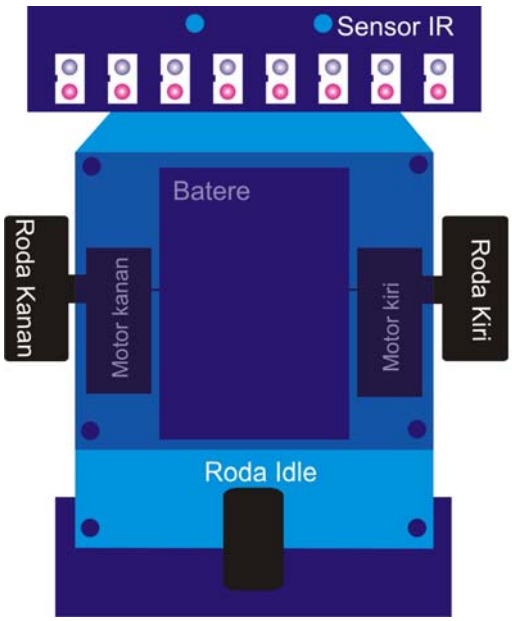
BAB III

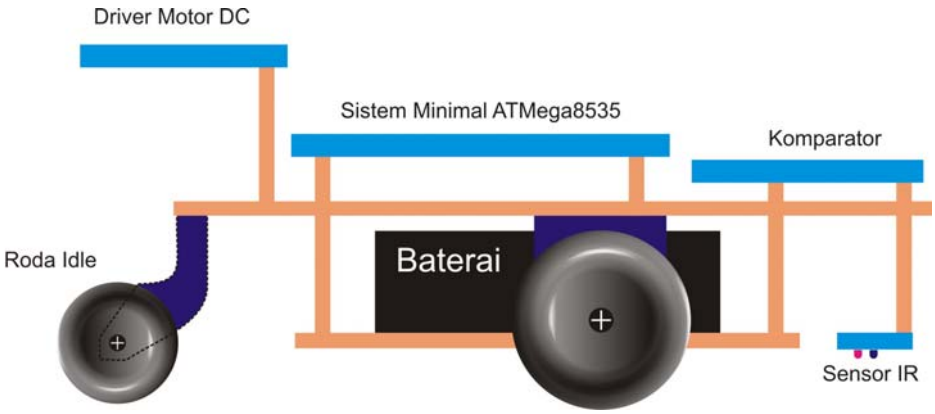
PERANCANGAN DAN IMPLEMENTASI ALAT

A. Perancangan Konstruksi Robot

Robot *line follower* yang akan dirancang ditunjukan pada gambar berikut ini :

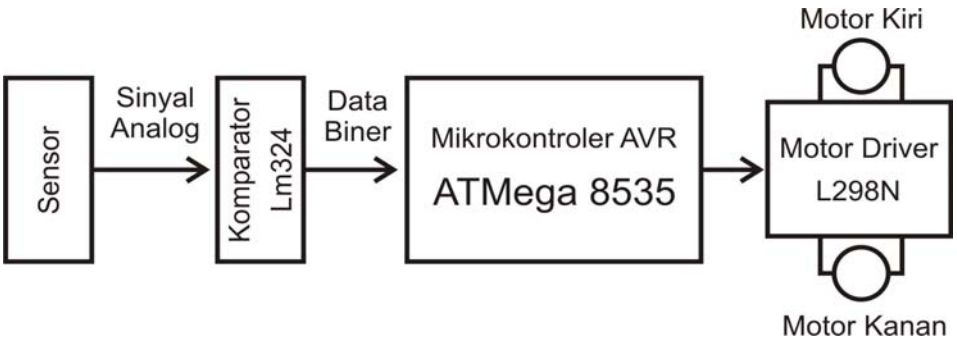


Gambar 3.1 Robot tampak dari bawah



Gambar 3.2 Robot tampak samping

Diagram blok robot *line follower* secara keseluruhan dapat dilihat pada gambar berikut :



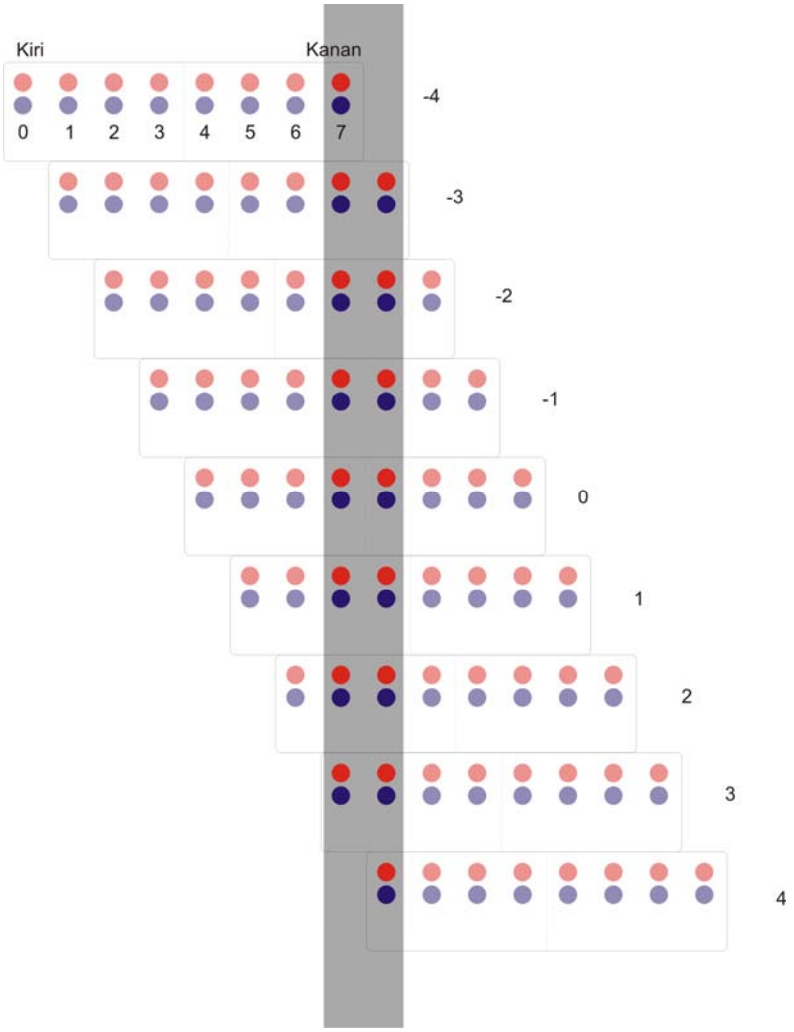
Gambar 3.3. Blok diagram robot line follower

