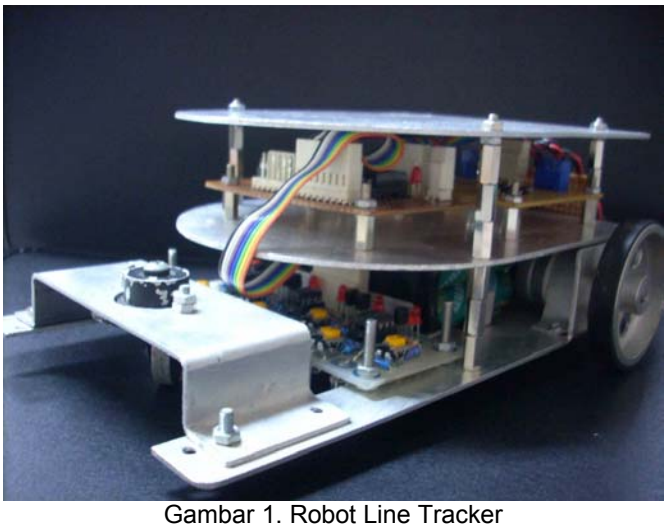
Membuat Sendiri Robot Line Tracker

Robot Line Tracker

Pada project kali ini kita akan membahas cara membuat robot line tracker yang dapat bergerak mengikuti track berupa garis hitam setebal 3 cm. Garis hitam tersebut disusun membentuk sejumlah persimpangan-persimpangan. Robot diprogram untuk dapat menghitung jumlah persimpangan yang sudah dilaluinya, kemudian belok sesuai dengan arah yang diinginkan. Untuk membaca garis, robot dilengkapi dengan sensor proximity yang dapat membedakan antara garis hitam dengan lantai putih. Sensor proximity ini dapat dikalibrasi untuk menyesuaikan pembacaan sensor terhadap kondisi pencahayaan ruangan. Sehingga pembacaan sensor selalu akurat.

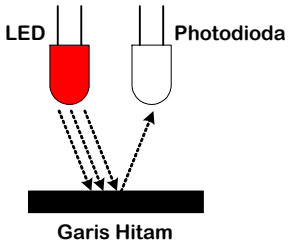
Agar pergerakan robot menjadi lebih halus, maka kecepatan robot diatur sesuai dengan kondisi pembacaan sensor proximity. Jika posisi robot menyimpang dari garis, maka robot akan melambat. Namun jika robot tepat berada diatas garis, maka robot akan bergerak cepat. Robot juga dapat kembali ke garis pada saat robot terlepas sama sekali dari garis. Hal ini bisa dilakukan karena robot selalu mengingat kondisi terakhir pembacaan sensor. Jika terakhir kondisinya adalah disebelah kiri garis, maka robot akan bergerak ke kanan, demikian pula sebaliknya.



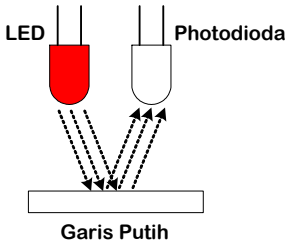
Gambar 1. Robot Line Tracker

Sensor Proximity

Sensor proximity bisa kita buat sendiri. Prinsip kerjanya sederhana, hanya memanfaatkan sifat cahaya yang akan dipantulkan jika mengenai benda berwarna terang dan akan diserap jika mengenai benda berwarna gelap. Sebagai sumber cahaya kita gunakan LED *(Light Emiting Diode)* yang akan memancarkan cahaya merah. Dan untuk menangkap pantulan cahaya LED, kita gunakan photodiode. Jika sensor berada diatas garis hitam maka photodioda akan menerima sedikit sekali cahaya pantulan. Tetapi jika sensor berada diatas garis putih maka photodioda akan menerima banyak cahaya pantulan. Berikut adalah ilustrasinya :



