BAB IV

ANALISA DAN PERANCANGAN

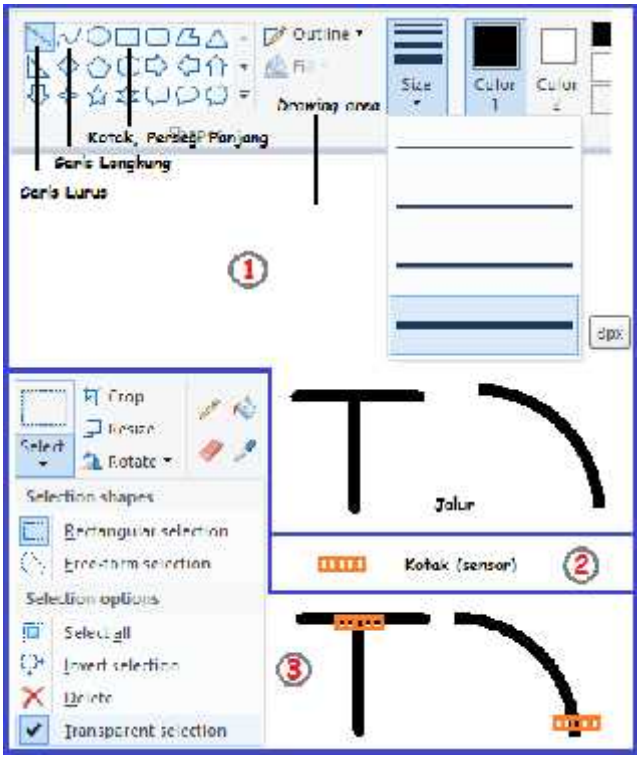
4.1 Analisa Algoritma dan *Maze* Lama

Algoritma *line maze solving* hanya dapat digunakan untuk *maze* dengan jalur lurus dan sudut belokan siku-siku. Sementara pada kenyataan yang ada, masih terdapat banyak bentuk jalur lain. Diantaranya adalah jalur lengkung dan *zig-zag*. Jadi algoritma serta *maze* lama ini masih belum dapat digunakan dengan lebih luas untuk tugas robot di dunia nyata.

4.2 Analisa *Maze Possibilities*

Analisa ini bertujuan untuk mendapatkan berbagai kemungkinan pembacaan dari kelima sensor yang ada terhadap jalur lengkung dan *zig-zag*.Analisa dilakukan dengan menggunakan aplikasi *Paint* yang terdapat pada sistem operasi *Windows*, dapat dilihat pada gambar 4.1. Berikut penjelasan singkatnya:

1. Buat bentuk jalur yang akan dianalisa. Apakah itu garis lurus atau lengkung. Pilih ketebalan garis 8 *pixels*.
2. Buat 5 buah kotak yang bersambungan sebagai sensor yang akan membaca garis.
3. Pada *selection options*(*Home* > *Image* > *Select*), klik *transparent selection*.*Select*gambar kotak (sensor) tersebut. Dan *copy*lalu *paste*, gambar akan muncul pada pojok kiri atas *drawing area*. Geser gambar dengan *mouse* ke arah jalur yang akan dianalisa. Lalu untuk menggeser dengan lebih detail, gunakan panah atas, bawah, kiri, atau kanan pada *keyboard*.



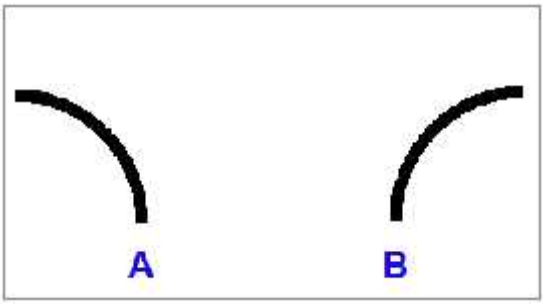
Gambar 4.1 Gambaran Penjelasan Singkat Tahap Analisa dengan *Paint*

4.2.1 *Maze Possibilities* Jalur Lengkung

Pada jalur lengkung terdapat 2*maze possibilities*. Adapun *maze possibilities* tersebut dapat dilihat pada gambar 4.1dengan nama bentuk tipe masing-masingnya adalah sebagai berikut:

A = kiri lingkar

B = kanan lingkar

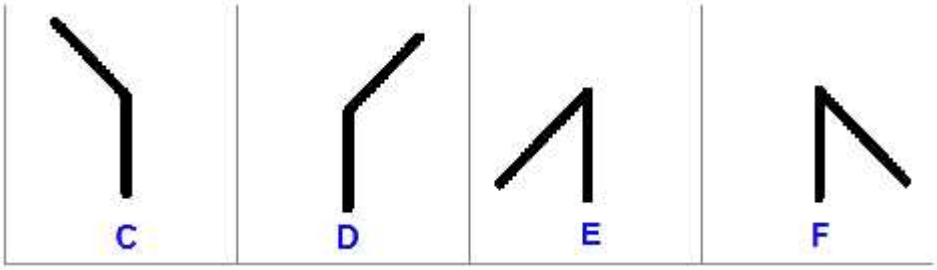


Gambar 4.2Dua*Maze Possibilities* Jalur Lengkung

4.2.2 *Maze Possibilities* Jalur Zig-zag

Dengan syarat kemiringan sudut pergeseran perubahan analisa garis adalah setiap 45°, maka untuk jalur *zig-zag* terdapat 4 macam *maze possibilities*. Hal ini bertujuan agar pembahasan tidak terlalu luas. Sebab jika tidak dibatasi, akan menimbulkan cukup banyak perbedaan hasil pembacaan sensor. Adapun *maze possibilities*tersebut dapat dilihat pada gambar 4.2 dan berikut adalah nama masing-masing tipe tersebut:

C = kiri tumpul
D = kanan tumpul
E = kiri lancip
F = kanan lancip



Gambar 4.3Empat*Maze Possibilities* Jalur *Zig-zag*

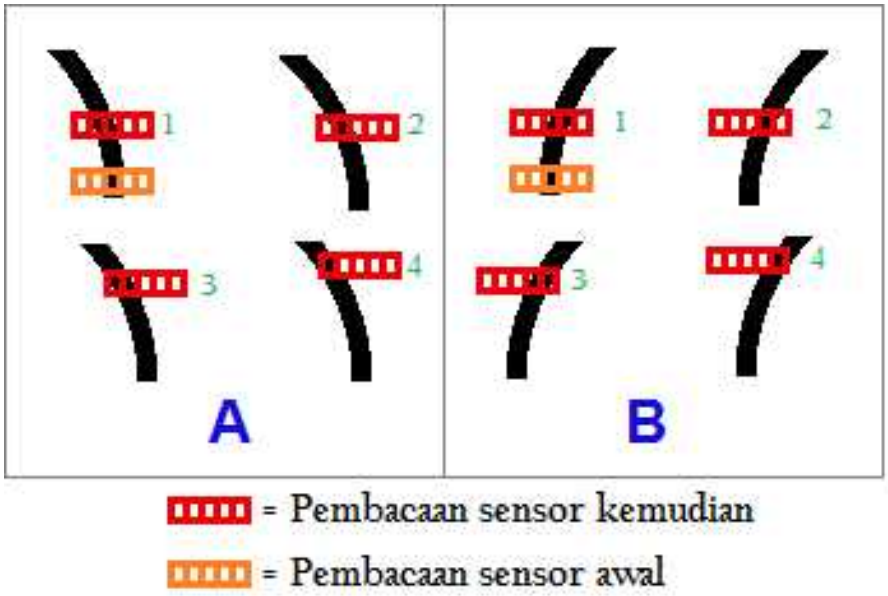
4.3 Analisa dan Perancangan Algoritma Kasus Jalur Lengkung dan *Zig-zag*

Sebelum melakukan perancangan algoritma, diperlukan analisa mengenai pembacaan sensor oleh robot saat berada pada jalur lengkung dan *zig-zag* yang telah dijelaskan sebelumnya. Penulis menjabarkannya sebagai berikut:

4.3.1 Analisa Algoritma

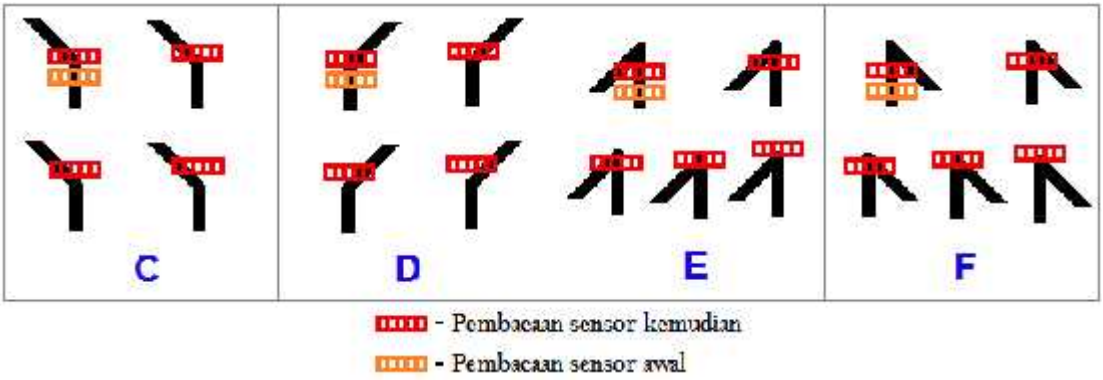
Berikut penjelasan mengenai algoritma atau tahap-tahap pembacaan sensor pada *maze possibilities* yang telah dijelaskan sebelumnya. Saat menemui kemungkinan tipe A dan B pembacaan sensor adalah sama seperti saat robot berjalan pada jalur lurus, yaitu "00100". Tugas robot pun sama seperti saat berjalan lurus di mana robot harus menjaga kestabilan posisi agar pembacaan sensor selalu "00100", baik itu saat menemukan *maze possibilities* tipe A dengan pembacaan "01100", “01000”, “11000”, dan “10000”, atau B dengan pembacaan

"00110", “00010”. “00011”, dan “00001”. Perbedaannya adalah bahwa robot mendapatkan tugas lebih dalam menjaga kestabilan.Apabila robot cukup cepat misalnya setelah berjalan di jalur lurus yang cukup panjang, lalu robot menemui jalur lengkung maka pembacaan sensor bisa hingga menjadi “00000” karena robot keluar jalur. Hal ini juga berlaku pada jalur *zig-zag* dengan sudut tumpul, yaitu *maze possibilities* C dan D. Penggambaran untuk pembacaan sensor terhadap jalur lengkung dapat dilihat pada gambar 4.3 di bawah ini.



Gambar 4.4 Perubahan Pembacaan Sensor Terhadap *Maze Possibilities* Jalur Lengkung

Dan pada jalur *zig-zag*, hal yang perlu dilakukan robot pada *maze possibilities*tipe C dan D adalah sama seperti pada jalur lurus, tipe A, dan tipe B, yaitu menjaga kestabilan posisi pada jalur.Untuk tipe E, kemungkinan-kemungkinan perubahan pembacaannya adalah “10100”, “11100", “01100”, “00100”, dan “00000”.Dan untuk tipe F adalah “00101”, “00111”, “00110”, “00100”, dan “00000”. Pembacaan “10100” dan “00101” tidak ada pada algoritma dari Vannoy II, sehinggaakan diabaikan.Dan kemudian didapatkan pembacaan yang sudah sesuai dengan algoritma *line maze solving*,yaitu “11100” pada tipe E dan “00111” pada tipe F.Penggambaran perubahan pembacaan sensor dapat dilihat pada gambar 4.4.



Gambar 4.5 Perubahan Pembacaan Sensor Terhadap *Maze Possibilities* Jalur *Zig-zag*

4.3.2 Perancangan Algoritma

Setelah didapatkan penggambaran untuk masing-masing alur pembacaan sensor, maka kemudian dilakukan perancangan lebih lanjut.Gambaran umum algoritma *line maze solving* dibagi menjadi dua program. Namun, penulis menyederhanakannya menjadi satu program. Secara rinci akan penulis jelaskan pada bagian ini.

Vannoy tidak menjelaskan mengenai *inchForward()*, maka penulis merancang algoritma untuk subprogram tersebut dengan nama "MajuSedikit()".Algoritma ini berfungsi menggerakkan robot beberapa inchi, beberapa sentimeter, atau beberapa milimeter ke depan. Lalu melakukan pengecekan pembacaan sensorterhadap garis untuk menentukan bahwa robot sedang berada di persimpangan mana dari *maze possibilities* yang dibuat oleh Vannoy II. Sehingga di dalam subprogram ini juga diletakkan perintah untuk membaca sensor. Berikut adalah algoritma tersebut:

```
MajuSedikit()
      Maju
      Delay 150
      BacaSensor
End MajuSedikit
```

Algoritma 4.1SubprogramUntuk Maju Beberapa Sentimeter

Namunalgoritma ini akan menghasilkan jarak tempuh yang berbeda. Sebab dalam waktu yang sama, motor yang digerakkan dengan baterai baru dan yang digerakkan dengan baterai yang sudah lemah akan menghasilkan putaran yang berbeda. Sehingga terkadang perlu dilakukan penyesuaian durasi *delay* dengan kondisi baterai.

