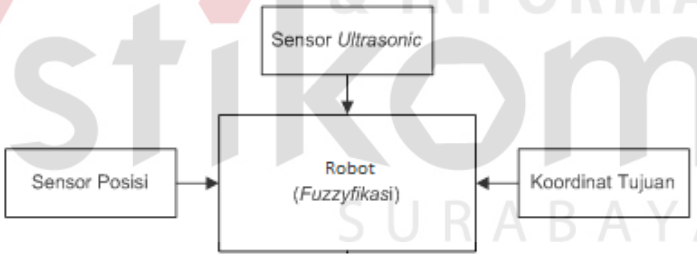
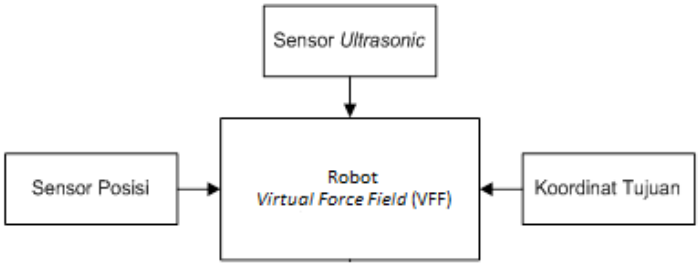
BAB III

METODE PENELITIAN

Metode penelitian yang digunakan meliputi studi kepustakaan dan penelitian laboratorium. Studi kepustakaan merupakan pengumpulan materi atau teori yang berkaitan dengan permasalahan, terutama metode penelitian yang akan digunakan, yang diperoleh dari buku, jurnal, maupun informasi dari internet. Dari informasi studi kepustakaan yang diperoleh, maka dilakukan penelitian laboratorium. Penelitian laboratorium adalah perancangan sistem keseluruhan seperti pada Gambar 3.1 dan Gambar 3.2, uji coba dan pengambilan data sistem yang akan dibahas pada Bab IV.

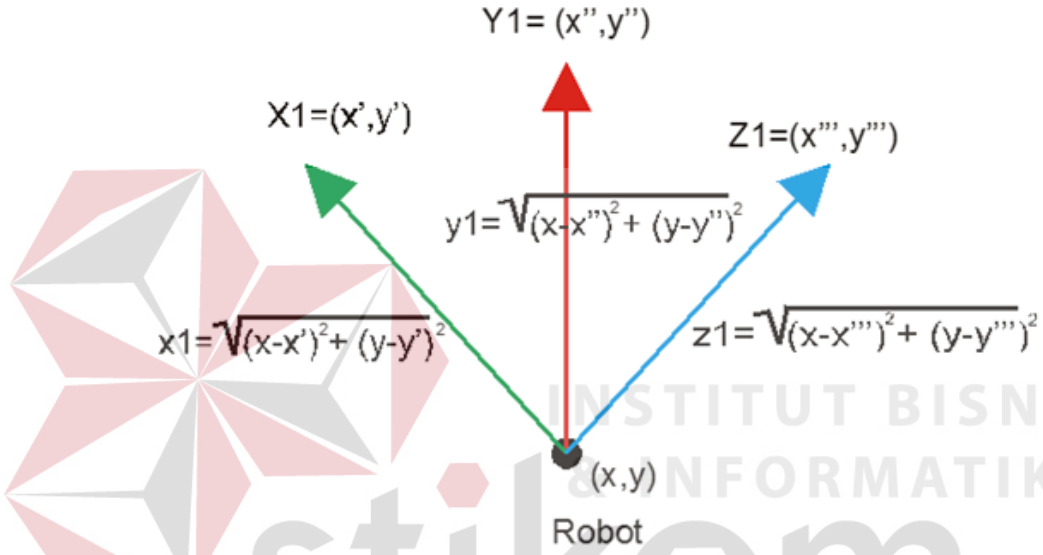


Gambar 3.1. Blok Diagram Sistem Menggunakan Metode *Fuzzy Logic Control*



Gambar 3.2. Blok Diagram Sistem Menggunakan Metode *Virtual Force Field* (VFF)

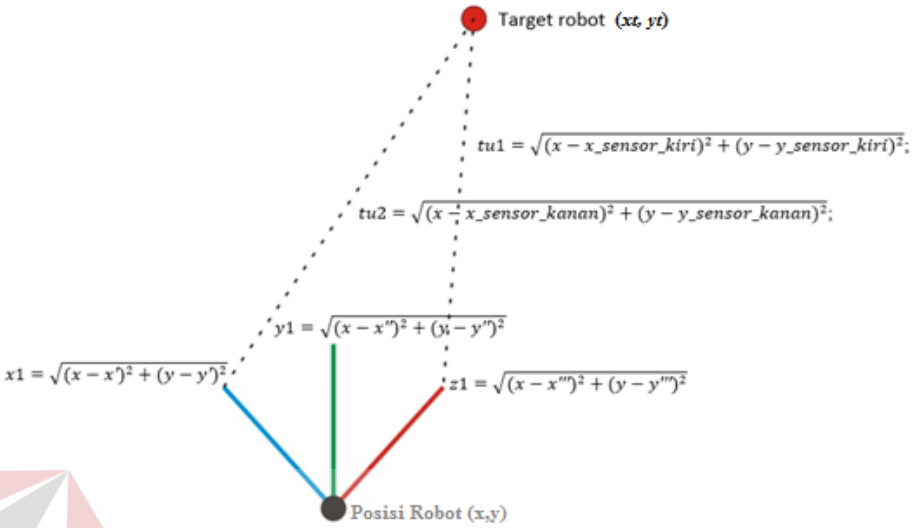
Dari blok diagram pada Gambar 3.1 dan Gambar 3.2 dapat dijelaskan bahwa robot memiliki sensor *ultrasonic*, sensor posisi, dan sensor tujuan sebagai masukan. Dimana masukan-masukan ini akan disimulasikan pada program. Untuk sensor *ultrasonic* yang dimanfaatkan untuk sensor jarak, akan digambarkan sebagai suatu garis yang memiliki koordinat-koordinat dan membentuk suatu vektor garis yang telah ditentukan (x1, y1, z1).



Gambar 3.3. Pembentukan koordinat sensor *Ultrasonic*

Pada gambar 3.3, terdapat garis vektor yang berwarna hijau, merah, dan biru (x1, y1, z1). Garis-garis tersebut merupakan *range* 3 sensor *ultrasonic* yang terdapat pada robot yang nilainya ditentukan menggunakan rumus *theorema phytagoras*. Ketiga garis tersebut akan menghasilkan nilai yang kemudian akan diolah menggunakan metode *Fuzzy Logic Control* dan metode *Virtual Force Field* (VFF), sehingga robot dapat menentukan sudut belok robot ketika terdapat *obstacle*, serta sebagai sensor yang dapat mendeteksi adanya *obstacle*. Selain sensor *ultrasonic*, terdapat pula sensor posisi, digambarkan dengan letak

koordinat (x, y) robot. Koordinat robot digunakan sebagai titik acuan untuk menggerakkan robot.



Gambar 3.4. Pembentukan Garis Bantu Menuju Target

Kemudian masukan selanjutnya adalah koordinat (*xt, yt*) tujuan. Koordinat tujuan adalah koordinat yang akan dituju robot. Untuk menuju ke target, dibuatlah garis bantu (*tu1* dan *tu2*). Nilai dari garis bantu tersebut akan diolah menggunakan metode *Fuzzy Logic Controller* dan *Virtual Force Field* bersama dengan nilai yang terhitung dari sensor *ultrasonic*. Sehingga robot dapat menentukan sudut belok menuju target maupun untuk menghindar ketika menemukan *obstacle*.

3.1 Alat dan Bahan Penelitian

3.1.1 Alat Penelitian

Alat yang digunakan untuk proses pengerjaan penelitian ini adalah *Personal Computer* (PC) / *Laptop*. Alat tersebut bertujuan untuk membuat simulasi 2 dimensi menggunakan aplikasi *Microsoft Visual Studio* 2008.

3.1.2 Bahan Penelitian

Bahan penelitian yang akan diteliti oleh penulis dalam penelitian ini adalah sebagai berikut :

1. Simulasi robot pada *Microsoft Visual Studio* 2008 dapat bergerak di setiap koordinat *layer*.
2. Koordinat target difungsikan sebagai titik acuan arah pergerakan dan pemberhentian pergerakan robot.
3. Pembentukan garis sensor *ultrasonic* pada simulasi sebagai pendeteksi *obstacle*.
4. Kemampuan robot untuk menghindari *obstacle obstacle* yang bergerak.
5. Perbandingan hasil perhitungan sudut belok robot ketika mendeteksi adanya *obstacle* menggunakan metode *Fuzzy Logic Controller* dan *Virtual Force Field* untuk menentukan jarak terpendek dan waktu tercepat menuju target.

3.2 Tahap Penelitian

Langkah-langkah yang dilakukan untuk menyelesaikan penelitian ini dibagi menjadi beberapa bagian yaitu:

3.2.1 Studi Literatur

Merupakan langkah yang bertujuan untuk mencari materi/teori dari buku penunjang, jurnal, artikel, maupun informasi dari internet yang berkaitan dengan permasalahan maupun metode penelitian yang digunakan, sehingga membantu dalam pembuatan sistem.

