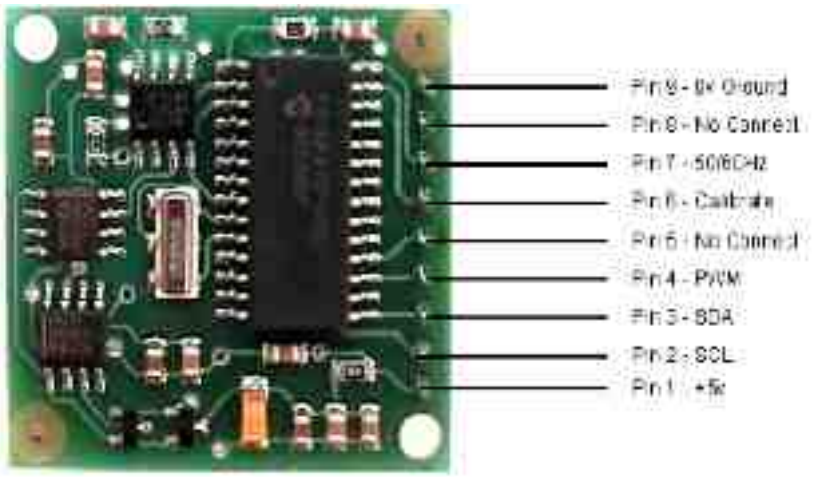
```
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

MCUCR=0x00;
MCUCSR=0x00;

TIMSK=0x00;

ACSR=0x80;
SFIOR=0x00;

lcd_init(16);

while (1)
    {
    // Place your code here
      if (UVITRON==0)        //UVTron dapet
      {
         lcd_gotoxy(0,0);
lcd_putsf("ada Api");
}
else
{
   lcd_clear();
}

    };
}
```

4. Sensor Kompas

Kompas modul telah secara spesifik dirancang untuk digunakan dalam robot sebagai bantuan untuk navigasi. Tujuannya adalah untuk menghasilkan data unik untuk mewakili arah robot menghadap. kompas menggunakan sensor KMZ51 Philips medan magnet, yang cukup sensitif untuk mendeteksi medan magnet Bumi. Hasil dari dua dari mereka dipasang pada sudut kanan satu sama lain digunakan untuk menghitung arah komponen horizontal medan Bumi magnet. Kami memiliki contoh menggunakan modul Kompas dengan berbagai kontroler populer. Koneksi ke kompas modul.



Gambar 6.9 Modul kompas

Kompas modul memerlukan power supply 5V pada 15mA nominal. Ada dua cara penggunaan dari modul. Sebuah sinyal PWM tersedia pada pin 4, atau antarmuka I2C disediakan pada pin 2,3. Sinyal PWM adalah pulsa lebar sinyal dimodulasi dengan lebar positif dari pulsa mewakili sudut. Lebar pulsa bervariasi dari 1mS (0 °) untuk 36.99mS (359,9 °) - dengan kata lain 100uS / ° dengan 1 mS offset. Sinyal rendah untuk 65mS antara pulsa, jadi waktu siklus adalah 65mS + lebar pulsa - ie.66ms-102ms. Denyut nadi dihasilkan oleh timer

16 bit di prosesor pemberian resolusi 1US, namun saya tidak akan merekomendasikan mengukur ini untuk sesuatu lebih baik dari 0,1 ° (10uS). Pastikan Anda menghubungkan pin I2C, SCL dan SDA, untuk pasokan 5V jika Anda menggunakan PWM, seperti tidak ada pull-up resistor pada pin ini. Pin 2,3 merupakan antarmuka I2C dan dapat digunakan untuk mendapatkan readout langsung dari modul. Jika antarmuka I2C tidak digunakan maka pin ini harus ditarik tinggi (ke +5 v) melalui beberapa resistor. Sekitar 47 kohm.

Berikut program sensor kompas menggunakan modul KMZ51 Philips.

```
#include <mega8535.h>
#include <delay.h>

// I2C Bus functions
#asm
  .equ __i2c_port=0x18 ;PORTB
  .equ __sda_bit=0
  .equ __scl_bit=1
#endasm
#include <i2c.h>

// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>

// Declare your global variables here


#define cmps_add 0xC0
unsigned char cmps;
unsigned char cmps_read(unsigned char add)
{
 unsigned char data;
 i2c_start();
```

```
  i2c_write(cmps_add);
  i2c_write(add);
  i2c_start();
  i2c_write(cmps_add|1);
  data=i2c_read(0);
  i2c_stop();
  return data;
}

void tampil(unsigned char dat)
{
        unsigned char data;
        //lcd_gotoxy(0,0);

        data = dat / 100;
        data+=0x30;
        lcd_putchar(data);

        dat%=100;
        data = dat / 10;
        data+=0x30;
        lcd_putchar(data);

        dat%=10;
        data = dat + 0x30;
        lcd_putchar(data);
        lcd_putchar(' ');
}



void main(void)
{
PORTA=0x00;
DDRA=0x00;
```

```
PORTB=0x00;
DDRB=0x00;

PORTC=0x00;
DDRC=0x00;

PORTD=0x00;
DDRD=0x00;

TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

MCUCR=0x00;
MCUCSR=0x00;

TIMSK=0x00;
```

```
ACSR=0x80;
SFIOR=0x00;

i2c_init();

lcd_init(16);

while (1)
    {
    cmps=cmps_read(1);
    lcd_gotoxy(0,0);
    tampil(cmps);
    delay_ms(20);
    };
}
```

6.2 Kontrol Embedded

Kontrol Embedded merupakan sebuah komponen elektronika yang tertanam dalam sebuah robot. Komponen elektronika tersebut disebut mikrokontroler. Mikrokontroler memiliki Berbagai jenis dan bentuk. Namun, yang sering digunakan dalam dunia robotic yaitu mikrokontroler berjenis Atmel, AVR, PIC, dll.

Perbedaan antara mikroprosesor dan mikrokontroler adalah sebagai berikut :

1. Mikroprosesor : adalah bagian dari CPU dari sebuah computer, tanpa memori tanpa I/O dan Peripheral. Contoh 8088 dan 80x86. Untuk dapat bekerja mikroprosesor membutuhkan perangkat pendukung berupa RAM, ROM, dan I/O
2. Mikrokontroler atau yang kita kenal dengan *Single Chip* mengkombinasikan CPU dengan memori dan I/O. Dengan demikian suatu mikrokontroler tidak membutuhkan tambahan RAM, ROM dan I/O

Perlengkapan Dasar Mikroprosesor/Mikrokontroler

1. CPU (Central Processing Unit)
2. Alamat/Address
3. Data

4. Pengendali
5. Memori
6. RAM
7. ROM
8. Input/Output

Berikut Penjelasan mengenai mikrokontroler tersebut.

6.2.1 Mikrokontroler Atmel (AVR ATmega 16)

AVR merupakan seri mikrokontroler CMOS 8-bit buatan Atmel, berbasis arsitektur RISC (Reduced Instruction Set Computer) yang ditingkatkan. Hampir semua instruksi dieksekusi dalam satu siklus clock. AVR mempunyai 32 register general-purpose, timer/counter fleksibel dengan mode compare, interrupt internal dan eksternal, serial UART, programmable Watchdog Timer, dan mode power saving. Mempunyai ADC dan PWM internal. AVR juga mempunyai In-System Programmable Flash on-chip yang mengijinkan memori program untuk diprogram ulang dalam sistem menggunakan hubungan serial SPI. ATmega16 adalah mikrokontroler CMOS 8-bit daya-rendah berbasis arsitektur RISC yang ditingkatkan. Untuk lebih jelas tentang arsitektur dari ATmega16 ditunjukan pada gambar. [5]

ATMega16 mempunyai throughput mendekati 1 MIPS per MHz membuat disainer sistem untuk mengoptimasi komsumsi daya versus kecepatan proses.

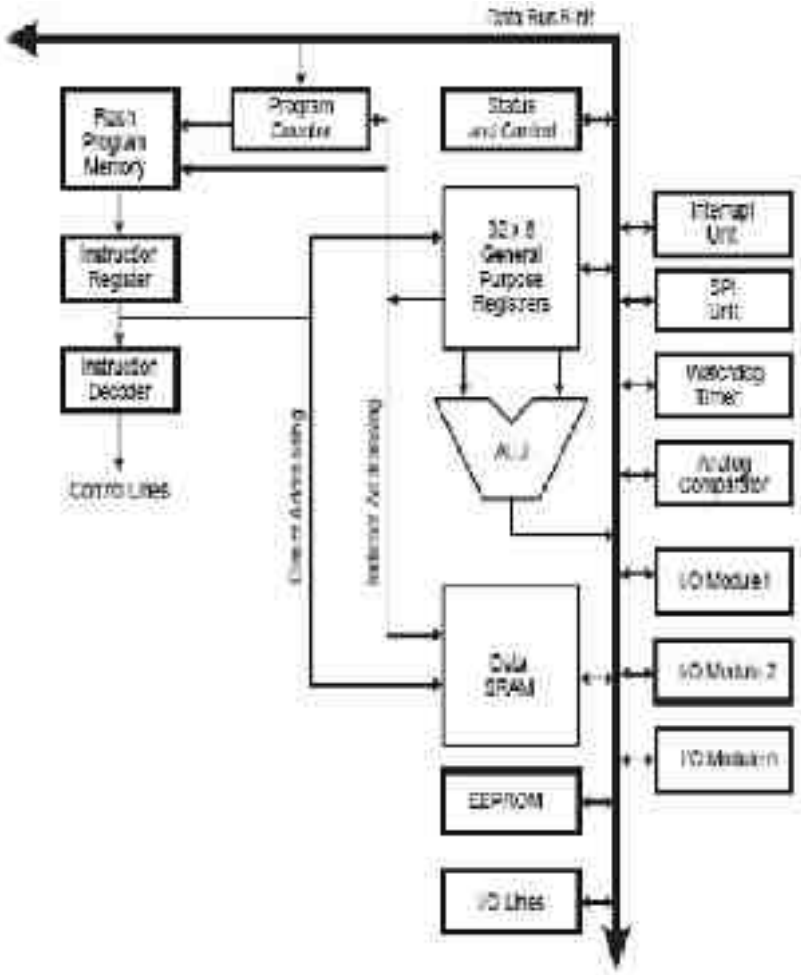
Beberapa keistimewaan dari AVR ATMega16 antara lain :

1. Advanced RISC Architecture
   - 130 Powerful Instructions – Most Single Clock Cycle Execution
   - 32 x 8 General Purpose Fully Static Operation
   - Up to 16 MIPS Throughput at 16 MHz
   - On-chip 2-cycle Multiplier
2. Nonvolatile Program and Data Memories
   - 8K Bytes of In-System Self-Programmable Flash
   - Optional Boot Code Section with Independent Lock Bits
   - 512 Bytes EEPROM

- 512 Bytes Internal SRAM

3. Peripheral Features
   - Two 8-bit Timer/Counters with Separate Prescalers and Compare Mode
   - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
   - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
   - Real Time Counter with Separate Oscillator
   - Four PWM Channels
   - 8-channel, 10-bit ADC
   - Byte-oriented Two-wire Serial Interface
   - Programmable Serial USART
4. Special Microcontroller Features
   - Power-on Reset and Programmable Brown-out Detection
   - Internal Calibrated RC Oscillator
   - External and Internal Interrupt Sources
   - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
5. I/O and Package
   3. 32 Programmable I/O Lines
   4. 40-pin PDIP, 44-lead TQFP, 44-lead PLCC, and 44-pad MLF
6. Operating Voltages
   1. 2.7 - 5.5V for ATmega16L

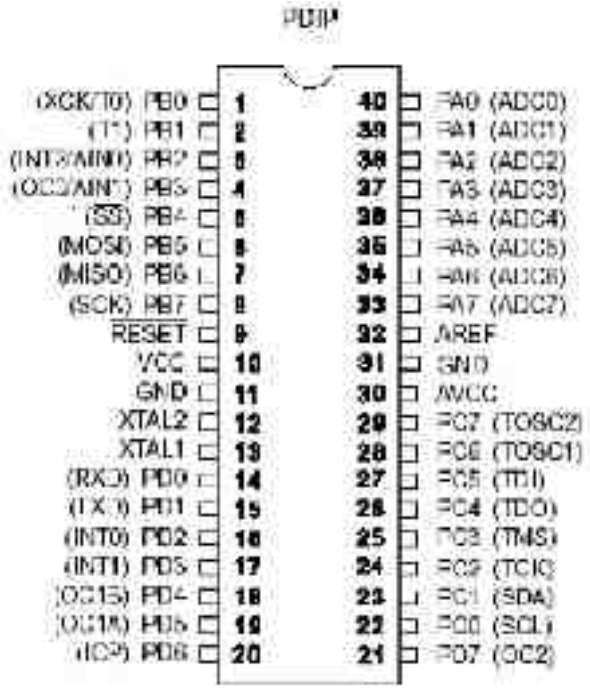
   4.5 - 5.5V for Atmega16



Gambar 6.10 Blok Diagram Arsitektur ATMega16

I. Konfigurasi Pin AVR ATMega 16

Pin-pin pada ATMega16 dengan kemasan 40-pin DIP (dual in- line package) ditunjukkan oleh gambar. Kemasan pin tersebut terdiri dari 4 PortyaituPort A, Port B, Port C,Port D yang masing masingPort terdiri dari 8 buah pin. Selain itu juga terdapat RESET, VCC, GND 2 buah, AVCC, XTAL1, XTAL2 dan AREF.



