BAB II

LANDASAN TEORI

2.1 Robot

Kemunculan pertama kalinya kata "robot" adalah dalam sebuah drama karangan Karel Capek yang berjudul "R.U.R (*Rossum's Universal Robot*)" pada tahun 1921. Asal kata "robot" sendiri adalah dari kata "robota" yang merupakan bahasa Ceko yang berarti tenaga kerja paksa (*forced labour*). Selain itu, dari sebuah karya Isaac Asimov berupa cerpen (cerita pendek) fiksi ilmiah yang berjudul "*Runaround*" pada tahun 1942 juga muncul kata "*robotics*". Isaac Asimov kemudian memasukkan cerita pendek tersebut ke dalam buku karangannya yang sangat terkenal, yaitu "*I, Robot*" (Siswaja, 2008).

Robotika merupakan satu cabang teknologi yang berhubungan dengan desain, konstruksi, operasi, disposisi struktural, pembuatan, dan aplikasi dari robot. Robotika terkait dengan ilmu pengetahuan di bidang elektronika, mesin, mekanika, dan perangkat lunak komputer (Maryani, 2013).

2.1.1 Definisi Robot

Ada beberapa definisi tentang robot yang dapat ditemukan (Hendrik, 2013), yaitu:

a. Kamus Webster

Sebuah alat otomatis yang melakukan fungsi dan kinerja sama dengan yang dilakukan oleh manusia.

b. Kamus Oxford

Sebuah mesin yang mampu melakukan serangkaian kompleks tindakan otomatis, terutama yang diprogram oleh komputer.

c. *International Standard Organization* (ISO 8373)

Sebuah pengontrol otomatis yang dapat diprogram ulang, manipulator serbaguna deprogram dalam tiga atau lebih poros, yang mana tetap di tempat atau bergerak untuk digunakan dalam otomasi industri.

2.1.2 Bentuk Robot

Berdasarkan bentuknya, robot dibagi ke dalam lima jenis (Siswaja, 2008), yaitu:

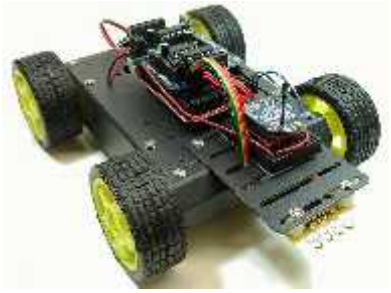
1. *Fixed Robot*

Merupakan bentuk robot yang tidak dapat berpindah dari satyu tempat ke tempat lain secara keseluruhan.Bentuk dari robot ini bersifat statis dari segi posisi dan sangat sulit dipindahkan. Contoh dari bentuk robot ini adalah: robot di bidang industri.

Gambar 2.1 Robot Industri di Sebuah Industri Mobil

2. *Mobile Robot*

Merupakan bentuk robot yang dapat secara dinamis berpindah tempat dari satu titik ke titik lainnya. Memiliki alat gerak untuk berpindah, seperti roda atau kaki.



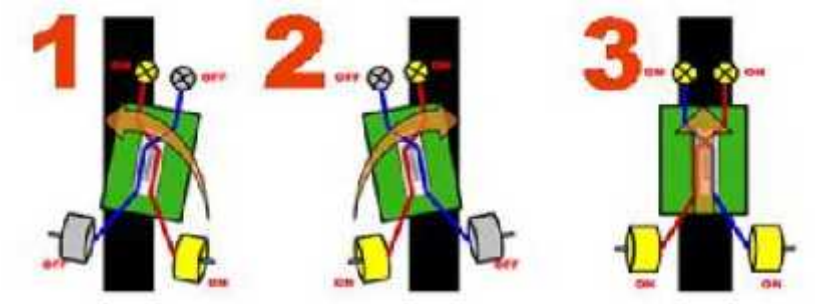
Gambar 2.2Mobile Robot

Robot *line tracer* atau *line follower*merupakan robot yang bergerak dengan mengikuti jalur berupa garis. Garis ini bisa berupa garis hitam di atas permukaan berwarna putih atau sebaliknya (Rudiyanto, 2010). Sensor yang digunakan untuk mendeteksi pola pada permukaan adalah sensor cahaya. Di mana warna garis dan permukaan sekitarnya memiliki perbedaan yang jelas seperti warna gelap dan

warna terang (Jatmiko dkk., 2010:47). Proses kerjanya dalam mendeteksi garis adalah informasi yang diterima sensor dikirim ke prosesor untuk diolah lalu diteruskan ke motor, sehingga motor dapat menggerakkan robot untuk mengikuti garis (Sholahuddinn & Hadi, 2013).

Menurut Sholahuddinn dan Hadi (2013), konstruksi paling sederhana pada robot line tracer setidaknya memiliki dua buah sensor (A-kiri dan B-kanan) yang terhubung dengan kedua motor (kiri dan kanan) secara bersilangan melalui sebuah processor/driver. Sensor A (kiri) mengendalikan motor kanan dan sensor B (kanan) mengendalikan motor kiri. Berikut penjabarannya:

a. Ketika sensor A mendeteksi garis sedangkan sensor B tidak, ini berarti posisi robot berada di sebelah kanan garis. Sehingga motor kanan akan aktif sementara motor kiri tidak, sehingga robot akan bergerak ke kiri.
b. Jika sensor B mendeteksi garis dan sensor A tidak, ini berarti robot berada di sebelah kiri garis. Maka motor kiri akan aktif dan motor kanan mati, sehingga robot akan bergerak ke kanan.
c. Jika sensor A dan sensor B sama-sama mendeteksi garis, artinya robot berada di atas garis. Maka kedua motor akan aktif dan robot bergerak lurus.