3.2.2 Perancangan Perangkat Lunak

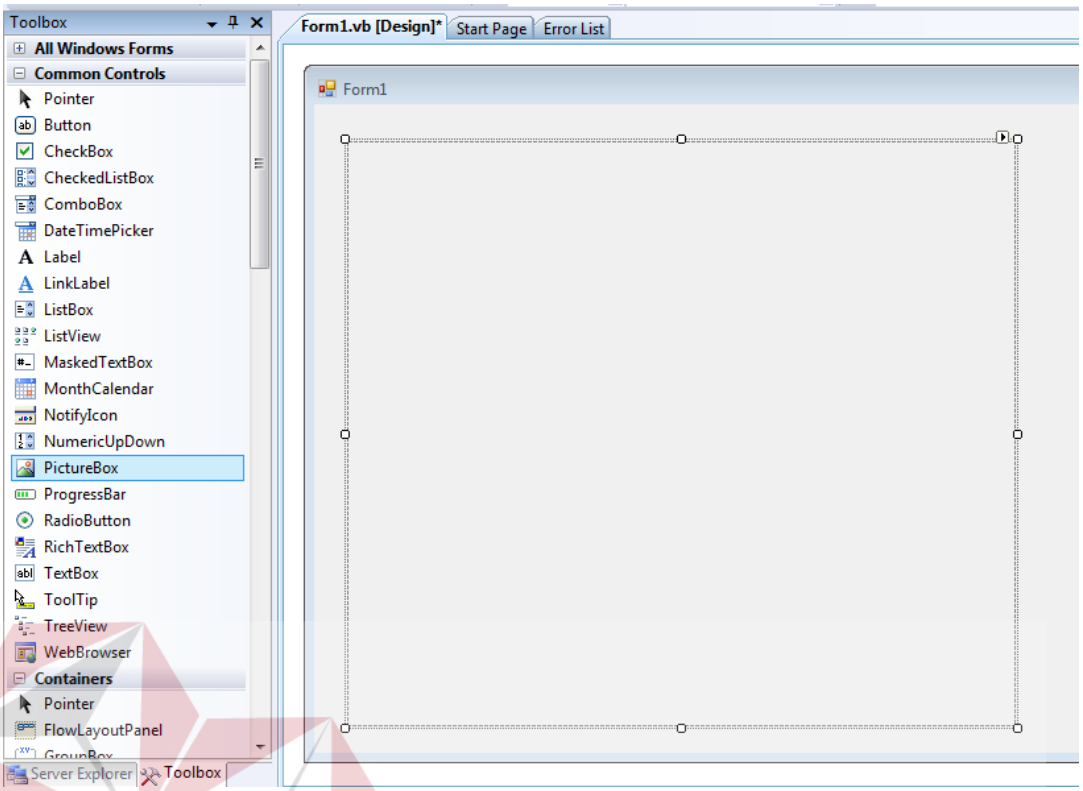
Perancangan perangkat lunak dibagi menjadi beberapa bagian pengerjaan yaitu sebagai berikut :

1. Desain Robot

Pada simulasi ini, robot didesain dengan ukuran 10 x 10 *pixel* dengan skala perbandingan 1 : 5. Sehingga jika simulasi diterapkan di kehidupan nyata, robot berukuran 50 x 50 cm. Kemudian robot mempunyai 3 sensor *ultrasonic* yang dipasang pada bagian depan robot yaitu kiri, tengah, dan kanan. Posisi sensor kiri dan kanan mempunyai sudut 30^0 dari sensor tengah. Setiap sensor didesain dengan jarak baca sensor maksimal sebesar 60 *pixel*. Artinya, dalam kondisi sebenarnya, jarak baca sensor *ultrasonic* maksimal 300 cm. Pada penelitian ini sensor digunakan untuk mengetahui adanya *obstacle* pada sudut dan jarak tertentu dari robot. Hasil tampilan yang diharapkan dapat dilihat pada Gambar 3.8.

2. Perancangan Arena Simulasi

Arena simulasi dibuat dengan menggunakan *picture box* yang merupakan *tools* pada *toolbox* yang disediakan *Microsoft Visual Studio* 2008. *Picture box* ini akan digunakan sebagai penampung gambar arena untuk robot. Gambar yang akan dimasukkan ke dalam *picture box* adalah gambar bertipe file *bitmap*, karena *bitmap* akan dapat dibaca oleh semua program grafis.



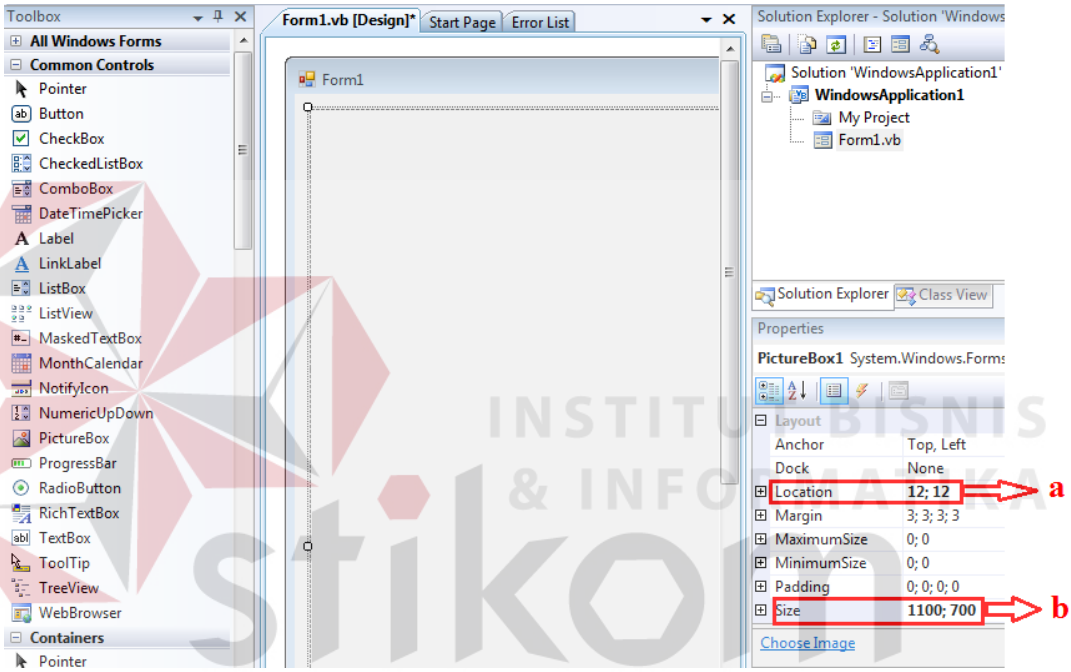
Gambar 3.5. Penggambaran *Picture Box* untuk Arena Simulasi

Hasil dari penggambaran arena menggunakan *picture box* dapat dilihat pada Gambar 3.5. Langkah-langkah penggambaran tersebut adalah sebagai berikut :

1. Setelah *form* siap, arahkan kursor pada panel *Toolbox*.
2. Pilih komponen *Picture Box*.
3. Arahkan kursor pada lembar *form*.
4. Klik pada koordinat tertentu, tahan kemudian *drag* kursor sampai ukuran tententu.

Setelah *picture box* berhasil tergambar pada *form* seperti pada Gambar 3.5, langkah selanjutnya adalah mengatur *properties* dari *picture box* tersebut yang disesuaikan dengan desain penulis. Untuk ukuran arena, penulis telah

mendesain arena dengan ukuran 1100 x 700 *pixel*. Artinya, ukuran sebenarnya adalah 5500 x 3500 cm. Ukuran tersebut sudah cukup untuk pergerakan robot dengan ukuran robot 10 x 10 *pixel* serta sesuai dengan resolusi pada *Personal Computer* (PC) / Laptop. Selain mengatur ukuran, diperlukan pula pengaturan gambar yang akan disisipkan pada *picture box*.