Gambar 6.11 Pin-pin ATMega16 kemasan 40-pin

Diskripsi dari pin-pin ATMega 16 adalah sebagai berikut :

1. VCC : Supply tegangan digital.

2. GND : Ground

3. Port A : Port A sebagai input analog ke A/D konverter. Port A juga sebagai 8-bit bi-directional port I/O, jika A/D konverter tidak digunakan. Pin-pin port dapat menyediakan resistor-resistor internal pull-up. Ketika port A digunakan sebagai input dan pull eksternal yang rendah akan menjadi sumber arus jika resistor-resistor pull-up diaktifkan. Pin-pin port A adalah tri-state ketika kondisi reset menjadi aktif sekalipun clock tidak aktif.

4. Port B : Port B adalah port I/O 8-bit bi-directional dengan resistor-resistor internal pull-up. Buffer output port B mempunyai karaketristik drive yang simetris dengan kemampuan keduanya sink dan source yang tinggi. Sebagai input, port B yang mempunyai pull eksternal yang rendah akan menjadi sumber arus jika resistor-

resistor pull-up diaktifkan. Pin-pin port B adalah tri-state ketika kondisi reset menjadi aktif sekalipun clock tidak aktif.

5. Port C : Port C adalah port I/O 8-bit bi-directional dengan resistor-resistor internal pull-up. Buffer output port C mempunyai karaketristik drive yang simetris dengan kemampuan keduanya sink dan source yang tinggi. Sebagai input, port C yang mempunyai pull eksternal yang rendah akan menjadi sumber arus jika resistor-resistor pull-up diaktifkan. Pin-pin port C adalah tri-state ketika kondisi reset menjadi aktif seklipun clock tidak aktif. Jika antarmuka JTAG enable, resistor-resistor pull-up pada pin-pin PC5(TDI), PC3(TMS), PC2(TCK) akan diktifkan sekalipun terjadi reset.

6. Port D : Port D adalah port I/O 8-bit bi-directional dengan resistor-resistor internal pull-up. Buffer output port D mempunyai karaketristik drive yang simetris dengan kemampuan keduanya sink dan source yang tinggi. Sebagai input, port D yang mempunyai pull eksternal yang rendah akan menjadi sumber arus jika resistor-resistor pull-up diaktifkan. Pin-pin port D adalah tri-state ketika kondisi reset menjadi aktif seklipun clock tidak aktif.

7. Reset : Sebuah low level pulsa yang lebih lama daripada lebar pulsa minimum pada pin ini akan menghasilkan reset meskipun clock tidak berjalan.

8. XTAL1 : Input inverting penguat Oscilator dan input intenal clock operasi rangkaian.

9. XTAL2 : Output dari inverting penguat Oscilator.

10. AVCC : Pin supply tegangan untuk PortA dan A/D converter . Sebaiknya eksternalnya dihubungkan ke VCC meskipun ADC tidak digunakan. Jika ADC digunakan seharusnya dihubungkan ke VCC melalui low pass filter.

11. AREF : Pin referensi analog untuk A/D konverter.

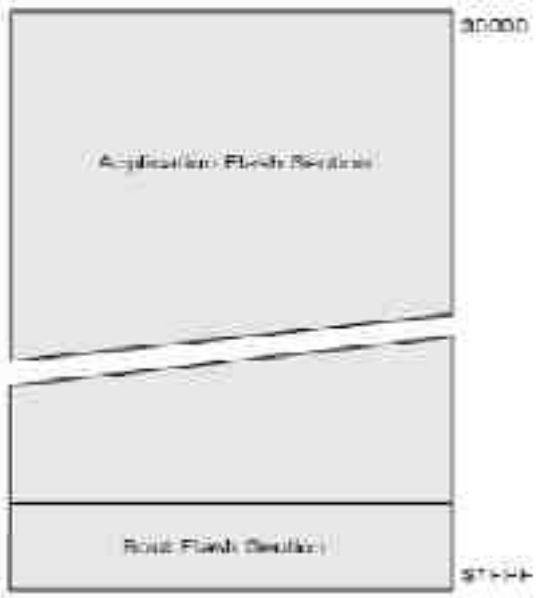
II. Struktur Memori

Untuk memaksimalkan performa dan paralelisme, AVR menggunakan arsitektur Harvard (dengan memori dan bus terpisah untuk program dan data). Instruksi pada memori program dieksekusi dengan pipelining single level.

Selagi sebuah instruksi sedang dikerjakan, instruksi berikutnya diambil dari memori program.

1. Memori Pengirim

ATMega16 memiliki 16K byte flash memori dengan lebar 16 atau 32 bit. Kapasitas memori itu sendiri terbagi manjadi dua bagian yaitu bagian mulai program dan bagian aplikasi program.



Gambar 6.12 Peta Memori Pengirim [5]

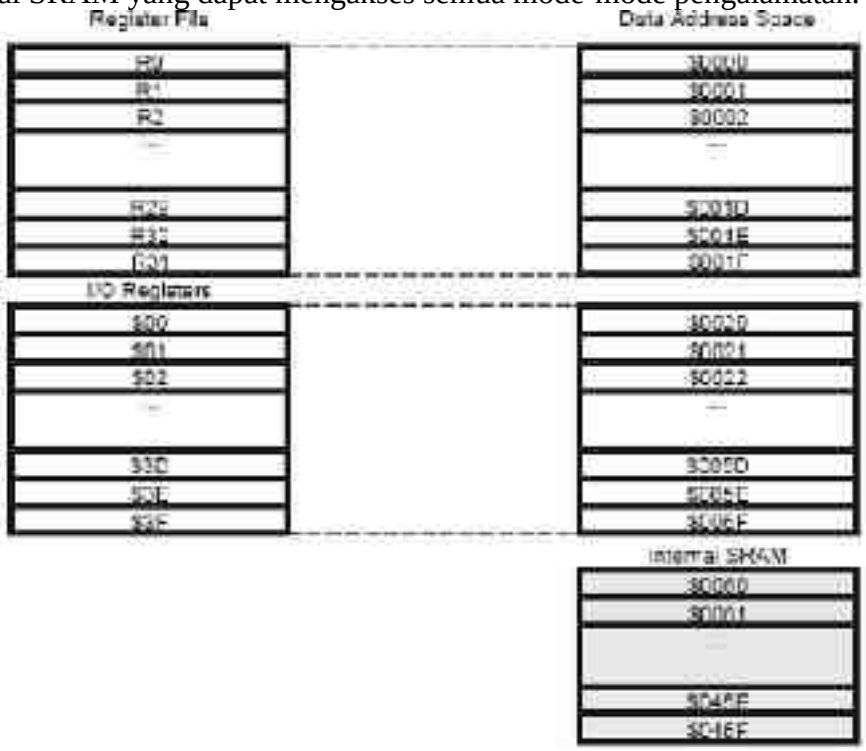
Memori pengirim memiliki kemampuan mencapai 10.000 tulis dan hapus.

2. Memori SRAM

Penempatan memori data yang lebih rendah dari 1120menunjukkan register, I/O memori, dan data internal SRAM. 96 alamat memori pertama untuk file register dan memori I/O, dan 1024 alamat memori berikutnya untuk data internal SRAM.

Lima mode pengalamatan yang berbeda pada data memori yaitu direct, indirect, indirect dis-placement, indirect pre-decreament dan indirect post-increament .Pada file register, mode indirect mulai dari register R26-R31. Pengalamatan mode direct mencapai keseuruhan kapasitas data. Pengalamatan mode indirect dis-placement mencapai 63 alamat memori dari register X atau Y. Ketika meggunakan mode pengalamatan indirect dengan pre-decrement dan post increment register X, Y, dan Z akan di-dicrement-kan atau di-increment-kan.

Pada ATmega16 memiliki 32 register, 64 register I/O dan 1024 data internal SRAM yang dapat mengakses semua mode-mode pengalamatan.



Gambar 6.13 Peta Memori SRAM

3. Memori EEPROM

Pada EEPROM ATmega16 memiliki memori sebesar 512 byte dengan daya tahan 100.000 siklus write/read.

Register-register pada memori EEPROM :

- Bit 15..9– Res:reserved bits

Bit ini sebagai bit-bit bank pada ATmega16 dan akan selalu membaca zero.

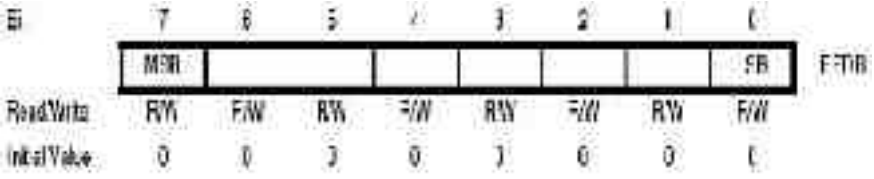
Gambar 6.14 Register alamat EEPROM Bit 15...8 [5]

- Bit8..0– EEAR8..0:EEPROM address

Bit-bit ini sebagai alamat EEPROM.

- Bit 7..0– EEDR7..0:EEPROM data

Bit-bit ini sebagai data EEPROM.

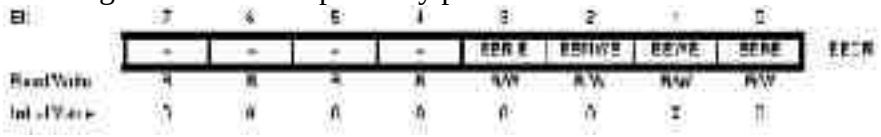


Gambar 6.15 Register data EEPROM Bit Bit 7...0 [5]

- Bit 7..4– Res:reserved bits

Bit-bit ini terdapat pada register kontrol.

Bit ini sebagai Enable Interupt Ready pada EEPROM.



Gambar 6.16 Register kontrol EEPROM Bit Bit 7...0 [5]

- Bit 2

Bit ini sebagai Enable Interupt Master pada EEPROM.

- Bit 1

Bit ini sebagai write enable pada EEPROM.

- Bit 0

Bit ini sebagai read enable pada EEPROM.

III. Interupsi

Interupsi merupakan suatu peristiwa yang menyebabkan mikrokontroller menghentikan program sejenak untuk mengerjakan proses interupsi tersebut. Setelah interrupt yang dikerjakan tersebut telah selesai maka mikrokontroller akan kembali ke rutin program yang telah dihentikan sejenak.

Interupsi yang mempunyai prioritas tertinggi tidak dapat diganggu oleh program yang lain maupun interupsi yang mempunyai prioritas lebih rendah. Selama proses interupsi dijalankan maka Program Counter (alamat dari instruksi yang sedang berjalan) disimpan ke stack. Berikut merupakan prioritas interupsi dari prioritas tinggi sampai prioritas rendah.

Tabel 6.1 Prioritas Interrupt [5]

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow
11	$014	SPI, STC	Serial Transfer Complete
12	$016	USART, RXC	USART, Rx Complete
13	$018	USART, UDRE	USART Data Register Empty
14	$01A	USART, TXC	USART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface
19	$024	INT2	External Interrupt Request 2
20	$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	$028	SPM_RDY	Store Program Memory Ready

IV. Port sebagai input/output digital

ATmega16 mempunyai empat buah port yang bernama PortA, PortB, PortC, dan PortD. Keempat port tersebut merupakan jalur bi-directional dengan pilihan internal pull-up. Tiap port mempunyai tiga buah register bit, yaitu DDxn, PORTxn, dan PINxn. Huruf ‘x’mewakili nama huruf dari port sedangkan huruf ‘n’ mewakili nomor bit. Bit DDxn terdapat pada I/O address DDRx, bit PORTxn terdapat pada I/O address PORTx, dan bit PINxn terdapat pada I/O address PINx. Bit DDxn dalam regiter DDRx (Data Direction Register) menentukan arah pin. Bila DDxn diset 1 maka Px berfungsi sebagai pin output. Bila DDxn diset 0 maka Px berfungsi sebagai pin input. Bila PORTxn diset 1 pada saat pin terkonfigurasi sebagai pin input, maka resistor pull-up akan diaktifkan. Untuk mematikan resistor pull-up, PORTxn harus diset 0 atau pin dikonfigurasi sebagai pin output. Pin port adalah tri-state setelah kondisi reset. Bila PORTxn diset 1 pada saat pin terkonfigurasi sebagai pin output maka pin port akan berlogika 1. Dan bila PORTxn diset 0 pada saat pin terkonfigurasi sebagai pin output maka pin port akan berlogika 0. Saat

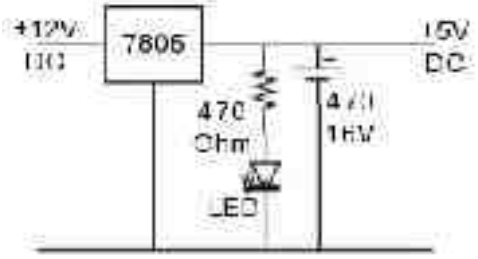
mengubah kondisi port dari kondisi tri-state (DDxn=0, PORTxn=0) ke kondisi output high (DDxn=1, PORTxn=1) maka harus ada kondisi peralihan apakah itu kondisi pull-up enabled (DDxn=0, PORTxn=1)atau kondisi output low (DDxn=1, PORTxn=0). Biasanya, kondisi pull-up enabled dapat diterima sepenuhnya, selama lingkungan impedansi tinggi tidak memperhatikan perbedaan antara sebuah strong high driver dengan sebuah pull-up. Jika ini bukan suatu masalah, maka bit PUD pada register SFIOR dapat diset 1 untuk mematikan semua pull-up dalam semua port. Peralihan dari kondisi input dengan pull-up ke kondisi output low juga menimbulkan masalah yang sama. Kita harus menggunakan kondisi tri-state (DDxn=0, PORTxn=0) atau kondisi output high (DDxn=1, PORTxn=0) sebagai kondisi transisi.

