 467 untuk sensor depan kanan, dan 581 sampai 703 untuk sensor belakang.

HASIL PENELITIAN

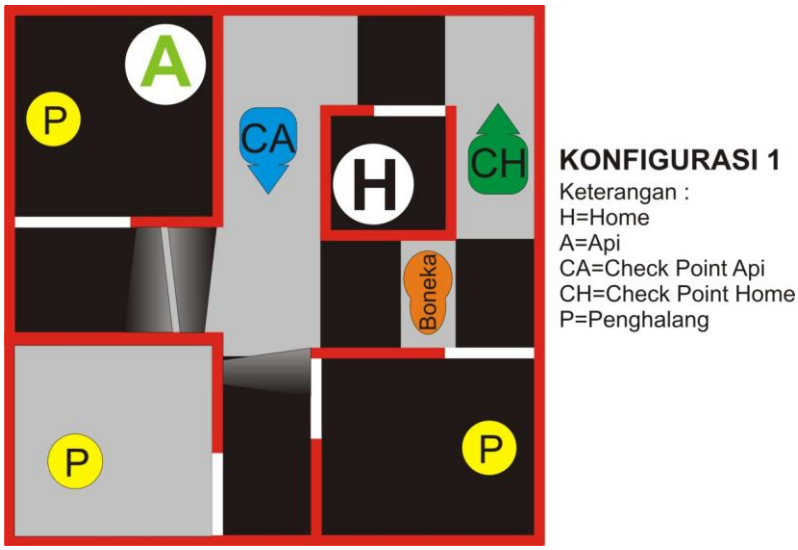
Pada bab ini dijelaskan tentang hasil penelitian berupa proses pengambilan data pengamatan untuk pemetaan maze pada *check point* api dan *check point home*, pengujian kemampuan robot beroda pemadam api, dan analisisnya.

IV.1 Pengujian Robot

Pada sub bab ini dilakukan pengujian robot pada lapangan arena kontes KRPAI 2013. Pengujian ini dilakukan terhadap 14 jenis kombinasi letak *home* dan posisi titik api. Untuk setiap jenis konfigurasi dilakukan pengujian sebanyak 5 kali percobaan.

1) Pengujian Robot Pada Konfigurasi Lapangan 1

Pada konfigurasi ini *home* berada pada ruang 4 dan posisi api berada pada ruang 3. Posisi *home*, api, boneka anjing, dan penghalang dapat dilihat pada Gambar 4.1. Data pengamatan hasil percobaan konfigurasi lapangan 1 dapat dilihat pada Tabel 4.1 dan Tabel 4.2.



Gambar 4.1 Konfigurasi Lapangan 1

Tabel 4.1 Data Pengujian Parameter *Mapping* Area dengan Konfigurasi Lapangan 1

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	79	38	62	46	84	32	76	31	72	47
Belakang (cm)	44	73	86	86	32	54	41	50	39	61
Kiri (cm)	8	9	7	12	13	9	14	12	8	11
Kanan (cm)	15	12	17	14	12	17	12	14	13	12
Warna (ADC)	398	411	402	388	387	401	419	391	410	413

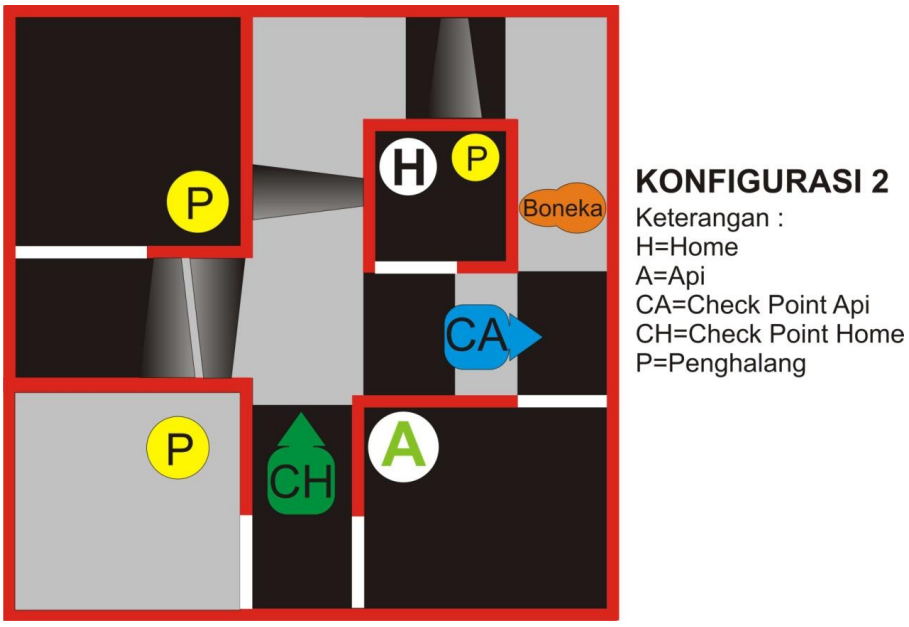
Tabel 4.2 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 1

No	***Check Point***		**Waktu (detik)**			**Keterangan**
	Api	***Home***	**Api**	***Home***	**Total**	
1	✓	✓	17.6	16.6	34.2	Berhasil
2	✓	×	27.5	-	-	Gagal *Home*
3	✓	✓	24.5	26.1	50.6	Berhasil
4	✓	✓	16.2	17.7	33.9	Berhasil
5	✓	✓	17.1	29.4	46.5	Berhasil

Berdasarkan Tabel 4.1 dan Tabel 4.2 dapat disimpulkan pada konfigurasi lapangan 1 robot dapat berhasil memadamkan api dan sampai ke *home* dengan waktu tempuh rata-rata 41.3 detik. Pada percobaan ke-2 robot gagal menemukan *home* disebabkan oleh pengukuran jarak belakang tidak sesuai dengan jarak yang ditetapkan pada program *mapping* sehingga robot tidak berpindah metoda berjalan saat melintasi *check point home*.

2) Pengujian Robot Pada Konfigurasi Lapangan 2

Pada konfigurasi ini *home* berada pada ruang 4 dan posisi api berada pada ruang 1. Posisi *home*, api, boneka anjing, dan penghalang dapat dilihat pada Gambar 4.2. Data pengamatan hasil percobaan konfigurasi 2 dapat dilihat pada Tabel 4.3 dan Tabel 4.4.



Gambar 4.2 Konfigurasi Lapangan 2

Tabel 4.3 Data Pengujian Parameter *Mapping* Area dengan Konfigurasi Lapangan 2

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	36	102	31	115	33	91	38	122	41	73
Belakang (cm)	73	45	41	44	75	52	72	53	71	41
Kiri (cm)	9	11	7	9	11	14	11	10	9	14
Kanan (cm)	14	15	16	12	17	14	14	16	12	11
Warna (ADC)	411	517	432	516	421	544	419	532	440	512

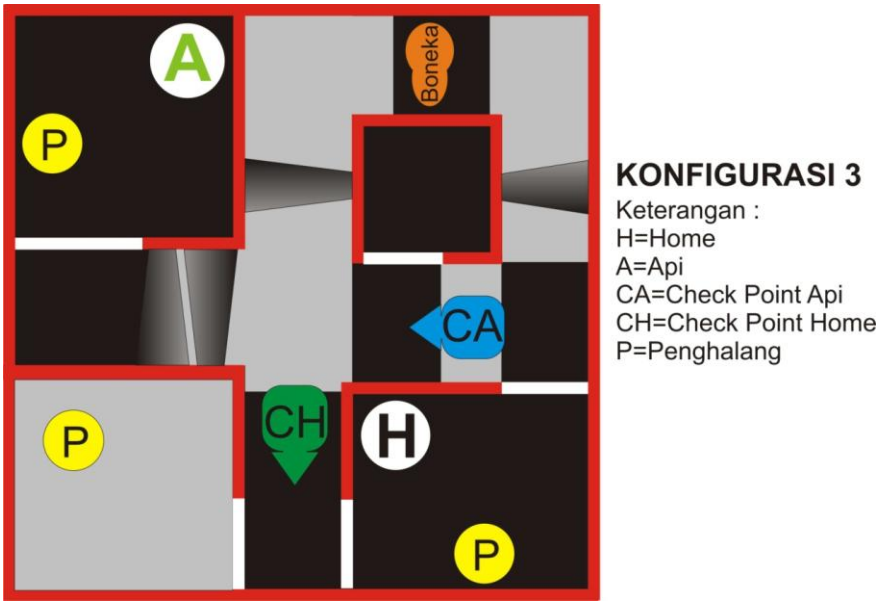
Tabel 4.4 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 2

No	***Check Point***		**Waktu (detik)**			**Keterangan**
	Api	***Home***	**Api**	***Home***	**Total**	
1	✓	✓	11.9	27.4	39.3	Berhasil
2	✓	×	12.0	-	-	Gagal *Home*
3	✓	✓	12.7	25.7	38.4	Berhasil
4	✓	×	12.5	-	-	Gagal *Home*
5	✓	✓	13.8	33.0	46.8	Berhasil

Berdasarkan Tabel 4.3 dan Tabel 4.4 dapat disimpulkan pada konfigurasi lapangan 1 robot dapat berhasil memadamkan api dan sampai ke *home* dengan waktu tempuh rata-rata 41.5 detik. Robot tidak menemukan *home* pada percobaan ke-2 dan ke-4 disebabkan oleh pengukuran jarak depan tidak sesuai dengan jarak yang ditetapkan pada program *mapping* sehingga robot tidak berpindah metoda berjalan saat melintasi *check point home*.

3) Pengujian Robot Pada Konfigurasi Lapangan 3

Pada konfigurasi ini *home* berada pada ruang 1 dan posisi api berada pada ruang 3. Posisi *home*, api, boneka anjing, dan penghalang dapat dilihat pada Gambar 4.3. Data pengamatan hasil percobaan konfigurasi 3 dapat dilihat pada Tabel 4.5 dan Tabel 4.6.



Gambar 4.3 Konfigurasi Lapangan 3

Tabel 4.5 Data Pengujian Parameter *Mapping* Area dengan Konfigurasi Lapangan 3

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	70	48	68	42	61	45	67	51	76	49
Belakang (cm)	35	75	39	82	32	60	40	64	37	63
Kiri (cm)	9	11	9	9	11	14	10	13	8	10
Kanan (cm)	14	14	16	12	14	13	13	12	15	14
Warna (ADC)	388	548	383	581	398	590	417	574	423	564

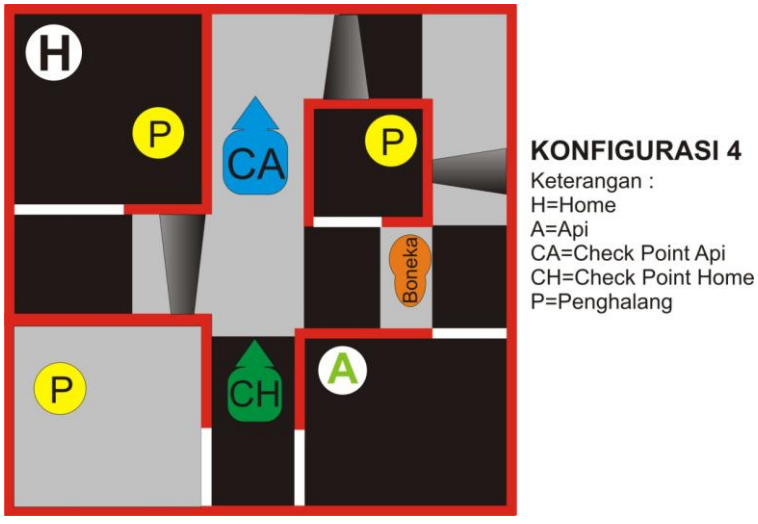
Tabel 4.6 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 3

No	*Check Point*		Waktu (detik)			Keterangan
	Api	*Home*	Api	*Home*	Total	
1	✓	✓	32.2	14.3	46.5	Berhasil
2	✓	✓	37.5	13.3	50.8	Berhasil
3	✓	✓	36.1	13.6	49.7	Berhasil
4	✓	✓	39.1	14.2	53.3	Berhasil
5	✓	✓	36.4	14.0	50.4	Berhasil

Berdasarkan Tabel 4.5 dan 4.6 dapat disimpulkan pada konfigurasi lapangan 3, robot dapat berhasil memadamkan api dan sampai ke *home* dengan waktu tempuh rata-rata 50.1 detik. Pada konfigurasi ini robot dapat menyelesaikan misi dengan baik .