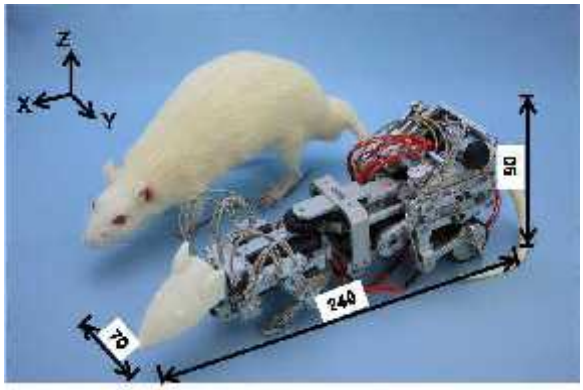
Gambar 2.3 Penjabaran Sederhana Cara Kerja *Line Tracer*

Chandra dkk. (2013) menjelaskan bahwa komponen-komponen pada *line tracer* terdiri dari: kapasitor, resistor, transistor, dioda, saklar, *Printed Circuit Board* (PCB),*Light Emiting Diode* (LED), *photodiode*, dan motor DC. Kapasitor berfungsi menyimpan muatan listrik sementara yang terdiri dari kapasitor polar dan non polar. Resistor bertugas mengendalikan arus yang masuk ke rangkaian listrik. Transistor adalah suatu bahan yang dapat merubah bahan yang tidak dapat menghantarkan arus listrik menjadi bahan pengantar atau setengah mengantar arus listrik (semi-konduktor).Dioda hanya melewatkan arus atau tegangan dalam satu

arah saja, mengubah arus bolak balik (AC) menjadi arus searah (DC). Saklar berfungsi sebagai pemutus atau penyambung aliran listrik. PCB sebagai tempat dipasangnya komponen-komponen elektronika. LED berfungsi memancarkan cahaya. *Photodiode* berfungsi sebagai sensor cahaya. Motor DC adalah komponen yang mengubah energi listrik menjadi energi mekanik.

3. Robot dengan Bentuk Tubuh Hewan (*Animaloid/Bug Robot*)

Merupakan robot yang memiliki bentuk menyerupai binatang. Disebut *bug robot* karena pada awalnya mengambil bentuk serangga.



Gambar 2.4*Animaloid* Berbentuk Tikus

4. *Humanoid*

Merupakan robot yang memiliki bentuk menyerupai manusia. Hal yang paling sulit dari bentuk robot ini adalah memikirkan bagaimana robot dapat berdiri tegak dan berjalan dengan seimbang.

Gambar 2.5*Humanoid Robot*

5. *Combination*

Merupakan robot dengan bentuk gabungan dari keempat robot sebelumnya.

Gambar 2.6Tokoh Utama Pada Film "*Wall E*"

2.1.3 Sistem Robot dan Fungsinya

Pitowarno (dalam Maryani, 2013:II-9) menjelaskan bahwa sistem robot dan fungsinya dalam bidang robotika adalah sebagai berikut:

1. *Controller System*

Controller system berupa rangkaian elektronik yang minimal terdiri dari rangkaian *processor*yang berupa *Central Processing Unit* (CPU), memori, dan komponen antarmuka masukan/keluaran, lalus*ignal conditioning* untuk sensor (analog maupun digital), dan *driver* untuk aktuator. Akan lebih baik jika ditambahkan dengan sistem monitor seperti *Liquid Crystal Display* (LCD), atau *Cathode Ray Tube* (CRT), dan *seven segment*.

2. Mekanik Robot

Mekanik robot merupakan sistem mekanik yang minimal terdiri dari satu buah sistem gerak. Jumlah fungsi gerak ini dikenal juga dengan *Degree Of Freedom* (DOF) atau derajat kebebasan. Sebuah aktuator mewakili satu sendi, dan sendi tersebut disebut dengan satu DOF. Sedangkan pada struktur kaki dan roda diukur berdasarkan fungsi *holonomic* atau *non-holonomic*.

3. Sensor

Merupakan perangkat yang berfungsi sebagai pendeteksi gerakan atau perubahan lingkungan yang dibutuhkan oleh *controller system*. Bisa berupa sistem paling sederhana seperti sensor ON/OFF menggunakan limit *switch*, sistem bus serial, sistem bus paralel, sistem analog, hingga sistem kamera sebagai mata.

4. Aktuator

Merupakan perangkat yang berfungsi menghasilkan gerakan. Bisa berupa perangkat hidrolik (perangkat kompresi berbasis bahan cair seperti oli), sistem *pneumatic* (perangkat kompresi berbasis udara), atapun sistem motor listrik (*stepper*,*solenoid*, motor DC, magnet permanen, *brushless*, motor DC servo, dll.).

Sumber gerakan dari aktuator dilanjutkan dengan kaki/roda dan tangan. Robot dengan roda setidaknya terdiri dari sebuah roda penggerak (*drive* atau *steer*). Beberapa jenis lain di antaranya adalah dua roda diferensial (roda kiri dan kanan *independent* atau dapat bergerak secara berbeda satu sama lain atau sistem *belt* seperti pada tank), tiga roda (*synchro drive* atau sistem *holonomic*), dan empat roda (*Ackermann model/car-like mobile robot* atau sistem *mecanum wheels*).