Tabel 6.2 Konfigurasi pin port [5]

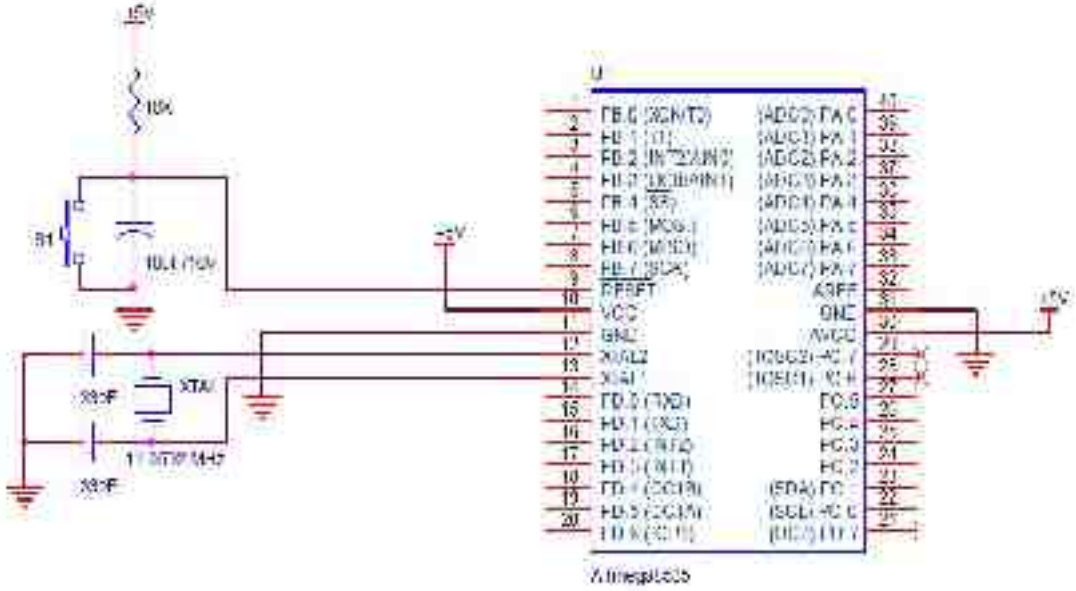
DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

V. Power Supply

Rangkaian Power Supply dapat menggunakan rangkaian berikut :



VI. Rangkaian Min-Sys AVR ATmega8535

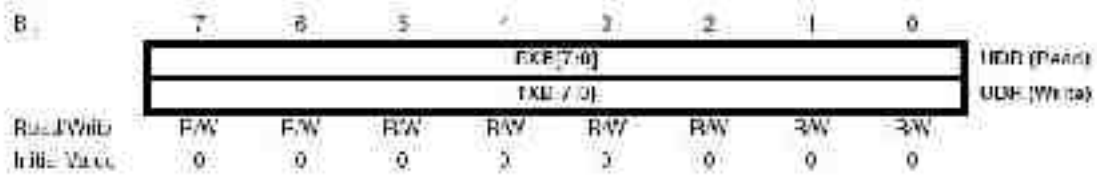


6.2.2 USART Pada Mikrokontroler AVR

Selain timer, fitur dan fasilitas yang terdapat dalam mikrokontroler AVR yang sering digunakan adalah USART. USART (Universal Synchronous Asynchronous serial Receiver and Transmitter) merupakan protokol komunikasi serial yang terdapat pada mikrokontroler AVR. Dengan memanfaatkan fitur ini kita dapat berhubungan dengan "dunia luar". Dengan USART kita bisa menghubungkan mikrokontroler dengan PC, hape, GPS atau bahkan modem, dan banyak lagi peralatan yang dapat dihubungkan dengan mikrokontroler dengan menggunakan fasilitas USART.

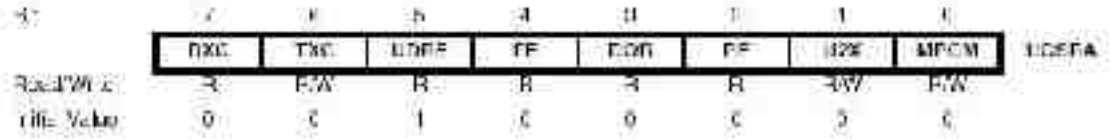
Komunikasi dengan menggunakan USART dapat dilakukan dengan dua cara yaitu dengan mode sinkron dimana pengirim data mengeluarkan

pulsa/clock untuk sinkronisasi data, dan yang kedua dengan mode asinkron, dimana pengirim data tidak mengeluarkan pulsa/clock, tetapi untuk proses sinkronisasi memerlukan inisialisasi, agar data yang diterima sama dengan data yang dikirimkan. Pada proses inisialisasi ini setiap perangkat yang terhubung harus memiliki baud rate (laju data) yang sama. Pada mikrokontroler AVR untuk mengaktifkan dan mengeset komunikasi USART dilakukan dengan cara mengaktifkan register-register yang digunakan untuk komunikasi USART. Register-register yang dipakai antara lain:



Gambar 6.17 Register UDR…. USART

UDR : merupakan register 8 bit yang terdiri dari 2 buah dengan alamat yang sama, yang digunakan sebagai tempat untuk menyimpan data yang akan dikirimkan (TXB) atau tempat data diterima (RXB) sebelum data tersebut dibaca.



Gambar 6.18 Register UCSRA

UCSRA: merupakan register 8 bit yang digunakan untuk mengendalikan mode komunikasi USART dan untuk membaca status yang sedang terjadi pada USART.

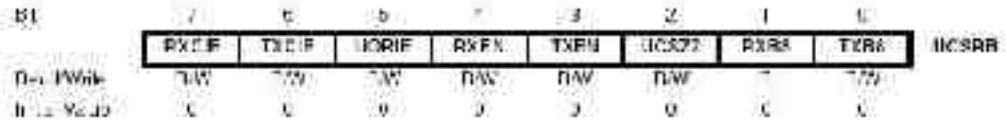
Bit **RXC** [status]—> akan “1″ bila ada data di UDR (RXB) yang belum terbaca. Dapat digunakan untuk sumber interupsi, dengan mengeset RXCIE

Bit **TXC**[status]—> akan “1″ bila ada data di UDR (TXB) yang sudah dikirimkan. Dapat digunakan untuk sumber interupsi, dengan mengeset TXCIE

Bit **UDRE**[status]**—>**akan “1″ bila UDR siap untuk menerima data baru.

Bit **U2X**[kendali]—> diisi “1″ bila kecepatan transmisi data ingin dinaikkan 2kali.

Bit **MPCM**[kendali]—>digunakan bila ingin menggunakan komunikasi multiprosesor.



Gambar 6.19 Register UCSRB

UCSRB: merupakan register 8 bit yang digunakan untuk mengendalikan mode komunikasi USART dan untuk membaca status yang sedang terjadi pada USART.

Bit **RXCIE**[kendali]—>digunakan untuk mengaktifkan interupsi yang bersumber dari RXC.

Bit **TXCIE**[kendali]—>digunakan untuk mengaktifkan interupsi yang bersumber dari TXC.

Bit **UDRIE**[kendali]—>digunakan untuk mengaktifkan interupsi yang bersumber dari UDRE.

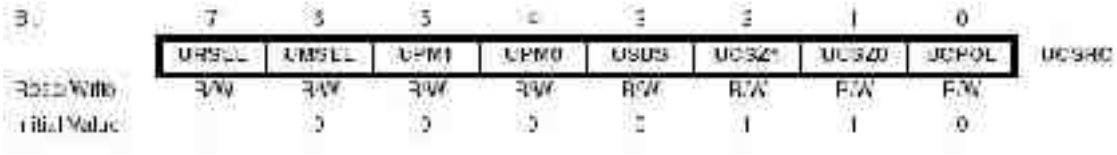
Bit **RXEN**[kendali]—>digunakan untuk mengaktifkan receiver.

Bit **TXEN**[kendali]—>digunakan untuk mengaktifkan transmitter.

Bit **UCSZ2**[kendali]—>digunakan untuk menentukan panjang data yang dikirim dalam sekali. Digunakan bersama2 dengan UCSZ1,UCSZ0 pada UCSRC.

Bit **RXB8**[status]—>digunakan sebagai penampung data ke 9 pada penerimaan data dengan 9 bit

Bit **TXB8**[status]—>digunakan sebagai penampung data ke 9 pada transmisi data dengan 9 bit



Gambar 6.20 Register UCSRC

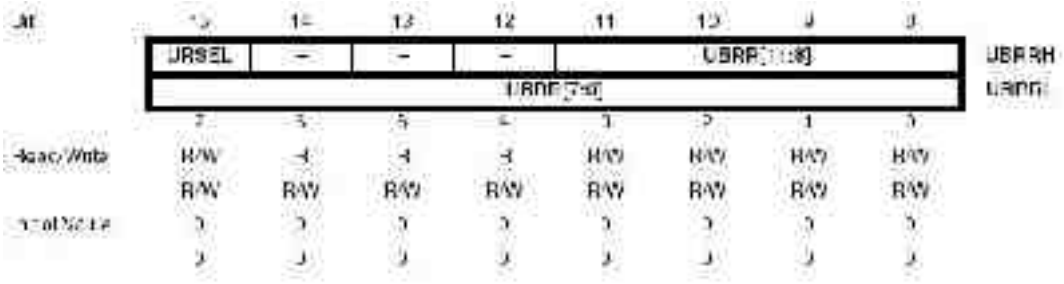
UCSC: merupakan register 8 bit yang digunakan untuk mengendalikan mode komunikasi USART dan untuk membaca status yang sedang terjadi pada USART. Bit **URSEL**[kendali]—>digunakan untuk memilih register pada UCSRC dan UBRRH. Dua register ini memiliki alamat yang sama, sehingga untuk proses penulisan memerlukan bantuan URSEL. Bila URSEL=1, maka register yang diisi adalah UCSRC, sedangkan bila URSEL=0, register yang

diisi adalah UBRRH. Tidak semua mikrokontroler AVR memiliki URSEL, karena ada yang memiliki register UBRRH dan UCSRC yang beda alamat

Bit **UMSEL**[kendali]—>bila “1″, maka mode yang dipilih adalah asinkron, “0″=sinkron

Bit **USBS**[kendali]—>bila “1″, maka stop bit berjumlah 2 bit

Bit **UCSZ1,UCSZ0**[kendali]—>bersama2 UCSZ2 digunakan untuk menentukan jumlah bit yang akan dikirimkan dalam sekali pengiriman data.



Gambar 6.21 Register UBRRL,H

UBRRL,H: merupakan register 16 bit yang digunakan untuk mengatur laju data (baud rate) pada saat mode komunikasi asinkron.

Berikut program USART 2 arah.

```
#include <mega8535.h>
char a;

// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x15 ;PORTC
#endasm
```

```
#include <lcd.h>

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
```

```
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer[rx_wr_index]=data;
  if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
  if (++rx_counter == RX_BUFFER_SIZE)
    {
    rx_counter=0;
    rx_buffer_overflow=1;
    };
  };
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// USART Transmitter buffer
#define TX_BUFFER_SIZE 8
char tx_buffer[TX_BUFFER_SIZE];

#if TX_BUFFER_SIZE<256
```

```
unsigned char tx_wr_index,tx_rd_index,tx_counter;
#else
unsigned int tx_wr_index,tx_rd_index,tx_counter;
#endif

// USART Transmitter interrupt service routine
interrupt [USART_TXC] void usart_tx_isr(void)
{
if (tx_counter)
  {
  --tx_counter;
  UDR=tx_buffer[tx_rd_index];
  if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
  };
}

#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
while (tx_counter == TX_BUFFER_SIZE);
#asm("cli")
if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
  {
  tx_buffer[tx_wr_index]=c;
  if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
  ++tx_counter;
  }
else
  UDR=c;
#asm("sei")
}
#pragma used-
```

```
#endif

// Standard Input/Output functions
#include <stdio.h>

// Declare your global variables here

void main(void)
{
// Declare your local variables here

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

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

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

// Port D initialization
```

```
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
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
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=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=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 19200
UCSRA=0x00;
UCSRB=0xD8;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x0C;
```

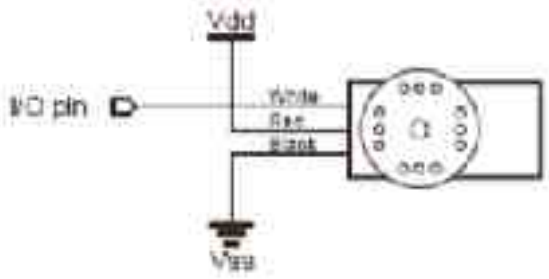
```
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);

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


while (1)
    {
    // Place your code here
    a=getchar();
    lcd_putchar(a);
    putchar(a);
    };
}
```

6.3 Aktuator

Aktuator adalah sebuah peralatan mekanis untuk menggerakkan atau mengontrol sebuah mekanisme atau sistem. Aktuator diaktifkan dengan menggunakan lengan mekanis yang biasanya digerakkan oleh motor listrik, yang dikendalikan oleh media pengontrol otomatis yang terprogram di antaranya mikrokontroler. Berikut actuator yang sering digunakan pada mobile robot.