Dan karena dalam algoritma subprogram “MajuSedikit” sudah ada perintah untuk membaca sensor, maka “BacaSensor” tidak perlu lagi ditulis jika letaknya setelah “MajuSedikit”. Dan pada saat robot berada di *end of maze*, ditambahkan perintah untuk megnhentikan putaran motor.Jadi rancangan algoritma untuk mengatur pergerakan robot pada saat *mapping* berdasarkan algoritma *line maze solving* dengan*right hand rule* adalah sebagai berikut:

```
Start
      BacaSensor
      If (Pembacaan sensor 00100) Then
            Maju
      ElseIf (Pembacaan sensor 01100) Then
            GeserKiri1
      ElseIf (Pembacaan sensor 01000) Then
            GeserKiri2
      ElseIf (Pembacaan sensor 10000or11000) Then
            GeserKiri3
      ElseIf (Pembacaan sensor 00110) Then
            GeserKanan1
      ElseIf (Pembacaan sensor 00010) Then
            GeserKanan2
      ElseIf (Pembacaan sensor 00011or00001) Then
            GeserKanan3
      Else{Ada persimpangan}
            {Jalan Buntu}
            If (Pembacaan sensor 00000) Then
                  Simpan U
                  PutarBalik
            {Simpang T, +, atau End of maze}
            ElseIf (Pembacaan sensor 11111) Then
                  MajuSedikit
                  {Simpang T}
                  If (Pembacaan sensor 00000) Then
                        Simpan R
                        BelokKanan
                  {End of maze}
                  ElseIf (Pembacaan sensor 11111) Then
                        Berhenti
                        KalkulasiJalurPendek
                        JalankanJalurPendek
                  {Simpang Empat}
                  Else
                        Simpan R
                        Belokkanan
                  EndIf
            {Kanan saja dan Kanan atau Lurus}
            ElseIf (Pembacaan sensor 00111) Then
                  MajuSedikit
                  {Kanan saja}
                  If (Pembacaan sensor 00000) Then
                        BelokKanan
                  {Ada garis, maka Kanan atau Lurus}
                  Else
                        Simpan R
                        BelokKanan
```

```
                EndIf
            {Kiri saja dan Kiri atau Lurus}
            ElseIf (Pembacaan sensor 11100) Then
                MajuSedikit
                {Kiri saja}
                If (Pembacaan sensor 00000) Then
                    BelokKiri
                {Ada garis, maka Kiri atau Lurus}
                Else
                    Simpan S
                    Maju
                EndIf
            EndIf
        EndIf
End
```

Algoritma 4.2 Algoritma Untuk *Mapping* Dengan *Right Hand Rule*

Sebagai penjelasan tambahan, hasil rancangan algoritma dengan menggunakan *left hand rute* dapat dilihat pada gambar A.1 dalam lampiran A. Untuk kalkulasi hasil rekaman jalur, penulis melakukan sedikit perubahan dari algoritma yang dibuat oleh Vannoy II. Pada algoritma yang dibuat oleh Vannoy II, tiga rekaman yang dibaca ini diambil mulai dari rekaman paling akhir. Sedangkan yang penulis buat, dimulai dari satu rekaman sebelum rekaman paling akhir. Sebab penulis menyimpan rekaman terakhir untuk “F” yang merupakan rekaman terhadap *end of maze*.