Robot yang menggunakan kaki sesungguhnya pengembangan dari sistem roda yang didesain sedemikian rupa hingga dapat bergerak seperti makhluk hidup. Robot yang berjalan dengan sistem dua kaki (*biped robot*) memiliki struktur seperti persendian pada kaki manusia di mana setidaknya terdapat sendi-sendi yang mewakili pinggul, lutut, dan pergelangan kaki. Konfigurasi yang ideal untuk pergerakan pada pinggul terdiri dari multi DOF dengan kemampuan gerakan mengangkat dan membuka kaki. Begitu juga pada pergelangan kaki yang idealnya juga memiliki kemampuan untuk melakukan gerakan polar, yaitu gerakan naik turun dengan satu titik pusat. Jumlah kaki dapat didesain lebih dari empat buah, misalnya untuk *animaloid* seperti serangga. Bahkan untuk robot ular memiliki DOF yang banyak tergantung pada panjang robot ular itu sendiri.

Robot yang memiliki tangan lebih dikenal juga sebagai *manipulator*, yaitu sistem gerak yang berfungsi memanipulasi objek dengan cara memegang, mengangkat, memindahkan, atau mengolah. Pada robot industri fungsi ini dapat berupa putaran (memasang mur atau baut dan mengebor), mengelas atau membubut (*tracking*), ataupun mengaduk (kontrol proses). Robot dengan manipulator memiliki *real world*, yaitu daerah kerjanya. Robot yang tersusun dari tangan saja memiliki area kerja yang terbatas sesuai panjang jangkauan tangannya. Untuk robot yang memiliki roda atau kaki, area terbatas menjadi lebih luas tergantung dari kemampuan jelajahnya. Dengan menggabungkan robot tangan dan robot beroda/berkaki maka daerah kerja dapat berupa jalur di jalan (seperti *line follower* atau *route-runner robot*). Robot berjalan menuju objek menggunakan sensor radar, sonar, cahaya, atau kamera.

2.2 Arduino

Margolis (2011:71) menjelaskan bahwa Arduino terbagi menjadi dua, yaitu *hardware*dan *software*. Djuandi (2011:4) menjelaskan bahwa *hardware* merupakan papan input/output (I/O) dan *software* meliputi penulis program berupa *Integrated Development Environment*(IDE), penghubung (bukan secara fisik) dengan computer berupa *driver*, serta pengembang program berupa contoh program dan *library*.

Gambar 2.7 Arduino Uno

2.2.1 *Hardware*: Arduino Uno (ATMega328)

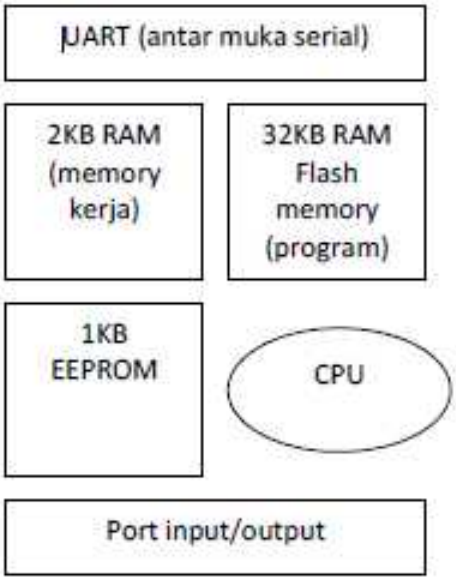
Robot *line tracer* yang dijadikan sebagai media penelitian ini menggunakan papan *microcontroller* Arduino Uno yang berbasis ATMega328. Berikut spesifikasi dari Arduino Uno:

Tabel 2.1 Spesifikasi Arduino Uno

Microcontroller	ATmega328
Tegangan pengoperasian	5V
Tegangan input yang disarankan	7-12V
Batas tegangan input	6-20V
Jumlah pin I/O digital	14 (6 di antaranya menyediakan keluaran PWM)
Jumlah pin input analog	6
Arus DC tiap pin I/O	40 mA
Arus DC untuk pin 3.3V	50 mA
Memori Flash	32 KB (ATmega328), sekitar 0.5 KB digunakan oleh bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Djuandi (2011:8) menggambarkan diagram blok sederhana *microcontroller* ATMega328 (Arduino Uno). Blok-blok tersebut terdiri dari:

1. *Universal Asynchronous Receiver/Transmitter* (UART), merupakan antar muka untuk komunikasi serial.
2. Memory kerja 2KB RAM yang digunakan oleh variable-variabel dalam program, bersifat *volatile* (hilang saat daya diputuskan).
3. *Flash memory* 32KB RAM yang bersifat *non-volatile*untuk menyimpan program yang di-*upload* dari computer dan juga *bootloader*, yaitu program yang dijalankan CPU saat daya dinyalakan.
4. EEPROM 1KB untuk menyimpan data yang tidak boleh hilang saat daya diputuskan. Bersifat *non-volatile*.
5. CPU untuk menjalankan setiap instruksi program.
6. Port input/output untuk menerima dan mengeluarkan data digital atau analog melalui pin-pin yang ada.