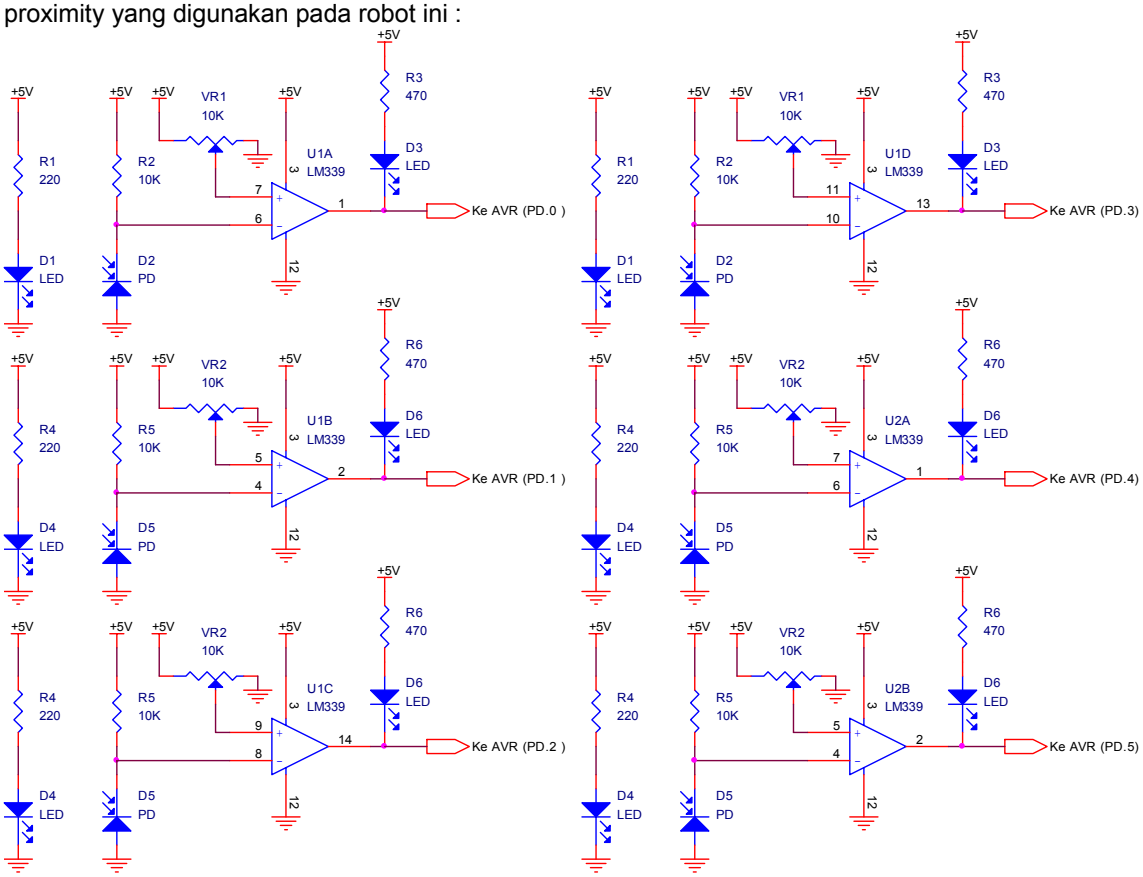
Gambar 1. Cahaya pantulan sedikit



Gambar 2. Cahaya pantulan banyak

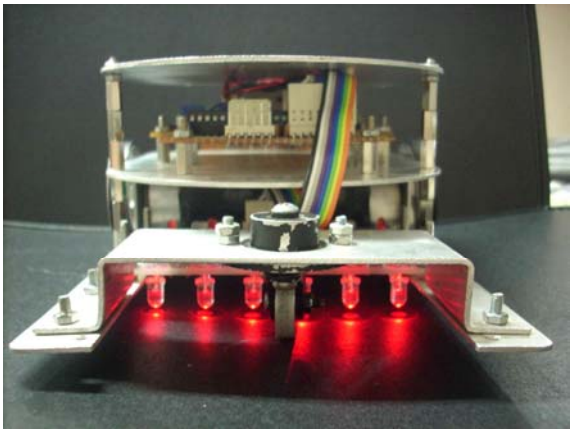
Gambar 2. Prinsip Kerja Sensor Proximity

Sifat dari photodioda adalah jika semakin banyak cahaya yang diterima, maka nilai resistansi diodanya semakin kecil. Dengan melakukan sedikit modifikasi, maka besaran resistansi tersebut dapat diubah menjadi tegangan. Sehingga jika sensor berada diatas garis hitam, maka tegangan keluaran sensor akan kecil, demikian pula sebaliknya. Berikut adalah gambar rangkaian sensor proximity yang digunakan pada robot ini :



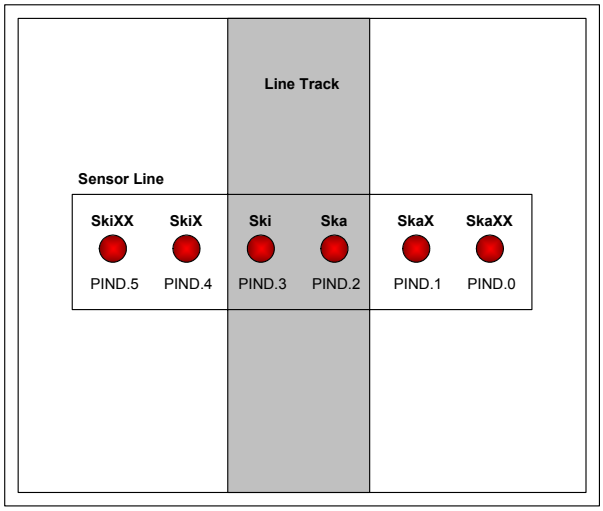
Gambar 3. Rangkaian Sensor Proximity

Agar dapat dibaca oleh mikrokontroler, maka tegangan sensor harus disesuaikan dengan level tegangan TTL yaitu 0 – 1 volt untuk logika 0 dan 3 – 5 volt untuk logika 1. Hal ini bisa dilakukan dengan memasang operational amplifier yang difungsikan sebagai komparator. Output dari photodiode yang masuk ke input inverting op-amp akan dibandingkan dengan tegangan tertentu dari variable resistor VR. Tegangan dari VR inilah yang kita atur agar sensor proximity dapat menyesuaikan dengan kondisi cahaya ruangan.



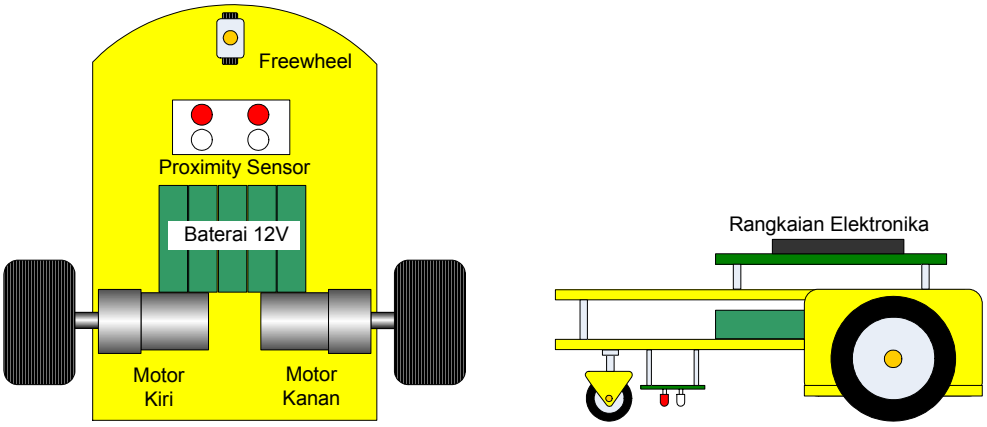
Gambar 4. Posisi Pemasangan Sensor Proximity Pada Robot