Robot menggunakan sensor *Infrared* (IR) untuk mendeteksi garis pembimbing. Masing-masing sensor terdiri dari pasangan *infrared emitting diode* (IRED) sebagai pemancar (Tx) dan *phototransistor* sebagai sensor (Rx). Jumlah sensor IR yang dipasang pada robot *line follower* ini terdiri dari 8 buah yang dipasang sejajar dan menghadap ke lantai. Keluaran dari sensor masih berupa sinyal analog yang bergantung dari jumlah pancaran sinar inframerah yang dipantulkan dan diterima oleh sensor *phototransistor*. Sensor dipasang pada rangkaian pengkondisi sinyal yang berfungsi sebagai pembanding (komparator), untuk menghasilkan keluaran berupa logika "0" dan "1". Keluaran dari rangkaian sensor ini dihubungkan ke mikrokontroler ATMega8535. Konfigurasi pemasangan sensor adalah sebagai berikut :

Gambar 3.4. Konfigurasi posisi sensor

Mulai dari kiri ke kanan, sensor diberi nama 0, 1, 2, 3, 4, 5, 6, 7. pemakaian sensor IR pada robot *line follower* ini diasumsikan bahwa, jika sensor berada pada garis, dalam hal ini adalah garis hitam, maka keluaran dari

sensor berlogika “0” dan jika sensor tidak berada pada garis, dalam hal ini latar warna putih, maka keluaran sensor berlogika “1”. Pemasangan sensor yang tampak seperti gambar 3.1 dan gambar 3.2 maka kemungkinan posisi sensor adalah :



Gambar 3.5 Kemungkinan posisi sensor pada lintasan

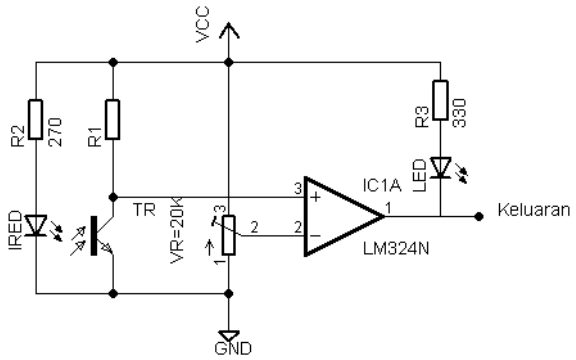
Mikrokontroler sebagai sistem navigasi dari robot akan memutar kedua motor DC secara differensial, dengan teknik PWM (*Pulse Widht Modulation*) untuk menggerakan robot mengikuti garis hitam. Keputusan mikrokontroler

untuk menggerakan motor DC berdasarkan kemungkinan posisi sensor seperti pada gambar 3.5 diatas, menggunakan Algoritma pemrograman *Fuzzy Logic Controller* (FLC).

B. Sensor *Infrared* (IR)

1. Rangkaian Sensor

Skematik rangkaian sensor ini bisa dilihat pada gambar berikut :



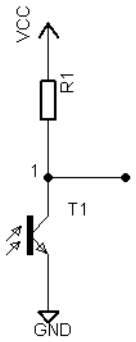
Gambar 3.6. Skematik rangkaian sensor IR

Resistansi dari sensor (*phototransistor*) akan meningkat dengan semakin banyaknya pancaran inframerah yang masuk ke sensor. Sensor yang ideal adalah sensor yang memiliki nilai resistansinya mendekati nol ketika ada pancaran inframerah. Sebaliknya, resistansi sensor akan sangat besar sekali ketika tidak ada pancaran inframerah yang masuk ke sensor.

2. Karakterisasi Sensor

Untuk menghasilkan *voltage swing* (perbedaan tegangan ketika *phototransistor* disinari dan ketika tidak disinari) yang bagus, maka pemilihan nilai resistansi R_1 harus benar-benar diperhatikan. Pemilihan

nilai R_1 bisa ditentukan dari prinsip pembagian tegangan. Perhatikan gambar dibawah ini :



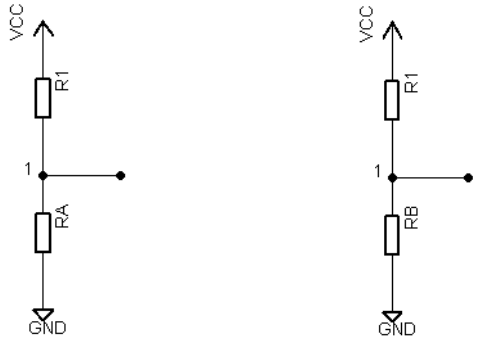
Gambar 3.7. Skema rangkaian pembagian tegangan

Kita asumsikan bahwa,

$R_{sensor} = R_A$; ketika *phototransistor* tidak disinari oleh inframerah.

$R_{sensor} = R_B$; jika *phototransistor* disinari oleh inframerah.