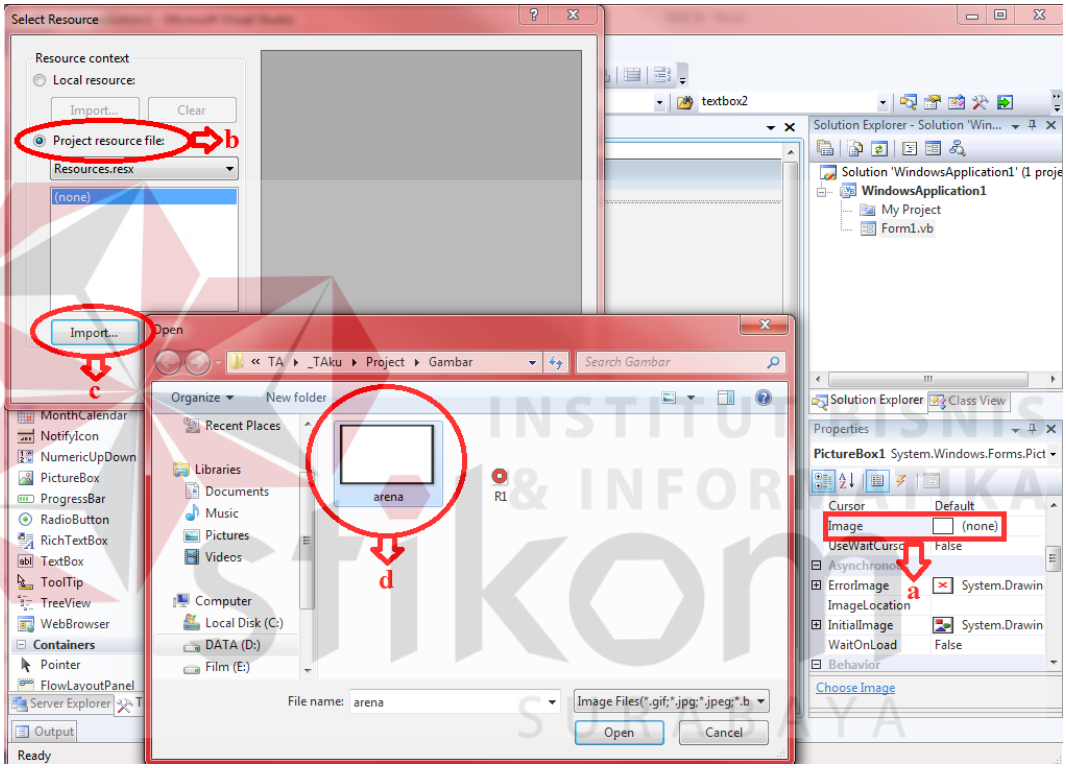


Gambar 3.6. Pengaturan *Location* dan Ukuran *Picture Box*

Penjelasan untuk pengaturan *location* dan ukuran *picture box* dapat dilihat pada Gambar 3.6. *Location* merupakan koordinat (x, y) dimana *picture box* akan diletakkan. Disini, penulis mengatur koordinat *location* untuk *picture box* adalah pada koordinat (12,12). Berikut adalah langkah-langkah untuk pengaturan *location* dan ukuran *picture box* :

1. Klik pada *picture box* yang telah dibuat dalam *form*.
2. Arahkan kursor pada panel *Properties*.

3. Klik pada item *location*, dan ubah nilainya menjadi 12:12 seperti yang ditandai dengan poin "a" pada Gambar 3.6.
4. Kemudian tetap pada *properties picture box*, klik pada item *size*, dan ubah nilainya menjadi 1100:700 seperti yang ditandai dengan poin "b" pada Gambar 3.6.



Gambar 3.7. Pengaturan *Properties Picture Box* untuk Menyisipkan Gambar

Berikut adalah langkah-langkah yang dilakukan untuk menyisipkan gambar pada *picture box* seperti pada Gambar 3.7 :

1. Klik pada *picture box* yang telah dibuat dalam *form*.
2. Arahkan kursor pada panel *Properties*.
3. Pilih *Image* seperti yang ditandai sebagai poin "a" pada Gambar 3.7.

4. Kemudian akan muncul jendela *Select Resources*, pilih *Project resources file* seperti yang ditandai sebagai poin "b" pada Gambar 3.7.
5. Klik tombol *Import* pada jendela *Select Resources* seperti yang ditandai sebagai poin "c" pada Gambar 3.7.
6. Selanjutnya akan muncul jendela *Open* untuk memilih gambar yang akan disisipkan. Masuk pada *directory* letak gambar disimpan, kemudian pilih gambar seperti yang ditandai sebagai poin "d" pada Gambar 3.7. kemudian klik tombol *Open*.

Tampilan arena simulasi yang dimasukkan pada *picture box* ketika program dijalankan dapat dilihat pada Gambar 3.8.

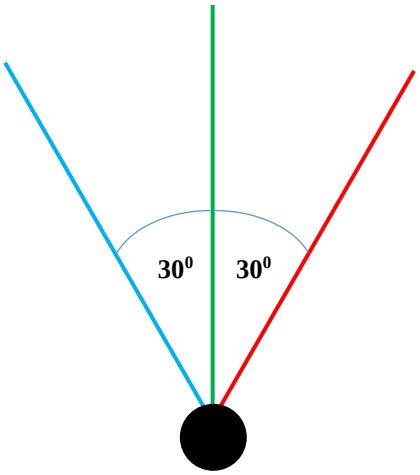
3. Perancangan Robot

Pembuatan robot pada simulasi ini juga menggunakan *picture box* sebagai penampung gambar. Langkah-langkah yang dilakukan untuk menampilkan gambar robot sama dengan langkah-langkah yang digunakn untuk menampilkan gambar arena simulasi. Perbedaannya hanya pada *location* dan ukuran robot. Untuk *location*, penulis meletakkan robot pada koordinat (276,470). Sedangkan untuk ukuran *picture box* yang digunakan untuk gambar robot adalah 10 x 10 *pixel*. Artinya, ukuran sebenarnya adalah 50 x 50 cm. Sehingga ketika program dijalankan, akan terlihat seperti pada Gambar 3.8.

Gambar 3.8. Tampilan Robot pada Arena Simulasi

4. Perancangan Program untuk Sensor *Ultrasonic*

Perancangan program untuk sensor *ultrasonic* merupakan proses penggambaran jarak baca maksimal sensor *ultrasonic* yang disimulasikan dengan pola garis. Garis tersebut memiliki panjang dan sudut tertentu, dimana garis tersebut akan digunakan untuk menghitung nilai jarak yang terbaca oleh sensor *ultrasonic*. Kemudian nilai tersebut akan digunakan sebagai titik acuan untuk menentukan sudut belok robot. Terdapat 3 buah sensor *ultrasonic* yang terpasang pada bagian depan robot. Yaitu sebelah kiri, tengah, dan kanan. Dimana setiap sensor memiliki jarak 30^0. Dan setiap pola garis memiliki panjang 60 *pixel*, yang artinya jarak baca maksimal sensor *ultrasonic* sebenarnya adalah 300 cm.



Gambar 3.9. Penggambaran Sensor *Ultrasonic*

Penggambaran sensor *ultrasonic* pada robot adalah seperti Gambar 3.9, dimana terdapat 3 sensor dengan jarak setiap sensor 30^0. Berikut adalah penulisan program untuk membentuk ketiga sensor tersebut :

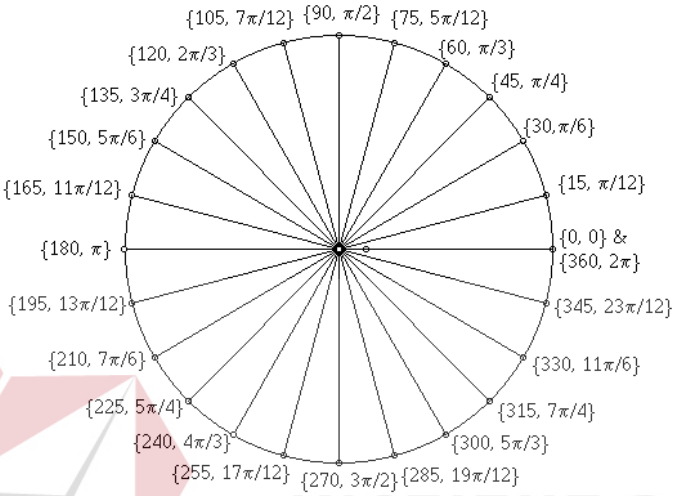
```
Dim TENGAH As Double = VEKTOR(0, X1, Y1, T1) 'GARIS SENSOR DeRAJAT 0
Dim KANAN As Double = VEKTOR(1, X1, Y1, T1) 'GARIS SENSOR DERAJAT 30
Dim KIRI As Double = VEKTOR(2, X1, Y1, T1) 'GARIS SENSOR DERAJAT -30
```

Variable TENGAH, KANAN, dan KIRI adalah *variable* yang digunakan untuk menampung nilai jarak baca sensor. Sedangkan VEKTOR merupakan sebuah *function* yang berisi proses penggambaran jarak baca sensor serta proses perhitungan nilai tiap sensor. Berikut adalah isi program dalam *function* VEKTOR :

```
RS = 0
If ID = 0 Then
    TS = T
End If
If ID = 1 Then
    TS = T + ((30 / 180) * Math.PI)
End If
If ID = 2 Then
```

```
        TS = T - ((30 / 180) * Math.PI)
    End If
```

Program diatas merupakan proses penentuan sudut peletakan sensor menggunakan rumus radian.



Gambar 3.10. Radian Dalam Lingkaran. (http://math.rice.edu/).