Sensor proximity terdiri dari 6 pasang LED dan photodiode yang disusun sedemikian rupa sehingga jarak antara satu sensor dengan yang lainnya lebih kecil dari lebar garis hitam. Perhatikan gambar berikut :



Gambar 5. Jarak Antar Sensor Proximity

Rancangan Mekanik Robot

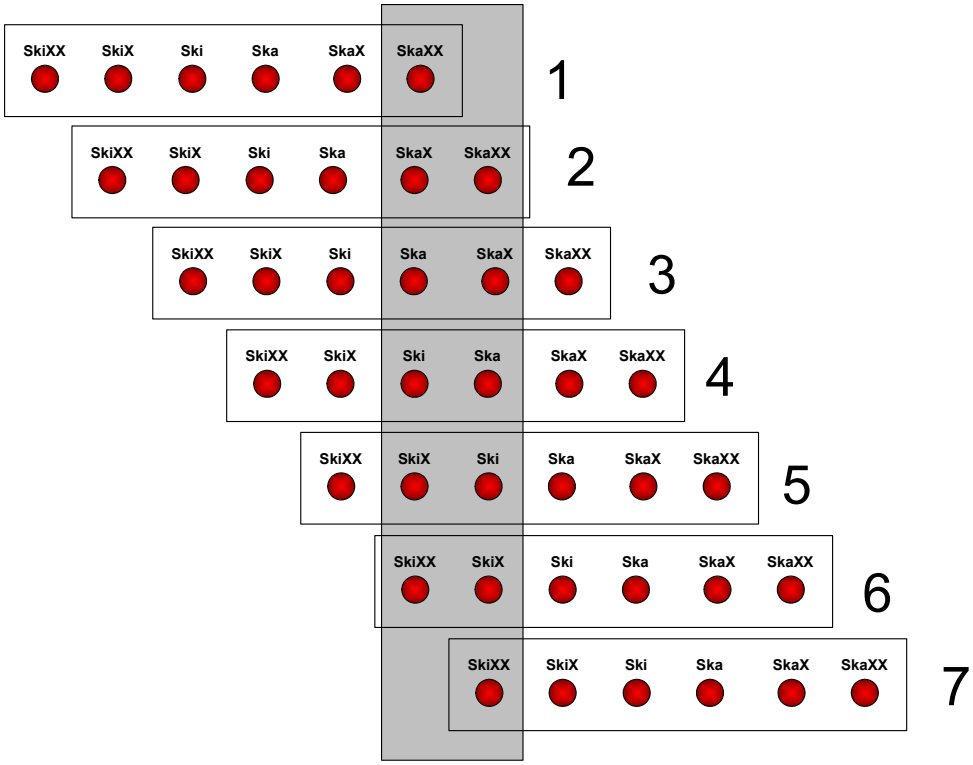


Robot Tampak Atas **Robot Tampak Samping**

Gambar 6. Mekanik Robot

Algoritma Pergerakan Robot

Sebelum membuat program, maka kita perlu mendefinisikan seluruh kemungkinan pembacaan sensor proximity. Dengan demikian kita dapat menentukan pergerakan robot yang tujuannya adalah menjaga agar robot selalu berada tepat diatas garis. Berikut adalah beberapa kemungkinan pembacaan garis oleh sensor proximity :



Gambar 7. Kemungkinan Posisi Sensor Proximity Pada Line

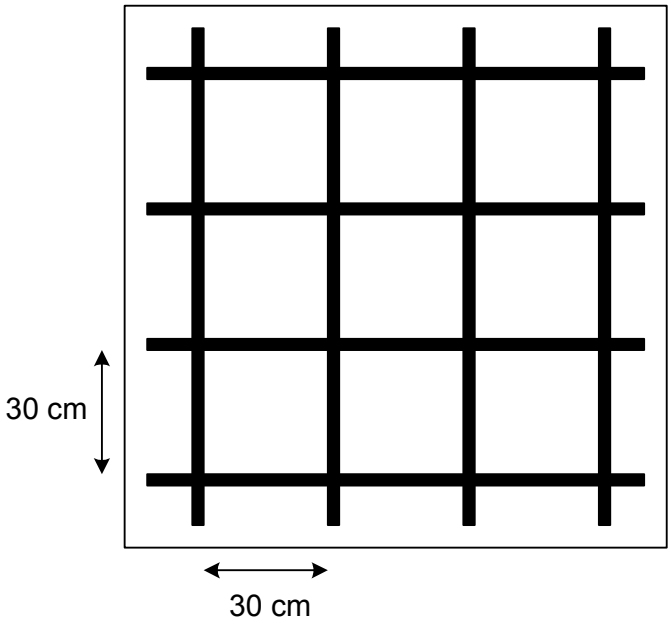
Setelah mengetahui kemungkinan-kemungkinan posisi sensor, maka selanjutnya harus didefinisikan aksi dari setiap kondisi tersebut. Perhatikan tabel berikut ini :