Maka skema rangkaian diatas dapat dirubah sebagai berikut :



(a) (*b*)

Gambar 3.8. Skema rangkaian pembagi tegangan ketika (a) disinari; (b) tidak disinari

Nilai resistansi total dari kedua resistor adalah R_1+R_{sensor}. Sedangkan tegangan yang melewati keduanya adalah V_{CC}. Dengan menggunakan hukum ohm, kita dapat menghitung besarnya arus, yaitu:

$$i = \frac{V_{CC}}{R_1 + R_{sensor}} \quad \text{.....(3.1)}$$

Ketika tidak disinari oleh inframerah, $R_{sensor} = R_A$, maka :

$$i = \frac{V_{CC}}{R_A + R_1} \quad \text{.....(3.2)}$$

Tegangan keluarannya adalah :

$$V_{out} = i \times R_A$$

$$V_{out} = V_{CC} \frac{R_A}{R_A + R_1} \quad \text{.....(3.3)}$$

Jika disinari, $R_{sensor} = R_B$, maka :

$$i = \frac{V_{CC}}{R_B + R_1} \quad \text{.....(3.4)}$$

Tegangan keluarannya adalah :

$$V_{out} = i \times R_B$$

$$V_{out} = V_{CC} \frac{R_B}{R_B + R_1} \quad \text{.....(3.5)}$$

Dari persamaan (3) dan (5), dapat ditentukan *voltage swing*, yaitu :

$$V_S = V_A - V_B$$

$$V_S = V_{CC}\left(\frac{R_A}{R_A + R_1} - \frac{R_B}{R_B - R_1} \right) \quad \text{.....(3.6)}$$

Relatif *voltage swing* adalah

$$V_S(R) = \frac{V_S}{V_{CC}}$$

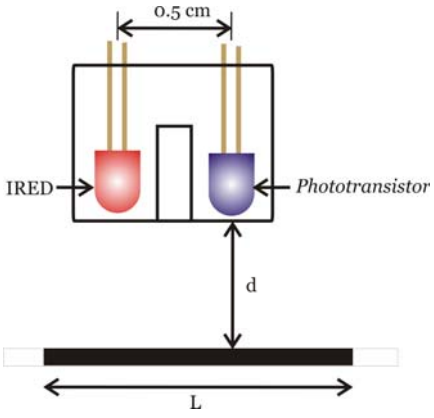
$$V_S(R) = \left(\frac{R_A}{R_A + R_1} - \frac{R_B}{R_B - R_1} \right) \text{.....(3.7)}$$

Nilai R_A dan R_B dapat diketahui dengan melakukan pengukuran resistivitas *phototransistor* menggunakan *ohmmeter*. Dengan diketahuinya nilai R_A dan R_B maka dapat dibuat grafik *voltage Swing* (V_S) tehadap range nilai resistansi R_1. Kemudian dari grafik tersebut dengan mudah dapat menentukan nilai resistansi R_1 untuk menghasilkan *voltage swing* yang paling besar.

Setelah dilakukan pengukuran nilai R_A adalah 65 KΩ dan nilai RB adalah 1920 KΩ.

a. Pengamatan Jarak Sensor Terhadap Lintasan

Pengamatan ini dilakukan dengan mengubah variabel jarak sensor ke lantai (d) seperti terlihat pada gambar berikut :



Gambar 3.9. Model percobaan untuk pengamatan karakteristik phototransistor

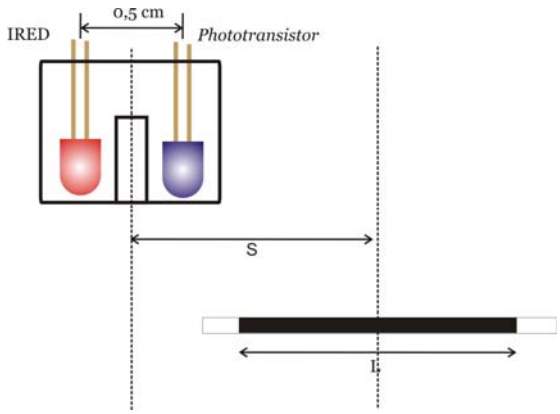
Pada saat dilakukan pengukuran IRED diberi arus panjar maju (I_d) sebesar 19,58 mA. Data hasil pengamatannya terlihat pada tabel berikut :

Tabel 3.1. Data hasil pengukuran karakteristik phototransistor

No.	Jarak sensor (d) - cm	Tegangan Keluaran *Phototransistor* (volt)	
		Lantai Putih	Lantai Hitam
1	1	0.16	2.52
2	1,5	0.14	2.76
3	2	0.17	3.89
4	2,5	0.21	4.28
5	3	0.4	4.32
6	3,5	1.76	4.57
7	4	2.3	4.48
8	4,5	2.7	4.68
9	5	3.09	4.66
10	5,5	3.58	4.68
11	6	3.62	4.68
12	6,5	3.66	4.68
13	7	3.68	4.68

b. Pengamatan Pengaruh Lebar Lintasan

Pengamatan pengarus lebar garis terhadap tegangan keluaran sensor dilakukan dengan merubah variabel lebar lintasan dan jarak sensor ke lintasan (s), seperti terlihat pada gambar berikut :



Gambar 3.10. Model percobaan pengamatan pengaruh lebar garis lintasan

Pada saat dilakukan pengukuran IRED diberi arus panjar maju sebesar 19,58 mA dengan jarak sensor ke lintasan adalah 2,5 cm karena berdasarkan hasil pengukuran diatas menunjukan jarak yang paling bagus adalah 2,5 cm sampai 3 cm.