4) Pengujian Robot Pada Konfigurasi Lapangan 4

Pada konfigurasi ini *home* berada pada ruang 3 dan posisi api berada pada ruang 1. Posisi *home*, api, boneka anjing, dan penghalang dapat dilihat pada Gambar 4.4. Data pengamatan hasil percobaan konfigurasi 4 dapat dilihat pada Tabel 4.7 dan Tabel 4.8.



Gambar 4.4 Konfigurasi Lapangan 4

Tabel 4.7 Data Pengujian Parameter *Mapping* Area dengan Konfigurasi Lapangan 4

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	34	101	41	97	29	91	32	94	35	104
Belakang (cm)	57	45	58	44	81	52	58	41	64	52
Kiri (cm)	10	11	11	9	9	13	10	12	8	11
Kanan (cm)	17	15	17	12	19	13	14	11	16	12
Warna (ADC)	482	509	488	523	486	535	497	522	487	541

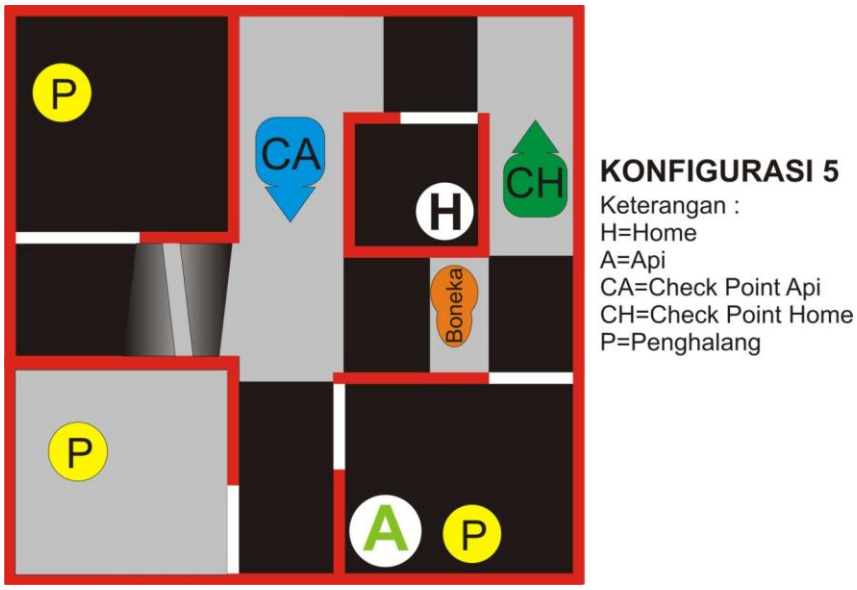
Tabel 4.8 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 4

No	***Check Point***		**Waktu (detik)**			**Keterangan**
	Api	***Home***	**Api**	***Home***	**Total**	
1	✓	✓	19.5	13.3	32.8	Berhasil
2	✓	✓	18.1	11.6	29.7	Berhasil
3	✓	✓	18.4	12.0	30.4	Berhasil
4	✓	✓	26.7	11.2	37.9	Berhasil
5	✓	✓	18.9	11.4	30.3	Berhasil

Berdasarkan Tabel 4.7 dan Tabel 4.8 dapat disimpulkan pada konfigurasi lapangan 4 robot dapat berhasil memadamkan api dan sampai ke *home* dengan waktu tempuh rata-rata 32.2 detik. Pada konfigurasi ini robot dapat menyelesaikan misi dengan baik .

5) Pengujian Robot Pada Konfigurasi Lapangan 5

Pada konfigurasi ini *home* berada pada ruang 4 dan posisi api berada pada ruang 1. Posisi *home*, api, boneka anjing, dan penghalang dapat dilihat pada Gambar 4.5. Data pengamatan hasil percobaan konfigurasi 5 dapat dilihat pada Tabel 4.9 dan Tabel 4.10.



Gambar 4.5 Konfigurasi Lapangan 5

Tabel 4.9 Data Pengujian Parameter *Mapping* Area dengan Konfigurasi Lapangan 5

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	86	34	90	38	63	47	77	44	82	43
Belakang (cm)	47	63	34	64	34	66	38	52	46	45
Kiri (cm)	8	12	12	11	13	9	11	12	12	10
Kanan (cm)	15	12	17	8	12	8	12	10	14	9
Warna (ADC)	380	432	392	443	402	435	395	467	379	432

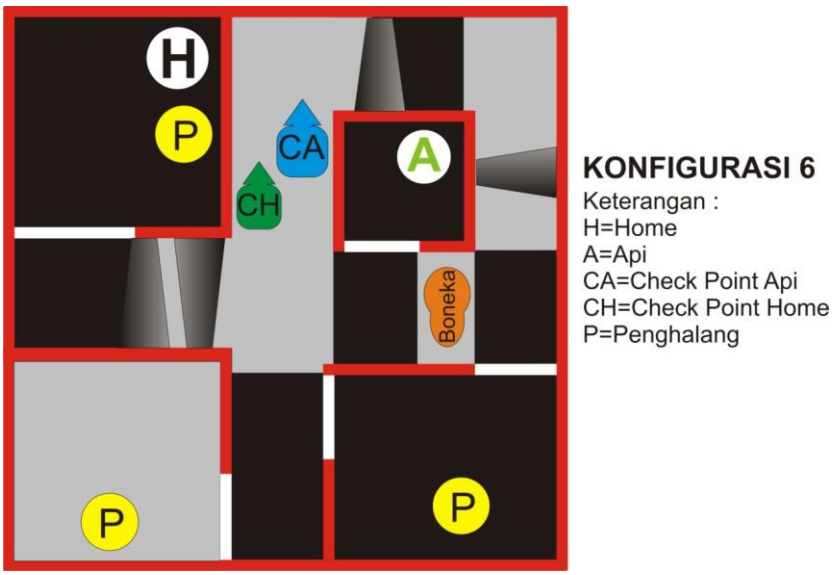
Tabel 4.10 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 5

No	***Check Point***		**Waktu (detik)**			**Keterangan**
	Api	***Home***	**Api**	***Home***	**Total**	
1	✓	✓	32.9	9.9	42.8	Berhasil
2	✓	✓	49.1	10.5	59.6	Berhasil
3	✓	✓	48.6	10.7	59.3	Berhasil
4	✓	✓	32.4	9.8	42.2	Berhasil
5	✓	✓	33.3	9.8	43.1	Berhasil

Berdasarkan Tabel 4.9 dan Tabel 4.10 dapat disimpulkan pada konfigurasi lapangan 5 robot dapat berhasil memadamkan api dan sampai ke *home* dengan waktu tempuh rata-rata 49.4 detik. Pada konfigurasi ini robot dapat menyelesaikan misi dengan baik .

6) Pengujian Robot Pada Konfigurasi Lapangan 6

Pada konfigurasi ini *home* berada pada ruang 2 dan posisi api berada pada ruang 4. Posisi *home*, api, boneka anjing, dan penghalang dapat dilihat pada Gambar 4.6. Data pengamatan hasil percobaan konfigurasi 6 dapat dilihat pada Tabel 4.11 dan Tabel 12.



Gambar 4.6 Konfigurasi Lapangan 6

Tabel 4.11 Data Pengujian Parameter *Mapping* Area dengan Konfigurasi Lapangan 6

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	36	44	44	43	-	-	47	62	46	51
Belakang (cm)	29	79	43	56	-	-	34	52	38	70
Kiri (cm)	9	10	9	9	-	-	8	12	9	14
Kanan (cm)	15	14	17	16	-	-	13	9	14	8
Warna (ADC)	388	422	379	430	-	-	428	453	427	446

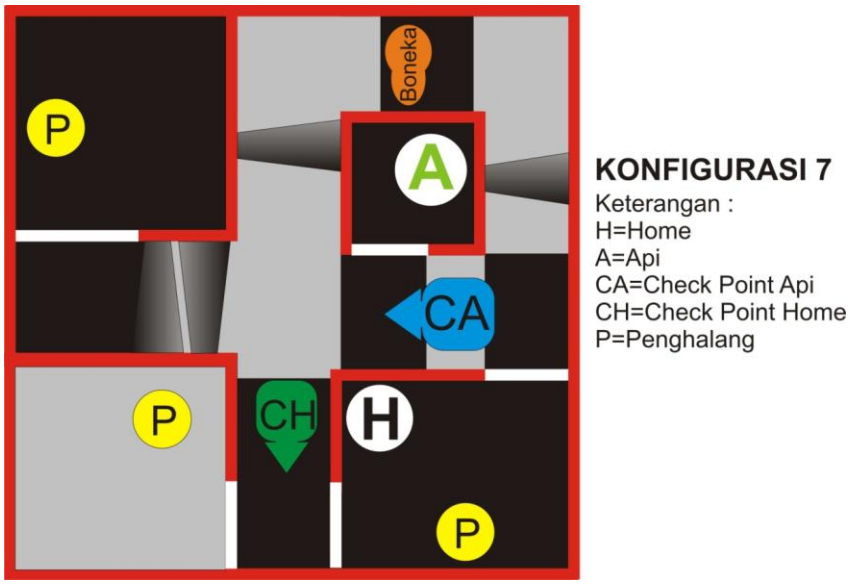
Tabel 4.12 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 6

No	*Check Point*		Waktu (detik)			Keterangan
	Api	*Home*	Api	*Home*	Total	
1	✓	×	24.1	-	-	Gagal *Home*
2	✓	✓	25.0	32.1	57.1	Berhasil
3	×	×	-	-	-	Gagal
4	✓	✓	23.4	32.0	55.4	Berhasil
5	✓	✓	23.2	31.9	55.1	Berhasil

Berdasarkan Tabel 4.11 dan Tabel 4.12 dapat disimpulkan pada konfigurasi lapangan 6 robot dapat berhasil memadamkan api dan sampai ke *home* dengan waktu tempuh rata-rata 55.8 detik. Pada percobaan ke-1 robot gagal menemukan *home* karena disebabkan oleh pengukuran jarak belakang tidak sesuai dengan jarak yang ditetapkan pada mapping area *check point home*. Kegagalan pada percobaan ke-3 karena robot terbalik pada saat melintasi rintangan yang berbentuk "polisi tidur" saat keluar dari ruang 3.

7) Pengujian Robot Pada Konfigurasi Lapangan 7

Pada konfigurasi ini *home* berada pada ruang 1 dan posisi api berada pada ruang 4. Posisi *home*, api, boneka anjing, dan penghalang dapat dilihat pada gambar 4.7. Data pengamatan hasil percobaan konfigurasi 7 dapat dilihat pada tabel 4.13 dan tabel 4.14.