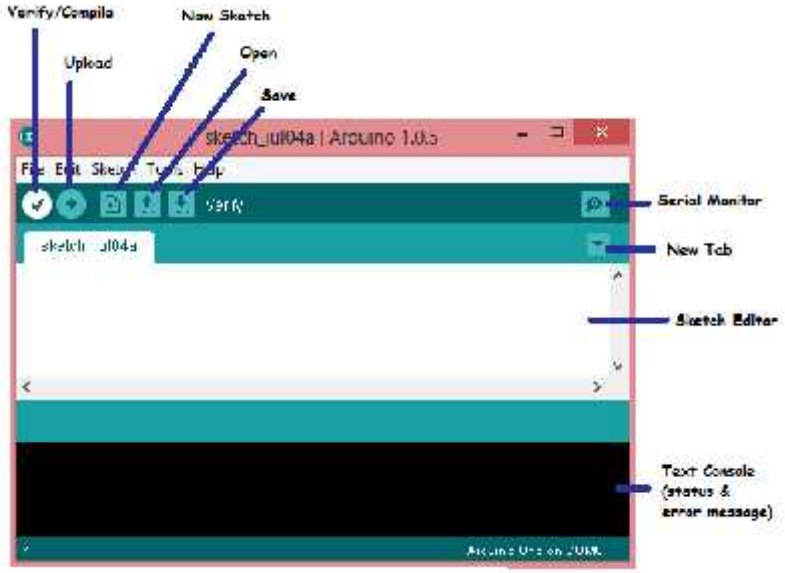
Gambar 2.8 Diagram Blok Sederhana Arduino Uno

Karvinen (2011:18) menjelaskan bahwa Arduino Uno dihubungkan melalui sebuah kabel USB-B dengan ujung USB-A yang terhubung ke komputer. Kabel dapat dilihat pada gambar 2.9 berikut.

Gambar 2.9 Kabel USB Arduino Uno: USB-B dan USB-A

2.2.2 *Software:* IDE Arduino 1.0.5

Menurut Djuandi (2011:12), IDE Arduino terdiri dari *editor* program, *compiler*, dan *uploader*. *Editor* program berfungsi sebagai tempat bagi pengguna untuk menulis dan mengedit program. Karena *microcontroller* hanya memahami kode biner, maka diperlukan *compiler* untuk mengubah kode program menjadi kode biner. Dan agar kode biner bisa dimuat ke dalam *microcontroller*, maka dibutuhkan sebuah modul yang bernama *uploader*. Kode program sering juga disebut dengan istilah *sketch*.



Gambar 2.10 Gambar Penjelasan Singkat *Tools* Arduino 1.0.5

2.3 Algoritma *Line Maze Solving*

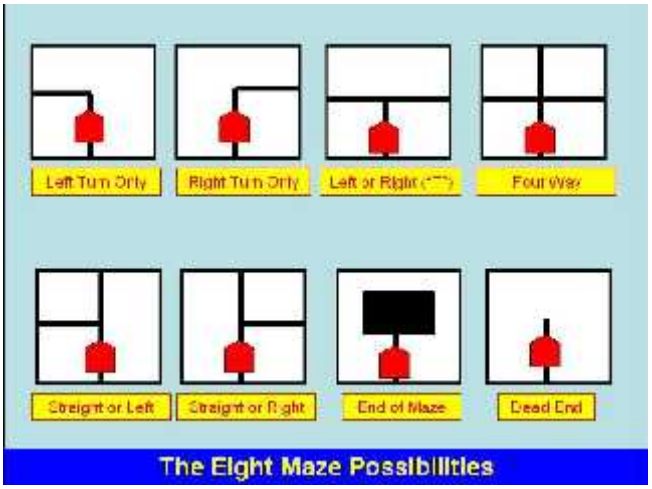
Algoritma*line maze solving* adalah salah satu algoritma yang digunakan pada robot untuk mencari jalur terpendek dari sebuah *line maze*. Terdapat 2 aturan dalam algoritma ini, yaitu *left hand rule* dan *right hand rule*. Dalam *left hand rule*, robot akan lebih memilih untuk belok kiri dari pada lurus atau belok kanan dan jika tidak ada belokan ke kiri akan lebih memilih lurus dari pada belok kanan. Sebaliknya dalam *right hand rule*, robot akan lebih memilih belok kanan dari pada lurus dan lebih memilih lurus dari pada belok kiri (Vanoy II, 2009).

Berdasarkan penjelasan Vanoy II (2009) mengenai pengujian algoritma *line maze solving*, robot yang telah diprogram dengan algoritma ini diletakkan di titik *start* sebuah *line maze*. Kemudian saat menjumpai persimpangan, robot akan memilih untuk berbelok sesuai dengan prioritas dari *rule* yang digunakan. Hal ini terus berlanjut sambil robot menyimpan/merekam jalur yang telah dilewati ke dalam memori hingga robot menemui akhir *maze*(*end of maze*). Mishra (dalam Hendriawan, 2010) menyebutkan proses perekaman jalur ini disebut sebagai *mapping*. Setelah selesai, robot langsung melakukan kalkulasi/perhitungan atas jalur yang telah tersimpan dalam memori robot untuk mendapatkan jalur terpendek dalam menyelesaikan *maze* tersebut pada penelusuran ke dua kalinya.

Menurut Vannoy II (2009), terdapat 8 kemungkinan dalam *maze* yang ia sebut dengan "*The Eight Maze Possibilities*". Kedelapan kemungkinan tersebut adalah sebagai berikut:

1. tidak ada pilihan lain selain belok kiri(*left turn only*),
2. tidak ada pilihan lain selain belok kanan(*right turn only*),
3. kiri atau kanan (*left or right*),
4. kiri, lurus, atau kanan (*four way*),
5. lurus atau kiri (*straight or left*),
6. lurus atau kanan(*straight or right*),
7. jalan buntu(*dead end*),
8. *end of maze.*

Left turn only dan *right turn only* bukanlah merupakan sebuah persimpangan, sebab hanya ada satu belokan. Sehingga robot tidak menyimpan apapun ke dalam memori. *Dead end* juga bukan sebuah persimpangan, namun karena *dead end* menandakan bahwa jalur yang dipilih adalah salah maka robot melakukan *U-turn* atau putar balik dan menyimpannya ke dalam memori.