Tabel 3.2. Data hasil pengukuran pengaruh jarak sensor terhadap lintasan

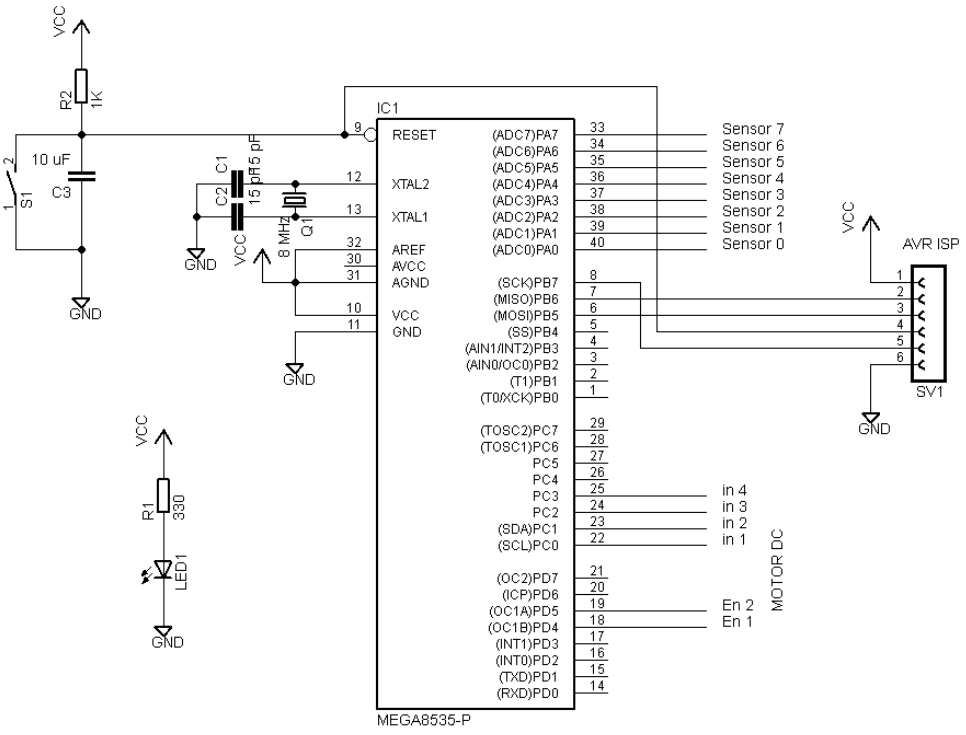
No.	Jarak sensor (s) - cm	Tegangan Keluaran *Phototransistor* (volt)		
		Lebar Garis 1,5 cm	Lebar garis 2 cm	Lebar garis 2.5
1	-1	0.15	0.17	0.16
2	-0.75	1.59	2.06	2.15
3	-0.5	2.66	3.29	3.34
4	0	3.85	4.26	4.46
5	0.5	3.52	3.75	4.12
6	0.75	2.18	2.83	3.12
7	1	0.18	0.2	0.19
8	1.25	0.16	0.17	0.17

C. Mikrokontroller ATMega8535

1. Rangkaian Sistem Minimal ATMega8535

Menurut Agus Bejo (2008;140) Disebut sistem minimal karena pemakaian komponen *hardware* yang digunakan merupakan kebutuhan yang paling minimal agar sebuah prosesor dapat bekerja.

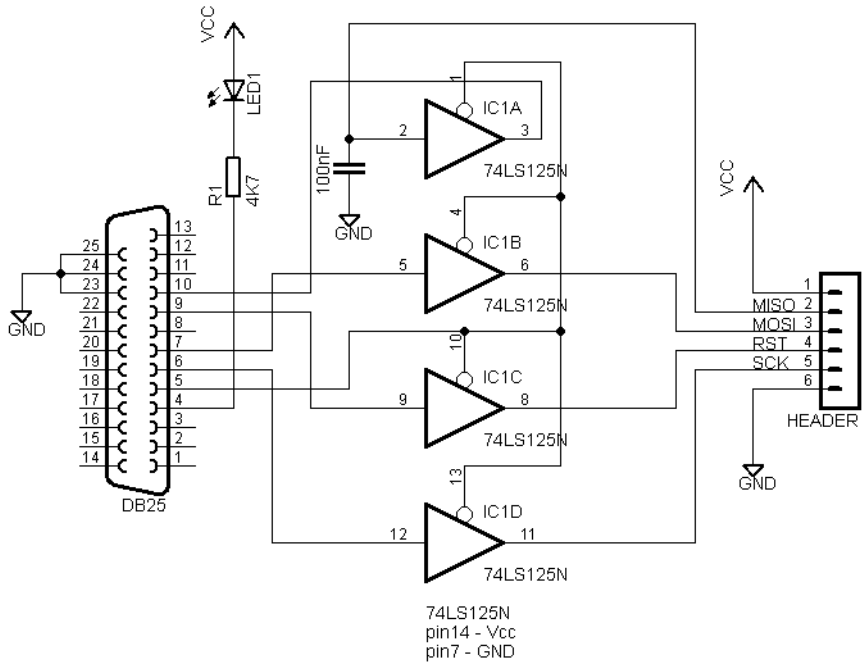
Gambar 3.8 adalah gambar skematik sistem minimal mikrokontroler ATMega8535 yang sudah ditambah dengan *header* ISP (*In System Programming*) yang memungkinkan mikrokontroler diprogram secara langsung di dalam *board* robot melalui kabel ISP, tanpa harus bongkar pasang. Sensor dihubungkan ke PORTA, dan input motor dihubungkan ke PORTC sedangkan PWM motor dihubungkan ke PORTD pin ke 4 dan 5 yang merupakan keluaran PWM.