Sedangkan untuk mengubah nilai dari radian ke derajat dapat menggunakan rumus 3.1 sebagai berikut:

$$derajat = \frac{radian \; x \; 180^{o}}{\pi} \quad \text{.....(3.1)}$$

Setelah menentukan sudut baca sensor, langkah selanjutnya adalah membuat pola garis baca sensor. Berikut adalah penulisan program pembuatan pola garis baca sensor :

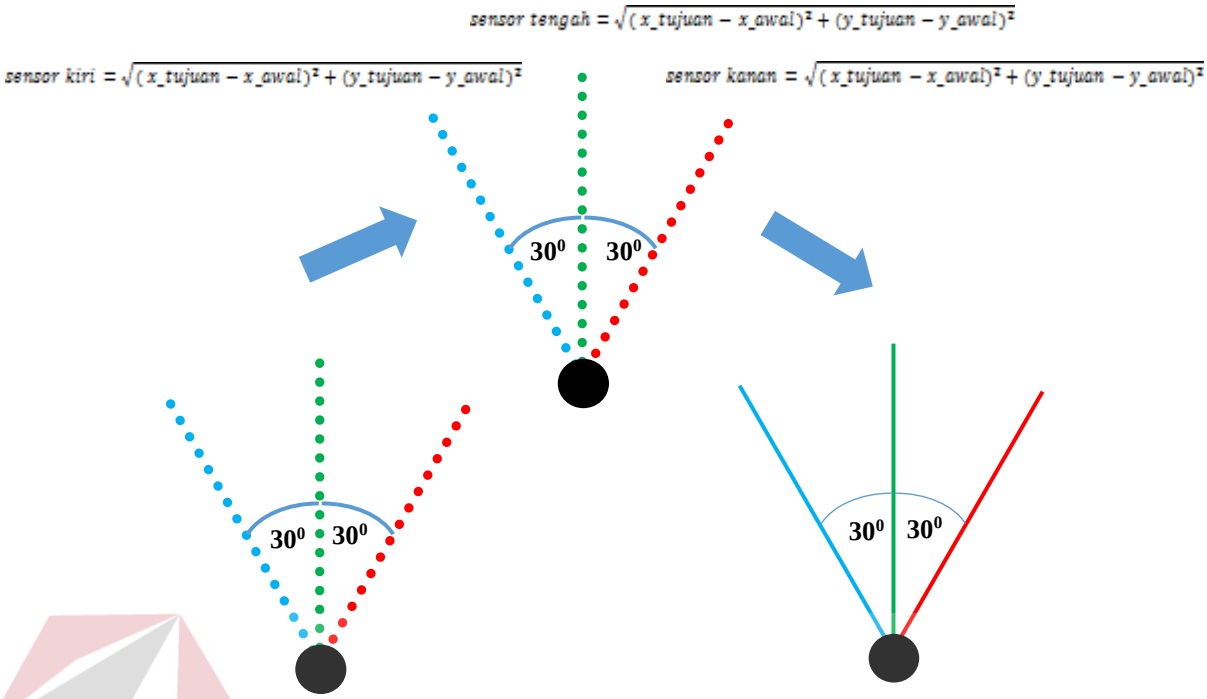
```
Do Until WARNA <> WARNA_REF Or PLUS = 300
        RS = RS + 0.2
        XS = X + RS * Math.Cos(TS)
        YS = Y + RS * Math.Sin(TS)
        WARNA = ARENA.GetPixel(CInt(XS), CInt(YS))
```

```
        PLUS = PLUS + 1
Loop
PLUS = 0
VEKTOR = Math.Sqrt((XS - X) * (XS - X) + (YS - Y) * (YS - Y))
```

Program diatas merupakan proses pembentukan pola garis jarak baca sensor, pola garis dibentuk berdasarkan titik-titik koordinat yang ditentukan menggunakan rumus *trigonometri*. Penarikan garis dilakukan setiap 0.2 *pixel* dan diulang selama 300 kali. Sehingga, panjang garis akan sama dengan 60 *pixel*. Untuk mengetahui jarak baca masing-masing sensor, digunakan rumus *phytagoras* seperti pada rumus 2.10. Sedangkan untuk menampilkan pola garis jarak baca sensor, menggunakan penulisan program seperti berikut :

```
If ID = 0 Then
        G.DrawLine(New Pen(Color.Red), CSng(X), CSng(Y), CSng(XS), CSng(YS))
End If
If ID = 1 Then
        G.DrawLine(New Pen(Color.Green), CSng(X), CSng(Y), CSng(XS), CSng(YS))
End If
If ID = 2 Then
        G.DrawLine(New Pen(Color.Blue), CSng(X), CSng(Y), CSng(XS), CSng(YS))
End If
```

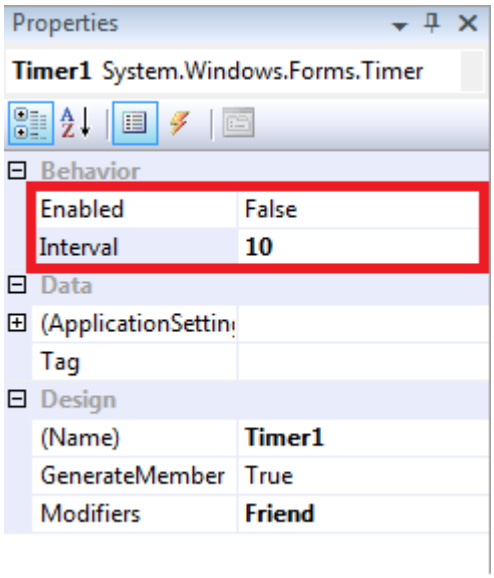

Perbedaan warna pada penggambaran garis sensor hanya dilakukan untuk mempermudah penulis untuk menyelesaikan penelitian. Proses keseluruhan penggambaran sensor *ultrasonic* dapat dilihat pada Gambar 3.11.



Gambar 3.11. Proses Penggambaran Sensor *Ultrasonic*

5. Perancangan Program Pergerakan Robot

Agar dapat menggerakkan robot, dibutuhkan komponen *Timer* yang digunakan sebagai penghitung waktu secara otomatis yang disediakan oleh *Microsoft Visual Studio* 2008. Komponen t*imer* terletak pada panel *toolbox*, sama seperti *picture box*. Ada beberapa *properties* yang harus disesuaikan, seperti yang terlihat pada Gambar 3.12 berikut.



Gambar 3.12. Tampilan *Properties Timer*

Properties pada *timer* yang perlu disesuaikan adalah :

1. *Enabled* diubah menjadi *False*, ini dimaksudkan agar pada awal program dijalankan, *timer* tidak langsung berjalan.
2. Nilai *interval* diubah menjadi 10 ms atau 0.01 detik. Nilai *interval* dibuat sekecil mungkin agar pergerakan robot terlihat lebih halus.

Berikut merupakan penulisan program untuk menggerakkan robot menggunakan *timer* :

```
X = X + 1 * Math.Cos(TR)
Y = Y + 1 * Math.Sin(TR)
```

Program untuk menggerakkan robot ditulis di dalam *function timer*. Karena ketika status *timer* berubah menjadi *enabled*, program yang dijalankan adalah program yang berada dalam *function timer*. Untuk menggerakkan robot, dibuatlah titik-titik koordinat yang ditentukan melalui rumus *trigonometri* seperti

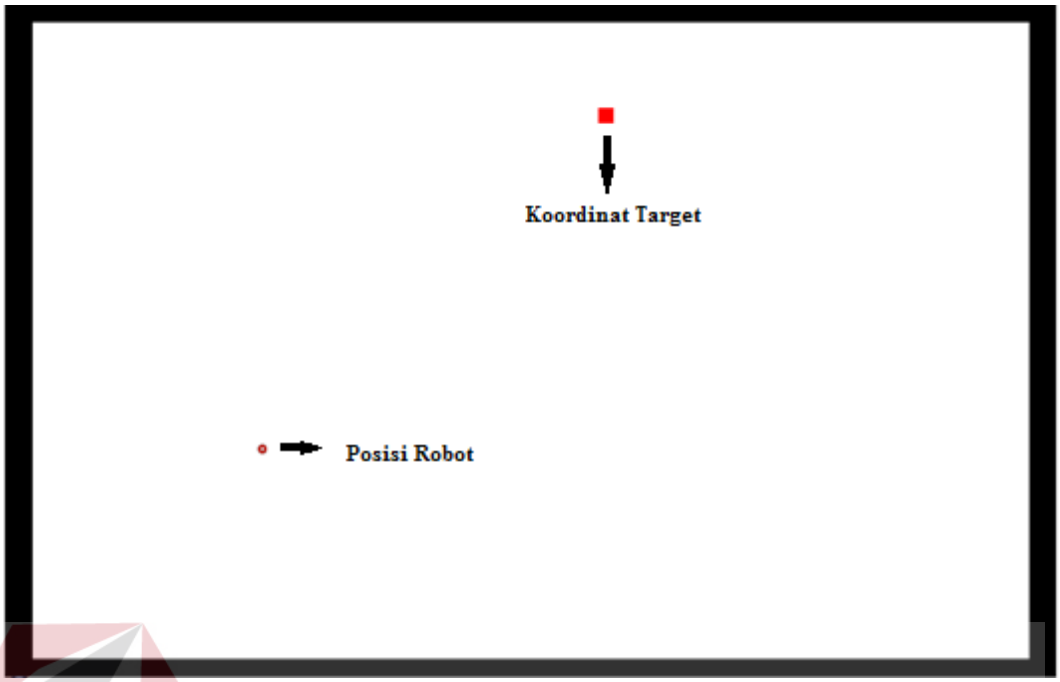
halnya membuat titik-titik pada pola garis jarak baca sensor *ultrasonic*. Sehingga robot dapat bergerak mengikuti titik-titik koordinat tersebut.