Gambar 4.7 Konfigurasi Lapangan 7

Tabel 4.13 Data Pengujian Parameter *Mapping* Area dengan Konfigurasi Lapangan 7

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	73	34	66	45	79	48	81	52	78	49
Belakang (cm)	34	72	32	60	41	57	38	68	37	64
Kiri (cm)	11	14	9	15	12	11	14	9	9	14
Kanan (cm)	10	14	11	13	12	9	9	14	13	12
Warna (ADC)	387	541	392	589	382	583	454	599	442	572

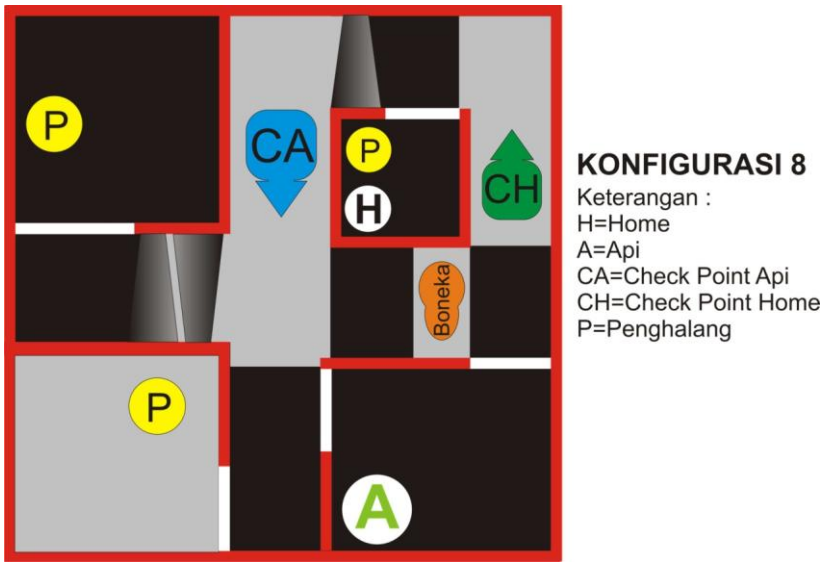
Tabel 4.14 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 7

No	***Check Point***		**Waktu (detik)**			**Keterangan**
	Api	***Home***	**Api**	***Home***	**Total**	
1	✓	✓	10.6	22.7	33.3	Berhasil
2	✓	✓	11.7	21.2	32.9	Berhasil
3	✓	✓	10.6	38.2	48.8	Berhasil
4	✓	✓	11.8	30.9	42.7	Berhasil
5	✓	✓	11.3	22.4	33.7	Berhasil

Berdasarkan Tabel 4.13 dan Tabel 4.14 dapat disimpulkan pada konfigurasi lapangan 7 robot dapat berhasil memadamkan api dan sampai ke *home* dengan waktu tempuh rata-rata 38.2 detik. Pada konfigurasi ini robot dapat menyelesaikan misi dengan baik .

8) Pengujian Robot Pada Konfigurasi Lapangan 8

Pada konfigurasi ini *home* berada pada ruang 4 dan posisi api berada pada ruang 1. Posisi *home*, api, boneka anjing, dan penghalang dapat dilihat pada Gambar 4.8. Data pengamatan hasil percobaan konfigurasi 8 dapat dilihat pada Tabel 4.15 dan 4.16.



Gambar 4.8 Konfigurasi Lapangan 8

Tabel 4.15 Data Pengujian Parameter *Mapping* Area dengan Konfigurasi Lapangan 8

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	79	39	-	-	88	43	-	-	89	37
Belakang (cm)	46	64	-	-	36	66	-	-	49	56
Kiri (cm)	13	9	-	-	9	12	-	-	10	9
Kanan (cm)	15	12	-	-	11	14	-	-	12	17
Warna (ADC)	388	432	-	-	411	455	-	-	413	443

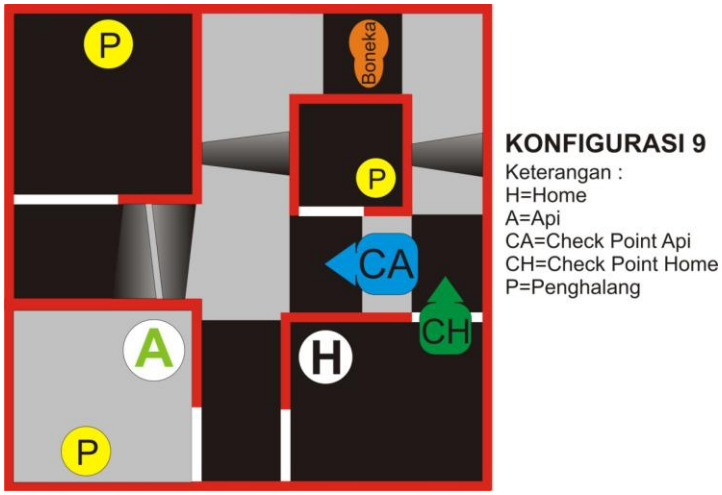
Tabel 4.16 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 8

No	*Check Point*		Waktu (detik)			Keterangan
	Api	*Home*	Api	*Home*	Total	
1	✓	✓	34.9	11.9	46.8	Berhasil
2	×	×	-	-	-	Gagal
3	✓	✓	47.6	10.3	57.9	Berhasil
4	✓	✓	39.4	12.8	52.2	Berhasil
5	×	×	-	-	-	Gagal

Berdasarkan tabel 4.15 dan tabel 4.16 dapat disimpulkan pada konfigurasi lapangan 8 robot dapat dapat berhasil memadamkan api dan sampai ke *home* dengan waktu tempuh rata-rata 52.3 detik. Kegagalan robot pada saat kembali *home* pada percobaan ke-2 dan 5 disebabkan oleh adanya furniture sebagai pengahalang yang terdapat di ruang-4 sehingga robot terhenti di dalam ruangan tersebut.

9) Pengujian Robot Pada Konfigurasi Lapangan 9

Pada konfigurasi ini *home* berada pada ruang 1 dan posisi api berada pada ruang 2. Posisi *home*, api, boneka anjing, dan penghalang dapat dilihat pada Gambar 4.9. Data pengamatan hasil percobaan konfigurasi 9 dapat dilihat pada Tabel 4.17 dan Tabel 4.18.



Gambar 4.9 Konfigurasi Lapangan 9

Tabel 4.17 Data Pengujian Parameter *Mapping* Area dengan Konfigurasi Lapangan 9

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	67	66	79	76	74	72	73	78	79	27
Belakang (cm)	33	59	42	77	40	74	32	64	41	69
Kiri (cm)	9	46	7	9	12	13	11	36	11	12
Kanan (cm)	14	14	16	11	12	12	15	11	10	15
Warna (ADC)	382	511	411	542	432	534	404	577	445	523

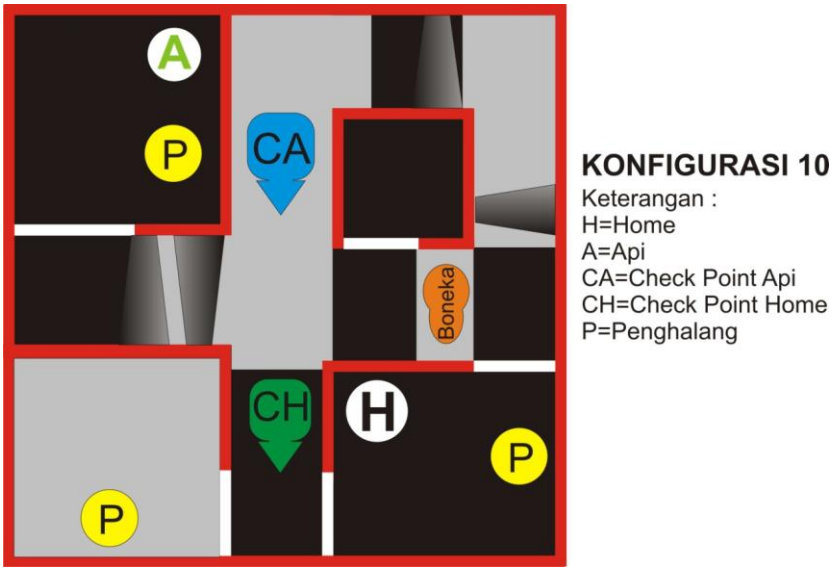
Tabel 4.18 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 9

No	***Check Point***		**Waktu (detik)**			**Keterangan**
	Api	***Home***	**Api**	***Home***	**Total**	
1	✓	×	38.1	-	-	Gagal *Home*
2	✓	✓	38.4	12.8	51.2	Berhasil
3	✓	✓	35.9	11.7	47.6	Berhasil
4	✓	×	36.2	-	-	Gagal *Home*
5	✓	×	37.0	-	-	Gagal *Home*

Berdasarkan Tabel 4.17 dan 4.18 dapat disimpulkan pada konfigurasi lapangan 9 robot dapat berhasil memadamkan api dan sampai ke *home* dengan waktu tempuh rata-rata 49.4 detik. Kegagalan robot pada saat kembali *home* pada percobaan ke-1,4, dan 5 disebabkan oleh pengukuran jarak oleh sensor depan dan belakang tidak sesuai dengan jarak pada pemetaan area sehingga robot tidak berpindah metoda berjalan saat melintasi *check point home*.

10) Pengujian Robot Pada Konfigurasi Lapangan 10

Pada konfigurasi ini *home* berada pada ruang 1 dan posisi api berada pada ruang 3. Posisi *home*, api, boneka anjing, dan penghalang dapat dilihat pada Gambar 4.10. Data pengamatan hasil percobaan konfigurasi 10 dapat dilihat pada Tabel 4.19 dan 4.20.



Gambar 4.10 Konfigurasi Lapangan 10

Tabel 4.19 Data Pengujian Parameter *Mapping* Area dengan Konfigurasi Lapangan 10

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	84	48	69	43	103	-	72	41	74	36
Belakang (cm)	41	74	39	60	34	-	36	66	43	72
Kiri (cm)	10	11	12	9	13	-	11	10	12	10
Kanan (cm)	11	14	14	9	12	-	12	10	14	9
Warna (ADC)	388	541	382	588	401	-	385	577	379	562

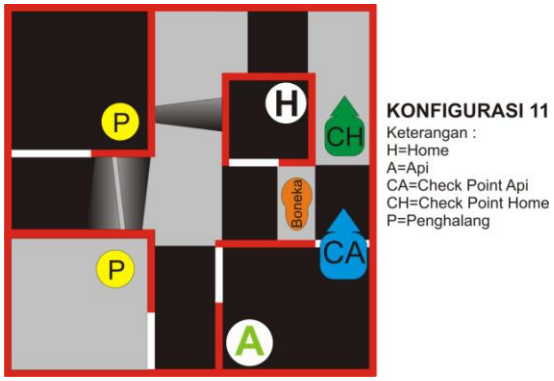
Tabel 4.20 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 10

No	*Check Point*		Waktu (detik)			Keterangan
	Api	*Home*	Api	*Home*	Total	
1	✓	✓	20.3	11.5	31.8	Berhasil
2	✓	✓	21.1	11.1	32.2	Berhasil
3	×	×	-	-	-	Gagal
4	✓	✓	19.3	11.2	30.5	Berhasil
5	✓	✓	20.8	11.1	31.9	Berhasil

Berdasarkan Tabel 4.19 dan Tabel 4.20 dapat disimpulkan pada konfigurasi lapangan 10, robot dapat berhasil memadamkan api dan sampai ke *home* dengan waktu tempuh rata-rata 31.6 detik. Kegagalan pada percobaan ke-3 disebabkan robot tidak berpindah metoda berjalan pada saat melintasi check point api yang telah ditentukan, hal ini disebabkan hasil pembacaan sensor jarak yang tidak sesuai dengan jarak yang ditetapkan pada pemetaan area, karena bentuk daerah pantul sensor yang tidak datar (banyak celah dan ruang).

11) Pengujian Robot Pada Konfigurasi Lapangan 11

Pada konfigurasi ini *home* berada pada ruang 4 dan posisi api berada pada ruang 1. Posisi *home*, api, boneka anjing, dan penghalang dapat dilihat pada Gambar 4.11. Data pengamatan hasil percobaan konfigurasi 11 dapat dilihat pada Tabel 4.21 dan Tabel 4.22.