6.3.1 Motor Servo

Motor servo adalah motor yang mampu bekerja 2 arah (searah jarum jam atau berlawanan jarum jam) dimana arah pergerakan motornya dapat dikendalikan hanya dengan memberikan pengaturan pulsa pada bagian pin kontrolnya.



Gambar 6.22 Motor Servo [4]

Motor servo sering kali digunakan sebagai alat penggerak dalam aplikasi robotika, karena di dalam motor servo telah terpasang roda gigi sehingga memiliki torsi relatif cukup kuat walaupun kecepatan motornya lambat.

Spesifikasi dari motor servo sebagai berikut:

- Catu daya : 6 VDC (maksimum).
- Waktu putar : 1,5 detik / 180 derajat (rata-rata).
- Berat fisik : 45 gram.
- Torsi putar : 3,40 kg-cm.

- Ukuran fisik : 40,5 mm (P) x 20,0 mm (L) x 38,0 mm (T).

Prinsip dasar :

Untuk membuat motor servo searah jarum jam atau berlawanan jarum jam maka harus dengan memberikan lebar pulsa 1 ms atau 2 ms yang diulang-ulang. Pulsa diatas terdiri dari dua bagian, yaitu bagian yang "tinggi/high" selebar 1 ms atau 2 ms dan bagian yang "rendah/low" selebar 20 ms .

Berikut program servo continuous buatan Parallax

```
#include <mega16.h>
#include <delay.h>
#define servo PORTC.0

// Declare your global variables here

void cw ()
{
unsigned int y;
for (y=1;y<=10;y++)
{
      servo=1;
      delay_ms(1);
      servo=0;
      delay_ms(20);
}

}

void ccw()
{
unsigned int x;
```

```
for (x=1;x<=10;x++)
   {
      servo=1;
      delay_ms(2);
      servo=0;
      delay_ms(20);
    }


}

// Declare your global variables here

void main(void)
{
// Declare your local variables here

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

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

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=Out
```

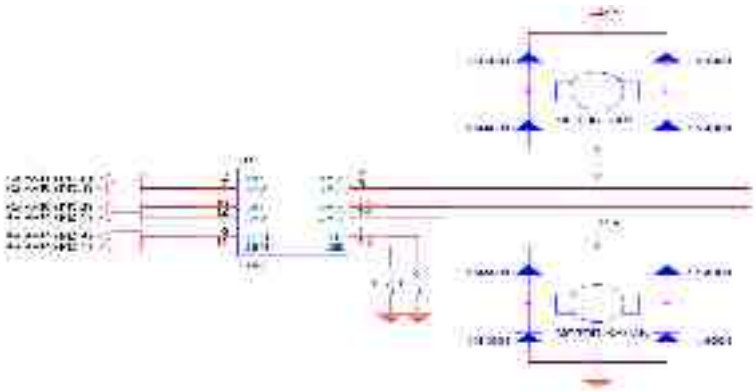
```
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=0
PORTC=0x00;
DDRC=0x01;

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

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
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
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
```

```
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=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=0x00;

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

while (1)
```

```
    {
    // Place your code here
cw();
      delay_ms(100);
ccw();
      delay_ms(100);
    };
}
```

6.3.2 Motor DC

Dalam menggerakkan motor DC dibutuhkan suatu rangkaian penggerak motor DC (Driver Motor DC). Berbagai macam jenis Driver Motor DC yaitu IC L293, L298, transistor atau Mosfet. Namun untuk mobile robot,driver motor dc yang paling umum digunakan yaitu 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. Berikut adalah rangkaian penggerak motor dc.



Gambar 6.23 Rangkaian Driver Motor DC

Berikut merupakan program untuk menggerakkan motor DC

```
#include <mega8535.h>
```

```
#include <delay.h>

#define EnKa        PORTD.2
#define DirKa_plus   PORTD.1
#define DirKa_min    PORTD.0
#define EnKi        PORTD.5
#define DirKi_plus   PORTD.4
#define DirKi_min    PORTD.3
#define sensor       PINA


void maju(){
EnKa=1;EnKi=1;
DirKa_plus=1;DirKi_plus=1;
DirKa_min=0;Di rKi_min=0;
}

void mundur(){
EnKa=1;EnKi=1;
DirKa_plus=0;DirKi_plus=0;
DirKa_min= 1;DirKi_min=1;
}

void bkir(){
EnKa=1;EnKi=1;
DirKa_plus=1;DirKi_plus=0;
DirKa_min=0; DirKi_min=1;
}

void bkan(){
```

```
EnKa=1;EnKi=1;
DirKa_plus=0;DirKi_plus=1;
DirKa_min=1; DirKi_min=0;
}

void stop(){
EnKa=1;EnKi=1;
DirKa_plus=0;DirKi_plus=1;
DirKa_min=1;DirK i_min=0;
}

void main(void)
{
// Declare your local variables here

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

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

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

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

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 500.000 kHz
// Mode: Fast PWM top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
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
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1 H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
O CR1BL=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;
[ BR]// 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
ACSR=0x80;
SFIOR=0x00;

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

while (1)
    {
  maju();
  delay_ms(100);
  mundur();
  delay_ms(100);
  bkir();
  delay_ms(100);
  bkan();
  delay_ms(100);
  stop();
  delay_ms(100);
    };
}
```

6.3.3 LCD

LCD (Liquid Crystal Display) adalah modul penampil yang banyak di gunakan karena tampilanya menarik. LCD yang paling banyak digunkan saat ini adalah LCD M1632 refrubusing refurbish karena harganya yang cukup murah. LCD M1362 merupakan modul LCD dengan tampilan 2 X 16 (2 baris dan 6 kolom) dengan asumsi daya rendah. Modul tersebut di lengkapi dengan mikrokontroler yang di desain khusus untuk menampilkan LCD. Chip HD44780 yang berfungsi sebagai pengendali LCD memiliki CGROM (Character Generator Read Only Memory), dan DDRAM (Display Data Random Acess Memory). LCD yang umum ada yang panjangnya hingga 40 karakter (2 X 40 dan 4 X 40), di mana kita menggunakan DDRAM untuk mengatur tempat penyimpanan karakter tersebut

Gambar 6.25 LCD 2x16

Tabel 6.5 LCD [3]

Display	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
Line 1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
Line 2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55
A alamat awal																						

Karakter 00H dan alamat akhir 39H, jadi alamat di baris kedua mulai dari 40H jika anda ingin meletakkan suatu karakter pada baris ke 2 kolom. Jika anda ingin meletakkan suatu karakter pada baris ke 2 kolom pertama, maka harus diset pada alamat 40H. Jadi meskipun LCD yang digunakan 2x16 atau 2x40. maka penulisan program umumnya sama aja. CGRAM merupakan memori untuk menggambarkan pola sebuah karakter dapat diubah – ubah sesuai dengan keinginan. Namun, memori akan hilang saat power supply tidak aktif sehinngga pola karakter akan hilang.

Berikut tabel pin untuk LCD M1632. perbedaan dengan LCD standar ialah pada kaki 1 VCC, dan kaki 2 GND, ini kebalikan dengan LCD standar. Anda dapat menguhubungkan pin data ke Port A. Sebagai contoh yang diberikan pada tabel pin data dihubungkan ke Port C mengikuti arsitektur SmartAVR ATmega8535.

Tabel 6.6 Susunan Kaki Led LCD Refrubish M1632

NO	Nama Pin	Deskripsi	Port
1	VCC	+ 5 V	VCC
2	GND	0 V	GND
3	VEE	Tegangan kontras LCD	Trimport
4	RS	Register select, 0 = input intruksi, 1 = input Data	PD 5
5	R/W	1 = read : 0 = write	PD 6
6	E	Enable clock	PD 7
7	D0	Data bus 0	PC 0
8	D1	Data bus 0	PC 1
9	D2	Data bus 0	PC 2
10	D3	Data bus 0	PC 3
11	D4	Data bus 0	PC 4
12	D5	Data bus 0	PC 5
13	D6	Data bus 0	PC 6
14	D7	Data bus 0	PC 7
15	Anode	Tegangan positif backlight	VCC
16	Katode	Tegangan Negatif backlight	GND

Driver LCD seperti HD44780 memiliki dua register yang aksesnya di atur menggunakan pin RS. Pada saat RS berlogika 0 register yang di akses

ialah perintah . sedangkan pada saat RS berlogika 1 register yang di akses adalah register data. D1 – D7 pada LCD berfungsi untuk menerima data pin 5 pada LCD (R/W) harus diberi logika 0, dan berlogika 1 untuk mengirimkan data ke mikrokontroler. Setiap kali menerima / mengirimkan data, untuk mengaktifkan LCD diperlukan sinya E (chip enable) dalam bentuk perpindahan logika 1 ke logika 0. Sedangkan pin RS (Registerd Selected), berguna untuk memilih instruction Registration (IR) atau data registered (DR), jika nilai RS 1 dan R/W 1, maka akan dilakukan operasi penulisan data ke DDRAM atau CGRAM. Sedangkan jika RS berlogika 1 dan R/W berlogika 1, akan membaca data dari DDRAM atau CGRAM ke register DR. karakter yang akan di tampilkan ke display mempunyai alamat tertentu pada memori DDRAM. Misalnya, alamat DDRAM 00H berisi data 30 H (nilai ASCII untuk angka 0), maka akan tampil di baris 1 kolom 1 angka 0. proses penampilannya dengan cara : controler LCD mengambil data pada alamat DDRAM 00H. data 30H digunakan sebagai alamat untuk mengambil data display LCD pada memori CGROM atau CGRAM, kemudian data pada CGROM / CGRAM di ambil dan di tampilkan ke display.

Fungsi busy flag pada LCD untuk indicator apakah controller LCD sudah siap menerima perintah atau data selanjutnya. Jika busy flag berlogika 1, maka perintah atau data yang dikirimkan oleh mikrokontroler tidak akan diproses, tetapi jika berlogika 0, maka perintah atau data yang dikrimkan oleh mikrokontroler akan diproses.

Berikut program LCD menggunakan perangkat LCD 2x16.

```
#include <mega8535.h>

// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>

// Declare your global variables here

void main(void)
{
PORTA=0x00;
DDRA=0x00;

PORTB=0x00;
DDRB=0x00;

PORTC=0x00;
DDRC=0x00;

PORTD=0x00;
DDRD=0x00;

TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
```

```
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

MCUCR=0x00;
MCUCSR=0x00;

TIMSK=0x00;

ACSR=0x80;
SFIOR=0x00;

lcd_init(16);

while (1)
    {
    // Place your code here

Lcd_putsf("Tes LCD");

    };
}
```

LATIHAN

1. Apa yang dimaksud dengan mobile robot? Sebutkan cirri-ciri mobile robot!
2. Sebutkan dan jelaskan macam-macam sensor yang digunakan pada mobile robot!
3. Jelaskan cara kerja sensor untrasonik dan inframerah! Sebutkan kelebihan dan kekurangannya masing-masing!
4. Sebutkan fungsi sensor kompas! Bagaimana cara kerja sensor kompas, jelaskan!
5. Jelaskan perbedaan antara mikroprosesor dengan mikrokontroler!
6. Pada mikrokontroler AVR ATmega 16 ada berapa port I/O? Apa cirri-ciri dan fungsi dari setiap port tersebut?
7. Apa yang dimaksud dengan USART? Bagaimana cara kerja USART dalam AVR ATmega 16?
8. Bagaimana struktur memori pada AVR ATmega16? Berapa kapasitas SRAM dan EEPROM-nya?

REFERENSI

http://id.wikipedia.org/wiki/Robot_Mobile

http://id.wikipedia.org/wiki/Aktuator

Sigit, Riyanto, “Robotika, Sensor & Aktuator”, Graha Ilmu, Yogyakarta, 2007

http://id.wikipedia.org/wiki/Robot_Mobile

www.innovativeelectronics.com, Februari 2008

www.acroname.com, April 2008

Sigit, Riyanto, “Robotika, Sensor &Aktuator”, GrahaIlmu, Yogyakarta, 2007

www.alldatasheet.com, Februari 2008

Eko Putra, Agfianto, *TeknikAntarmukaKomputer :KonsepdanAplikasi,*GrahaIlmu, Yogyakarta, 2002.

BAB VII

ROBOT VISION

7.1 Pengenalan Robot Vision

Perancangan pengolahan citra sebuah mobil robot tidak bisa dilepaskan dari perangkat sensor dan kecerdasan robot. Fitur umum semua robot yang akan dibahas lebih jauh mempunyai kemampuan untuk menerjemahkan sekuen dari rangkaian operasi penangkapan gambar selama alat itu dibutuhkan. Bagaimanapun, semua robot tersebut tidak mampu merasakan dan memberi tanggapan terhadap perubahan sesaat pada lingkungan.

Level yang lebih tinggi dari hierarki sebuah sensor ialah kemampuan untuk merasakan perubahan sesaat pada lingkungan tersebut yang kemudian diasosiasikan dengan antarmuka dan algoritma artificial intelligent. Bila digambarkan dalam bentuk diagram, siklus yang didapat adalah tertutup melalui algoritma interpolator yang kemudian merespons perintah program asli yang berisi serangkaian instruksi dengan factor koreksi yang didapat dari algoritma artificial intelligent yang sebelumnya.