6. Perancangan Program untuk Menentukan Koordinat Target Robot dan Koordinat *Obstacle*

Koordinat target yang dituju robot ditentukan oleh *user*, namun hanya bisa dilakukan sekali saja. Yaitu dengan cara mengarahkan kursor ke koordinat tertentu, kemudian klik kiri pada *mouse*, maka akan terbentuk objek yang merupakan target yang dituju robot. Berikut adalah penulisan program untuk menentukan koordinat target :

```
If e.Button = Windows.Forms.MouseButtons.Left Then
    c = Color.Red
End If
If e.Button = Windows.Forms.MouseButtons.Right Then
    c = Color.White
End If

If c <> Color.Green Then
    PictureBox1.Image = ARENA.Clone
    TextBox6.Text = CInt(e.X)
    TextBox5.Text = CInt(e.Y)
    XT = CInt(TextBox6.Text)
    YT = CInt(TextBox5.Text)
    PictureBox3.BackColor = Color.Red
    PictureBox3.Location = UPDATE_LOKASI(XT, YT)
    TUJUAN = 1
End If
```

Ketika program dijalankan, hasilnya akan terlihat seperti pada Gambar 3.13.



Gambar 3.13. Penggambaran Koordinat Target

Selanjutnya adalah menentukan koordinat *obstacle* pada jalur yang akan dilalui oleh robot. Sehingga robot mendapatkan kendala dalam proses bergerak menuju target, selain itu *obstacle* juga digunakan sebagai sarana untuk mengetahui respon robot terhadap *obstacle*. Berikut adalah penulisan program untuk menentukan koordinat target dan *obstacle* diam :

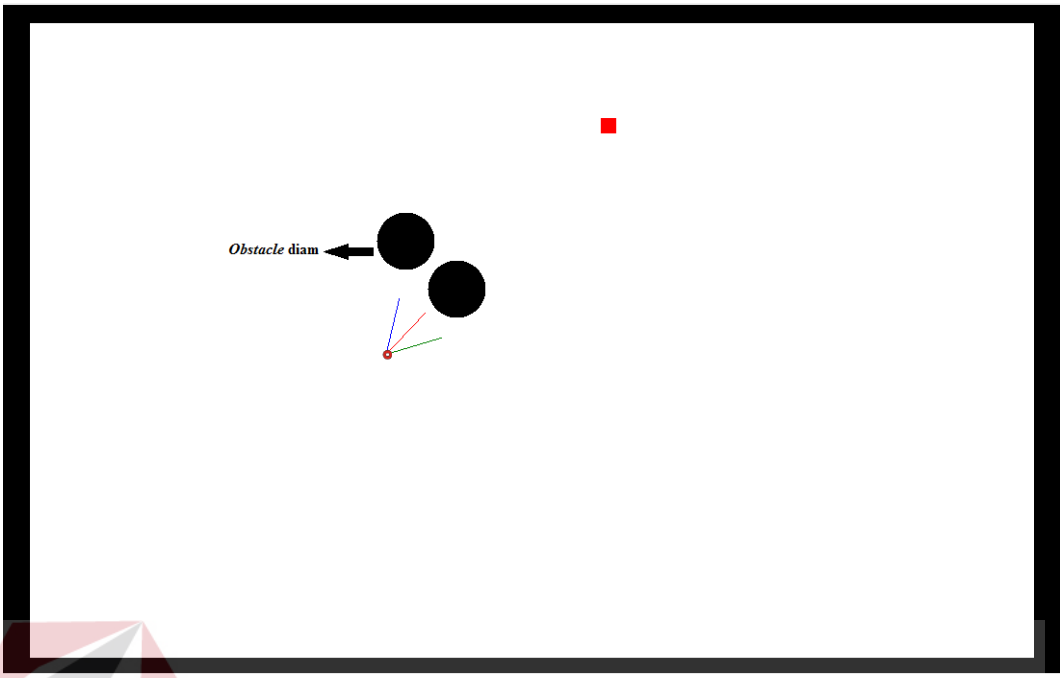
```
        If e.Button = Windows.Forms.MouseButtons.Right Then
                c = Color.Black
            End If
            If e.Button = Windows.Forms.MouseButtons.Left Then
                c = Color.White
            End If

            If c <> Color.Green Then
                g.FillEllipse(New SolidBrush(c), New Rectangle(e.X
- 10, e.Y - 10, 60, 60))
                XO(j) = e.X
                YO(j) = e.Y
                g.FillEllipse(New SolidBrush(d), New Rectangle(e.X
- 10, e.Y - 10, 60, 60))
                g.DrawImage(ARENA, 0, 0)
                PictureBox1.Image = ARENA.Clone
                TUJUAN = 3
                j = j + 1
```

```
End If
```

Proses pembuatan target dan *obstacle* diam ini diletakkan pada *event mouse move* yang dimiliki oleh *picture box* yang menampung gambar arena simulasi. *Mouse move* merupakan *event* yang akan aktif ketika *mouse* melewati *picture box*.

. Untuk menentukan koordinat *obstacle* diam, caranya hampir sama dengan menentukan koordinat terget. Namun, *user* hanya dapat menentukan koordinat *obstacle* diam apabila koordinat target sudah ditentukan. Dengan cara mengarahkan kursor ke koordinat yang diinginkan, kemudian klik kanan pada *mouse*. Apabila *user* menekan klik kanan pada *mouse*, maka akan tercipta sebuah objek berwarna hitam. Objek inilah yang disimulasikan sebagai *obstacle* diam. Sedangkan apabila *user* klik kiri pada *mouse*, akan tercipta objek berwarna putih. Tidak akan terlihat karena arena simulasi juga berwarna putih. Sehingga ini berfungsi sebagai penghapus *obstacle*.

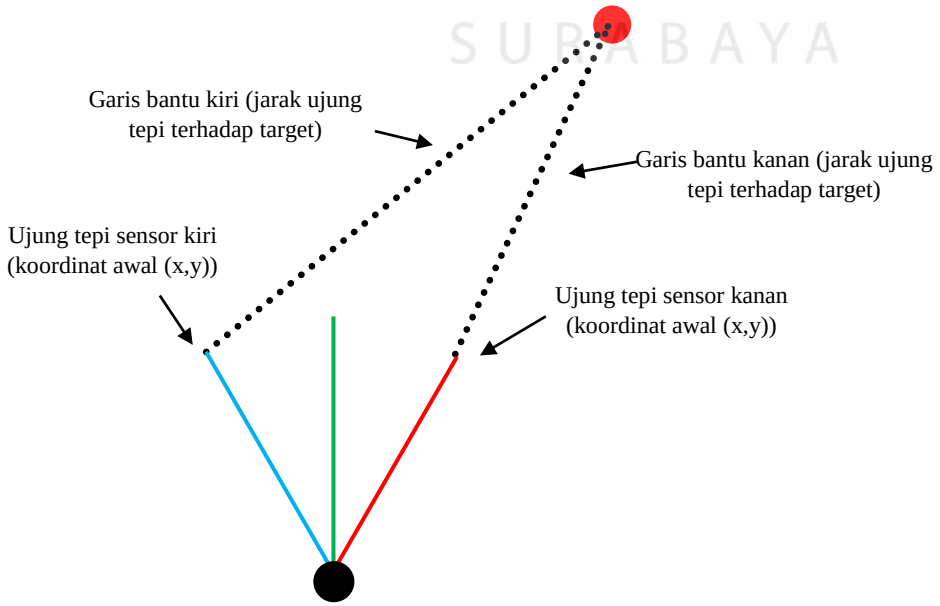
Ketika program dijalankan, hasilnya akan terlihat seperti pada Gambar 3.14.



Gambar 3.14. Penggambaran Koordinat *Obstacle*

7. Perancangan Program untuk Menentukan Arah yang Dituju Robot

Untuk menentukan arah yang dituju robot, penulis memanfaatkan jarak dari koordinat tepi sensor (sensor *ultrasonic* kanan dan kiri) menuju koordinat target. Untuk lebih jelasnya dapat dilihat pada Gambar 3.15 berikut.