Gambar 4.11 Konfigurasi Lapangan 11

Tabel 4.21 Data Pengujian Parameter *Mapping* Area dengan Konfigurasi Lapangan 11

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	75	37	69	-	82	33	76	50	85	48
Belakang (cm)	66	63	67	-	54	48	59	54	61	52
Kiri (cm)	14	12	10	-	9	12	13	12	12	13
Kanan (cm)	15	12	11	-	12	10	11	9	15	8
Warna (ADC)	556	430	532	-	543	460	566	443	512	456

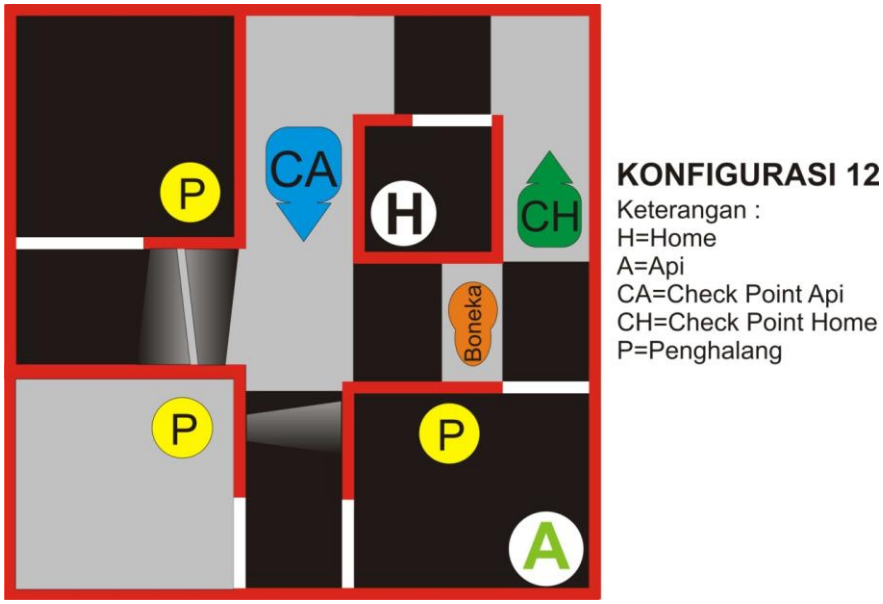
Tabel 4.22 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 11

No	***Check Point***		**Waktu (detik)**			**Keterangan**
	Api	***Home***	**Api**	***Home***	**Total**	
1	✓	✓	50.4	28.3	78.7	Berhasil
2	×	×	-	-	-	Gagal
3	✓	✓	49.9	25.6	75.5	Berhasil
4	✓	✓	48.4	26	74.4	Berhasil
5	✓	✓	48.7	29.5	78.2	Berhasil

Berdasarkan Tabel 4.21 dan Tabel 4.22 dapat disimpulkan pada konfigurasi lapangan 11 robot dapat berhasil memadamkan api dan sampai kembali ke *home* dengan waktu tempuh rata-rata 76.7 detik. Kegagalan pada percobaan ke-2 diakibatkan robot langsung berpindah metoda berjalan sebelum sampai pada *check point* api, hal ini terjadi karena pembacaan nilai paramter yang sama dengan ditetapkan tetapi tidak pada area *check point* yang ditentukan.

12) Pengujian Robot Pada Konfigurasi Lapangan 12

Pada konfigurasi ini *home* berada pada ruang 4 dan posisi api berada pada ruang 1. Posisi *home*, api, boneka anjing, dan penghalang dapat dilihat pada Gambar 4.12. Data pengamatan hasil percobaan konfigurasi 12 dapat dilihat pada Tabel 4.23 dan Tabel 4.24.



Gambar 4.12 Konfigurasi Lapangan 12

Tabel 4.23 Data Pengujian Parameter *Mapping* Area dengan Konfigurasi Lapangan 12

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	75	37	69	-	82	33	76	50	85	48
Belakang (cm)	66	63	67	-	54	48	59	54	61	52
Kiri (cm)	14	12	10	-	9	12	13	12	12	13
Kanan (cm)	15	12	11	-	12	10	11	9	15	8
Warna (ADC)	556	430	532	-	543	460	566	443	512	456

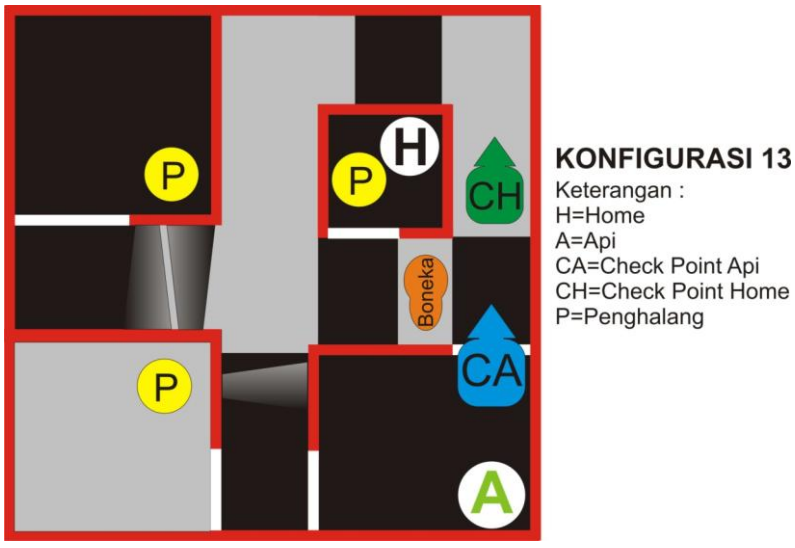
Tabel 4.24 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 12

No	*Check Point*		Waktu (detik)			Keterangan
	Api	*Home*	Api	*Home*	Total	
1	✓	✓	44.3	9.9	54.2	Berhasil
2	✓	✓	45.1	11.0	56.1	Berhasil
3	×	×	-	-	-	Gagal
4	✓	✓	44.2	9.7	51.9	Berhasil
5	✓	✓	46.7	9.1	55.8	Berhasil

Berdasarkan Tabel 4.23 dan Tabel 4.24 dapat disimpulkan pada konfigurasi lapangan 12 robot dapat berhasil memadamkan api dan sampai ke *home* dengan waktu tempuh rata-rata 54.5 detik. Kegagalan pada percobaan ke-3 diakibatkan robot terjungkir pada saat melintasi rintangan yang berupa polisi tidur sebelum memasuki ruang 3.

13) Pengujian Robot Pada Konfigurasi Lapangan 13

Pada konfigurasi ini *home* berada pada ruang 4 dan posisi api berada pada ruang 1. Posisi *home*, api, boneka anjing, dan penghalang dapat dilihat pada Gambar 4.13. Data pengamatan hasil percobaan konfigurasi 13 dapat dilihat pada Tabel 4.25 dan Tabel 4.26.



Gambar 4.13 Konfigurasi Lapangan 13

Tabel 4.25 Data Pengujian Parameter *Mapping* Area dengan Konfigurasi Lapangan 13

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	73	37	69	34	69	34	76	-	74	41
Belakang (cm)	42	34	36	64	33	64	34	-	38	53
Kiri (cm)	10	12	13	12	12	11	11	-	13	11
Kanan (cm)	10	13	11	9	11	14	12	-	9	12
Warna (ADC)	386	432	375	452	387	443	400	-	379	432

Tabel 4.26 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 13

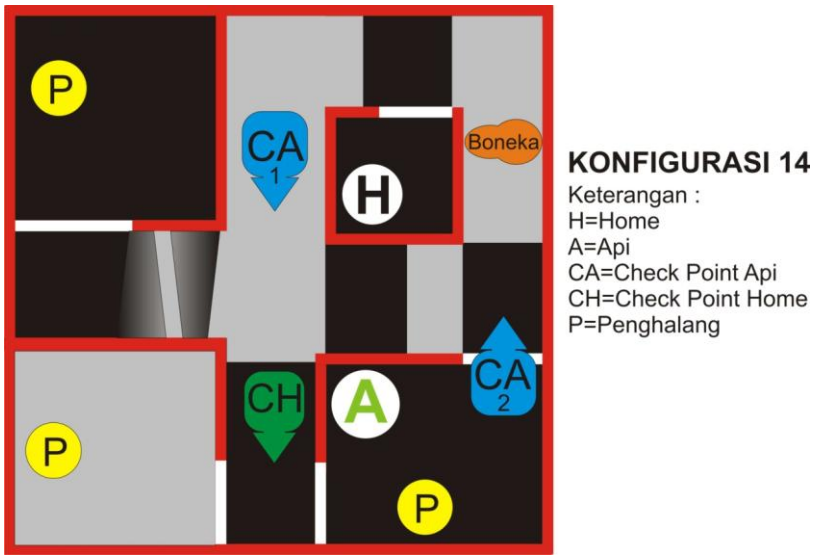
No	***Check Point***		**Waktu (detik)**			**Keterangan**
	Api	***Home***	**Api**	***Home***	**Total**	
1	✓	✓	49.3	28.0	77.3	Berhasil
2	✓	✓	52.8	21.0	73.8	Berhasil
3	✓	✓	47.0	25.3	72.3	Berhasil
4	×	×	-	-	-	Gagal
5	✓	✓	47.3	21.7	69.0	Berhasil

Berdasarkan Tabel 4.25 dan Tabel 4.26 dapat disimpulkan pada konfigurasi lapangan 13 robot dapat berhasil memadamkan api dan sampai ke *home* dengan waktu tempuh rata-rata 73.1 detik. Kegagalan pada percobaan ke-4 disebabkan robot terlebih dahulu berpindah metoda berjalan sebelum sampai di *check point* api yang telah ditentukan, hal ini terjadi karena pembacaan nilai paramter yang sama dengan ditetapkan tetapi tidak pada area *check point* yang ditentukan.

14) Pengujian Robot Pada Konfigurasi Lapangan 14

Pada konfigurasi ini *home* berada pada ruang 4 dan posisi api berada pada ruang 1. Posisi *home*, api, boneka anjing, dan penghalang dapat

dilihat pada Gambar 4.14. Data pengamatan hasil percobaan konfigurasi 14 dapat dilihat pada Tabel 4.27, Tabel 4.28, dan Tabel 4.29.



Gambar 4.14 Konfigurasi Lapangan 14

Tabel 4.27 Data Pengujian Parameter *Mapping* Area *Check Point* Api 1 dengan Konfigurasi Lapangan 14

Percobaan ke -	1	2	3	4	5
	Api	Api	Api	Api	Api
Depan (cm)	77	74	88	81	82
Belakang (cm)	44	48	52	55	32
Kiri (cm)	7	7	10	9	12
Kanan (cm)	14	13	14	12	11
Warna (ADC)	384	392	430	421	420

Tabel 4.28 Data Pengujian Parameter *Mapping* Area *Check Point* Api 2 dan *Check Point Home* dengan Konfigurasi Lapangan 14

Percobaan ke -	1		2		3		4		5	
	Api	Home	Api	Home	Api	Home	Api	Home	Api	Home
Depan (cm)	98	46	104	-	78	44	86	40	84	-
Belakang (cm)	64	72	49	-	53	60	69	66	46	-
Kiri (cm)	12	13	11	-	11	10	18	9	34	-
Kanan (cm)	14	12	12	-	11	12	12	14	10	-
Warna (ADC)	489	531	512	-	532	578	560	567	555	-