Ada dua metode dasar dalam penggunaan sensor, yaitu sensor statik dan sensor siklus tertutup. Umumnya sensor-sensor yang digunakan dalam sistem robotika adalah yang dapat merasakan dan memanipulasi keadaan dengan fungsi matriks dan algoritma Artificial Intelligent (AI). Manipulator tersebut akan terus dalam keadaan statik ketika proses sensing telah dilaksanakan. Kemudian, dapat diawasi gerak yang telah dikerjakantanpa referensi yang lebih jauh dari sensor. Metode ini mengacu pada sensor statik.

Sebaliknya, dalam sensor siklus tertutup robot dikendalikan oleh alat sensing selama pergerakan manipulator. Kebanyakan sistem vision beroperasi pada mode siklus tertutup ini. Di sini sistem vision mengawasi faktor koreksi antara posisi aktual robot dengan posisi yang diinginkan. Faktor error ini kemudian digunakan untuk menggerakkan penggerak robot. Dengan sensor

siklus tertutup, bahkan ketika objek masih dalam pergerakan, sebuah robot pasti mampu melakukan transfer informasi aktual yang diinginkan.

Kebanyakan sistem vision dilengkapi dengan satu atau lebih kamera yang dihubungkan dengan prosesor vision. Prosesor ini kemudian mendigitalisasi citra yang ditangkap melalui kamera dan menganalisisnya untuk mendefinisikan objek tersebut. Aplikasi utama sistem vision adalah handling, assembly, pengklasifikasi terpisah, dan juga inspeksi. Dalam handling dan assembly, sistem digunakan untuk mengenali posisi dan orientasi dari objek yang akan menjalani assembly. Sistem tersebut juga akan mendeterminasi presensi bagian-bagian dan mendeteksi tambahan partikular objek (misalnya diameter) untuk mengecek bagian mana yang tidak memenuhi kriteria fungsionalnya lagi.

Pada pengenalan perancangan pengolahan citra pada mobil robot nantinya, citra yang telah terdigitalisasi akan disimpan dalam memori komputer vision dan diproses secara subsekuen. Komputer dapat mendeterminasi tipe objek dan orientasinya, dan mentransfer informasi yang didapat ke dalam pengendali komputer robot. Pada dasarnya, citra sebuah objek dapat direpresentasi dan disimpan dalam citra biner ataupun citra grey-scale. Dalam sistem pengolahan citra secara biner, peta duawarna dari pemandangan visual disimpan dalam komputer dengan bentukrepresentasi 1’s untuk bayangan (silhouette) dan 0’s untuk latarnya(background).

Perangkat keras preprosesor video digunakan dalam sistem vision pada citra biner untuk memperbaiki perbandingan dari derau sinyal dan untuk mengurangi sejumlah informasi yang dikirimkan ke komputer. Salah satu perintah umum yang akan dibuat oleh perangkat ini adalah mendeteksi sudut.

Perangkat video preprosesor hanya mengirimkan titik-titik transisi kepada komputer, diikuti dengan pengurangan sejumlah informasi yang akan disimpan ke dalam komputer utama. Perangkat lunak komputer utama terdiri atas algoritma pendeteksian sudut yang akan mendeterminasi sudut mana yang dimilikioleh masing-masing objek yang dilihat, dan memasukkan titik yang didapat ke dalam daftar sudut yang telah dibuat sebelumnya.

Pada perancangan pengolahan citra untuk mobile robot ini, akan banyak ditemui pemrograman perangkat keras untuk masing-masing alat yang dibutuhkan. Juga diperlukan pemrograman perangkat lunak, khususnya dalam hal pembangkitan model objek ke dalam bentuk virtualnya. Dalam kasus ini kami memakai pemrograman **C/C++, Watcom C/C++**, dan **VRML** (Virtual Reality Modelling Language), yang telah banyak digunakan karena kemudahannya membangkitkan model objek tiga dimensi.

7.1.1 Elemen Pengolahan Citra Mobil Robot

Elemen pengolahan citra pada sebuah mobile robot yang akan diimplementasikan semuanya berbasis visual. Dengan menggunakan sensor visual, maka robot ini diharapkan dapat bekerja sesuai dengan cara kerja manusia. Sensor visual dianalogikan sebagai indera penglihatan pada manusia. Sistem visual yang digunakan pada mobil robot ini diharapkan dapat melakukan berapa fungsi, yaitu :

- Pemetaan lingkungan (*environment mapping*)
- Pengenalan objek dan penghalang (*object and obstacle recognition*)
- Penjejakan objek (*object tracking*)
- Pembentukan lintasan (*path generating*)
- Pemindahan objek (*pick and place*)

Robot ini dilengkapi dengan sensor visual dan sensor ultrasonik. Sensor visual memberikan informasi visual yang kemudian diolah menjadi deskripsi objek dalam lingkungan tempat robot tersebut bekerja. Sensor ultrasonik akan memberikan data berupa informasi jarak relatif robot dengan objek disekitarnya.

Masalah utama dalam sistem visual pada autonomous mobile robot dalam melakukan fungsinya adalah proses pengenalan objek. Pada sebagian besar penelitian yang telah dilakukan, model objek yang akan dikenali atau dijejaki telah diketahui untuk mempermudah proses yang diperlukan untuk beberapa masukan data. Dengan demikian, masalahnya adalah mencocokkan model dengan objek yang ada. Hasil yang diperoleh adalah informasi mengenai

posisi objek tersebut. Dengan menggunakan model dalam 3D (tiga dimensi), informasi orientasi objek juga dapatdiperoleh.

Pada sebagian besar penelitian dalam pembentukan model objek 3D, sistem visual yang digunakan dilengkapi dengan dua buah kamera (*stereovision* / *binocular camera*). Sistem ini lebih kompleks dan membutuhkan biaya lebih besar dibandingkan dengan sistem yang menggunakan satu kamera (*monocular camera*). Pada penelitian yang hanya menggunakan sebuah kamera, proses pembentukan model objek 3D dilakukan dengan menggunakan urutan citra. Proses ini membutuhkan waktu yang cukup lama karena harus memproses beberapa citra masukan.

Pada tulisan yang diperoleh melalui penelitian yang panjang padapenginderaan mobil robot ini dikembangkan suatu sistem visual pada *autonomous mobile robot* yang dapat melakukan proses pembentukan model 3D objek yang memberikan informasi posisi dan orientasi objek. Sistem ini diusahakan sederhana, murah, dan dapat melakukan prosesdengan cepat.

Sepanjang perjalanannya, *autonomous mobile robot* akan memetakan secara langsung lingkungan yang sedang dijelajahinya sertadapat mendeteksi dan memodelkan objek dan rintangan yang ada dalamlingkungan tersebut. Hal ini diperlukan untuk menentukan lintasan*autonomous mobile robot* tersebut. Sistem visual yang dikembangkanadalah sistem yang menggunakan model objek 3D dan dapat memetakanatau memodelkan lingkungan maupun objek yang ada secara cepat(meskipun dengan model yang tidak terlalu akurat) dengan hanyamenggunakan sebuah citra masukan yang diperoleh dari kameramonokular.

Tujuan penelitian ini adalah untuk mengembangkan suatu metode pendeteksian titik–titik kontrol yang dapat merepresentasikan model 3D suatu objek. Metode ini diharapkan dapat memberikan hasil yang baik dan dapat diproses dengan cepat. Selain itu, penelitian ini juga bertujuan untuk mengembangkan suatu sistem yang mempunyai kecerdasan buatan sehingga dapat melakukan fungsinya meskipun hanya memiliki informasi masukan yang terbatas.

7.2 Definisi, Representasi, dan Pemodelan Citra

Pada pembahasan berikutnya, kita akan banyak menggunakan istilah citra maupun gambar, baik dalam pemberian penjelasan, maupun pemrograman citra. Tapi apakah sebenarnya definisi citra dan gambar itu?

Citra adalah representasi informasi dua dimensi yang diciptakan atau dibuat dengan melihat atau lebih tepatnya merasakan sebuah gambar atau pemandangan. Pembuatan citra pada spektrum visual manusia tidaklah begitu penting mengingat kompleksnya penginderaan manusia itu sendiri. Tapi, bagi sebuah alat, atau katakanlah sebuah instrument pencitraan, pembuatan citra adalah proses yang tidak mudah. Alat pencitraan harus terlebih dahulu memiliki sensor yang mengkonversikan informasi radiometri sebuah gambar atau pemandangan menjadi sebuah citra. Kemudian, proses tadi diteruskan dengan berbagai proses transformasi, analisis, dan pemrosesan lanjut, serta penginterpretasian citra itu sendiri nantinya. Berbeda halnya dengan gambar, yang merupakan kumpulan objek tiga dimensi dengan beberapa pengaturan geometri dan biasanya disusun oleh hukum fisika yang berlaku di alam.

Representasi sebuah citra biasanya dikaitkan dengan karakteristik kuantitas representasi masing-masing elemen gambar (*pixel*). Sebuah citra juga dapat merepresentasi karakteristik penyerapan jaringan tubuh dengan proses pencitraan sinar x, target menyilang dari pencitraan radar, keadaan suhu suatu daerah dengan pencitraan infra merah, maupun medan gravitasi suatu wilayah dengan pencitraan geofisika. Berikut ini diberikan bagan mengenai beberapa buah representasi citra dan permasalahan dalam pemodelannya .

Gambar 7.1 Representasi dan pemodelan citra

Bila dikaitkan dengan fungsi matematika, citra dapat direpresentasikan dengan fungsi, yaitu :

f (x,y)
f (x1,y2)
f (I), dan seterusnya

Argumen fungsi citra tersebut adalah variabel yang bebas (x, y, atau x1, x2, atau I) dan koordinat ruangan yang berkaitan dengan lokasi fisik dalam badan sensor citra tersebut. Lebih jauh lagi, f menyatakan intensitas (dalam

besaran lumen, volt, dan sebagainya) di berbagai lokasi tersebut. Kisaran nilai fungsi f biasanya disebut kisaran *greyscale*. Jika nilai f beradasetidaknya diantara 2 nilai, citra disebut sebagai citra yang *binary*. Masing-masing kuantitas mempunyai unit set asosiasi; x1 dan x2 biasanya dihitung atau dicari dengan mengalikan bentuk dan dimensi permukaan citra. Dalam kondisi ini, biasanya f diekspresikan dengan kondisi unit intensitas sensor (misalnya volt, coulomb) yang jarang dipakai pada kondisi yang diharapkan.

Citra *greyscale* membolehkan dua nilai intensitas. Citra *binary* adalah sebuah kelas citra yang menghasilkan beberapa buah proses inisial dari data citra *greyscale*. Sebagai ilustrasinya, HVS mempunyai persepsi *greyscale* yang mempunyai kisaran kira-kira 64 level perbedaan (6 bit). Dengan kata lain, pengamatan manusia menunjukkan skala intensitas *greyscale* 6 bit (atau malah lebih besar) dengan intensitas batang (*bar*) yang berkisar dalam intensitas dari level yang paling kecil ke level yang paling besar pada variasi yang lebih kecil dan halus.

Selanjutnya, representasi f(x) akan menitikberatkan pada hal berikut :

- f mempunyai nilai antara yang terbatas
- x mempunyai daerah tertentu (permukaan citra ataupun sensor mempunyai *field of view* yang terbatas).
- Baik f maupun x adalah faktor terkuantisasi yang diskret pada proses dalam komputer.

Sebelum melangkah ke fungsi matematika yang lebih lanjut dan rumit, ada baiknya dipahami bahwa baik versi kontinu maupun diskret dari fungsi *f(x1 , x2)* bergantung pada :

- Geometri dan bentuk pandangan. Terdapat hubungan geometri darisensor ke objek tinjauan. Juga terdapat karakteristik sensor (misalnya magnifikasi, intensitas tanggapan, dan karakteristik geometri)
- Komposisi objek yang dilihat. Isi objek dan hubungan geometri direfleksikan dalam spektra yang berkaitan.

Lebih jauh lagi, diketahui bahwasanya fungsi citra terdiri atas sejumlah argumen (x1 dan x2) yang terdisribusi merata dan nilai intensitas f yang dikenal sebagai ‘dasar dua transformasi’. Di dalam domain (daerah asal) argumen tersebut, karena dua dimensi transformasi itu sifatnya alamiah, maka

hal-hal tersebut akan dispesifikasikan ke dalam bentuk geometri. Hierarki pemrosesannya selanjutnya didasari oleh hubungan geometri matematika berikut ini,

Fitur x1, x2 ⇒bentuk dasar ⇒identifikasi atau deskripsi objek ⇒deskripsi objek.

Kemudian, kita akan mendefinisikan masukan dan keluaran fungsi citra sebagai f(x) dan g(x). Selanjutnya, kita dapat mengklasifikasikan usaha pemrosesan level rendah dengan menggunakan dua tipe transformasi dengan struktur sebagai berikut :

- *Intensity-based*, g(x) = O [f(x)] (yang akan dioperasikan dengan prinsip intensitas).
- *Geometric-based*, g(x) = f(h(x))

Pada pembahasan bab-bab berikutnya mengenai citra , kita akan banyak mengunakan transformasi dan algoritma fourier yang diimplementasikan pada peningkatan kontras sebuah atau beberapa citra. Juga akan dipakai transformasi *affine* dan transformasi perspektif yang digunakan pada manipulasi geometri citra, seperti rotasi, translasi, dan sebagainya. Lagi pula, representasi data umumnya menggunakan pemrosesan 2 dimensi terlebih dahulu, termasuk pencitraan pada spectrum yang *visible*, pencitraan infra merah, serta pencitraan yang merupakan *aperture* radar buatan.