Tabel 1. Aksi Pergerakan Robot

Posisi Sensor	Aksi Robot	Roda Kiri	Roda Kanan
1	Belok Kanan Tajam	Maju cepat	Berhenti
2	Belok Kanan Sedang	Maju cepat	Maju lambat
3	Belok Kanan Ringan	Maju cepat	Maju sedang
4	Maju Lurus	Maju cepat	Maju cepat
5	Belok Kiri Ringan	Maju sedang	Maju cepat
6	Belok Kiri Sedang	Maju lambat	Maju cepat
7	Belok Kiri Tajam	Berhenti	Maju cepat

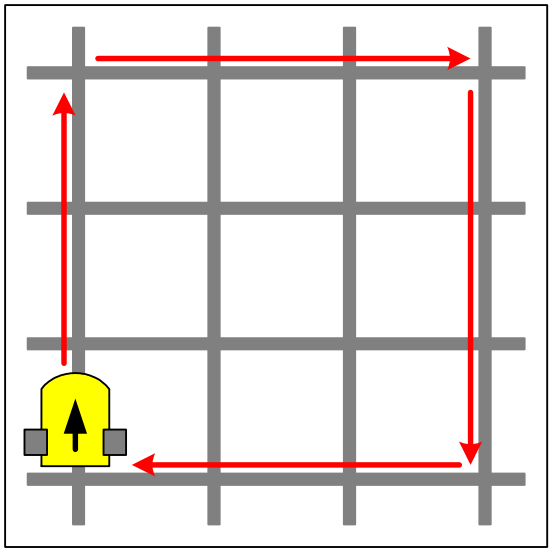
Lapangan Uji Coba

Lapangan berupa garis-garis hitam diatas lantai berwarna putih. Garis hitam disusun membentuk banyak persimpangan. Ukuran tiap kotak adalah 30 cm x 30 cm. Ketebalan garis hitam adalah 3 cm. Garis hitam ini bisa dibuat menggunakan isolasi hitam kemudian ditempel pada lantai atau kertas karton berwarna putih.



Gambar 8. Lapangan Uji Coba

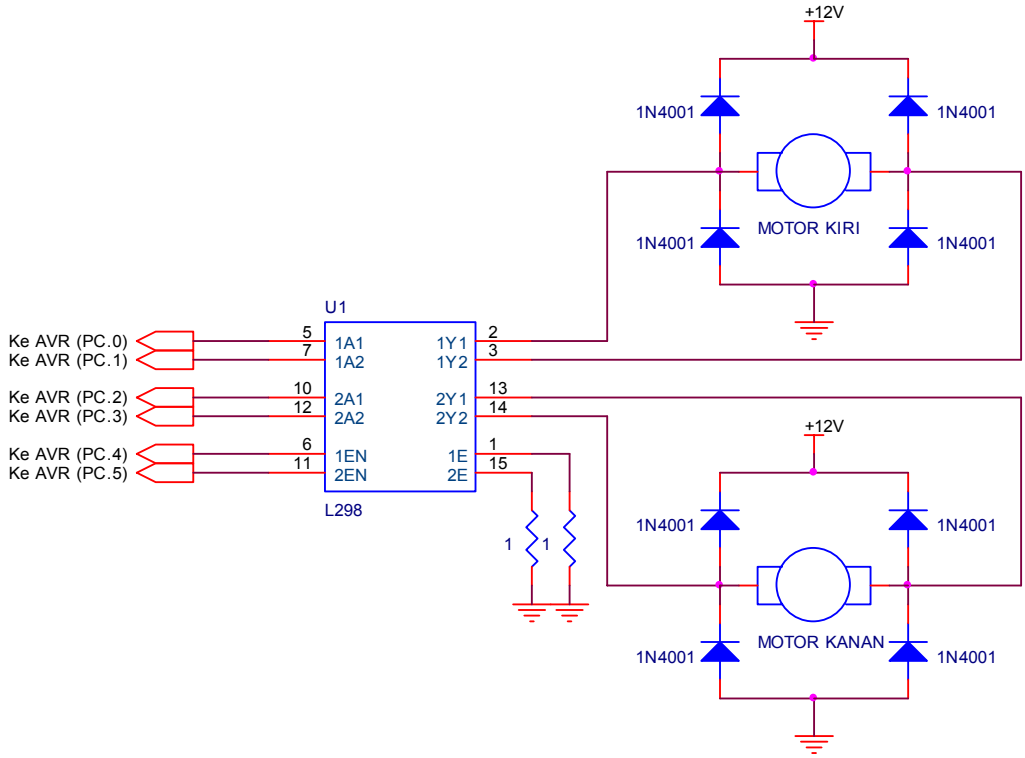
Dalam aplikasi ini, robot akan bergerak mengikuti kotak terluar lapangan. Posisi awal robot seperti terlihat pada gambar berikut ini :



Gambar 9. Pergerakan Robot di Lapangan

Driver Motor DC

Untuk menggerakkan dua buah motor dc, digunakan IC H-Bridge Motor Driver L298, yang mampu memberikan arus maksimum sebesar 1A ke tiap motor. Input L298 ada 6 jalur, terdiri dari input data arah pergerakan motor dan input untuk PWM (Pulse Width Modulation). Untuk mengatur kecepatan motor, pada input PWM inilah akan diberikan lebar pulsa yang bervariasi dari mikrokontroler.