Tabel 4.29 Pengujian Robot Pada Arena KRPAI 2013 dengan Konfigurasi Lapangan 14

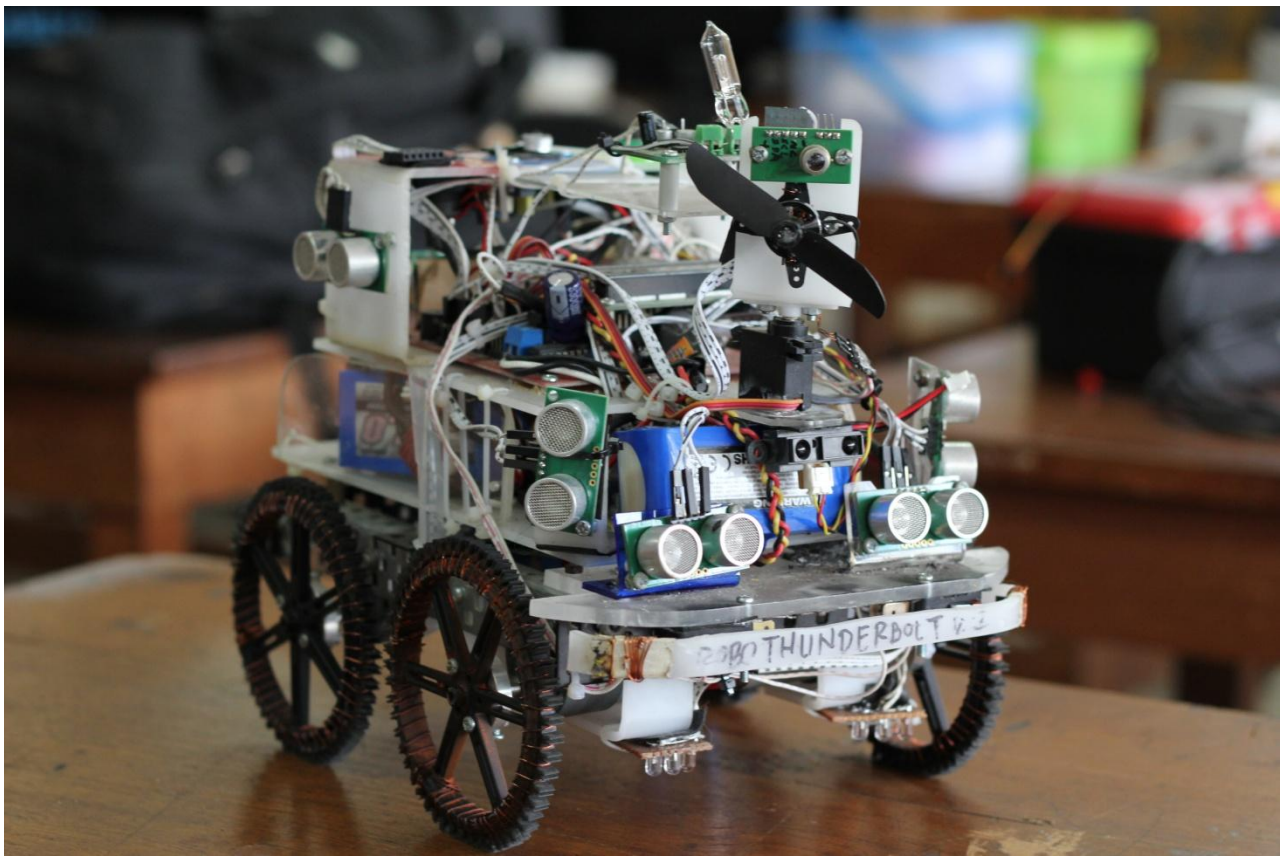
No	***Check Point***		**Waktu (detik)**			**Keterangan**
	Api	***Home***	**Api**	***Home***	**Total**	
1	✓	✓	36.4	15.5	51.9	Berhasil
2	×	×	-	-	-	Gagal
3	✓	✓	37.2	14.4	51.6	Berhasil
4	✓	✓	35.8	16.7	52.5	Berhasil
5	×	×	-	-	-	Gagal

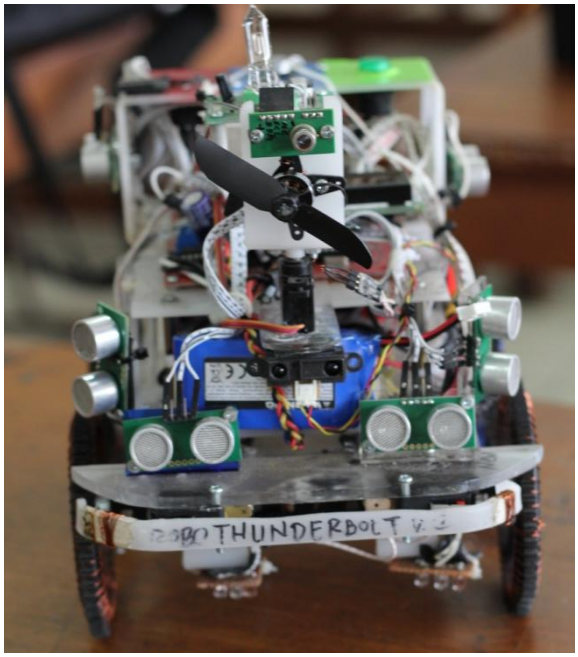
Berdasarkan tabel 4.27, tabel 4.28, dan tabel 4.29 dapat disimpulkan pada konfigurasi lapangan 14 robot dapat berhasil memadamkan api dan sampai ke *home* dengan waktu tempuh rata-rata 52 detik. Kegagalan pada percobaan ke-2 dan 5 disebabkan karena pengukuran jarak pada check point ke-2 tidak sesuai dengan parameter jarak yang ditetapkan pada pemetaan area.

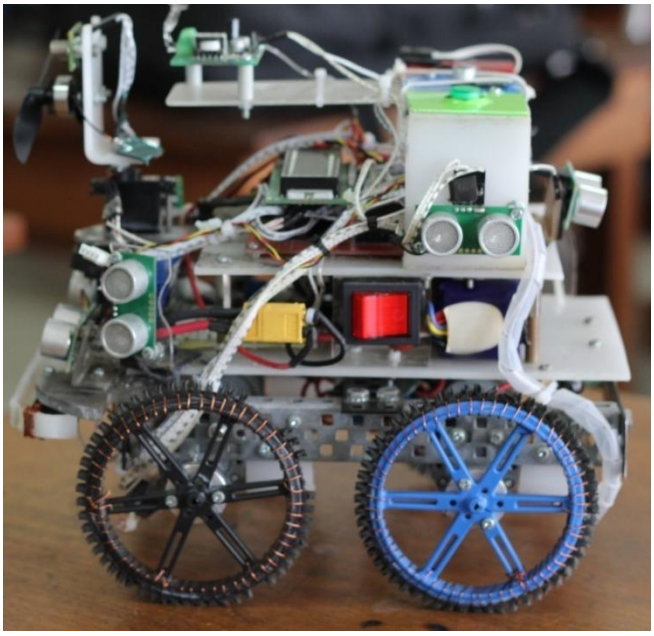
DAFTAR PUSTAKA

1. Andrianto, H. 2008. *Buku Panduan : Pelatihan Mikrokontroler AVR ATmega16*.
2. Darmawan, Aan dan Theresia, Novie. 2008. *Diktat Pengantar Sistem Cerdas*.
3. Datasheet "*8-bit AVR Microcontroller with 128Kbytes In-System Programable Flash" ATMEL*.
4. Datasheet "*Flame Sensor UV TRON*", HAMAMATSU.
5. Ensikopledia Wikipedia. Kecerdasan Buatan. (online), (http://id.wikipedia.org/wiki/Kecerdasan_buatan, diakses 2 Mei 2013)
6. Ensikopledia Wikipedia. *Maze Solving Algorithm*. (online), (http://id.wikipedia.org/wiki/Maze_solving_algorithm,diakses 2 Mei 2013)
7. Fahmizal, Rusdianto Effendi, Eka Iskandar. 2011. *Implementasi Sistem Navigasi Behavior Based dan Kontroler PID pada Manuver Robot Maze*. Surabaya : Institut Teknologi Sepuluh November.
8. M. Iqbal Nugraha, Akhmad Hendriawan, Ressa Akbar. 2010. *Penerapan Algoritma Maze Mapping Untuk Menyeleseaikan Maze Pada Line Tracer*. Surabaya : Politeknik Elektronika Negri Surabaya.
9. SRF-05 *Technical Document*. (online), (http://www.robot-electronics.co.uk/htm/srf05tech.htm, diakses 8 Juli 2013)
10. Vannoy Richard T. 2009. *Teaching a Robot to Solve a Line Maze*.

LAMPIRAN A
FOTO ROBOT


ROBO THUNDERBOLT V.2


ROBO THUNDERBOLT V.2



LAMPIRAN B
PROGRAM PADA PENGONTROL MIKRO ATMEGA 128

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.0 Professional
Automatic Program Generator
© Copyright 1998-2010 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 5/3/2013
Author  :
Company :
Comments:


Chip type               : ATmega128
Program type            : Application
AVR Core Clock frequency: 11.059200 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 1024
*****************************************************/

#include <mega128.h>
#include <stdio.h>
#include <delay.h>
#include <math.h>
#include <stdlib.h>

unsigned char text[20];
unsigned char varServo1, varKipas;
float
sinyalControl,proporsional,integral,derivative,processVariable,error,setPointSensorKiri,setPointPWM,leftPWM,rig
htPWM,rate,rateInt,lastError;
float ki,kp,kd,Ts;
int detect, padam, counterA, counterJurus;
bit jurus, varA,varC,counterC,island;
int ADCdepanKiri,ADCbelakangKiri, ADCbelakangKanan, ADCboneka,sound;
int counterB,varB,interlock,limit,ruangTerakhir,balikRuangTerakhir,ADCkanan;
int a,start,homeIsland1,island1,suhuReferensi;
int homeIsland2, island2,counterApi,B;

int data,i;
unsigned char Msg1[20],Msg2[20],reg,revision,ambient,pixel[8];
// I2C Bus functions
#asm
  .equ __i2c_port=0x12 ;PORTD
  .equ __sda_bit=1
  .equ __scl_bit=0
#endasm
#include <i2c.h>
// Alphanumeric LCD Module functions
#include <alcd.h>

#ifndef RXB8
#define RXB8 1
```

```
#endif

#ifndef TXB8
#define TXB8 0
#endif

#ifndef UPE
#define UPE 2
#endif

#ifndef DOR
#define DOR 3
#endif

#ifndef FE
#define FE 4
#endif

#ifndef UDRE
#define UDRE 5
#endif

#ifndef RXC
#define RXC 7
#endif

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<DOR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART1 Receiver buffer
#define RX_BUFFER_SIZE1 8
char rx_buffer1[RX_BUFFER_SIZE1];

#if RX_BUFFER_SIZE1 <= 256
unsigned char rx_wr_index1,rx_rd_index1,rx_counter1;
#else
unsigned int rx_wr_index1,rx_rd_index1,rx_counter1;
#endif

// This flag is set on USART1 Receiver buffer overflow
bit rx_buffer_overflow1;
int pointer=0,i=0,flag=0;char kata[20],ambil[10] ;
int ir;
// USART1 Receiver interrupt service routine
interrupt [USART1_RXC] void usart1_rx_isr(void)
{
char status,data;
status=UCSR1A;
data=UDR1;
kata[pointer]==data;
if(pointer >=4 && pointer <=7){kata[pointer-4]=data;}

pointer++;
```

```
if(data=='\r'){pointer=0;for(i=0;i<8;i++){kata[i]=0x00;}}
ir=atoi(kata);


if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
   rx_buffer1[rx_wr_index1++]=data;
#if RX_BUFFER_SIZE1 == 256
   // special case for receiver buffer size=256
   if (++rx_counter1 == 0)
      {
#else
   if (rx_wr_index1 == RX_BUFFER_SIZE1) rx_wr_index1=0;
   if (++rx_counter1 == RX_BUFFER_SIZE1)
      {
      rx_counter1=0;
#endif
      rx_buffer_overflow1=1;
      }
   }
}

// Get a character from the USART1 Receiver buffer
#pragma used+
char getchar1(void)
{
char data;
while (rx_counter1==0);
data=rx_buffer1[rx_rd_index1++];
#if RX_BUFFER_SIZE1 != 256
if (rx_rd_index1 == RX_BUFFER_SIZE1) rx_rd_index1=0;
#endif
#asm("cli")
--rx_counter1;
#asm("sei")
return data;
}
#pragma used-
// Write a character to the USART1 Transmitter
#pragma used+
void putchar1(char c)
{
while ((UCSR1A & DATA_REGISTER_EMPTY)==0);
UDR1=c;
}
#pragma used-


#define ADC_VREF_TYPE 0x00
// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
```