Algoritma ini berfungsi mengubah setiap tiga rekaman apabila pada rekaman ke-2 dari setiap tiga rekaman yang dibaca adalah “U”. Jika tidak, maka rekaman yang dibaca akan langsung digeser ke satu rekaman lebih awal. Sementara jika ya, maka rekaman pertama dari tiga rekaman tersebut diubah sesuai *replacement rules* yang telah diberikan Vannoy II.Dan rekaman ke-2 dan ke-3 diubah menjadi “O” untuk kemudian diabaikan saat pemilihan jalur sesuai rekaman saat robot menemui persimpangan. Berikut adalah algoritmanya:

```
panjangjalur = integer
KalkulasiJalurPendek()
      int x = (panjangjalur-2) {karena terakhir Fdan tak ada
                                U sebelum F}
      While (x > 0)
            If (jalur[x]=‘U’)Then
                  If jalur[x-1]=’L’ and jalur[x+1]=’L’
                        Jalur[x-1]=’S’
                        Jalur[x]=’O’
                        Jalur[x]+1]=’O’
                  ElseIfjalur[x-1]=’L’ and jalur[x+1]=’S’
                        Jalur[x-1]=’R’
                        Jalur[x]=’O’
```

```
                        Jalur[x]+1]='O'
                    ElseIfjalur[x-1]='R' and jalur[x+1]='R'
                        Jalur[x-1]='S'
                        Jalur[x]='O'
                        Jalur[x]+1]='O'
                    ElseIfjalur[x-1]='R' and jalur[x+1]='S'
                        Jalur[x-1]='L'
                        Jalur[x]='O'
                        Jalur[x]+1]='O'
                    ElseIfjalur[x-1]='S' and jalur[x+1]='L'
                        Jalur[x-1]='R'
                        Jalur[x]='O'
                        Jalur[x]+1]='O'
                    ElseIfjalur[x-1]='S' and jalur[x+1]='R'
                        Jalur[x-1]='L'
                        Jalur[x]='O'
                        Jalur[x]+1]='O'
                    ElseIfjalur[x-1]='L' and jalur[x+1]='R'
                        Jalur[x-1]='U'
                        Jalur[x]='O'
                        Jalur[x]+1]='O'
                    ElseIfjalur[x-1]='R' and jalur[x+1]='L'
                        Jalur[x-1]='U'
                        Jalur[x]='O'
                        Jalur[x]+1]='O'
                    ElseIfjalur[x-1]='S' and jalur[x+1]='S'
                        Jalur[x-1]='U'
                        Jalur[x]='O'
                        Jalur[x]+1]='O'
                    EndIf

                    x--
                Else
                    x--
                EndIf
            EndWhile
End KalkulasiJalurPendek
```

Algoritma 4.3 Algoritma Kalkulasi Hasil Rekaman Jalur

Vannoy tidak menuliskan algoritma untuk menelusuri *maze* setelah didapatkan jalur terpendek, maka penulis membuat algoritmanya berdasarkan algoritma untuk*mapping*. Berikut adalah algoritmanya:

```
JalankanJalurPendek()
    BacaSensor
    If (Pembacaan sensor 00100) Then
        Maju
    ElseIf (Pembacaan sensor 01100) Then
        GeserKiri1
    ElseIf (Pembacaan sensor 01000) Then
        GeserKiri2
    ElseIf (Pembacaan sensor 10000or11000) Then
        GeserKiri3
    ElseIf (Pembacaan sensor 00110) Then
        GeserKanan1
    ElseIf (Pembacaan sensor 00010) Then
        GeserKanan2
    ElseIf (Pembacaan sensor 00011or00001) Then
```

```
            GeserKanan3
      Else{Ada persimpangan}
            {Kanan saja dan Kanan atau Lurus}
            ElseIf (Pembacaan sensor 00111) Then
                  MajuSedikit
                  {Kanan saja}
                  If (Pembacaan sensor 00000) Then
                        BelokKanan
                  {Ada garis, maka Kanan atau Lurus}
                  Else
                        PilihJalur
                  EndIf
            {Kiri saja dan Kiri atau Lurus}
            ElseIf (Pembacaan sensor 11100) Then
                  MajuSedikit
                  {Kiri saja}
                  If (Pembacaan sensor 00000) Then
                        BelokKiri
                  {Ada garis, maka Kiri atau Lurus}
                  Else
                        PilihJalur
                  EndIf
            {Simpang T, +, atau End of maze}
            If (Pembacaan sensor 11111) Then
                  PilihJalur
            EndIf
            JalankanJalurPendek
      EndIf
End JalankanJalurPendek
```

Algoritma 4.4 Algoritma MenelusuriJalur Terpendek

Dalam menelusuri hasil jalur terpendek, akan dilakukan pemilihan jalur mana yang akan ditelusuri saat menjumpai persimpangan. Algoritma ini diperlukan untuk menghilangkan efek dari rekaman “O” terhadap pemilihan jalur. Berikut adalah algoritmanya:

```
bacajalur = integer
PilihJalur()
      Ifjalur[bacajalur]='F'Then
            Berhenti
            Selesai
      ElseIfjalur[bacajalur]='R'Then
            BelokKanan
      ElseIfjalur[bacajalur]='S'Then
            Maju
      ElseIfjalur[bacajalur]='L'Then
            BelokKiri
      ElseIfjalur[bacajalur]='O'Then
            Bacajalur++
            PilihJalur
      EndIf
      Bacajalur++
      JalankanJalurPendek
End PilihJalur
```

Algoritma 4.5 Algoritma Pemilihan Jalur