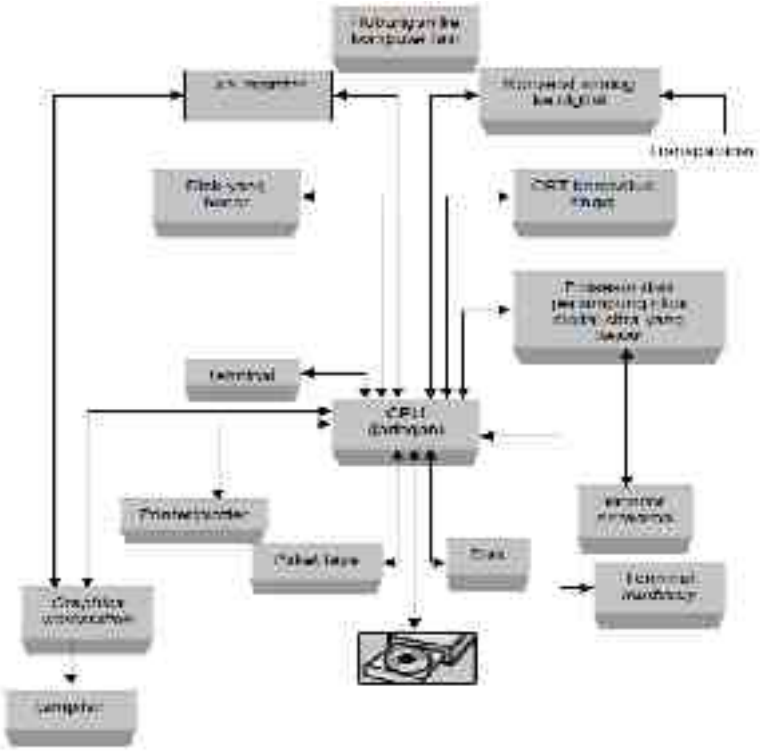
7.2.1 PENGOLAHAN CITRA DIGITAL : SEBUAH PENDEKATAN

Frase pengolahan citra digital atau dalam bahasa Inggrisnya *digital image processing* sering dikaitkan dengan pemrosesan gambar dua dimensi oleh komputer digital atau pemrosesan digital data dua dimensi. Citra digital adalah sekumpulan bilangan kompleks maupun real yang direpresentasikan oleh bit yang terhingga.

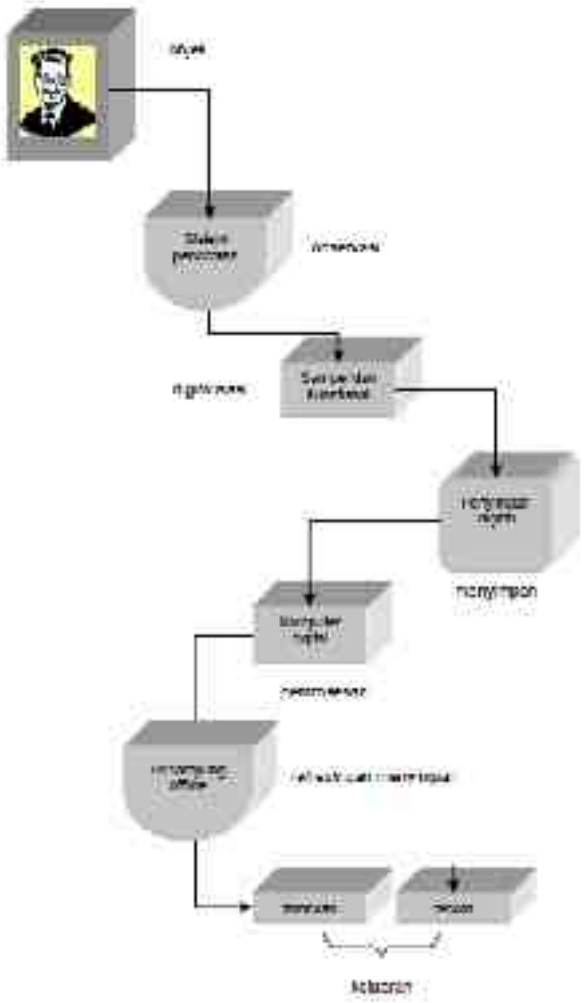
Untuk sebuah pengolahan citra digital yang sederhana, tingkat pemrosesannya dapat dijabarkan sebagai berikut :

1. Sebuah citra dalam bentuk transparansi, slide, foto, ataupun *chart* didigitalkan terlebih dahulu dan disimpan sebagai sebuah matriks yang berisikan digit biner dalam memori komputer.
2. Citra kemudian dapat diproses dan ditampilkan pada monitor yang mempunyai resolusi tinggi.
3. Untuk tampilan, citra disimpan dalam sebuah penampung memori yang dapat diakses dengan cepat (*rapid access buffer memory*) yang akan me-*refresh* monitor dengan 30 frame/detik untuk memproduksi sebuah tampilan kontiniu yang dapat dilihat dengan jelas.
4. Komputer mikro maupun komputer mini digunakan untuk berkomunikasi dan mengendalikan semua proses digitalisasi, penyimpanan, dan operasi tampilan melalui komputer jaringan.
5. Masukan ke komputer yang berupa program dibuat melalui sebuah terminal, dan keluaran juga tersedia pada terminal, monitor, ataupun printer.

Untuk lebih jelasnya, perhatikan bagan proses dan periferal standar serta tipikal tahapan proses yang dibutuhkan pada pengolahan citra secara digital.



Gambar 7.2 Hubungan periferal pada pengolahan citra digital yang sederhana



Gambar 7.3 Tipikal tahapan pengolahan citra digital

Meskipun pada kenyataannya sangat banyak aplikasi pengolahan citra, tetapi pada pembahasan pengolahan citra pada mobil robot ini akan diberi batasan pada beberapa kelas permasalahan dasar yang ada, yakni :

1. Proses persepsi citra
2. Proses kuantisasi dan sampling citra
3. Proses representasi stokastik citra
4. Proses peningkatan citra
5. Proses restorasi dan penyaringan citra

6. Proses analisis citra
7. Proses pengkompresian data citra

Berikutnya akan diberikan gambaran ringkas mengenai tahapan ataupun kelas dasar permasalahan pada pengolahan citra secara digital sebagaimana diuraikan di atas.

7.2.2 Proses Persepsi Citra

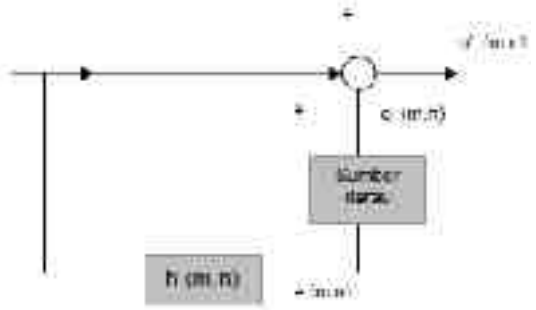
Dalam menampilkan keluaran sebuah sistem pencitraan kepada manusia, perlu dinyatakan bagaimana citra itu ditransformasikan ke dalam bentuk paket informasi kepada manusia. Dalam proses mempersepsikan sebuah citra, banyak hal yang harus dilaksanakan, antara lain bagaimana kita mengatur cahaya, tingkat kejelasan, maupun tingkat kekontrasan sumber pencahayaan citra yang akan kita olah tersebut. Setelah itu, tahapan yang harus dicapai adalah memahami bagaimana proses kontras cahaya itu bila datang secara bersamaan; di bagian ini kita akan bermain dengan persamaan hukum Weber. Kemudian, untuk interaksi yang spasial dari tingkat pencahyaan dan persepsi citra, kita akan dapat menemui fenomena yang disebut efek Mach Band. Efek ini akan menunjukkan bahwa terang-gelapnya pencahayaan bukanlah fungsi monoton dari luminansi. Pada tahapan berikutnya, efek Mach Band akan mengukur tanggapan sistem visual dalam koordinat spasial. Transformasi Fourier dari tanggapan.Impuls akan memberikan tanggapan frekuensi sistem tadi. Setelah itu, MTF dapat dideterminasi.

Dalam banyak sistem pengolahan citra, fungsi visibilitas akan sangat membantu. Misalnya dalampengkodean citra, keluaran citra u‘ (m,n) terdiri atas derau tambahan (*additive noise*) q (m,n), yangbergantung pada e (m,n), sebuah fungsi masukan citra u (m,n). Urutan atau tahapan e (m,n) kadang-kadang disebut fungsi masking . Fungsi masking adalah sebuah fitur citra yang akan diobservasi atau diproses pada aplikasi yang akan dilakukan berikutnya. Contohnya adalah , bila e (m,n) = u (m,n)dalam transmisi PCM dari sejumlah citra. Contoh yang lain adalah :

1. e (m,n) = u (m,n) - u (m-1,n)
2. e (m,n) = u (m,n) – a1u (m-1,n) - a2u (m,n-1) + a3u (m-1,n-1)

3. e (m,n) = u (m,n) - α[u (m-1,n) + u (m+1,n) + u (m,n-1) +u (m,n+1)

Di bawah ini, diberikan contoh diagram fungsi visibilitas dari model sumber derau.



Gambar 7.4 Fungsi visibilitas model sumber derau

Pada tahap persepsi citra ini, bagian yang kelak menjadi sangat penting adalah representasi warna. Pengetahuan mengenai warna menjadi sangat penting, mengingat semua proses desain dan pengembangan citra selalu berkaitan dengan sistem warna *vision*. Penggunaan warna untuk tampilan citra bukanlah sekadar untuk memperindah saja, melainkan juga untuk mempermudah kita menerima informasi visual . Di satu pihak kita menyadari hanya sedikit level *grey* yang dapat ditampilkan, sementara dilain pihak kita memiliki kemampuan untuk membedakan ratusan warna lainnya, yang menjadikan proses demi proses pencitraan menjadi lebih menarik. Atribut perseptual warna adalah *brightness, hue,* dan saturasi. *Brightness* merepresentasikan luminansi yang diterima sebelumnya. *Hue* mengacu pada kemerahan, kehijauan, dan lain sebagainya. Untuk sumber cahaya yang monokromatik, perbedaan warna dimanifestasikan oleh perbedaan panjang gelombang. Saturasi adalah aspek persepsi yang memvariasikan cahaya putih yang lebih kuat yang akan ditambahkan ke dalam cahaya monokromatik.

Warna yang direpresentasikan berdasarkan teori klasik Thomas Young (102), yang menyatakan bahwa warna apa pun dapat direproduksi dengan cara mencampurkan dan mengkombinasikan tiga warna primer.

7.2.3 Proses Kuantisasi dan Sampling Citra

Persyaratan dasar pengolahan citra dengan komputer adalah citra harus tersedia dalam bentuk digital, yang direpresentasikan oleh sebuah *array* yang mempunyai panjang kata biner tertentu (tidak tak terhingga). Citra yang diberikan disampel pada kisi yang diskret dan masing-masing sampel atau piksel dikuantisasi menggunakan sejumlah tertentu bit. Citra yang didigitalkan kemudian diproses pada komputer tersebut. Untuk menampilkan sebuah citra digital, citra pertama sekali dikonversikan menjadi sinyal analog, yang di-*scan* melalui sebuah alat atau instrument tertentu.

Bagannya ditunjukkan berikut ini :

Gambar 7.5 Proses digitalisasi citra

Gambar 7.6 Proses menampilkan citra

Salah satu bagian proses sampling dan kuantisasi citra adalah peng-*scan*-an citra . Metode sampling citra yang paling umum adalah melakukan *scan* baris demi baris, lalu mencuplik masing-masing baris tersebut. Contohnya adalah sebuah objek, film, ataupun transparansi yang secara kontiniu diiluminasi untuk membentuk citra elektron pada sebuah plat yang bercahaya sensitif yang disebut target. Dalam tabung *vidicon*, target adalah fotokonduktif, sedangkan sebuah tabung citra *dissector* adalah fotoemisif.

Pada proses sampling dan kuantisasi ini akan banyak dijumpai juga persamaan matematis seperti teori sampling dua dimensi, replikasi, rataan Nyquist, Aliasing, dan frekuensi Foldover. Aliasing merupakan suatu proses yang menyebabkan distorsi atau gerigi pada gambar. Sebagai gambaran adalah menembakkan suatu cahaya lewat pusat sebuah pixel, melihat perpotongan, kemudian mewarnai seluruh pixel melalui warna apa pun yang berhubungan dengan benda yang dikenai. Hasilnya mungkin tidak sangat akurat dan kadang-kadang gambarnya kabur.

Contoh yang berikutnya adalah dengan mengambil lantai berubin putih dan hitam yang lebarnya berukuran 0.25 meter, kemudian disampling pada interval 0.5 meter. Akan didapat semua sampel berwarna putih atau berwarna hitam. Hal ini sebenarnya tidak mewakili warna lantai, yang seharusnya menjadi gabungan keduanya, atau dengan kata lain berwarna abu-abu. Contoh lainnya adalah membuat animasi dengan menggerakkan beberapa gambar dengan menggerakkan objek dan memperagakan semua *frame* dengan cepat (misalnya 24 *frame* tiap detik). Pada pemodelan, jika suatu objek bergerak cukup cepat, mungkin akan muncul lompatanlompatan sekitar layar. Contohnya, jika objek berukuran 1 cm tapi bergerak begitu cepat di dalam 1/24 detik (lamanya sebuah *frame*), sekitar 5 cm menjauh dari sebelumnya, objek akan muncul seperti melompat dengan drastis sekali.