```
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}


int i;

void maju(char speedA, char speedB)
{
PORTB.0=0;
PORTB.1=1;
PORTB.3=0;
PORTB.4=1;
OCR1A=speedA;    //kiri
OCR1B=speedB;    //kanan
}


void mundur(char speedA, char speedB)
{
PORTB.0=1;
PORTB.1=0;
PORTB.3=1;
PORTB.4=0;
OCR1A=speedA;    //kiri
OCR1B=speedB;    //kanan
}

void stop(char speedA, char speedB)
{
PORTB.0=1;
PORTB.1=1;
PORTB.3=1;
PORTB.4=1;
OCR1A=speedA;    //kiri
OCR1B=speedB;    //kanan
}

void kiri(char speedA, char speedB)
{
PORTB.0=1;
PORTB.1=1;      //motor kiri
PORTB.3=0;
PORTB.4=1;      //motor kanan
OCR1A=speedA;    //kiri
OCR1B=speedB;    //kanan
}

void kanan(char speedA, char speedB)
{
PORTB.0=0;
PORTB.1=1;
PORTB.3=1;
PORTB.4=1;
OCR1A=speedA;    //kiri
```

```
OCR1B=speedB;    //kanan
}

void bantingKiri(char speedA, char speedB)
{
PORTB.0=1;
PORTB.1=0;
PORTB.3=0;
PORTB.4=1;
OCR1A=speedA;    //kiri
OCR1B=speedB;    //kanan
}

void bantingKanan(char speedA, char speedB)
{
PORTB.0=0;
PORTB.1=1;
PORTB.3=1;
PORTB.4=0;
OCR1A=speedA;    //kiri
OCR1B=speedB;    //kanan
}

unsigned int getSrf(unsigned char i)
{
   int jarak = 0, bacaJarak;
   switch(i)
   {
         case 0 :      //sensor kiri depan
               {
                     PORTA.1 = 0;
                     DDRA.1 = 1;
                     PORTA.1 = 1;
                     delay_us(15);
                     PORTA.1 = 0;
                     DDRA.1 = 0;
                     delay_us(750);
                     while(PINA.1 == 0){delay_us(1);};
                     while(PINA.1 == 1)
                     {
                           jarak++;
                           delay_us(1);
                           if(jarak>25000)   break;
                     }
                     delay_us(5);
               }
               break;
         case 1 :         //sensor depan kiri
               {
                     PORTA.2 = 0;
                     DDRA.2 = 1;
                     PORTA.2 = 1;
                     delay_us(15);
                     PORTA.2 = 0;
                     DDRA.2 = 0;
                     delay_us(750);
```

```
            while(PINA.2 == 0){delay_us(1);};
            while(PINA.2 == 1)
            {
                jarak++;
                delay_us(1);
                if(jarak>25000)  break;
            }
            delay_us(5);
        }
        break;
case 2 :        //sensor depan kanan
        {
            PORTA.3 = 0;
            DDRA.3 = 1;
            PORTA.3 = 1;
            delay_us(15);
            PORTA.3 = 0;
            DDRA.3 = 0;
            delay_us(750);
            while(PINA.3 == 0){delay_us(1);};
            while(PINA.3 == 1)
            {
                jarak++;
                delay_us(1);
                if(jarak>25000)  break;
            }
            delay_us(5);
        }
        break;
case 3 :        //sensor kanan depan
        {
            PORTA.0 = 0;
            DDRA.0 = 1;
            PORTA.0 = 1;
            delay_us(15);
            PORTA.0 = 0;
            DDRA.0 = 0;
            delay_us(750);
            while(PINA.0 == 0){delay_us(1);};
            while(PINA.0 == 1)
            {
                jarak++;
                delay_us(1);
                if(jarak>25000)  break;
            }
            delay_us(5);
        }
        break;
    case 4 :        //sensor kiri belakang
          {
              PORTA.6 = 0;
              DDRA.6 = 1;
              PORTA.6 = 1;
              delay_us(15);
              PORTA.6 = 0;
              DDRA.6 = 0;
```

```
                delay_us(750);
                while(PINA.6 == 0){delay_us(1);};
                while(PINA.6 == 1)
                {
                    jarak++;
                    delay_us(1);
                    if(jarak>25000)  break;
                }
                delay_us(5);
            }
            break;
        case 5 :
            {
                PORTA.4 = 0;
                DDRA.4 = 1;
                PORTA.4 = 1;
                delay_us(15);
                PORTA.4 = 0;
                DDRA.4 = 0;
                delay_us(750);
                while(PINA.4 == 0){delay_us(1);};
                while(PINA.4 == 1)
                {
                    jarak++;
                    delay_us(1);
                    if(jarak>25000)  break;
                }
                delay_us(5);
            }
            break;
        case 6 :
            {
                PORTA.5 = 0;
                DDRA.5 = 1;
                PORTA.5 = 1;
                delay_us(15);
                PORTA.5 = 0;
                DDRA.5 = 0;
                delay_us(750);
                while(PINA.5 == 0){delay_us(1);};
                while(PINA.5 == 1)
                {
                    jarak++;
                    delay_us(1);
                    if(jarak>25000)  break;
                }
                delay_us(5);
            }
            break;
//          case 7 :
//              {
//                  PORTA.7 = 0;
//                  DDRA.7 = 1;
//                  PORTA.7 = 1;
//                  delay_us(15);
//                  PORTA.7 = 0;
```

```
//                          DDRA.7 = 0;
//                          delay_us(750);
//                          while(PINA.7 == 0){delay_us(1);};
//                          while(PINA.7 == 1)
//                          {
//                                jarak++;
//                                delay_us(1);
//                                if(jarak>25000)  break;
//                          }
//                          delay_us(5);
//                    }
//                    break;
          default :
                {
                     return 0;
                }
   }
   bacaJarak = jarak/29.034;
   return bacaJarak;
}

void servo(unsigned int sdt)
{
int k; int s;
for (k=0;k<20;k++)
   {
   PORTD.5=1;
   delay_us(500);

   for(s=0;s<sdt;s++)
   {delay_us(11);}

   PORTD.5=0;
   delay_us(17500);

   for(s=0;s<(180-sdt);s++)
   {delay_us(11);}
   }

}

int coba=150; int flag2=2;
char kata[20];

void servoTPA()
{

    if(flag2==1){coba=coba+30;}
    if(coba>120){flag2=2;}
    if(flag2==2){coba=coba-30;}
    if(coba<20){flag2=1;}
    servo(coba);

}

void kipasBaru()
```

```
{

   for   ( varKipas=0;varKipas<=50;varKipas++ ){
        PORTD.6=1;
        delay_us(1500);
        PORTD.6=0;
        delay_us(18500);
        }

     for( varKipas=0;varKipas<=100;varKipas++ ){
        PORTD.6=1;
        delay_us(2000);
        PORTD.6=0;
        delay_us(18000);
        }
    if(flag2==1){coba=coba+30;}
    if(coba>120){flag2=2;}
    if(flag2==2){coba=coba-30;}
      if(coba<20){flag2=1;}
   servo(coba);




   sprintf(kata,"%d",coba);lcd_gotoxy(0,1);lcd_puts(kata);

}

void kipasTPA()
{

   for   ( varKipas=0;varKipas<=50;varKipas++ ){
        PORTD.6=1;
        delay_us(1500);
        PORTD.6=0;
        delay_us(18500);
        }

     for( varKipas=0;varKipas<=100;varKipas++ ){
        PORTD.6=1;
        delay_us(2000);
        PORTD.6=0;
        delay_us(18000);
        }
}

void kipas(void){

   for( varServo1=0;varServo1<=10;varServo1++ ){
        PORTD.5=1;
        delay_us(1500);
        PORTD.5=0;
        delay_us(18500);
   }
   for( varKipas=0;varKipas<=50;varKipas++ ){
        PORTD.6=1;
```

```
    delay_us(1500);
    PORTD.6=0;
    delay_us(18500);
    }
for( varKipas=0;varKipas<=100;varKipas++ ){
    PORTD.6=1;
    delay_us(2000);
    PORTD.6=0;
    delay_us(18000);
    }
for( varServo1=0;varServo1<=10;varServo1++ ){
    PORTD.5=1;
    delay_us(1200);
    PORTD.5=0;
    delay_us(18800);
}
for( varKipas=0;varKipas<=50;varKipas++ ){
    PORTD.6=1;
    delay_us(1500);
    PORTD.6=0;
    delay_us(18500);
    }
for( varKipas=0;varKipas<=100;varKipas++ ){
    PORTD.6=1;
    delay_us(2000);
    PORTD.6=0;
    delay_us(18000);
    }
for( varServo1=0;varServo1<=10;varServo1++ ){
    PORTD.5=1;
    delay_us(1500);
    PORTD.5=0;
    delay_us(18500);
}
for( varKipas=0;varKipas<=50;varKipas++ ){
    PORTD.6=1;
    delay_us(1500);
    PORTD.6=0;
    delay_us(18500);
    }
for( varKipas=0;varKipas<=100;varKipas++ ){
    PORTD.6=1;
    delay_us(2000);
    PORTD.6=0;
    delay_us(18000);
    }
for( varServo1=0;varServo1<=10;varServo1++ ){
    PORTD.5=1;
    delay_us(1800);
    PORTD.5=0;
    delay_us(18200);
}
for( varKipas=0;varKipas<=50;varKipas++ ){
    PORTD.6=1;
    delay_us(1500);
    PORTD.6=0;
```

```
        delay_us(18500);
        }
    for( varKipas=0;varKipas<=100;varKipas++ ){
        PORTD.6=1;
        delay_us(2000);
        PORTD.6=0;
        delay_us(18000);
        }
    for( varServo1=0;varServo1<=10;varServo1++ ){
        PORTD.5=1;
        delay_us(1500);
        PORTD.5=0;
        delay_us(18500);
    }

}

void TPA_read()
{
delay_ms(40);
i2c_start();
i2c_write(0xD0);
i2c_write(reg);
i2c_start();
i2c_write(0xD1);
data=i2c_read(0);
i2c_stop();

}


// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x80;
DDRB=0x7F;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;
```

```
// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0xFF;

// Port E initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTE=0xFF;
DDRE=0x00;

// Port F initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTF=0x00;
DDRF=0x00;

// Port G initialization
// Func4=In Func3=In Func2=In Func1=In Func0=In
// State4=T State3=T State2=T State1=T State0=T
PORTG=0x00;
DDRG=0x00;


// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
ASSR=0x00;
TCCR0=0x4F;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 10.800 kHz
// Mode: Fast PWM top=ICR1
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// OC1C output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR1A=0xA1;
TCCR1B=0x01;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
```

```
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
TCCR2=0x4D;
TCNT2=0x00;
OCR2=0x00;

// Timer/Counter 3 initialization
// Clock source: System Clock
// Clock value: Timer3 Stopped
// Mode: Normal top=0xFFFF
// OC3A output: Discon.
// OC3B output: Discon.
// OC3C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer3 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// INT3: Off
// INT4: Off
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x00;
EIMSK=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
```

```
TIMSK=0x00;

ETIMSK=0x00;

// USART0 initialization
// USART0 disabled
UCSR0B=0x00;

// USART1 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART1 Receiver: On
// USART1 Transmitter: On
// USART1 Mode: Asynchronous
// USART1 Baud Rate: 9600
UCSR1A=0x00;
UCSR1B=0x98;
UCSR1C=0x06;
UBRR1H=0x00;
UBRR1L=0x47;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 172.800 kHz
// ADC Voltage Reference: AREF pin

ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x87;


// SPI initialization
// SPI disabled
SPCR=0x00;

// I2C Bus initialization
i2c_init();

// TWI initialization
// TWI disabled
TWCR=0x00;

// Alphanumeric LCD initialization
// Connections specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTC Bit 0
// RD - PORTC Bit 1
// EN - PORTC Bit 2
// D4 - PORTC Bit 4
// D5 - PORTC Bit 5
// D6 - PORTC Bit 6
// D7 - PORTC Bit 7
// Characters/line: 16
```