Gambar 10. Rangkaian Driver Motor DC

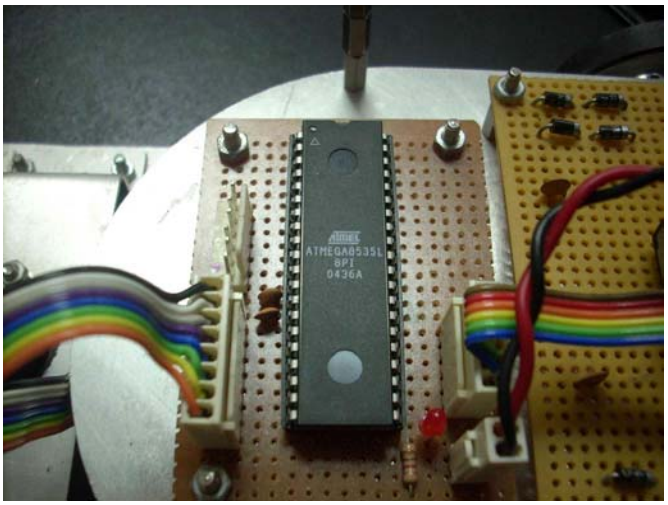
Untuk menentukan arah pergerakan motor maka pada input L298 harus diberikan kondisi sesuai dengan tabel berikut :

Tabel 2. Tabel Kebenaran Driver Motor

Motor 1						Motor 2					
Input			Output			Input			Output		
I1A1	1A2	1EN	1Y1	1Y2	Aksi Motor	2A1	2A2	2EN	2Y1	2Y2	Aksi Motor
0	0	1	0	0	Free Running Stop	0	0	1	0	0	Free Running Stop
0	1	1	0	12V	CW	0	1	1	0	12V	CW
1	0	1	12V	0	CCW	1	0	1	12V	0	CCW
1	1	1	12V	12V	Fast Stop	1	1	1	12V	12V	Fast Stop
x	x	0	0	0	Free Running Stop	x	x	0	0	0	Free Running Stop

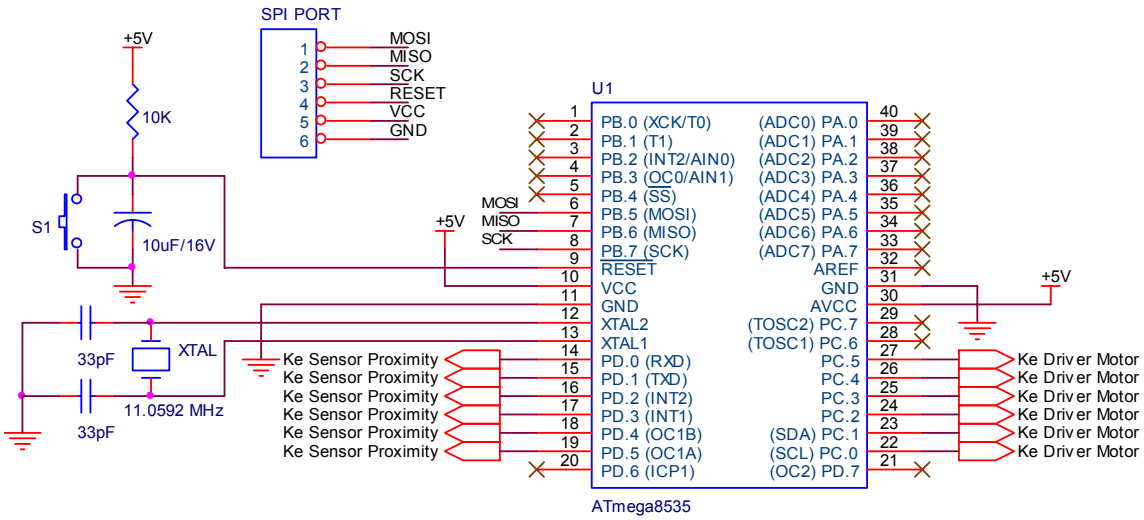
AVR Microcontroller

Sebagai ”otak” robot digunakan mikrokontroler AVR jenis ATmega8535 yang akan membaca data dari sensor proximity, mengolahnya, kemudian memutuskan arah pergerakan robot.



Gambar 11. Mikrokontroler ATmega8535 Pada Robot

Pada robot line track ini, keluaran sensor proximity dihubungkan ke PortD.0 dan PortD.5 pada mikrokontroler. Sedangkan driver motor dihubungkan ke PortC.0 s/d PortC.5 seperti terlihat pada gambar berikut :

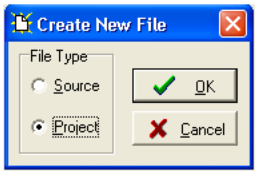


Gambar 12. Mikrokontroler ATmega8535

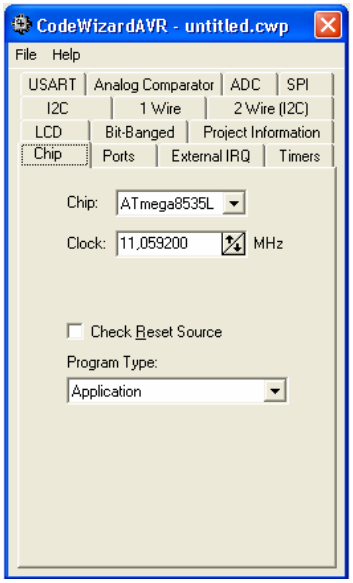
Membuat Source Code

Source code secara lengkap dapat dilihat pada Lampiran A. Source code dibuat dengan menggunakan software CodeVisionAVR dengan langkah-langkah sebagai berikut :