Gambar 3.11. Sistem minimal AVR ATMega8535

2. Rangkaian *In System Programming* (ISP) AVR

Salah satu kelebihan dari CodeVison AVR adalah tersedianya fasilitas untuk mendownload program ke mikrokontroler yang telah terintegrasi sehingga dengan demikian CodeVision AVR ini selain dapat berfungsi sebagai *software* kompiler juga dapat berfungsi sebagai *software programmer/downloader*. Untuk dapat menggunakan fasilitas *downloader* ini kita membutuhkan tambahan modul *hardware* khusus seperti *Atmel STK200*, *Kanda System* STK200+/300. Modul hardware tersebut harganya relatif mahal sehingga hal inilah yang menjadi pertimbangan bagi kebanyakan orang untuk tidak menggunakannya. Ada alternatif modul *hardware* lain yang lebih mudah dan murah untuk dibuat namun tetap kompatibel dengan modul *hardware* STK200. modul tersebut skematiknya seperti pada gambar berikut :



Gambar 3.12. Rangkaian ISP AVR kompatibel STK200

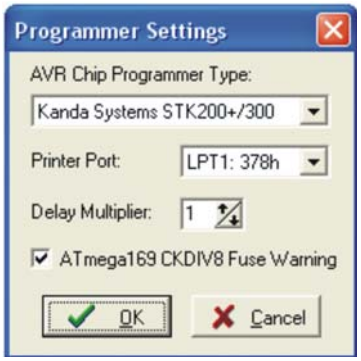
3. Konfigurasi ISP CODEVISION AVR

Agar fasilitas CodeVision AVR *Chip Programmer* dengan menggunakan modul *hardware* seperti pada gambar diatas dapat bekerja maka pada pengaturan *programmer*-nya harus dipilih modul STK200. caranya adalah sebagai berikut :

a. Buka program CodeVision AVR

b. Klik menu *Settings*

c. Pilih *Programmer*, maka akan muncul kotak dialog berikut :

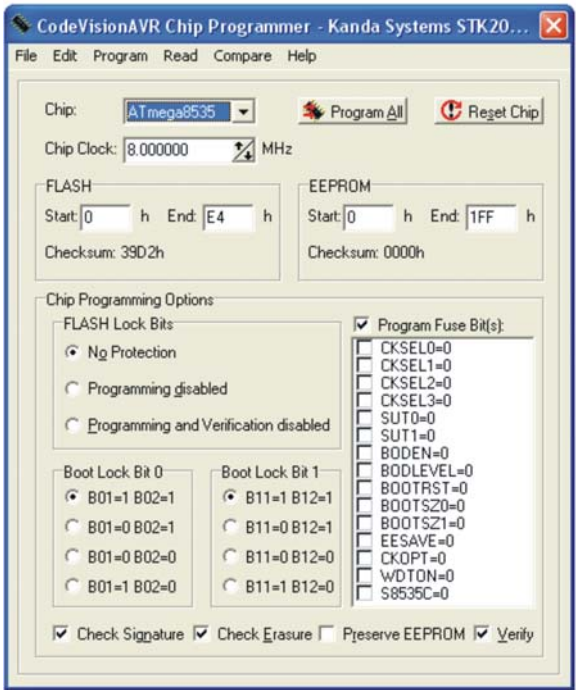


Gambar 3.13. Programmer Settings

Pada combo dialog *AVR Chip Programmer Type*, pilih *Kanda System* STK200+/300.

d. Selanjutnya untuk membuka program CodeVision AVR *Chip Programmer*, pilih menu *Tools*

e. Klik *Chip Programmer*, maka akan keluar kotak dialog sebagai berikut:



Gambar 3.14. Kotak dialog CodeVision AVR Chip Programmer

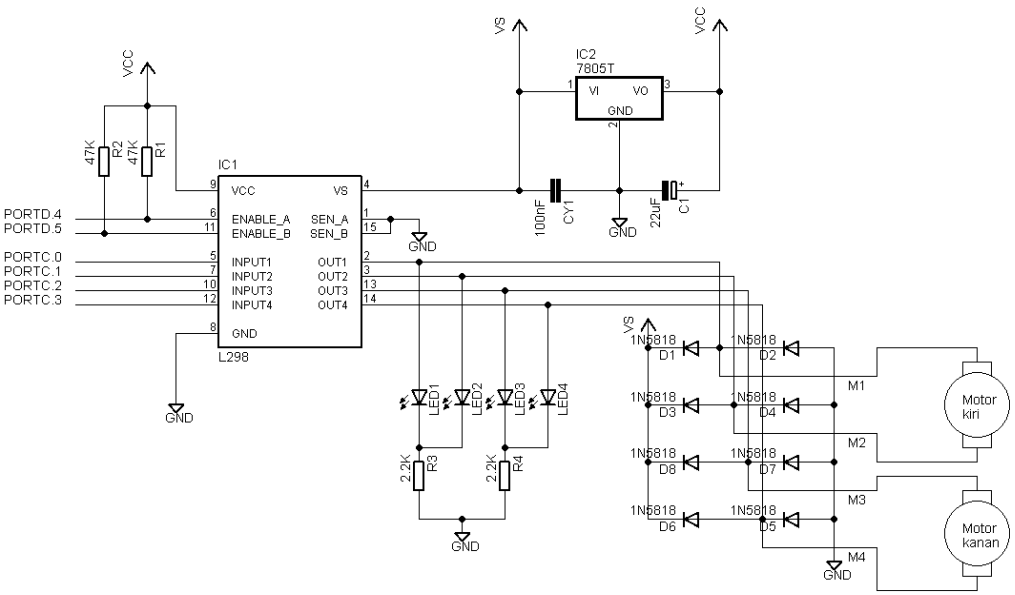
Keterangan :

- *Chip*, untuk menentukan jenis mikrokontroler
- *Chip Clock,* menentukan frekuensi osilator mikrokontroler yang digunakan.
- *Reset Chip*, untuk memberikan sinyal*reset* ke mikrokontroler.
- *Program All*, untuk melakukan *erase chip*, *blank chek*, *program flash*, *program eeprom*, program *lock bits* dan program *fuse bits* sekaligus dengan satu langkah.