```
lcd_init(16);
putchar1('a');
// Global enable interrupts
#asm("sei")
//delay_ms(2000);
//putchar1('\r');
//putchar1('\r');
//putchar1('L');

counterA=0;
counterB=0;
counterC=0;
detect=0;
padam=0;
jurus=0;
counterJurus=0;
varA=0;
varB=0;
varC=0;
interlock=0;
limit=0;
ruangTerakhir=0;
balikRuangTerakhir=0;
island=0;
a=0;
start=0;
homeIsland1=0;
homeIsland2=0;
island1=0;
island2=0;
suhuReferensi=65;
counterApi=0;
B=0;

while (1)
{

  //  ======================================Program
Utama================================================================
ADCdepanKiri=read_adc(0);
ADCbelakangKiri=read_adc(1);
ADCbelakangKanan=read_adc(2);
ADCboneka=read_adc(3);
//ADCkanan=read_adc(5);


if(PINB.7==0 || start==1) // tombol start
{
   if (PINF.4==1)           //if ga ada api
   {
      PORTD.4=0; //buzzer
      if ((padam==0 && island1==0)||(padam==1 && homeIsland1==1 ) || (padam==1 && island1==0 && island
== 0))
      {
         if(getSrf(1) > 8 && limit==0 && ADCboneka < 340) //BONEKA--------
              {
```

```
kp=4;         //4                    //awal PID
kd=2;         //1
//ki=0.1;
setPointPWM = 150;
processVariable = getSrf(0);                  //================== proporsional program
setPointSensorKiri = 7;
error = setPointSensorKiri - processVariable;
proporsional = kp * error;                    //================== proporsional end
rate = error - lastError;                   //================== derivative program
//rateInt = error + lastError;
Ts=1;                                 //==============>>>> Ts = time sampling
//integral = Ts * (ki * rateInt);
derivative = (kd/Ts) * rate;                  //==============>>>> derivative
lastError = error;                            //==============>>>> error sebelumnya
sinyalControl =  proporsional + derivative; // + integral;       //sinyal control , derivative, integral
rightPWM = setPointPWM - sinyalControl;
leftPWM = setPointPWM + sinyalControl;

if(rightPWM >= 255)
      {
      rightPWM = 150;
      }
if(leftPWM >= 255)
      {
      leftPWM = 150;
      }
if(rightPWM <= 0)
      {
      rightPWM = 0;
      }
if(leftPWM <= 0)
      {
      leftPWM = 0;
      }
leftPWM;
rightPWM;
if(leftPWM == 0 && rightPWM == 150)
      {
       bantingKiri(leftPWM,rightPWM);

      }
else if(rightPWM == 0 && leftPWM == 150)
      {
       bantingKanan(leftPWM,rightPWM);

      }
  if( (rightPWM <255 || leftPWM < 255)  && (rightPWM > 0 || leftPWM > 0) ){
              if( (leftPWM > rightPWM) && (rightPWM < 30) ){
                   kanan(leftPWM,rightPWM);
                   }
              if( (leftPWM < rightPWM) && (leftPWM < 30)){
                   kiri(leftPWM,255);
                    lcd_clear();
            lcd_gotoxy(0,0);
            sprintf(text,"S.K=%3d S.D=%3d ",getSrf(0),getSrf(1));
            lcd_puts(text);
```

```
                lcd_gotoxy(0,1);
                sprintf(text,"r=%3d l=%3d",rightPWM,processVariable);
                lcd_puts(text);
                      }


                 else{
                      maju(leftPWM,rightPWM);
                 }



                }

      if( leftPWM > rightPWM )
         {
         kanan(leftPWM,rightPWM);
         }
      else if( leftPWM < rightPWM )
         {
         kiri(leftPWM,rightPWM);
         }
      else if( leftPWM == rightPWM )
         {
         maju(leftPWM,rightPWM);
         }


    if(ADCdepanKiri<170 && ADCbelakangKiri<350 && padam==1 && counterB >= 2 ) //PUTIH 2-
2nya ----------------
      {
        stop(255,255);
        delay_ms(1000);
        PORTD.4=1;
        delay_ms(500);
        PORTD.4=0;
        delay_ms(500);
        PORTD.4=1;
        delay_ms(500);
        PORTD.4=0;
        delay_ms(500);
        PORTD.4=1;
        delay_ms(500);
        PORTD.4=0;
        delay_ms(500);
        PORTD.4=1;
        delay_ms(500);
        PORTD.4=0;
        delay_ms(500);
        stop(255,255);
        lcd_clear();
        lcd_gotoxy(0,0);
        sprintf(text,"CounterB=%2d", counterB);
        lcd_puts(text);
        lcd_gotoxy(0,1);
        sprintf(text,"HOME! Detect=%1d",detect);
```

```
                lcd_puts(text);
                stop(255,255);
              }


            } //end if PID

            else if (getSrf(1) <= 7|| ADCboneka>=340 || limit==1) // BONEKA ----------
             {

            bantingKanan(250,250);

             }

      }


      else if((padam==0 && island1==1) || (padam==1 && island1==1 ))
      {
      if(getSrf(2) > 10 && limit==0 && ADCboneka < 280)  //BONEKA----------
            {
            kp=6;          //4                    //wall kanan
            kd=3;         //1
            //ki=0.5;
            setPointPWM = 150;
            processVariable = getSrf(3);                  //=================== proporsional program
            setPointSensorKiri = 9;
            error = setPointSensorKiri - processVariable;
            proporsional = kp * error;                    //=================== proporsional end
            rate = error - lastError;                   //=================== derivative program
            //rateInt = error + lastError;
            Ts=1;                                 //===============>>>> Ts = time sampling
            //integral = Ts * (ki * rateInt);
            derivative = (kd/Ts) * rate;                  //===============>>>> derivative
            lastError = error;                          //===============>>>> error sebelumnya
            sinyalControl =  proporsional + derivative; // + integral;       //=================== sinyal control ,
derivative end
            rightPWM = setPointPWM - sinyalControl;
            leftPWM = setPointPWM + sinyalControl;

            if(rightPWM >= 255)
                  {
                  rightPWM = 150;
                  }
            if(leftPWM >= 255)
                  {
                  leftPWM = 150;
                  }
            if(rightPWM <= 0)
                  {
                  rightPWM = 0;
                  }
            if(leftPWM <= 0)
                  {
                  leftPWM = 0;
                  }
```

```
leftPWM;
rightPWM;
if(leftPWM == 0 && rightPWM == 150)
      {
       bantingKiri(leftPWM,rightPWM);
      }
else if(rightPWM == 0 && leftPWM == 150)
      {
       bantingKanan(leftPWM,rightPWM);
      }
if( (rightPWM <255 || leftPWM < 255)  && (rightPWM > 0 || leftPWM > 0) )
  {
            if( (leftPWM > rightPWM)&& (rightPWM < 60)  )
                {
                 kiri(leftPWM,rightPWM);
                }
            if( (leftPWM < rightPWM)&& (leftPWM < 60) )
                {
                kanan(255,rightPWM);
                lcd_clear();
                lcd_gotoxy(0,0);
                sprintf(text,"kanan=%3d",getSrf(3) );
                lcd_puts(text);
                }
            else
                {
                 maju(leftPWM,rightPWM);
                }
             lcd_gotoxy(0,1);
             sprintf(text,"kiri=%3d",leftPWM,);
             lcd_puts(text);
             lcd_gotoxy(0,1);
             sprintf(text,"r=%3d l=%3d",rightPWM,processVariable);
             lcd_puts(text);

  }
   if( leftPWM > rightPWM )
      {
      kiri(leftPWM,rightPWM);
      }
   else if( leftPWM < rightPWM )
      {
      kanan(leftPWM,rightPWM);
      }
   else if( leftPWM == rightPWM )
      {
      maju(leftPWM,rightPWM);
      }

   if(ADCdepanKiri<150 && ADCbelakangKiri< 350 && padam==1 && counterB >= 2 ) //PUTIH 2-
2nya-------
    {
      stop(255,255);
      delay_ms(1000);
      PORTD.4=1;
      delay_ms(500);
```

```
                PORTD.4=0;
                delay_ms(500);
                PORTD.4=1;
                delay_ms(500);
                PORTD.4=0;
                delay_ms(500);
                PORTD.4=1;
                delay_ms(500);
                PORTD.4=0;
                delay_ms(500);
                PORTD.4=1;
                delay_ms(500);
                PORTD.4=0;
                delay_ms(500);
                stop(255,255);
                lcd_clear();
                lcd_gotoxy(0,0);
                sprintf(text,"CounterB=%2d", counterB);
                lcd_puts(text);
                lcd_gotoxy(0,1);
                sprintf(text,"Detect=%1d",detect);
                lcd_puts(text);
                stop(255,255);
              }

          } //end if PID

          else if (getSrf(2) <= 9 || ADCboneka>=280 || limit==1 ) //--------BONEKA----------
           {

          bantingKiri(200,200);

           }

    }

}




 else     //jika nemu api
 {
    PORTD.4=1;  //buzzer
        if( ADCdepanKiri < 150 && detect == 0 && counterA>=1 )     //-------PUTIH---------
        {
              maju(55,55);
              delay_ms(250);
              detect=1;
        }

        else if((ADCdepanKiri < 150||ADCbelakangKanan <120 )  && detect == 1)   //---PUTIH 2-2nya--------
        {
              stop(255,255);

              while (PINF.4==0 && counterApi<=16 )
```

```
                {
                reg=0x00;
                TPA_read();
                revision=data;

                reg=0x01;
                TPA_read();
                ambient=data;

                for (i=0;i<8;i++)
                {
                reg=reg+1;
                TPA_read();
                pixel[i]=data;
                }
    sprintf(Msg1,"%3d %3d %3d %3d", pixel[0],pixel[1],pixel[2],pixel[3]);
    sprintf(Msg2,"%3d %3d %3d %3d", pixel[4],pixel[5],pixel[6],pixel[7]);
    lcd_gotoxy(0,0);
    lcd_puts(Msg1);
    lcd_gotoxy(0,1);
    lcd_puts(Msg2);

                if (pixel[0] >= suhuReferensi || pixel[1] >= suhuReferensi || pixel[2] >= suhuReferensi || pixel[3] >=
suhuReferensi || pixel[4] >= suhuReferensi || pixel[5] >= suhuReferensi || pixel[6] >= suhuReferensi || pixel[7] >=
suhuReferensi)

                  {
                  kipasTPA();
                  padam=1;
                  }
                servoTPA();
                counterApi+=1;

//      lcd_clear();
//      lcd_gotoxy(0,0);
//      sprintf(text,"%1d",counterApi);
//      lcd_puts(text);
                stop(255,255);
                }


          }  // end klo dapat juring lingkaran putih


          else if (island1==1)
          {
          if(getSrf(2) > 10 && limit==0 && ADCboneka < 280)  //BONEKA----------
             {
             kp=6;          //4                    //wall kanan
             kd=3;         //1
             //ki=0.5;
             setPointPWM = 150;
             processVariable = getSrf(3);                  //================== proporsional program
             setPointSensorKiri = 9;
             error = setPointSensorKiri - processVariable;
             proporsional = kp * error;                    //================== proporsional end
```

```
            rate = error - lastError;                    //================== derivative program
            //rateInt = error + lastError;
            Ts=1;                                  //===============>>>> Ts = time sampling
            //integral = Ts * (ki * rateInt);
            derivative = (kd/Ts) * rate;                 //===============>>>> derivative
            lastError = error;                           //===============>>>> error sebelumnya
            sinyalControl =  proporsional + derivative; // + integral;       //================== sinyal control ,
derivative end
            rightPWM = setPointPWM - sinyalControl;
            leftPWM = setPointPWM + sinyalControl;

            if(rightPWM >= 255)
                  {
                  rightPWM = 150;
                  }
            if(leftPWM >= 255)
                  {
                  leftPWM = 150;
                  }
            if(rightPWM <= 0)
                  {
                  rightPWM = 0;
                  }
            if(leftPWM <= 0)
                  {
                  leftPWM = 0;
                  }
            leftPWM;
            rightPWM;
            if(leftPWM == 0 && rightPWM == 150)
                  {
                   bantingKiri(leftPWM,rightPWM);
                  }
            else if(rightPWM == 0 && leftPWM == 150)
                  {
                   bantingKanan(leftPWM,rightPWM);
                  }
            if( (rightPWM <255 || leftPWM < 255)  && (rightPWM > 0 || leftPWM > 0) )
              {
                        if( (leftPWM > rightPWM)&& (rightPWM < 30)  )
                            {
                             kiri(leftPWM,rightPWM);
                            }
                        if( (leftPWM < rightPWM)&& (leftPWM < 30) )
                            {
                            kanan(255,rightPWM);
                            lcd_clear();
                            lcd_gotoxy(0,0);
                            sprintf(text,"kanan=%3d",getSrf(3) );
                            lcd_puts(text);
                            }
                        else
                            {
                             maju(leftPWM,rightPWM);
                            }
                         lcd_gotoxy(0,1);
```