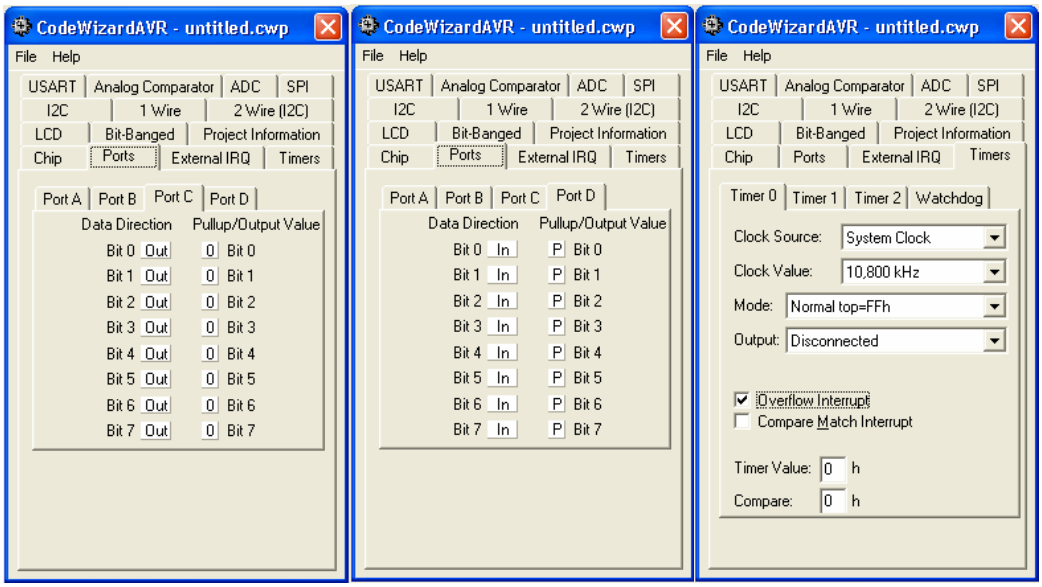
1. Jalankan CodeVisionAVR, kemudian klik **File** -> **New**, Pilih Project



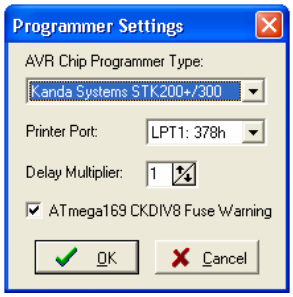
2. **"Do you want to use the CodeWizardAVR?"** Klik Yes
3. Pilih Chip yang digunakan, chip : ATmega8535L, clock : 11.059200 MHz



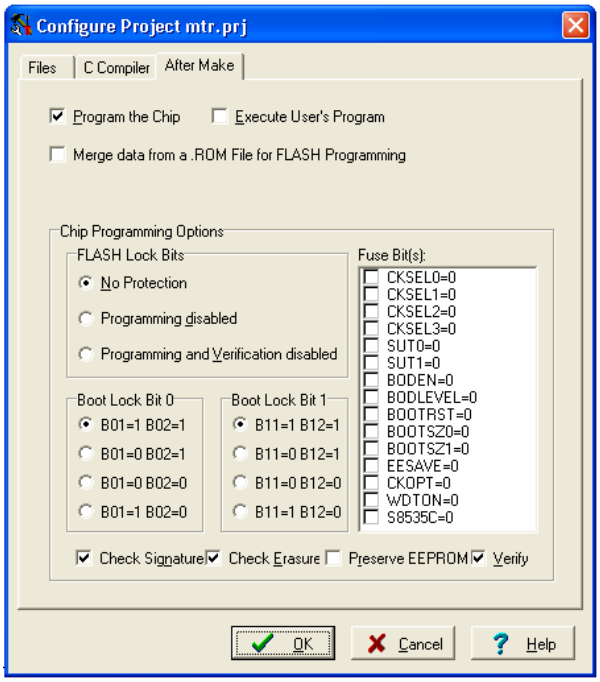
4. Lakukan setting sebagai berikut :
   Port : Port C sebagai Output dan Port D sebagai Input Pullup
   Timers : Timer 0 dengan Clock Value 10,800 KHz**,** aktifkan Overflow Interrupt



5. Klik **File** -> **Generate, Save and Exit**
6. Buatlah source code seperti pada Lampiran A.
7. Setelah selesai membuat source code, klik **Setting** -> **Programmer**
8. Pilih AVR Chip Programmer Type : **Kanda System STK200+/300** dan pilih Printer Port pada **LPT1 : 378h**



9. Klik **Project** -> **Configure**, kemudian pilih menu **After Make** dan aktifkan **Program the Chip**. Klik OK jika sudah.
   PERHATIAN ! Jangan mengubah setting apapun pada menu ini. Jika salah memilih, chip Anda tidak bisa digunakan lagi !!



10. Untuk meng-compile project, klik **Project** -> **Make**
11. Jika tidak ada error maka file siap didownload ke chip. Pastikan koneksi kabel downloader dan chip sudah terpasang dengan benar.
12. Nyalakan power supply dan klik **Program**. Tunggu hingga proses download selesai.

Penjelasan Source Code

Berikut adalah penjelasan tiap bagian dari source code.

1. Membuat definisi port yang digunakan sebagai berikut :

```
#define SkiXX          PIND.0          // Sensor kiri terluar
#define SkiX           PIND.1          // Sensor kiri luar
#define Ski            PIND.2          // Sensor kiri tengah
#define Ska            PIND.3          // Sensor kanan tengah
#define SkaX           PIND.4          // Sensor kanan luar
#define SkaXX          PIND.5          // Sensor kanan terluar

#define EnKi           PORTC.4         // Enable L298 untuk motor kiri
#define dirA_Ki        PORTC.0         // Direction A untuk motor kiri
#define dirB_Ki        PORTC.1         // Direction B untuk motor kiri
#define EnKa           PORTC.5         // Enable L298 untuk motor kanan
#define dirC_Ka        PORTC.2         // Direction C untuk motor kanan
#define dirD_Ka        PORTC.3         // Direction D untuk motor kanan
```