Gambar 2.11*The Eight Maze Possibilities*

Dalam penjelasannya, Vannoy II menggunakan robot dengan 5 sensor. Di mana berjalan lurus adalah saat pembacaan sensor "00100". Jika saat lurus terjadi perubahaan pembacaan sensor menjadi "00000", maka ini berarti robot menemukan jalan buntu.

Dan dalam pembacaan sensor tersebut terdapat beberapa kemungkinan persimpangan yang mirip. Yang pertama antara *turn right only* dengan *straight or right*. Pada persimpangan ini sensor akan membacanya sebagai "00111". Maka untuk membedakannya adalah dengan melakukan "maju satu inchi". Ini dilakukan dengan membuat *subroutine* bernama "inch()" yang akan memerintahkan robot untuk maju sejauh satu inchi. Sehingga akan terjadi dua pembacaan yang berbeda, yaitu "00000" pada *turn right only* dan "00100" pada *straight or right*.Yang ke dua adalah antara *turn left only* dengan *straight or left*. Hal ini hampir sama dengan penjelasan pada kemungkinan antara *turn right only* dan *straight or right.* Bedanya pembacaan awal adalah "11100" dan akhir dari pembacaan *straight or left* adalah "00100".Yang ke tiga adalah antara *left or right*, *four* way, dan *end of maze* yang pembacaan awalnya adalah "11111".Setelah dilakukan "maju satu inchi", barulah terlihat perbedaan hasil pembacaan yang dilakukan oleh sensor. Yaitu "00000" pada *left or right*, "00100" pada *four way*, dan "11111" pada *end of maze*.Berikut adalah gambaran umum algoritma dari Vannoy II (2009):

```
      If first time through maze, go to Program One, otherwise, go
to Program Two
Program One:
- - Perform this loop until end of maze found:
At an intersection?
      If No, adjust speed and direction to go as fast as possible
      centered on the line.
      If Yes, determine the intersection type and decide which way
      to turn.
      Based on intersection type go straight, turn left, turn
      right or end program.
- - End Program One loop
Program Two:
- - Perform this loop until end of maze found:
Get direction of next turn from memory.
Go as fast as possible to next intersection.
At the end of the maze?
      If No, make the indicated turn.
      If Yes, end the program.
- - End Program Two loop
```

Algoritma 2.1 Gambaran Umum Algoritma *Line Maze Solving*

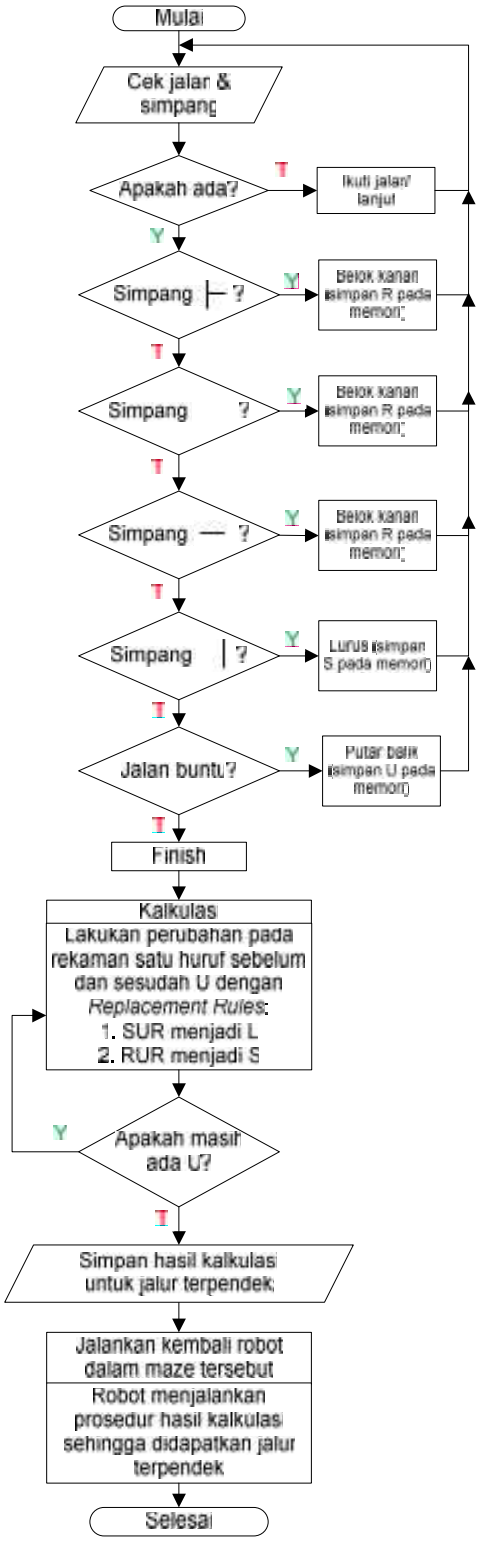
Dan ini adalah algoritma dari Vannoy II (2009) yang dituliskannya dengan *left hand rule*:

```
Do
  Gosub ReadSensors
  if sensors = %00100 then GoFastForward (Dead center - Go FAST!)
  elseif sensors = %01100 then GoSlightLeft (Just a little off to
      the right)
  elseif sensors = %01000 then GoMediumLeft
  elseif sensors = %10000 or %11000 then GoHardLeft
  elseif sensors = %00110 then GoSlightRight
  elseif sensors = %00010 then GoMediumRight
  elseif sensors = %00001 or %00011 then GoHardRight
  else (At an intersection if here)
    if   sensors   =   %00000   then   (Ran   off   the   line)
      storeThisTurn(U)
      Gosub U-Turn
    if sensors = %11111 then Gosub inchForward (Go forward to
      check)
      If sensors = %00000 then (Only happens at a T intersection)
        storeThisTurn(L)
        Gosub LeftTurn
      If sensors = %11111 then (Only happens at end of maze)
        endProgram
      else
        storeThisTurn(Left)  (Found  a  Four-Way  intersection)
        Gosub LeftTurn
    if sensors = %11100 then gosub leftTurn (This can be a Left-
Only or a Left-Straight intersection, but in either case, we go
left. But only one gets stored. Find out which.)
      inchForward()
      if sensors = %00000 then gosub leftTurn
      else
        storeThisTurn(Left)
        gosub leftTurn
    if sensors = %00111 then (This can be a Right-Only or a Right-
Straight intersection, so determine which)
      inchForward
      if sensors = %00000 then gosub rightTurn (This is Right-
Only, so go right)
```

```
      else(Keep    going    straight    -    or    do    nothing)
      storeThisTurn(Straight) (But still need to record it)

Loop
leftTurn()
leftMotor = backwards; rightMotor = forwards
delay 500 (Keep turning for 1/2 second)
End leftTurn

rightTurn()
leftMotor = forwards; rightMotor = backwards
delay 500 (Keep turning for 1/2 second)
End rightTurn

U-Turn()
gosub leftTurn
gosub leftTurn
End U-Turn

storeThisTurn(whichWay)
Add whichway to the stored string of turns
(Check to see if next-to-last character is "U".)
If storedString, length - 1 = "U" then
  Record last three characters
  Chop off last three characters
    If lastThree = LUL then replace with S
    If lastThree = LUR then replace with U
    If lastThree = LUS then replace with R
    If lastThree = RUL then replace with U
    If lastThree = RUR then replace with S
    If lastThree = RUS then replace with L
    If lastThree = SUL then replace with R
    If lastThree = SUR then replace with L
    If lastThree = SUS then replace with U
End storeThisTurn
```