```
                sprintf(text,"kiri=%3d",leftPWM,);
                lcd_puts(text);
                lcd_gotoxy(0,1);
                sprintf(text,"r=%3d l=%3d",rightPWM,processVariable);
                lcd_puts(text);

     }
      if( leftPWM > rightPWM )
         {
         kiri(leftPWM,rightPWM);
         }
      else if( leftPWM < rightPWM )
         {
         kanan(leftPWM,rightPWM);
         }
      else if( leftPWM == rightPWM )
         {
         maju(leftPWM,rightPWM);
         }



  } //end if PID

  else if (getSrf(2) <= 9 || ADCboneka>=280 || limit==1 ) //--------BONEKA----------
   {

  bantingKiri(200,200);

   }
 }

else  //  else klo blm dapat juring, jalan trs nyari juring putih+klo ada api
{

if(getSrf(1) > 8 && limit==0 && ADCboneka < 280) //BONEKA--------
    {
    kp=4;          //4                    //awal PID
    kd=2;          //1
    //ki=0.1;
    setPointPWM = 150;
    processVariable = getSrf(0);                  //================== proporsional program
    setPointSensorKiri = 7;
    error = setPointSensorKiri - processVariable;
    proporsional = kp * error;                    //================== proporsional end
    rate = error - lastError;                    //================== derivative program
    //rateInt = error + lastError;
    Ts=1;                                  //==============>>>> Ts = time sampling
    //integral = Ts * (ki * rateInt);
    derivative = (kd/Ts) * rate;                  //==============>>>> derivative
    lastError = error;                           //==============>>>> error sebelumnya
    sinyalControl =  proporsional + derivative; // + integral;       //sinyal control , derivative, integral
    rightPWM = setPointPWM - sinyalControl;
    leftPWM = setPointPWM + sinyalControl;

    if(rightPWM >= 255)
```

```
      {
      rightPWM = 150;
      }
if(leftPWM >= 255)
      {
      leftPWM = 150;
      }
if(rightPWM <= 0)
      {
      rightPWM = 0;
      }
if(leftPWM <= 0)
      {
      leftPWM = 0;
      }
leftPWM;
rightPWM;
if(leftPWM == 0 && rightPWM == 150)
      {
       bantingKiri(leftPWM,rightPWM);

      }
else if(rightPWM == 0 && leftPWM == 150)
      {
       bantingKanan(leftPWM,rightPWM);

      }
  if( (rightPWM <255 || leftPWM < 255)  && (rightPWM > 0 || leftPWM > 0) ){
                if( (leftPWM > rightPWM) && (rightPWM < 30) ){
                     kanan(leftPWM,rightPWM);
                     }
                if( (leftPWM < rightPWM) && (leftPWM < 30)){
                     kiri(leftPWM,255);
              lcd_clear();
              lcd_gotoxy(0,0);
              sprintf(text,"S.K=%3d S.D=%3d ",getSrf(0),getSrf(1));
              lcd_puts(text);
                     }
                 else{
                     maju(leftPWM,rightPWM);
                     }




               }

  if( leftPWM > rightPWM )
     {
     kanan(leftPWM,rightPWM);
     }
  else if( leftPWM < rightPWM )
     {
     kiri(leftPWM,rightPWM);
     }
  else if( leftPWM == rightPWM )
```

```
                {
                maju(leftPWM,rightPWM);
                }

            } //end if PID

            else if (getSrf(1) <= 7|| ADCboneka>=280 || limit==1) // BONEKA ------------
            {
            bantingKanan(250,250);
            }

          }  //end dari else klo blm dapat juring, jalan trs nyari juring putih+klo ada api


}     // end dari jika nemu api




if (ADCdepanKiri < 150 && padam==0)  //counter sebelum madamin api  //---PUTIH------
    {
    if ( ADCbelakangKiri> 480 && varA==0)       //-----ABU------
            {
             counterA +=1;
             varA=1;
             lcd_clear();
             lcd_gotoxy(0,0);
             sprintf(text,"CounterA=%2d", counterA);
             lcd_puts(text);
             lcd_gotoxy(0,1);
             sprintf(text,"Detect=%1d",detect);
             lcd_puts(text);
            }
    }

else if (ADCdepanKiri > 400 && padam==0)      //-----ABU------
    {
    varA=0;
    }



if (ADCdepanKiri < 150 && padam==1)          //counter setelah padamin api     //---PUTIH------
    {
    if ( ADCbelakangKiri> 480 && varB==0)    //----ABU-----
            {
             counterB +=1;
             varB=1;
             lcd_clear();
             lcd_gotoxy(0,0);
             sprintf(text,"CounterB=%2d", counterB);
             lcd_puts(text);
             lcd_gotoxy(0,1);
             sprintf(text,"Detect=%1d",detect);
```

```
            lcd_puts(text);
           }
      }

  else if (ADCdepanKiri > 400 && padam==1)     //-----ABU------
      {
      varB=0;
      }


  if(PINB.2==1)
  {
  limit=0;
  PORTD.4=0;
  }

  else if (PINB.2==0)
  {
  limit=1;
  PORTD.4=1;
  }

 //============================== ADC KALIBRASIIIIIII JANGAN LUPAAA=================
  if ( padam==0 && ADCbelakangKanan>400 && ADCbelakangKanan<620 && getSrf(5)<18 && getSrf(6)<18
&& getSrf(1)>50 && getSrf(1)<75 && getSrf(4)>38 && getSrf(4)<55 )
  {
  island1=1; //posisi no 1
  }

  if ( padam==1 && counterB>=1 && ADCbelakangKanan>400 && ADCbelakangKanan<620 && getSrf(3)<18
&& getSrf(5)<18 && getSrf(1)>35 && getSrf(1)<50 && getSrf(4)>50  )
  {
  if(ADCbelakangKiri>600 && ADCbelakangKiri<750 && island1==1)
  {
  homeIsland1=1;  //posisi no 2
  }
  }

  if (padam==0 && ADCbelakangKanan>400 && ADCbelakangKanan<620  && getSrf(6)<18 && getSrf(5)<18
&& getSrf(1)>55 && getSrf(1)<75 && getSrf(4)>32 && getSrf(4)<45)
  {
  island1=1; // posisi no 3
  island2=1;
  }

  if (padam==1 && island2==1 && ADCbelakangKanan>660 && getSrf(6)<18 && getSrf(5)<18 &&
getSrf(1)>70 && getSrf(1)<85 && getSrf(4)>38 && getSrf(4)<52   )
  {
  homeIsland1=1; //posisi no 4
  }

  if (padam==0 && ADCbelakangKanan>400 && ADCbelakangKanan<620  && getSrf(6)<18 && getSrf(5)<18
&& getSrf(1)>37 && getSrf(4)>50 && getSrf(4)<75 )
  {
  island1=1;  //posisi 5
  }
```

```
  if (padam==1 && ADCbelakangKanan>650 && getSrf(6)<18 && getSrf(5)<18 && getSrf(1)>60 &&
getSrf(4)>40 && getSrf(4)<58  )   //Gambar No 1
  {
  homeIsland1=1; //posisi 6
  }

  if (padam==1 && ADCbelakangKanan>650 && getSrf(6)<18 && getSrf(5)<18 && getSrf(1)>42 &&
getSrf(4)>50 && getSrf(4)<65  )   //Gambar No 1
  {
  homeIsland1=1; //posisi 7
  }


  if (padam==0 && ADCbelakangKanan>400 && ADCbelakangKanan<620  && getSrf(6)<18 && getSrf(5)<18
&& getSrf(1)>30 && getSrf(4)>30 && getSrf(4)<75 )
  {
  island1=1;  //posisi 5
  }

  if (padam==1 && island1==1 && ADCbelakangKanan>660 && getSrf(6)<18 && getSrf(5)<18 &&
getSrf(1)>70 && getSrf(1)<85 && getSrf(4)>38 && getSrf(4)<52   )
  {
  island1=0; //posisi no 4   //kasus no 12b
  }

if (padam==1 && counterB>=1 && ADCbelakangKanan>700 && getSrf(1)>90 && getSrf(4)>20)    Gambar No
1
{
homeIsland1=1;
}


if (counterApi==16  )
 {
 B++;
 }

  if (counterApi > 15)
  {
  maju(70,70);
  delay_ms(150);
  PORTD.4=1;
  delay_ms(1000);
  island1=1;
  padam=0;
  counterApi=0;
  }

 }                //=====end if TOMBOL START=======//

else
  {

  sound=read_adc(7);
  if(sound<10)
```

```
      {
       while(sound<10)
       {a++;
       lcd_clear();
       lcd_gotoxy(0,0);
       sprintf(text,"SOUND = %d",a);
       lcd_puts(text);
       sound=read_adc(7);
       delay_ms(10);
       if(a>=60)
       {start=1;break;}
       }
      a=0;
      }

   stop(255,255);
//   if(PINB.2==1)
//   {
//   PORTD.4=0;
//   }
//
//   else if (PINB.2==0)
//   {
//   PORTD.4=1;
//   }

    if(PINE.7==1 && PINE.5==1)
    {
    //=========================Test Sensor Warna===================================
        ADCdepanKiri=read_adc(0);
        ADCbelakangKiri=read_adc(1);
        ADCbelakangKanan=read_adc(2);
        ADCboneka=read_adc(3);
//        ADCkanan=read_adc(5);

      lcd_clear();
      lcd_gotoxy(0,0);
      sprintf(text,"%3d %3d %3d
%3d",ADCdepanKiri,ADCbelakangKiri,ADCbelakangKanan,ADCkanan);//,read_adc(1),read_adc(2));  //
adc(0)=depan kiri, adc(1)=belakang kiri, adc(2)=belakang kanan
      lcd_puts(text);
      lcd_gotoxy(0,1);
      sprintf(text,"JarakBoneka=%3d",ADCboneka);  // adc(3)= sensor boneka
      lcd_puts(text);
      delay_ms(200);
    }

    else if (PINE.7==0)
    {
    //  ============================Test Sensor Jarak Jek================================
        lcd_clear();
        lcd_gotoxy(0,0);
        sprintf(text,"%3d %3d %3d %3d ",getSrf(0),getSrf(1),getSrf(2),getSrf(3));
        lcd_puts(text);
        lcd_gotoxy(0,1);
        sprintf(text,"%3d %3d %3d",getSrf(4),getSrf(5),getSrf(6));
```

```
    lcd_puts(text);
    delay_ms(200);
}

if (PINE.5==0 && PINE.7==1)
{
 reg=0x00;
 TPA_read();
 revision=data;

 reg=0x01;
 TPA_read();
 ambient=data;

 for (i=0;i<8;i++)
{
 reg=reg+1;
 TPA_read();
 pixel[i]=data;
}
 lcd_clear();
 sprintf(Msg1,"%3d %3d %3d %3d", pixel[0],pixel[1],pixel[2],pixel[3]);
 sprintf(Msg2,"%3d %3d %3d %3d", pixel[4],pixel[5],pixel[6],pixel[7]);
 lcd_gotoxy(0,0);
 lcd_puts(Msg1);
 lcd_gotoxy(0,1);
 lcd_puts(Msg2);

}
```