Gambar 3.15. Garis Bantu Menuju Target

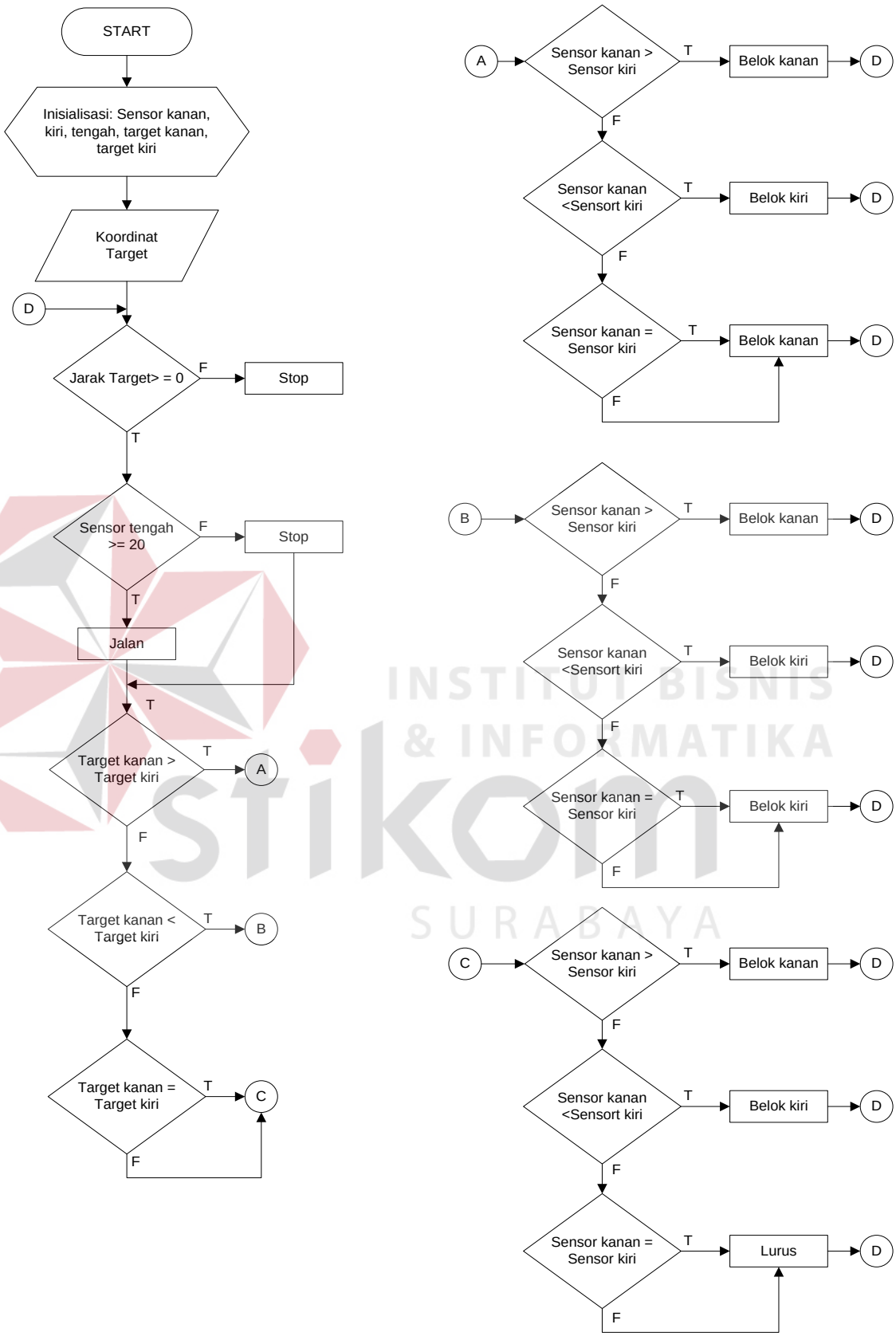
Berikut adalah penulisan program untuk membuat garis bantu menuju target :

```
G = Graphics.FromImage(DIS_ARENA)
If ID = 1 Then
   G.DrawLine(New Pen(Color.Green), CSng(X1), CSng(Y1), CSng(XS1), CSng(YS1))
   PTR = TARGET1(1, XS1, YS1)
End If
 If ID = 2 Then
    G.DrawLine(New Pen(Color.Blue), CSng(X1), CSng(Y1), CSng(XS1), CSng(YS1))
    PTL = TARGET1(2, XS1, YS1)
 End If

Private Function TARGET1(ByVal ID As Integer, ByVal X1 As Double, ByVal Y1 As
Double) As Double
      TARGET1 = Math.Sqrt((XT - X1) * (XT - X1) + (YT - Y1) * (YT - Y1))
End Function
```

Cara yang digunakan untuk menarik garis dari setiap ujung sensor kanan dan kiri menuju target adalah menggunakan rumus *theorema phytagoras*. Setiap ujung sensor akan memiliki nilai koordinat x dan y. Nilai tersebut yang akan dimasukkan ke perhitungan dalam *function* TARGET1 sebagai X1 dan Y1. Sementara XT dan YT adalah nilai koordinat x dan y dari target.

8. Perancangan Program untuk Menuju Target dan Menghindari Halangan

Perancangan program ditulis dalam bentuk *flowchart* yang dapat dilihat pada Gambar 3.16 berikut.



Gambar 3.16. *Flowchart* Program untuk Menuju Target dan Menghindari Halangan

Ketika simulasi dijalankan, pertama kali robot melakukan inisialisasi (konfigurasi sensor dan penentuan koordinat tujuan). Setelah inisialisasi, diperoleh nilai yang digunakan sebagai penggerak dan penentuan arah belok robot. Jika jarak dari posisi robot menuju target masih jauh maka robot akan memprioritaskan sensor tengah sebagai acuan pergerakan robot. Robot akan memeriksa arah robot terhadar target tujuan menggunakan jarak dari ujung sensor kanan dan kiri terhadap koordinat target, ketika terdapat *obstacle* robot akan lebih memprioritaskan sensor kanan dan kiri untuk menentukan arah belok robot guna menghindari *obstacle*. Setelah robot dapat menghindari *obstacle*, robot akan kembali menentukan arah dengan menggunakan parameter jarak antara ujung tepi sensor terhadap koordinat target (target kanan dan target kiri). Berikut merupakan program untuk arah target dan proses penghindaran *obstacle*:

Program menggunakan metode *Fuzzy* :

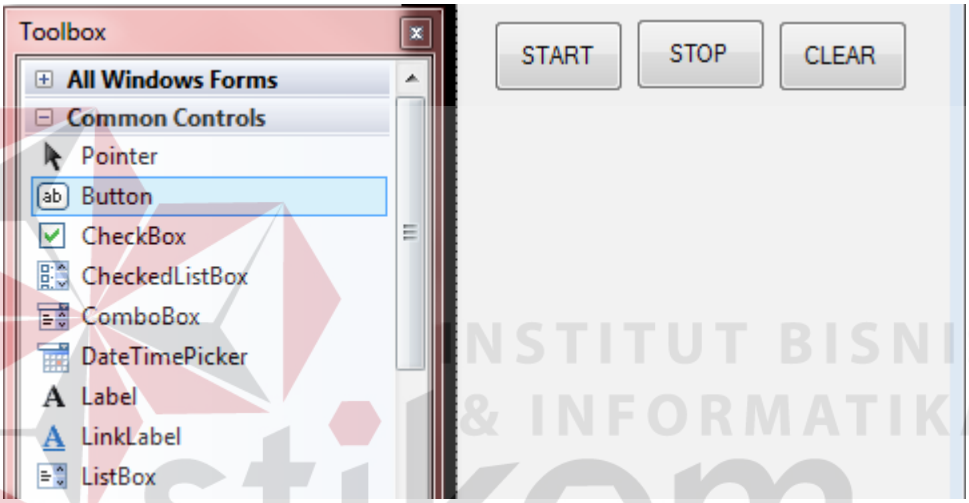
```
eror1 = KANAN1 - KIRI1
eror11 = PTL1 - PTR1
belok1 = belok1 + fuzzy_eror(eror1, eror11)
TR1 = (Math.PI / 350) * belok1
XR1 = XR1 + KECEPATAN1 * Math.Cos(TR1)
YR1 = YR1 + KECEPATAN1 * Math.Sin(TR1)
```

Program menggunakan metode VFF :

```
eror1 = KANAN1 - KIRI1
eror11 = PTL1 - PTR1
belok1 = belok1 + Math.Atan(eror1) + Math.Atan(eror11)
TR1 = (Math.PI / 350) * belok1
XR1 = XR1 + KECEPATAN1 * Math.Cos(TR1)
YR1 = YR1 + KECEPATAN1 * Math.Sin(TR1)
```

9. Perancangan Program Kontrol Sistem

Perancangan program kontrol sistem merupakan perancangan program untuk mengontrol jalannya simulasi. Dimana simulasi akan dikontrol oleh 3 tombol. Diantaranya adalah tombol *START*, *STOP*, dan *CLEAR*. Pembuatan 3 tombol tersebut memanfaatkan komponen *button* yang disediakan oleh *Microsoft Visual Studio* 2008.