Teknik dan teori dalam proses sampling ini sangat banyak antara lain sampling dan *interlasing grid* yang *nonrectangular*, sampling heksagonal, sampling optimal. Pembatasan praktis pada proses sampling dan rekonstruksi antara lain sampling apertur, fungsi interpolasi, dengan tambahan fungsi seperti fungsi interpolasi Lagrange serta tanggapan lapangan datar (*flat field response*) dan efek Moiré.

Langkah berikutnya untuk melakukan sampling citra yang telah terdigitalisasi adalah mengkuantisasi. Pengkuantisasian sebenarnya bertujuan memetakan sembarang variabel yang kontinyu menjadi sebuah variabel diskret. Proses mengkuantisasi citra banyak jenis dan tahapannya, antara lain pengkuantisasian Lloyd-Max dan pengkuantisasian optimal uniform. Kemudian dikenal *compandor* yakni *compressor expander*, yakni pengkuantisasi uniform yang dihasilkan dari transformasi nonlinear.

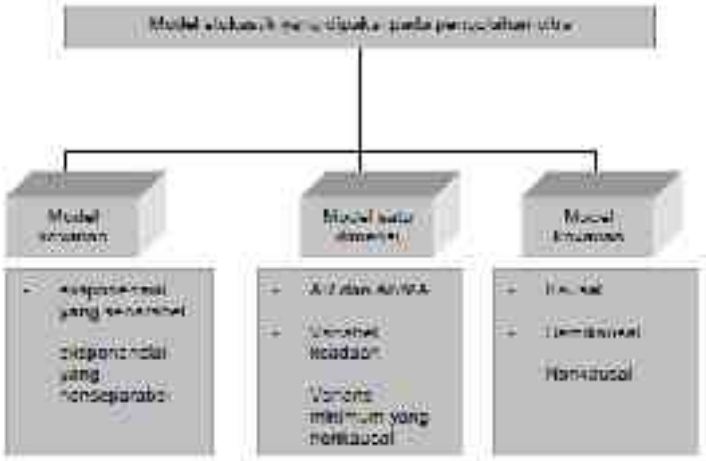
7.2.4 Proses representasi stokastik citra

Dalam representasi yang stokastik, citra sering sekali dianggap sebagai sebuah fungsi sample suatu *array* dari variabel acak yang disebut medan acak (*random field*).

Model yang kita perkenalkan di bidang bahasan ini, terbagi atas 2 jenis

1. Model Kovarian
   Dalam banyak aplikasi, termasuk restorasi citra dan kompresi data, seringkali citra dikarakterisasi sebagai fungsi kovarian, dan terkadang dimulai dengan representasi medan acak yang berarti fungsi kovarian itu dinyatakan oleh model eksponensial yang separabel ataupun yang non separabel. Model kovarian yang separabel adalah model yang baik untuk dianalisis pada algoritma pengolahan citra dan komputer yang atraktif, sedangkan model yang satu lagi tidak begitu nyaman untuk dianalisis. Model kovarian dianggap model yang sangat berguna pada pengkodean transformasi citra. Fungsi kovarian itu digunakan untuk mencari variasi koefisien transformasi yang bersangkutan.

2. Model Sistem Linier (*Linear System Models*)
   Alternatif untuk merepresentasikan medan acak oleh fungsi kovarian adalah dengan mengkarakterisasi semua fungsi tersebut sebagai keluaran sistem linier. Masukannya merupakan medan acak dengan mengetahui medan itu berdasarkan data statistic (misalnya derau masukan). Seperti yang kita ketahui, sistem linier dinyatakan dengan persamaan perbedaan dan biasanya berguna dalam mengembangkan algoritma pengolahan citra pada computer dengan efisien. Juga, algortima adaptif yang didasarkan pada parameter persamaan perbedaan yang *update* lebih mudah diimplementasikan. Permasalahan yang timbul dalam usaha menemukan model persamaan perbedaan yang stokastik adalahmerealisasikan kovariansi yang dikenal sebagai masalah faktorisasi yang spektral . Di bawah ini ditampilkan diagram model stokastik yang digunakan pada pengolahan citra.



Keterangan :
AR = *Auto Regressive*
ARMA = *Auto Regressive Moving Average*

Gambar 7.7 Diagram model stokastik pada pengolahan citra

7.2.5 Proses peningkatan kualitas citra

Peningkatan kualitas citra menitikberatkan satu proses pada pengolahan citra yang bertujuan menajamkan fitur citra seperti sudut (*edges*), batas (*boundaries*), ataupun tingkat kontras untuk membuat tampilan grafik lebih mudah dianalisis dan ditampilkan. Proses peningkatan kualitas citra bukanlah berupa penambahan informasi yang inheren pada data citra tersebut, melainkan menambahkan *range* yang dinamik dari fitur-fitur yang terplilih agar dapat dideteksi dengan mudah. Peningkatan kualitas citra terdiri atas level *gray* dan manipulasi kontras, reduksi derau, menegaskan sudut, penajaman, penyaringan, interpolasi dan magnifikasi, pewarnaan semu, dan lain-lain.

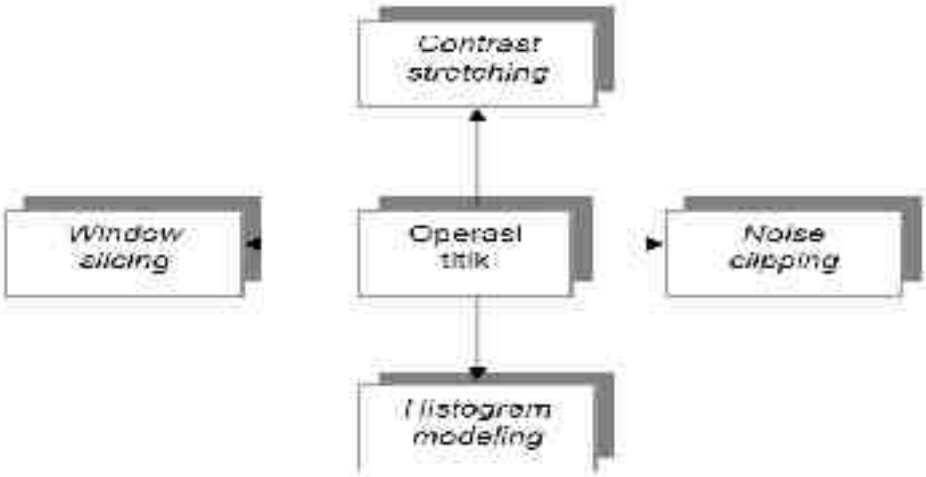
Kesulitan paling besar dalam proses peningkatan citra adalah mengkuantisasi kriteria yang akan ditingkatkan kualitasnya. Oleh karena itu, sejumlah teknik proses peningkatan citra bersifat empirik dan membutuhkan prosedur yang interaktif untuk mendapatkan hasil yang memuaskan. Bagaimanapun, proses peningkatan citra menyisakan sebuah topik yang sangat

penting karena kegunaannya di dalam aplikasi pemrosesan citra yang bersifat virtual. Di bawah ini disajikan pembagian teknik dalam proses peningkatan kualitas citra yang umum.

Gambar 7.8 Pembagian teknik dalam proses peningkatan kualitas citra

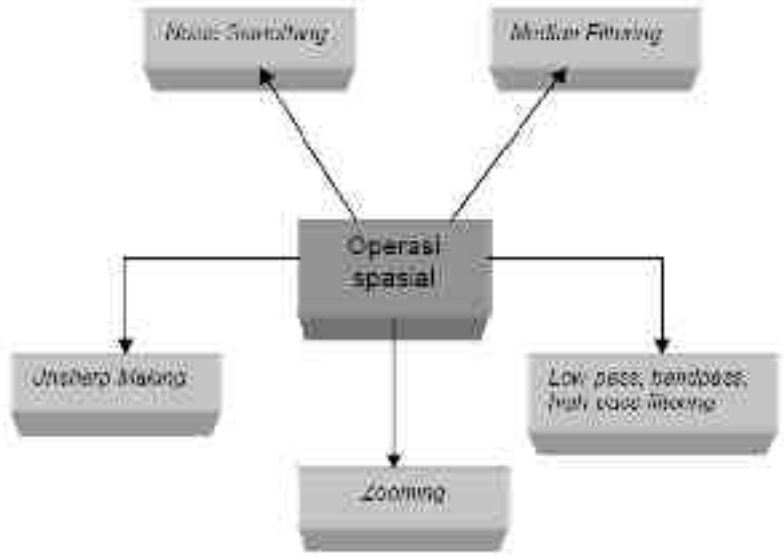
Masing-masing teknik tersebut mempunyai pembagian lain.

- Untuk operasi titik



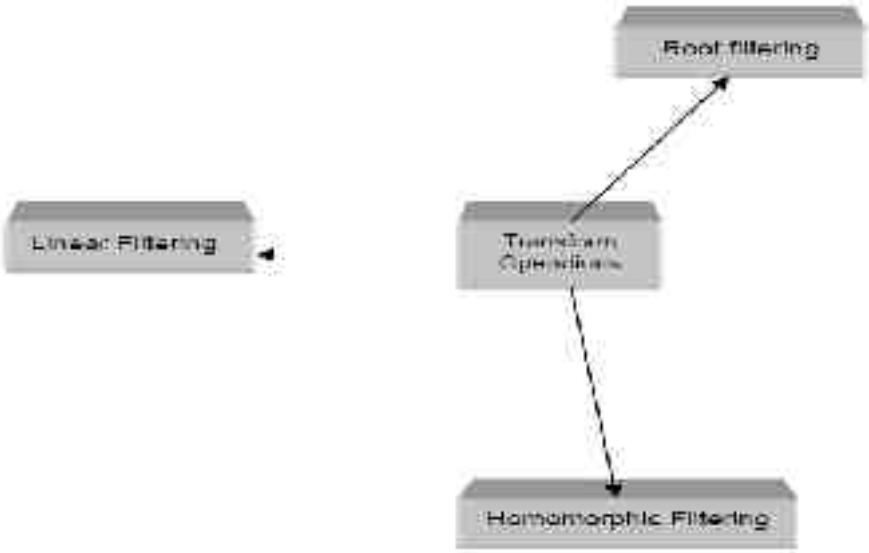
Gambar 7.9 Pembagian operasi titik

- Untuk operasi spasial



Gambar 7.10 Pembagian operasi spasial

•Untuk operasi transformasi



Gambar 7.11 Pembagian operasi transformasi

- Untuk pewarnaan semu

Gambar 7.12 Pembagian pewarnaan semu

7.2.6 Proses restorasi dan penyaringan citra

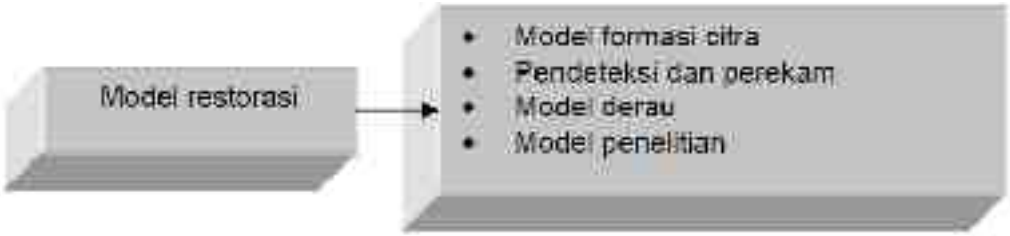
Sembarang citra diperoleh melalui optik atau optik elektrik, yang tentu saja menurun derajatnya akibat lingkungan. Penurunan ini mungkin dalam bentuk derau sensor, kabur karena kamera yang kurang fokus, pergerakan objek kamera yang relatif, turbulensi atmosfer acak, dan lain-lain. Restorasi citra ini berkaitan dengan penyaringan citra yang diobservasi untuk mengurangi efek penurunan ini. Faktor efektivitas penyaringan serta restorasi citra tergantung pada tingkat akurasi pengetahuan kita mengenai proses penurunan itu sendiri dan juga kriteria desain penyaringan. Kriteria yang seringkali dipakai adalah '*mean square error'*. Meskipun begitu, validitasnya sebagai pengukuran yang global dari fidelitas visual, masih dipertanyakan.

Restorasi citra berbeda dengan peningkatan kualitas citra (*image enhancement*) dalam hal keterkaitannya lebih lanjut dengan aksentuasi ataupun ekstraksi fitur citra. Permasalahan dalam restorasi citra ini sulit diprediksikan. Peningkatan kualitas yang mana yang dianggap sejalan dengan proses restorasi ini sangatlah sulit dinyatakan dalam bentuk matematis. Sebagai konsekuensinya, berbagai teknik restorasi yang digunakan seringkali

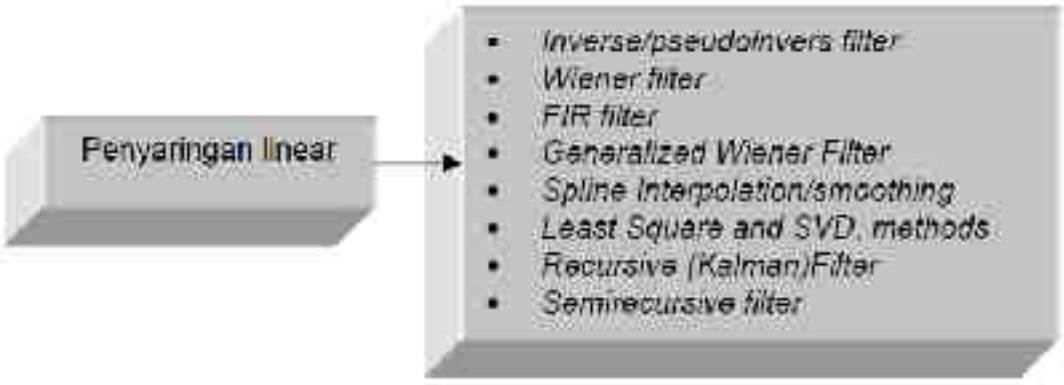
bergantung hanya pada sifat-sifat kelas sebuah set data. Di bawah ini diberikan diagram pembagian teknik restorasi dan penyaringan.