D. *Driver* Motor DC

1. Rangkaian Driver

Skematik rangkaian *driver* motor DC menggunakan IC L298, adalah sebagai berikut :



Gambar 3.15. Skema rangkaian driver motor DC

Arah putaran motor diatur oleh *input1* sampai *input4*, yang dihubungkan ke mikrokontroler lewat PORTC. *Input1* dan *input2* berfungsi untuk mengatur arah putaran motor sebelah kiri. Sedangkan *input3* dan *input4* berfungsi untuk mengatur arah putaran motor sebelah kanan. Selain arah lewat keempat input diatas dapat mengatur supaya motor berhenti berputar (*Brake*).

Dengan memberikan tegangan masukan (VS) sebesar 12,69 volt, maka data perputaran motor terlihat pada tabel berikut :

Tabel 3.3. Data pengukuran masukan dan keluaran driver motor L298

En_A (volt)	In1 (volt)	Inp2 (volt)	Out1 (volt)	Out2 (volt)	En_B (volt)	Inp3 (volt)	Inp4 (volt)	Out3 (volt)	Out4 (volt)
5,13	0	0	0	0	5,13	0	0	0	0
5,13	0	5,13	0	12,54	5,13	0	5,13	0	12,55
5,13	5,13	0	12,56	0	5,13	5,13	0	12,55	0
5,13	5,13	5,13	12,56	12,54	5,13	5,13	5,13	12,56	12,54
0	x	x	0	0	0	x	x	0	0

Keterangan : x = 5 volt atau 0 volt

2. PWM (*Pulse Width Modulation*) Motor

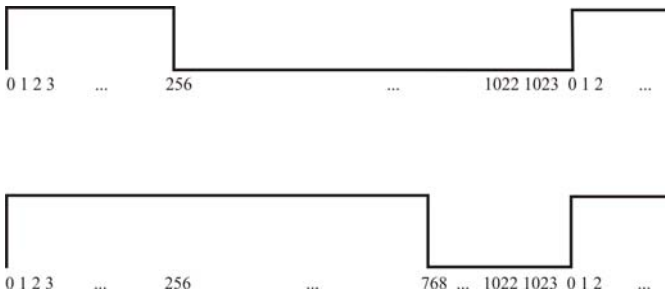
Kecepatan putaran Motor DC dikontrol lewat PWM (*Pulse Modulation Wave*). PWM dibangkitkan oleh mikrokontroler ATMega8535 dengan memanfaatkan fasilitas *Timer/Counter1*, dengan mode *fast PWM*. Alasan penggunaan Timer/Counter1 karena PWM dapat bekerja masing-masing berdiri sendiri atau tidak bergantung satu sama lain. PWM ini dikeluarkan lewat PORTD bit ke-4 dan bit ke-5, yang dihubungkan secara langsung ke pin *Enable A* dan *Enable B*. fungsi dari pin *Enable A* adalah untuk mengaktifkan *Out1* dan *Out2* yang dihubungkan ke motor DC bagian sebelah kiri. Artinya Enable A berfungsi mengatur kecepatan putaran motor Kiri. Begitu juga E*nable B* berfungsi mengaktifkan *Out3* dan *Out4* yang berfungsi mengatur kecepatan motor sebelah kanan. Dengan begitu, penggunaan PWM juga berfungsi sebagai *Steering*.

Berikut adalah contoh listing program untuk mengatur *timer/counter1* sebagai pembangkit PWM dengan mode *Fast* PWM 10-

bit. Keluaran pin OC1A merupakan PWM A, misal dengan *duty cycle* 25% dan keluaran OC1B merupakan PWM B misal dengan *duty cycle* 75%.

```
#include <mega8535.h>
Void main (void)
{
DDRD=0xFF;  //Port D sebagai output
/*inisialisasi TIMER1 sebagai Fast PWM 10-bit*/
TCCR1A=0xA3;
TCCR1B=0x0B;
TCNT1=0x0000; //setting inisialisai counter1
OCR1A=0x0100; //setting duty cycle 25% (PWM A)
OCR1B=0x0300; //setting duty cycle 75% (PWM B)
While(1);
}
```

Secara garis besar, cara kerja mode *Fast* PWM diatas adalah dengan membandingkan isi *register* TCNT1 dengan *register* OCR1A dan OCR1B untuk menghasilkan keluaran PWM. Isi *register* TCNT1 akan mencacah naik setiap interval waktu tertentu dari detak *clock* kristal sesuai pengaturan *register* TCCR1B sampai bernilai maksimum 0x03FF (1023). Selama nilai *register* TCNT1 lebih kecil dari data pembanding yaitu OCR1A atau OCR1B maka keluaran OC1A (PORTD.5) dan OC1B (PORTD.4) akan *high* dan jika nilai TCNT1 sudah melebihi data pembanding OCR1A atau OCR1B maka keluaran pin OC1A dan OC1B akan *low*. Jika nilai TCNT sudah mencapat maksimum yaitu 0x03FF (1023) maka nilai TCNT1 akan *reset* kembali menjadi 0x0000.