Gambar 3.17. Letak Komponen *Button* dan Penempatan pada *Form*

Proses pembuatan 3 tombol pada Gambar 3.17 sama dengan pembuatan *picture box*, yakni komponen *button* diambil dari panel *toolbox* kemudian di klik pada *form*. Berikut adalah fungsi dari masing-masing tombol :

a. Tombol *START*

Tombol *START* merupakan tombol yang digunakan untuk memulai menjalankan robot. Berikut adalah penulisan program pada tombol *START* :

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
```

```
        Timer1.Enabled = True
End Sub
```

Terlihat bahwa ketika tombol START di tekan, maka *Timer1* dijalankan. Dimana program untuk menjalankan robot telah ditulis di dalam *function Timer1*. Sehingga, ketika tombol *START* ditekan, maka robot akan berjalan sesuai dengan *interval* yang diatur pada *Timer1*.

b. Tombol *STOP*

Tombol *STOP* merupakan tombol yang digunakan untuk menghentikan jalannya robot. Berikut adalah penulisan program pada tombol *STOP* :

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Timer1.Enabled = False
End Sub
```

Tombol *STOP* adalah kebalikan dari tombol *START*. Jika di dalam tombol *START* terdapat perintah mengaktifkan *Timer1*, maka di dalam tombol *STOP* terdapat perintah untuk menon-aktifkan *Timer1*. Sehingga ketika tombol *STOP* ditekan, maka robot akan berhenti bergerak.

c. Tombol *CLEAR*

Tombol *CLEAR* berfungsi sebagai *reset*, yaitu menghapus seluruh objek yang berada di arena, kecuali robot. Serta mengembalikan sistem ke tahap awal seperti program pertama dijalankan. Namun, posisi robot tetap pada koordinat pada saat tombol *CLEAR* ditekan. Berikut adalah penulisan program pada tombol *CLEAR* :

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Picture box1.Image = Picture box1.InitialImage
        PictureBox5.BackColor = Color.White
        ARENA = Picture box1.Image.Clone
        DIS_ARENA = ARENA.Clone
End Sub
```

Picturebox1 kembali diisi dengan gambar inisialisasi, dimana gambar inisialisasi adalah gambar arena yang masih kosong tanpa koordinat target. Kemudian *picturebox5* yang merupakan *picturebox* yang menampung *obstacle* diubah menjadi warna putih yang berarti menghapus *obstacle*.

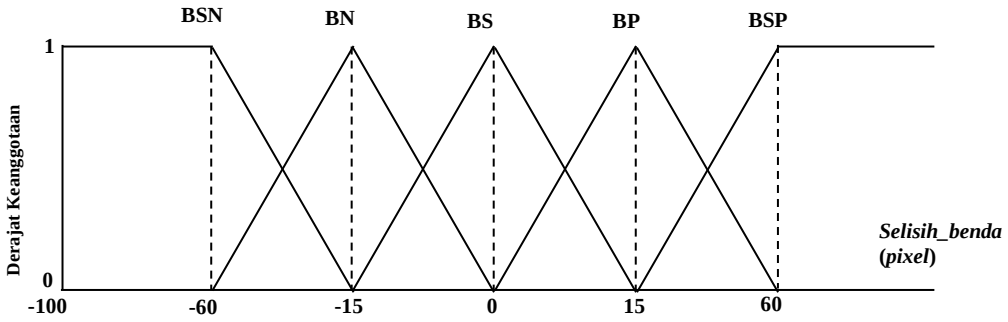
10. Perancangan Program untuk Metode *Fuzzy Logic Controller*

Metode *Fuzzy Logic Controller* adalah metode yang mempunyai *rule*, dimana *rule* tersebut dapat didesain sendiri oleh peneliti. Sehingga sistem dapat berjalan sesuai dengan konsep pemikiran yang dituangkan peneliti ke dalam algoritma *Fuzzy Logic Controller*. Dalam simulasi ini, *Fuzzy Logic Controller* dimanfaatkan sebagai metode penelitian untuk menentukan nilai sudut belok robot untuk berjalan menuju target, maupun untuk menghindari *obstacle*. Terdapat toga langkah dalam pengerjaan metode *Fuzzy Logic Controller*, diantarana sebagai berikut :

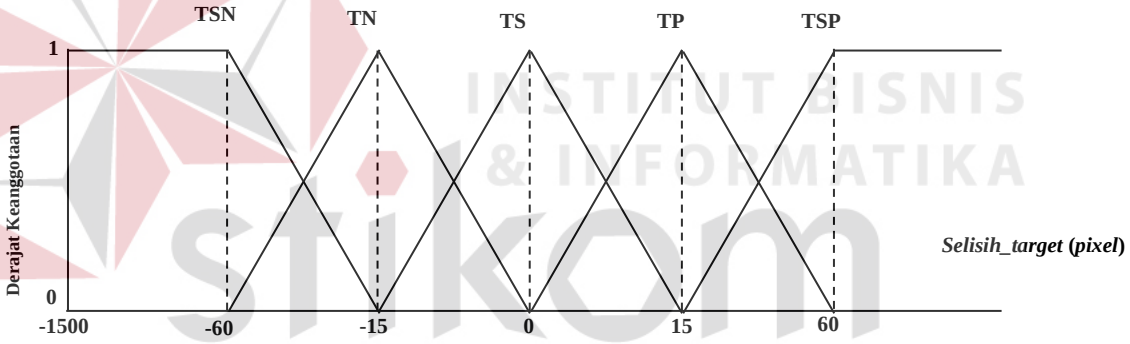
A. Fuzzifikasi

Fuzzifikasi adalah proses mengubah nilai-nilai *input* yang bersifat pasti (*crisp set*) ke dalam bentuk *fuzzy input*. Ada dua *input* untuk menentukan arah belok robot, yaitu selisih nilai jarak sensor *ultrasonic* kanan dan kiri

(*selisih_benda*) dan selisih jarak dari ujung sensor *ultrasonic* kanan dan kiri menuju target (*selisih_target*). Berikut adalah fungsi keanggotaan *selisih_benda* dan *selisih_target* :



Gambar 3.18. Fungsi Keanggotaan *Input* Selisih (Delta) Jarak Sensor



Gambar 3.19. Fungsi Keanggotaan *Input* Selisih (Delta) Jarak Target

Keterangan :

BSN	= Benda Sangat Negatif	TSN	= Target Sangat Negatif
BN	= Benda Negatif	TN	= Target Negatif
BS	= Benda Sedang	TS	= Target Sedang
BP	= Benda Positif	TP	= Target Positif
BSP	= Benda Sangat Positif	TSP	= Target Sangat Positif

Langkah perhitungannya adalah dengan menghitung selisih antara jarak dari ujung sensor *ultrasonic* kanan dan kiri menuju target. Kemudian menghitung selisih nilai jarak yang terbaca sensor *ultrasonic* kanan dan kiri. Selanjutnya, kedua hasil tersebut akan dipetakan dalam fungsi keanggotaan *Fuzzy*. Berikut adalah penulisan perhitungan dalam program :

```
selisih_benda = KANAN - KIRI
selisih_target = TKIRI - TKANAN
```

Setelah mendapatkan kedua nilai *input* untuk *Fuzzy*, langkah selanjutnya adalah menghitung nilai derajat keanggotaan dari masing-masing fungsi keanggotaan. Berikut adalah penulisan program untuk proses perhitungan nilai derajat keanggotaan :


```
'--------selisih_benda----------
BSN = FUZZYFIKASI(-100, -70, -60, -15, selisih)
BN = FUZZYFIKASI(-60, -15, -15, 0, selisih)
BS = FUZZYFIKASI(-15, 0, 0, 15, selisih)
BP = FUZZYFIKASI(0, 15, 15, 60, selisih)
BSP = FUZZYFIKASI(15, 60, 70, 100, selisih)

'----------selisih_target----------
TSN = FUZZYFIKASI(-1500, -1500, -60, -15, dselisih)
TN = FUZZYFIKASI(-60, -15, -15, 0, dselisih)
TS = FUZZYFIKASI(-15, 0, 0, 15, dselisih)
TP = FUZZYFIKASI(0, 15, 15, 60, dselisih)
TSP = FUZZYFIKASI(15, 60, 1500, 1800, dselisih)

Private Function FUZZYFIKASI(ByVal A As Double, ByVal B As Double, ByVal C As
Double, ByVal D As Double, ByVal X As Double) As Double
      If X <= A Then
```