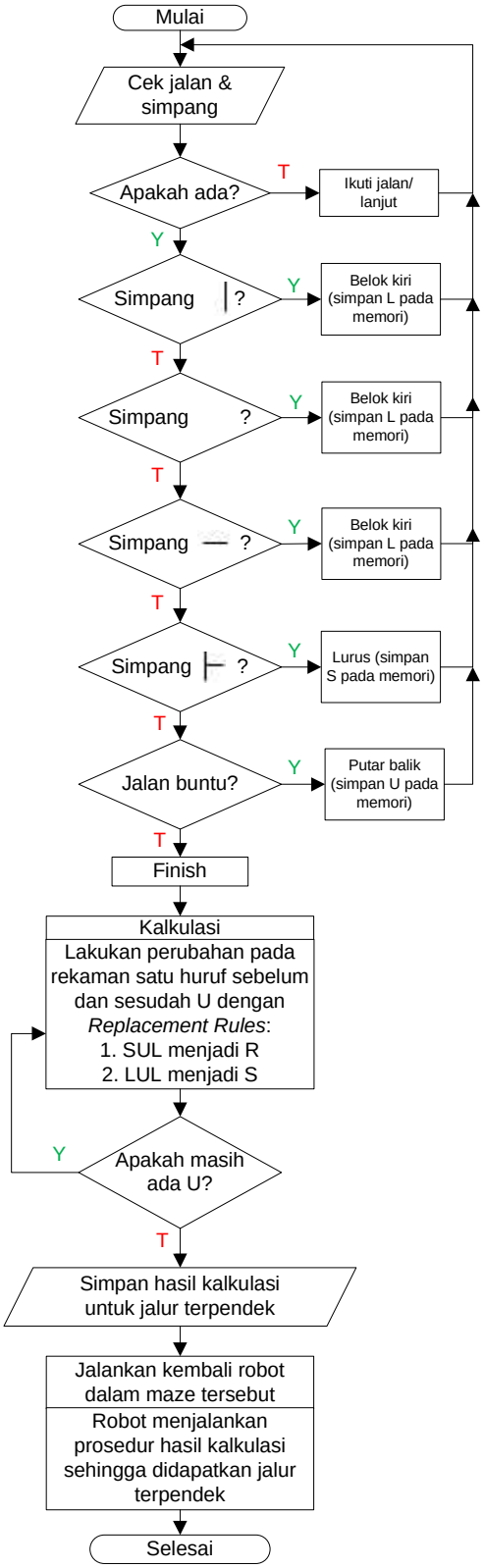
Algoritma 2.2 Algoritma *Line Maze Solving*(Vanoy II, 2009) dengan *Left Hand Rule*

Berikut adalah *flowchart*algoritma *line maze solving* dengan *right hand rule*:

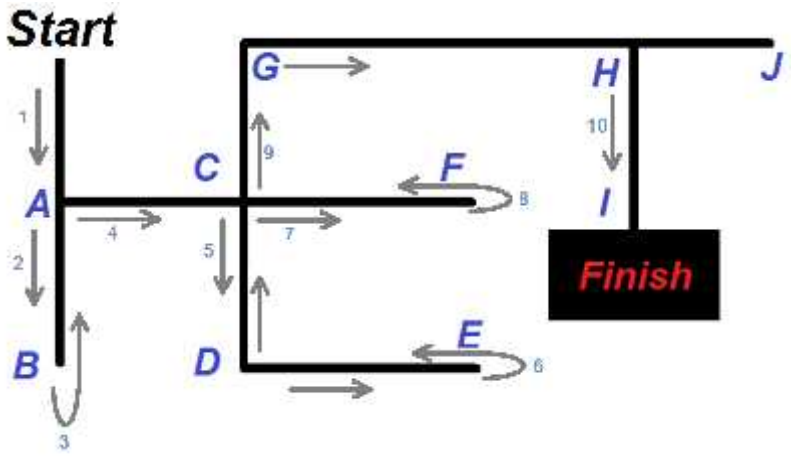


Gambar 2.12*Flowchart*Algoritma*Line Maze Solving* dengan *Right Hand Rule*

Berikut adalah *flowchart* algoritma *line maze solving* dengan *left hand rule*:



Gambar 2.13*Flowchart*Algoritma*Line Maze Solving* dengan *Left Hand Rule*



Gambar 2.14 Contoh Permasalahan *Line Maze* Sederhana

Sebagai contoh berdasarkan gambar 2.11, sebuah *line maze* untuk diselesaikan robot yang menggunakan algoritma *line maze solving* dengan *right hand rule*. Saat robot menemui persimpangan di titik A (lurus atau kiri).Maka robot memilih lurus dan menyimpan dalam memori sebagai "S" (*straight*). Kemudian saat di titik B robot mengidentifikasi bahwa ini adalah jalan buntu lalu melakukan *U-turn* serta menyimpan dalam memori sebagai "U" (*U-turn*). Robot kembali ke titik A dan berdasarkan *rule* maka robot berbelok ke kanan lalu menyimpannya sebagai "R" (*right*). Robot telah merekam "SUR", namun karena U merupakan jalur yang salah maka langkah sebelumnya yaitu S adalah langkah yang salah. Sehingga pada persimpangan pertama robot harus berbelok ke kiri dan menyimpannya sebagai "L" (*left*). Jadi aturan pertama yang didapatkan adalah "SUR" harus diganti dengan "L". Richard T. Vannoy II (2009) menyebut aturan ini sebagai "*Replacement Rules*". Jika dirangkum, maka *replacement rules* pada *right hand rule* adalah mengganti "SUR" dengan "L" dan "RUR" dengan "S". Dan pada *left hand rule* adalah mengganti "SUL" dengan "R" dan "LUL" dengan "S".

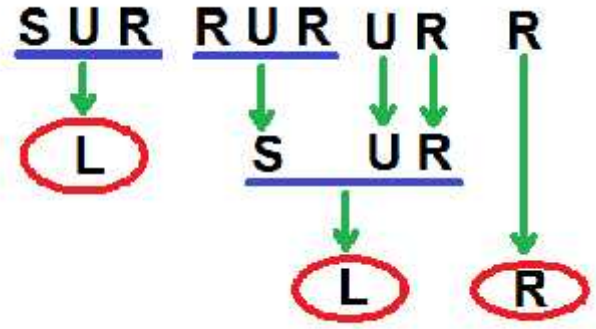
Setelah robot sampai di persimpangan titik C (kanan, lurus, atau kiri). Robot berbelok ke kanan dan menyimpannya sebagai "R" lalu sampai di titik D. Titik D bukan merupakan persimpangan sehingga robot tidak menyimpannya ke dalam memori. Setibanya di titik E, robot mengidentifikasikannya sebagai jalan buntu lalu berputar balik dan menyimpannya sebagai "U". Robot terus bergerak hingga kembali melewati titik D dan langsung berbelok kanan. Ketika robot

kembali ke titik C, robot berbelok ke kanan dan menyimpannya sebagai R. Maka robot sudah merekam "RUR", dan karena U merupakan jalan yang salah jadi langkah sebelumnya yaitu R adalah salah. Sehingga ketika selanjutnya robot menemui persimpangan C, robot akan langsung lurus dan menyimpannya sebagai "S". Maka didapatkan aturan selanjutnya, yaitu "RUR" diganti menjadi "S".

Setelah menyimpan jalur sebelumnya sebagai "S", robot yang bergerak dari titik C menuju titik F menemukan bahwa F adalah jalan buntu. Sehingga robot memutar arah (putar balik) dan menyimpannya sebagai "U". Robot kembali ke titik C dan berbelok ke kanan lalu menyimpannya sebagai "R". Jadi robot sudah merekam "SUR", dan sesuai aturan pertama dalam *replacement rules* maka untuk selanjutnya robot menemui titik C robot akan langsung berbelok ke kiri dan menyimpannya sebagai "L".

Seperti halnya titik D, titik G bukan merupakan sebuah persimpangan sehingga robot tidak menyimpan apapun ke dalam memori. Kemudian robot terus bergerak hingga titik H (kanan atau lurus) dan berbelok ke kanan serta menyimpannya sebagai "R".Dan akhirnya robot sampai di titik I yang merupakan *end of maze*.