Gambar 3.16. Sinyal PWM pada pin OC1A dan OC1B

Register TCCR1A diisi 0xA3 berarti mode *Fast* PWM 10-bit. *Register* TCCR1B diisi 0x0B berarti skala clock 64. sehingga jika frekuensi kristal yang digunakan adalah 8 MHz maka akan dihasilkan PWM dengan frekuensi 122 Hz atau perioda 8,192 ms, dengan perhitungan sebagai berikut :

$$f_{PWM} = \frac{f_{OSC}}{N \times (1 + TOP)}$$

$$f_{PWM} = \frac{800000}{64 \times (1 + 1023)}$$

$$f_{PWM} = 122Hz$$

atau dinyatakan dalam perioda adalah :

$$T_{PWM} = \frac{1}{f_{PWM}}$$

$$T_{PWM} = \frac{1}{122Hz}$$

$$T_{PWM} = 8{,}192ms$$

Register OCR1A diisi dengan 0x0100 (256) maka akan diperoleh *duty cycle* PWM A sebesar 25%, dengan perhitungan sebagai berikut :

$$DutyCycle(D) = \frac{OCR1A}{TOP} \times 100\%$$

$$DutyCycle(D) = \frac{256}{1023} \times 100\% = 25\%$$

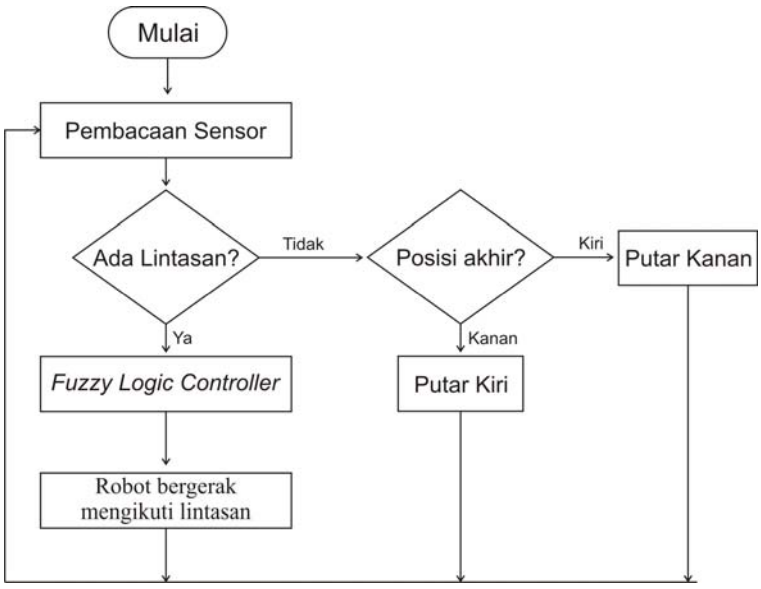
Register OCR1B diisi dengan 0x0300 (768) maka akan diperoleh *duty cycle* PWM B sebesar 75 %, dengan perhitungan sebagai berikut :

$$DutyCycle(D) = \frac{OCR1B}{TOP} \times 100\%$$

$$DutyCycle(D) = \frac{768}{1023} \times 100\% = 75\%$$

E. *Implementasi Fuzzy Logic Controller*

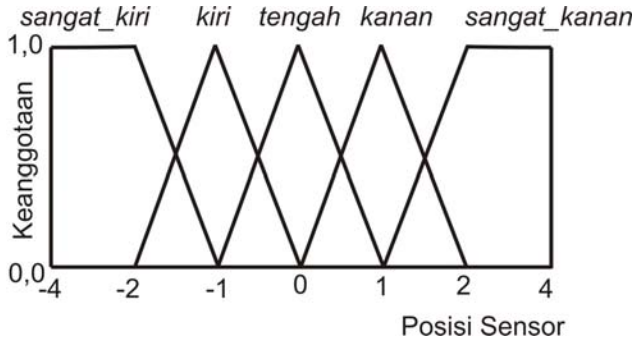
Perangkat lunak sistem robot *line follower* berupa program yang didalamnya meliputi proses pembacaan sensor, proses *fuzzy logic controller* untuk navigasi dan proses pengendalian gerakkan robot *line follower*. Diagram alir program secara keseluruhan dapat dilihat pada gambar berikut :



Gambar 3.17. Diagram alir program

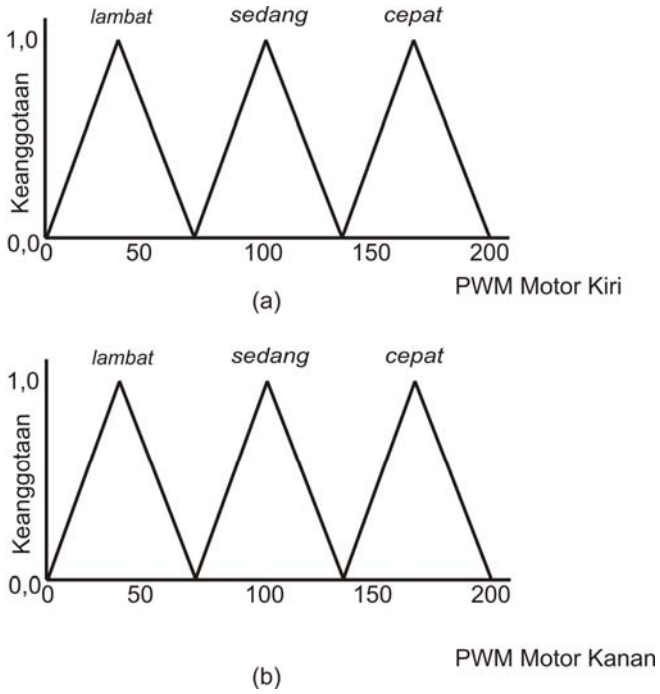
Proses Implementasi *fuzzy logic controller* meliputi proses fuzzifikasi (kuantisasi *Crisp Input* sensor dan derajat keanggotaan), evaluasi *rule* (aturan) dan defuzzifikasi. Proses *fuzzy logic controller* dalam bentuk program menggunakan bahasa C dengan kompiler CodeVision AVR.