```
            FUZZYFIKASI = 0
      ElseIf A <= X And X <= B Then
            FUZZYFIKASI = (X - A) / (B - A)
      ElseIf B <= X And X <= C Then
            FUZZYFIKASI = 1
      ElseIf C <= X And X <= D Then
            FUZZYFIKASI = (D - X) / (D - C)
      ElseIf D <= X Then
            FUZZYFIKASI = 0
      End If
End Function
```

B. *Rule Set*

Jumlah *variable* yang digunakan untuk membuat *Fuzzy* belok berjumlah dua, dan masing-masing *variable* memiliki lima himpunan *fuzzy* dalam fungsi keanggotaannya. Sehingga, terdapat dua puluh lima aturan. Untuk lebih jelasnya dapat dilihat pada Tabel 3.1 :

Tabel 3.1. *Rule Set* Arah Belok

		Selisih_target				
selisih_benda		**TSN**	**TN**	**TS**	**TP**	**TSP**
	BSN	KIB	KIB	KIB	KIB	KIB
	BN	KIB	KIB	KIS	KIS	KIS
	BS	KIS	KIS	T	KAS	KAS
	BP	KAS	KAS	KAS	KAB	KAB
	BSP	KAB	KAB	KAB	KAB	KAB

Keterangan :

KIB = Kiri Besar

KIS = Kiri Sedang

T = Tengah

KAS = Kanan Sedang

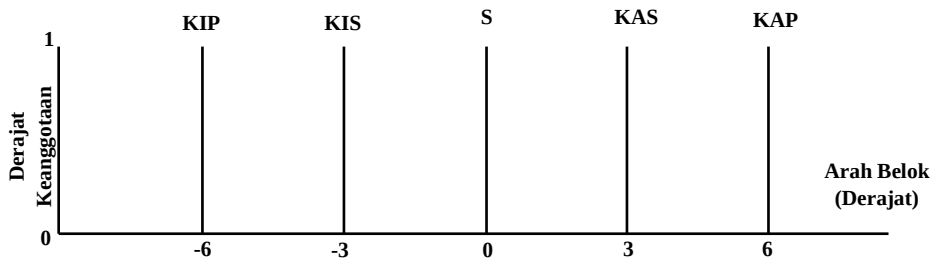
KAB = Kanan Besar

C. Defuzzifikasi

Defuzzifikasi merupakan kebalikan dari fuzzifikasi. Jika fuzzifikasi adalah proses mengubah nilai *input* yang bersifat pasti ke dalam *input Fuzzy*, defuzzifikasi adalah proses mengubah *output* himpunan *Fuzzy* menjadi *output* pasti atau tegas (*crisp*). Hal ini perlu dilakukan karena konstanta kendali *Fuzzy* hanya mengenal nilai tegas sebagai *variable* sinyal kontrol. Nilai dari *crisp output* inilah yang menjadi penentu arah robot. Proses pembentukan *crisp output* menggunakan metode sugeno menjadi bentuk *crisp output* dapat dituliskan dalam bentuk rumus sebagai berikut :

$$Crisp\_out = \frac{\sum_i FuzzyOutput_i \times (PosisiSingletonXaxis_i)}{\sum_i FuzzyOutput_i} \quad \text{.....(3.2)}$$

Berikut ini adalah *output Fuzzy* setelah melalui tahap proses defuzzifikasi:

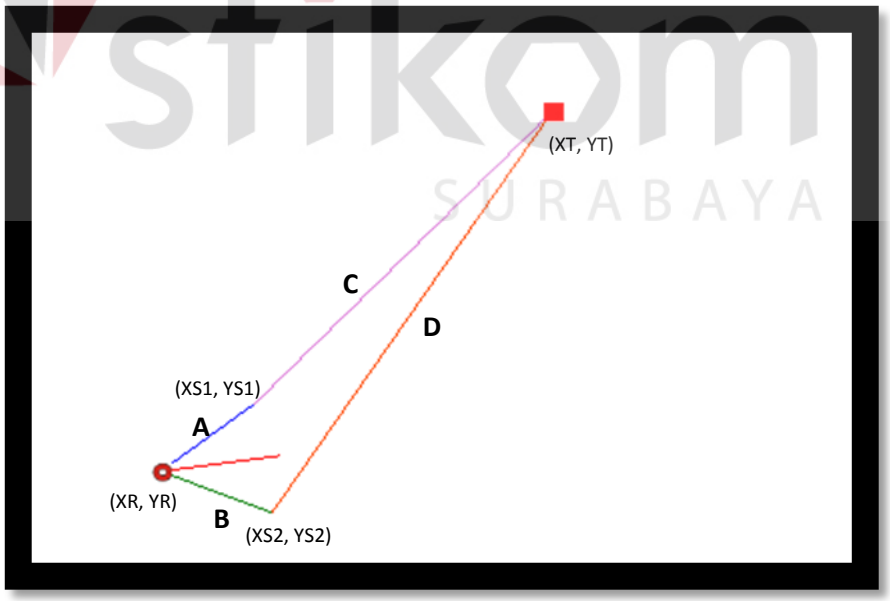


Gambar 3.20. *Output Fuzzy* Arah Belok

Output Fuzzy, yang dapat dilihat pada Gambar 3.20, adalah nilai yang akan menjadi sudut belok untuk pergerakan robot menuju target maupun menghindari *obstacle*.

11. Perancangan Program untuk Metode *Virtual Force Field*

Metode *Virtual Force Field* (VFF) yang digunakan dalam penelitian ini adalah metode VFF yang telah disederhanakan. Sehingga hanya diambil konsepnya sebagian yang berkaitan dengan vektor. Untuk mencapai target yang dituju dan dapat menghindari *obstacle*, ada beberapa parameter *input* yang akan diolah oleh metode VFF. Diantaranya adalah koordinat sensor kanan, koordinat sensor kiri, koordinat target kanan, dan koordinat target kiri.



Gambar 3.21. Penggambaran *Input* VFF

Keterangan :

XR	= Koordinat X Robot	XS1	= Koordinat X Sensor Kiri
YR	= Koordinat Y Robot	YS1	= Koordinat Y Sensor Kiri
XT	= Koordinat X Target	XS2	= Koordinat X Sensor Kanan
YT	= Koordinat Y Target	YS2	= Koordinat Y Sensor Kanan

Dari beberapa nilai *input* tersebut, akan diolah menggunakan rumus berikut :

$$\tan\theta = \frac{F_y}{F_x} \quad \text{.....(3.3)}$$

$$\theta = \tan^{-1}\frac{F_y}{F_x} \quad \text{.....(3.4)}$$

Vektor A merupakan jarak baca sensor kiri, vektor B merupakan jarak baca sensor kanan. Vektor C merupakan jarak dari ujung sensor kiri menuju target, dan vektor D merupakan jarak dari ujung sensor kanan menuju target.

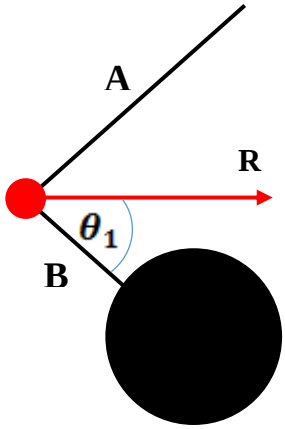
$$A = \sqrt{(XR - XS1)^2 + (YR - YS1)^2} \quad \text{.....(3.5)}$$

$$B = \sqrt{(XR - XS2)^2 + (YR - YS2)^2} \quad \text{.....(3.6)}$$

$$C = \sqrt{(XT - XS1)^2 + (YT - YS1)^2} \quad \text{.....(3.7)}$$

$$D = \sqrt{(XT - XS2)^2 + (YT - YS2)^2} \quad \text{.....(3.8)}$$

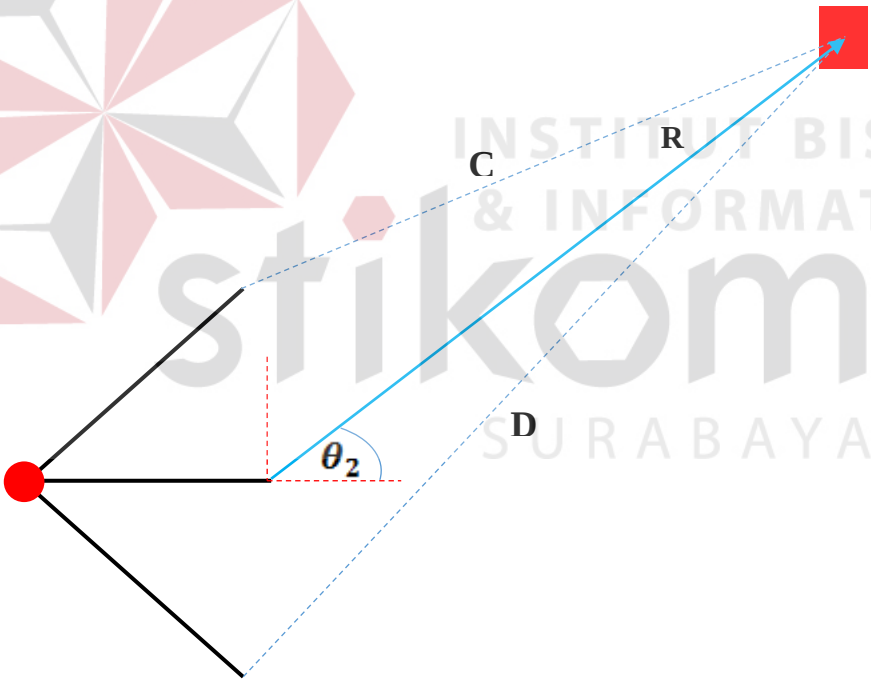
Untuk menentukan arah belok robot ketika robot mendeteksi adanya *obstacle,* digunakan perhitungan selisih vektor A dan B. Sedangkan untuk menentukan arah belok robot menuju target, digunakan perhitungan selisih vektor C dan D. Sehingga, dapat dimasukkan ke dalam perhitungan berikut.



Gambar 3.22. Sensor Robot Mendeteksi *Obstacle*

Untuk menghitung sudut arah belok robot untuk menghindari *obstacle* :

$$\theta_1 = \tan^{-1}(B - A) \quad \text{.....(3.9)}$$



Gambar 3.22. Garis Bantu Menuju Target

Untuk menghitung sudut arah belok robot untuk menuju target :

$$\theta_2 = \tan^{-1}(C - D) \quad \text{.....(3.10)}$$