2. Menentukan library yang digunakan :

```
#include <mega8535.h>                  // Library untuk chip ATmega8535
#include <delay.h>                     // Library delay
```

3. Membuat variable sebagai pengingat kondisi pembacaan sensor line terakhir.

```
bit x;
```

4. Membuat ISR Timer 0 / Interrupt Service Routine Timer 0.
   - ISR ini digunakan untuk menghasilkan pulsa PWM untuk mengendalikan motor kiri dan kanan melalui bit EnKi dan EnKa.
   - ISR Timer 0 dieksekusi secara periodik ketika Timer 0 overflow. Lamanya tergantung nilai Timer/Counter 0 (TCNT0).
   - Periode pulsa ditentukan oleh TCNT0. Nilai maksimumnya 0xFF atau 255d.
   - Duty cycle PWM untuk motor kiri ditentukan oleh nilai **lpwm**. Maksimum 255.
   - Duty cycle PWM untuk motor kanan ditentukan oleh nilai **rpwm**. Maksimum 255.

```
unsigned char xcount,lpwm,rpwm;        // Definisi variable
// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Place your code here
 xcount++;                             // xcount=xcount+1
   if(xcount<=lpwm)EnKi=1;             // EnKi=1 jika xcount <= lpwm
   else EnKi=0;                        // EnKi=0 jika xcount >  lpwm
   if(xcount<=rpwm)EnKa=1;             // EnKa=1 jika xcount <= rpwm
   else EnKa=0;                        // EnKa=0 jika xcount >  rpwm
   TCNT0=0xFF;                         // Timer0 Value Menentukan periode pulsa PWM
}
```

5. Membuat sub rutin agar robot bergerak maju

```
void maju()
       {
       dirA_Ki=1;dirB_Ki=0;            // Motor kiri maju
       dirC_Ka=1;dirD_Ka=0;            // Motor kanan maju
       }
```

6. Membuat sub rutin agar robot belok ke kiri

```
void belok_kiri()
       {
       unsigned int i;
       lpwm=50;       rpwm=50;              // Kecepatan pelan
       delay_ms(60);                        // Robot dimajukan sedikit
              dirA_Ki=0;dirB_Ki=1;          // Motor kiri mundur
              dirC_Ka=1;dirD_Ka=0;          // Motor kanan maju
       for(i=0;i<=1000;i++)   while (!SkiXX ||!SkiX) {};
       for(i=0;i<=1000;i++)   while ( SkiXX || SkiX) {};
       lpwm=0; rpwm=0;                      // Robot berhenti
       }
```

Pada program diatas, tampak ada 2 “for while” yang masing-masing diulang 1000 kali untuk memastikan bahwa sensor benar-benar membaca sebuah garis, bukan noda atau kotoran yang ada di lapangan.

- `for(i=0;i<=1000;i++) while (!SkiXX ||!SkiX) {};`
  Robot akan terus belok kiri selama sensor SkiXX=0 atau SkiX=0 (sensor berada diatas garis hitam)
- `for(i=0;i<=1000;i++) while ( SkiXX || SkiX) {};`
  Selanjutnya robot tetap belok kiri selama sensor SkiXX=1 atau SkiX=1 (sensor berada diatas permukaan putih)

7. Membuat sub rutin agar robot belok ke kanan

```
void belok_kanan()
        {
        unsigned int i;
        lpwm=50;        rpwm=50;                // Kecepatan pelan
        delay_ms(60);                           // Robot dimajukan sedikit
                dirA_Ki=1;dirB_Ki=0;            // Motor kiri maju
                dirC_Ka=0;dirD_Ka=1;            // Motor kanan mundur
        for(i=0;i<=1000;i++)    while (!SkaXX ||!SkaX) {};
        for(i=0;i<=1000;i++)    while ( SkaXX || SkaX) {};
        lpwm=0; rpwm=0;                         // Robot berhenti
        }
```

8. Membuat sub rutin membaca line

```
unsigned char sensor;
void scan_rule1()
{       maju();                 // Robot bergerak maju
        sensor=PIND;            // PIND diberi nama sensor
        sensor&=0b00111111;     // sensor di-AND-kan dengan 0b00111111
        switch(sensor)
        {       case 0b00111110:        rpwm=0;         lpwm=200;       x=1;    break;
                case 0b00111100:        rpwm=50;        lpwm=200;       x=1;    break;
                case 0b00111101:        rpwm=75;        lpwm=200;       x=1;    break;
                case 0b00111001:        rpwm=100;       lpwm=200;       x=1;    break;
                case 0b00111011:        rpwm=150;       lpwm=200;       x=1;    break;
                case 0b00110011:        rpwm=200;       lpwm=200;               break;
                case 0b00110111:        rpwm=200;       lpwm=150;       x=0;    break;
                case 0b00100111:        rpwm=200;       lpwm=100;       x=0;    break;
                case 0b00101111:        rpwm=200;       lpwm=75;        x=0;    break;
                case 0b00001111:        rpwm=200;       lpwm=50;        x=0;    break;
                case 0b00011111:        rpwm=200;       lpwm=0;         x=0;    break;
                case 0b00111111:                                                break;
                        if(x)           {lpwm=50;       rpwm=0;                 break;}
                        else            {lpwm=0;        rpwm=50;                break;}
        }
}
```

Variabel x ini berfungsi sebagai pengingat posisi terakhir robot terhadap garis. Jika robot berada di kanan garis, maka x=0. Jika robot berada di kiri garis, maka x=1. Ketika robot lepas dari track, maka program akan membaca kondisi variable x, sehingga dapat ditentukan arah gerak robot agar robot dapat kembali ke garis, seperti terlihat pada instruksi berikut :

```
        if(x)           {lpwm=50;       rpwm=0;                 break;}
        else            {lpwm=0;        rpwm=50;                break;}
```

Jika x=1 maka robot belok kanan, jika x=0 maka robot belok kiri.