Dalam sistem logika *fuzzy*, sebagai *crisp input* (masukan) adalah posisi dari 8 sensor IR yang dipasang didepan robot menghadap ke lantai yang akan mendeteksi lintasan. *Crisp output* (keluaran) dari sistem logika fuzzy adalah kecepatan putaran motor DC dengan teknik PWM dan arah kemudi robot. *Crisp input* posisi sensor hanya dirancang dalam tiga arah sudut pandang dari robot mobil. Lima arah tersebut adalah kiri, tengah, dan kanan. Dengan begitu terdapat lima label derajat keanggotaan (MF) yaitu : *sangat_kiri*, *kiri*, *tengah*, *kanan* dan *sangat_kanan*. Derajat keanggotaannya dapat dilihat pada gambar berikut :



Gambar 3.18. Derajat keanggotaan (MF) input posisi sensor

Derajat keanggotaan *crisp output* (keluaran) untuk kecepatan putaran motor memiliki enam label yaitu : *sangat_lambat*, *lambat, sedang, agak_cepat, cepat* dan *sangat_cepat.* Derajat keanggotaan kedua *crisp output* ini dapat dilihat pada gambar berikut :



Gambar 3.19. Derajat keanggotaan (MF) crisp output untuk (a) kecepatan PWM motor kiri; (b) kecepatan PWM motor kanan

Sedangkan untuk arah kemudi robot dapat ditentukan dari perbedaan kecepatan putaran motor berdasarkan derajat keanggotaan masing-masing PWM motor. Arah navigasi ini didefinisikan sebagai berikut :

Tabel 3.4. Arah Navigasi robot

Aksi Robot (Arah Navigasi)	PWM Motor Kiri	PWM Motor Kanan
Belok Kanan Tajam	*cepat*	*lambat*
Belok Kanan	*cepat*	*sedang*
Maju	*cepat*	*cepat*
Belok Kiri	*sedang*	*cepat*
Belok Kiri Tajam	*lambat*	*cepat*

Rule (aturan) yang digunakan adalah sistem *fuzzy* berbasis aturan, aturan ini diekstak dari pemikiran manusia. Untuk robot *line follower* yang dirancang menggunakan *rule* sebagai berikut :

- **Jika** posisi sensor *sangat_kiri* **Maka** arah navigasi *belok kanan Tajam*
- **Jika** Posisi sensor *kiri* **Maka** arah navigasi *belok kanan.*
- **Jika** Posisi sensor *tengah* **Maka** *maju.*
- **Jika** Posisi sensor *kanan* **Maka** *belok kiri.*
- **Jika** Posisi sensor *sangat_kanan* **Maka** *belok kiri tajam.*

Realisasi dalam bahas C untuk proses *fuzzy* diatas, sangat sederhana mengingat bahwa input sensor berupa data digital dalam bentuk biner. Begitu juga keluarannya untuk menggerakan motor berupa data digital. Teknik pemrogramannya pun menggunakan teknik *scanning*. Jadi *fuzzy logic* tidak sepenuhnya diterapkan mengingat kuantisasi *input* dan *output*, jadi hanya ada beberapa kemungkinan yang akan terjadi. Potongan listing program kuantisasi input sebagai berikut :

```
Unsigned char sensor;
Void fuzzifikasi()
{
 Sensor=PIND;
 if(sensor!=255) //jika tidak ada lintasan
       {
        // kuantisasi crisp input sensor
        if(sensor.0==0)
         kiri=1;
        if(sensor.1==0)
         kiri=2;
        if(sensor.2==0)
         kiri=3;
        if(sensor.3==0)
         kiri=4;
        if(sensor.4==0)
         kanan=1;
        if(sensor.5==0)
         kanan=2;
```

```
        if(sensor.6==0)
         kanan=3;
        if(sensor.7==0)
         kanan=4;
       }
```

Proses kuantisasi *output* yang merupakan PWM motor kiri dan kanan, juga arah navigasinya didefinisikan,

```
Unsigned char lpwm, rpwm
#include <delay.h>
void maju()
 {
  PORTC=0xAA;
 }
void belok_kiri()
 {
  lpwm =50; rpwm =50;
  delay_ms(60);          // dimajukan sedikit
  PORTC=0x88;
 }
void belok_kanan()
 {
  lpwm =50; rpwm =50;
  delay_ms(60);
  PORTC=0x22;
 }
Void putar_kanan()
 {
  lpwm =50; rpwm =50;
  delay_ms(60);
  PORTC=0x99;
 }
Void putar_kiri()
 {
  lpwm =50; rpwm =50;
  delay_ms(60);
  PORTC=0x66;
 }

//output PWM
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 1000.000 kHz
// Mode: Fast PWM top=00FFh
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0xA1;
TCCR1B=0x0A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
```

```
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0xFF;
OCR1BH=0x00;
OCR1BL=0xFF;
```

Rule yang diterapkan pada robot ini, listingnya adalah sebagai berikut :

```
Unsigned char sensor;
void rule()
{
  switch(sensor)
 { case sensor.0=0:  rpwm=0;   lpwm=200;  break;
   case (sensor.0&sensor.1)=0:  rpwm=50;  lpwm=200;
         break;
   case sensor.1:  rpwm=75;  lpwm=200;  break;
   case (sensor.1&sensor.2)=0:  rpwm=100; lpwm=200;
         break;
   case sensor.2:  rpwm=150; lpwm=200;  break;
   case (sensor.2&sensor.3)=0:  rpwm=200; lpwm=200;
         break;
   .
   .
   .
   dst
}
}
```

Jika sensor keluar dari lintasan, artinya berada pada lantai putih semuanya, maka rutin yang akan dieksekusi adalah :

```
Unsigned char dirl, MAX, history[i],rotpow,HMAX
Unsigned char putar_kiri, putar_kanan
Void move
for(i=0,dirl=0;i<MAX;i++) {
        if(history[i]==L)
        {dirl++;}
        }
         if(rotpow<160) {rotpow=160;}
         if(rotpow<255) {rotpow++;}

        if(dirl>HMAX)
        {putar_kanan;}
        else
        {putar_kiri;}
        }
```