Pada penyelesaian ke dua *maze*, robot akan langsung bergerak berdasarkan hasil kalkulasi yang dilakukan setelah proses penyelesaian *maze* pertama kali. Sehingga robot telah mendapatkan jalur terpendek dalam menyelesaikan *maze* tersebut. Penjabaran proses kalkulasinya dapat dilihat pada gambar 2.12.



Gambar 2.15 Gambaran Kalkulasi Jalur yang Telah dilewati

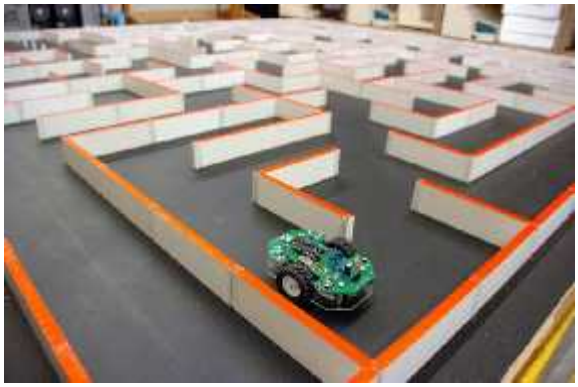
2.4 *Maze*

Menurut Maryani (2013), *maze* adalah tempat yang terdiri dari jalan ataupun lorong yang berliku-liku dan berbelit-belit. Juga dapat didefinisikan sebagai sistem rongga atau saluran yang berhubungan. *Maze* umumnya dikenal sebagai sebuah taman dengan jalan yang sangat berliku-liku hingga sangat rumit untuk menemukan jalan keluar.

Ada dua jenis *maze* yang umum digunakan pada bidang robotika (Maryani, 2013), yaitu:

1. *Wall Maze*

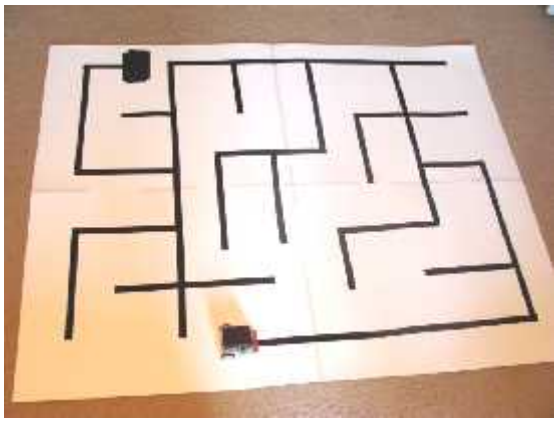
Secara umum dikenal dengan istilah labirin, yaitu rangkaian jalan yang terbentuk atas lorong-lorong tanpa atap. Contoh *wall maze* dapat dilihat pada gambar 2.13.



Gambar 2.16 Contoh *Wall Maze*

2. *Line Maze*

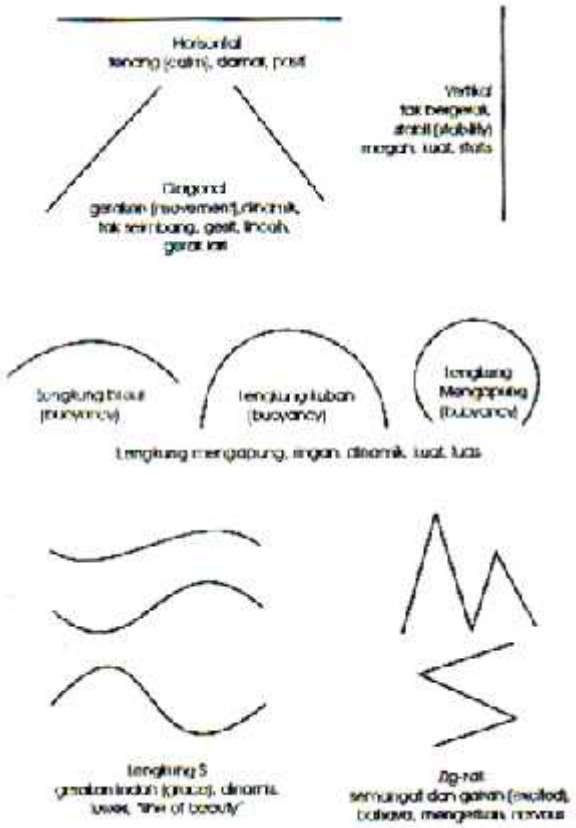
Merupakan rangkaian jalan yang terbentuk dari garis hitam di atas media putih atau sebaliknya. Gambar 2.14 berikut adalah contoh dari *line maze*.



Gambar 2.17 Contoh *Line Maze*

Sanyoto (dalam Santosa, 2008) menjelaskan bahwa secara garis besar garis hanya terdiri atas dua macam, yaitu garis lurus dan garis lengkung. Namun garis terbagi menjadi empat macam jika dirinci lebih khusus, yaitu:

1. Garis lurus yangdapat berupa garis horizontal, diagonal, atau vertikal.
2. Garis lengkung yang dapat berupa garis lengkung kubah, garis lengkung busur, atau garis lengkung mengapung.
3. Garis majemuk yang dapat berupa garis zig-zag atau garis berombak/lengkung S. Sebenarnya garis zig-zag adalah garis-garis lurus dengan arah yang berbeda namun tersambung, dan garis berombak/lengkung S adalah garis-garis lengkung yang saling tersambung.
4. Garis gabungan yang merupakan garis hasil gabungan antara garis lurus, garis lengkung, dan garis majemuk.



Gambar 2.18Gambar Macam, Jenis, dan Karakter Garis