9. Membuat sub rutin membaca persimpangan

```
void scan_count(unsigned char count)
{ unsigned int i;
  unsigned char xx=0;

  while(xx<count)
  {       for(i=0;i<1000;i++) while((sensor & 0b00011110)!=0b00000000) scan_rule1();
          for(i=0;i<1000;i++) while((sensor & 0b00011110)==0b00000000) scan_rule1();
          xx++;
```

```
    }
}
```

Variable count digunakan untuk menentukan jumlah persimpangan yang harus dilewati. Variable xx berisi jumlah persimpangan yang telah dilewati. Nilainya akan bertambah 1 ketika kondisi 4 sensor tengah membaca garis hitam semua kemudian membaca garis putih semua.

10. Membuat main program

```
void main(void)
{
.
.
.
.
while (1)
      {
      // Place your code here
       scan_count(3);                // Maju 3 persimpangan
       belok_kanan();                // Belok kanan

      };
}
```

Lampiran A. Source Code " Line Tracker Robot"

```
/*****************************************
This program was produced by the
CodeWizardAVR V1.24.0 Standard
Automatic Program Generator
© Copyright 1998-2003 HP InfoTech s.r.l.
http://www.hpinfotech.ro
e-mail:office@hpinfotech.ro

Project :
Version :
Date    : 17/11/2007
Author  : hendawan
Company :
Comments:


Chip type           : ATmega8535L
Program type        : Application
Clock frequency     : 11,059200 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 128
*****************************************/
#define SkiXX          PIND.0
#define SkiX           PIND.1
#define Ski            PIND.2
#define Ska            PIND.3
#define SkaX           PIND.4
#define SkaXX          PIND.5

#define EnKi           PORTC.4
#define dirA_Ki        PORTC.0
#define dirB_Ki        PORTC.1
#define EnKa           PORTC.5
#define dirC_Ka        PORTC.2
#define dirD_Ka        PORTC.3
#include <mega8535.h>
#include <delay.h>
bit x;
unsigned char xcount,lpwm,rpwm;
// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Place your code here
 xcount++;
   if(xcount<=lpwm)EnKi=1;
   else EnKi=0;
   if(xcount<=rpwm)EnKa=1;
   else EnKa=0;
   TCNT0=0xFF; // Timer0 Value Menentukan periode pulsa PWM
}
void maju()
        {
        dirA_Ki=1;dirB_Ki=0;
        dirC_Ka=1;dirD_Ka=0;
        }
void belok_kiri()
        {
        unsigned int i;
        lpwm=50;       rpwm=50;
        delay_ms(60);          // dimajukan sedikit
        dirA_Ki=0;dirB_Ki=1;
        dirC_Ka=1;dirD_Ka=0;
        for(i=0;i<=1000;i++)   while (!SkiXX ||!SkiX) {};
        for(i=0;i<=1000;i++)   while ( SkiXX || SkiX) {};
        lpwm=0; rpwm=0;
        }
void belok_kanan()
        {
        unsigned int i;
        lpwm=50;       rpwm=50;
        delay_ms(60);          // dimajukan sedikit
        dirA_Ki=1;dirB_Ki=0;
```

```
        dirC_Ka=0;dirD_Ka=1;
        for(i=0;i<=1000;i++)   while (!SkaXX ||!SkaX) {};
        for(i=0;i<=1000;i++)   while ( SkaXX || SkaX) {};
        lpwm=0; rpwm=0;
        }
// Declare your global variables here
unsigned char sensor;
void scan_rule1()
{       maju();
        sensor=PIND;
        sensor&=0b00111111;
        switch(sensor)
        {       case 0b00111110:       rpwm=0;         lpwm=200;       x=1;    break;
                case 0b00111100:       rpwm=50;        lpwm=200;       x=1;    break;
                case 0b00111101:       rpwm=75;        lpwm=200;       x=1;    break;
                case 0b00111001:       rpwm=100;       lpwm=200;       x=1;    break;
                case 0b00111011:       rpwm=150;       lpwm=200;       x=1;    break;
                case 0b00110011:       rpwm=200;       lpwm=200;               break;
                case 0b00110111:       rpwm=200;       lpwm=150;       x=0;    break;
                case 0b00100111:       rpwm=200;       lpwm=100;       x=0;    break;
                case 0b00101111:       rpwm=200;       lpwm=75;        x=0;    break;
                case 0b00001111:       rpwm=200;       lpwm=50;        x=0;    break;
                case 0b00011111:       rpwm=200;       lpwm=0;         x=0;    break;
                case 0b00111111:                                               break;
                        if(x)          {lpwm=50;       rpwm=0;                 break;}
                        else           {lpwm=0;        rpwm=50;                break;}
        }
}
void scan_count(unsigned char count)
{       unsigned int i;
        unsigned char xx=0;

        while(xx<count)
        {       for(i=0;i<1000;i++) while((sensor & 0b00011110)!=0b00000000) scan_rule1();
                for(i=0;i<1000;i++) while((sensor & 0b00011110)==0b00000000) scan_rule1();
                xx++;
        }
}
void main(void)
{
// Declare your local variables here

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

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

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

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

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 10,800 kHz
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x05;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
```

```
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

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

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x01;

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

// Global enable interrupts
#asm("sei")

while (1)
      {
      // Place your code here
        scan_count(3);
        belok_kanan();

      };
}
```