Gambar 7.13 Pembagian teknik restorasi dan penyaringan

Masing-masing sub permasalahan pada Gambar 7.13 itu, dibagi lagi menjadi :



Gambr 7.14 Pembagian model restorasi



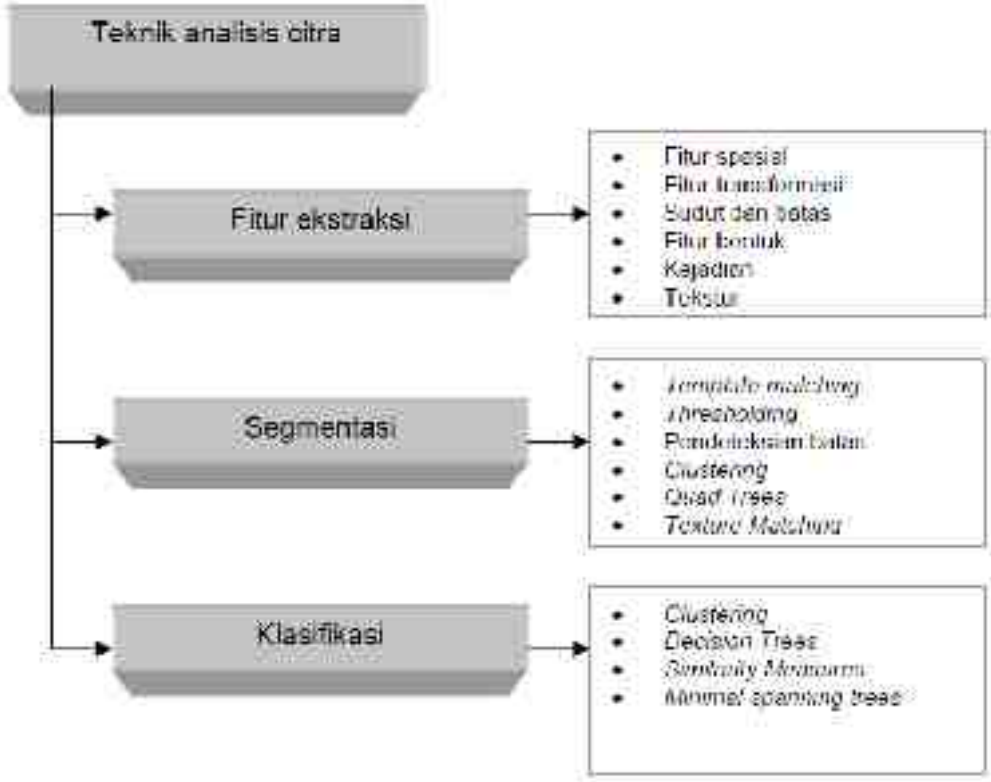
Gambar 7.15 Pembagian penyaringan linier

Gambar 7.16 Pembagian metode lain

7.2.7 Proses analisis citra

Tujuan terakhir dalam sejumlah aplikasi pengolahan citra adalah mengekstraksi fitur-fitur penting dari data citra. Deskripsi , interpretasi, ataupun pemahaman dari *scene* dapat disediakan oleh mesin. Sebagai contohnya, sebuah sistem *vision* dapat membedakan bagian-bagian pada sebuah proses asembli garis dan fiturnya, seperti ukuran dan jumlah *holes* yang ada. Sistem *vision* yang lebih canggih mampu menginterpretasi hasil analisis dan menjabarkan objek yang bermacam-macam dan hubungan antarmereka dalam sebuah kerangka.

Analisis citra, pada dasarnya terdiri atas fitur ekstraksi, segmentasi, pengklasifikasian teknik. Bagian-bagian yang terdapat pada analisis citra diperlihatkan pada **Gambar 7.17**.



Gambar 7.17 Pembagian teknik analisis citra

7.3 Sensor Image

Bayangkan kita memiliki sebuah sensor yang bisa melihat banyak warna yang berbeda, seperti photoresistor. Bagaimana kita menggunakan sensor ini untuk mendeteksi apel merah dan apel hijau? Benar dengan meninjau perbedaan keduanya. Apel merah akan memantulkan cahaya merah tetapi menyerap cahaya hijau. Begitu pula apel hijau akan memantulkan cahaya hijau dan menyerap cahaya merah. Jika kita menyinari cahaya merah (seperti dari LED) pada kedua apel, apel merah akan memantulkan lebih banyak cahaya daripada apel hijau. Sehingga apel hijau akan kelihatan lebih cerah dibandingkan dengan apel hijau. Begitupula ketika kita menyinari cahaya cahaya hijau dari LED hijau pada kedua apel, maka apel hijau akan kelihatan lebih cerah.

Bayangkan saja kita memiliki beberapa permen coklat M&M dan kita ingin robot kita mengetahui perbedaan antara permen biru, hijau, kuning dan oranye. Bagaimana cara yang akan kita gunakan? Benar!!! Yakni dengan menggunakan LED biru, LED hijau dan juga LED merah. Kemudian sinari permen – permen tersebut dengan masing – masing secara bergantian dan catat nilai kecerahannya.

Tampak sekali bahwa permen M&M biru akan membaca nilai hijau yang lebih cerah ketika LED biru dinyalakan tetapi didapatkan nilai yang sangat rendah untuk LED lain. Hijau dan kuning bisa dideteksi dengan cara yang serupa. Lalu timbul pertanyaan, bagaimana caranya membedakan kuning dan oranye pada M&M? gampang! Orange akan lebih dekat ke merah dalam spektrum cahaya dan akan memantulkan lebih banyak cahaya merah daripada cahaya kuning.

Tetapi hal ini akan agak sedikit lebih sulit, pertama photoresistor memiliki perbedaan tingkat sensitivitas untuk warna – warna yang berbeda. Biasanya photoresistor optimal dan memiliki puncak pada daerah sekitar 520nm. Pada nilai ini photoresistor sangat sensitif pada jarak yang jauh dari target. Dan seperti yang dilihat pada grafik, sensor tertentu bisa mendeteksi sebagian besar cahaya dan memberikan pembacaan yang terbaik pada jarak 12 inchi. Informasi ini didapatkan dari datasheet

Jika kita memindah benda pada jarak yang berbeda atau merubah sudut benda, maka pembacaan juga akan berubah

Dan ini masih lebih rumit, yakni cahaya latar (sebagai contoh cahaya matahari yang datang dari jendela) bisa merubah keseluruhan pembacaan. Jadi kita memerlukan semacam shielding untuk sensor dari cahaya latar tersebut.

Dan yang masih tak kalah lebih rumit lagi perbedaan warna cahaya LED pada panjang gelombang dan kecerahan yang berbeda dapat menyebar sehingga sering terjadi pembauran beberapa panjang gelombang (didapatkan dari datasheet)

7.3.1 Ellips RIO *cards*

Banyak jenis dan macam piranti penangkap gambar yang terdapat di masyarakat luas.

Beberapa alasan menggunakan Ellips RIO *cards* ini adalah:

1. RIO merupakan generasi ketiga dari PCI *bus master* dengan pengontrol yang fleksibel. Beberapa alat pengambil gambar yang didukung oleh pengontrol PCI SVGA umumnya tidak cocok untuk aplikasi pencitraan yang *real- time*.
2. Enam buah CVBS atau kamera monokrom maupun 3 Y/C dapat dihubungkan ke Ellips RIO *cards*
3. RIO dapat mendukung kamera standar hitam putih CCIR (50 *fields per second*) dan EIA (60 *fields per second*), maupun format keluaran yang berbeda seperti *binary* ataupun *grayscale*.
4. RIO *cards* juga dapat dipakai untuk menghubungkan NTSC (60 *fields per second*), PAL, dan SECAM (50 *fields per second*) yang dapat menghasilkan keluaran warna digital.
5. RIO dapat melakukan pengambilan gambar secara serempak antara 2 kamera hitam putih.

Selain 5 faktor di atas, masih banyak keuntungan lain dari penggunaan Ellips RIO *cards*, tergantung pada banyaknya aplikasi yang akan menggunakan RIO tersebut.Ellips RIO cards mempunyai dua versi, yaitu:

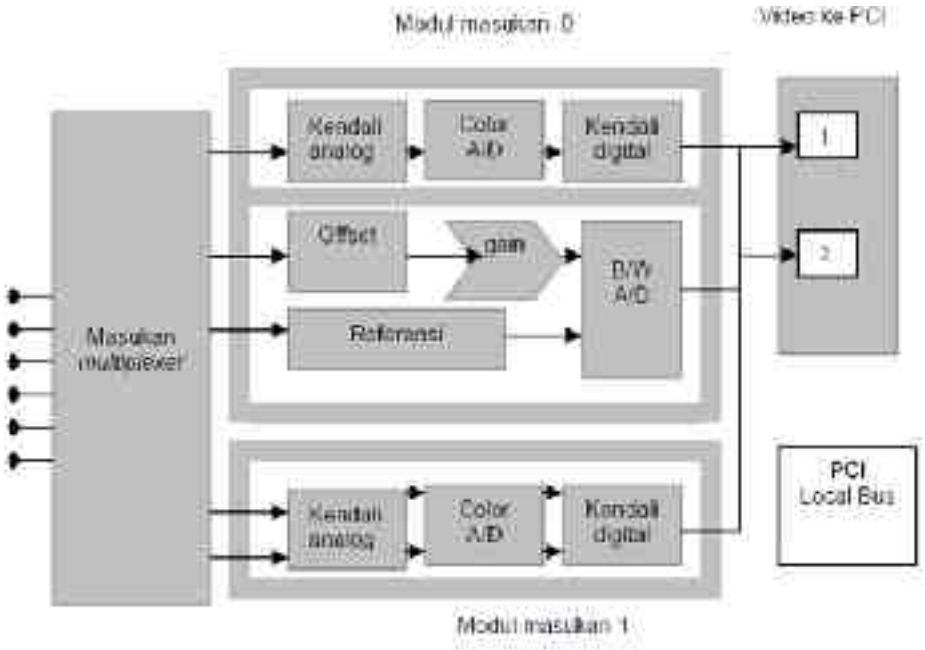
1. Versi dasar dengan satu modul input.
2. Versi lengkap dengan dua atau lebih modul input.

Untuk kamera pengambil gambar, kami menggunakan 2 buah kamera berwarna untuk mendukung kinerja pencitraan dalam mobil robot ini.

7.3.2. ARSITEKTUR PERANGKAT

Seperti yang sudah dijelaskan, perangkat utama pengambilan gambar ini adalah RIO *cards* dan dua buah kamera berwarna. Yang akan menjadi pembahasan teknik pengambilan gambar di mobil robot ini adalah

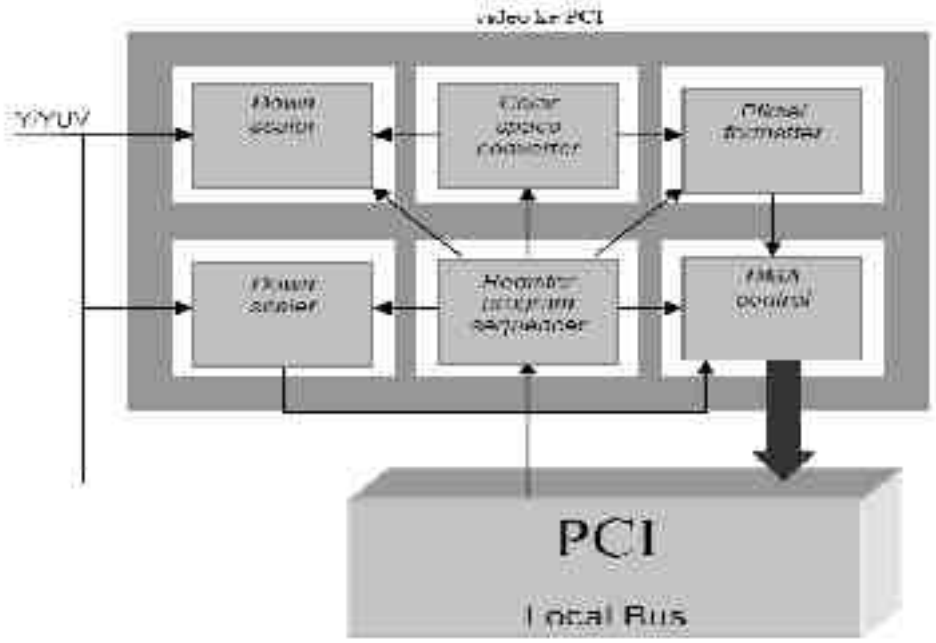
pemrograman perangkat keras RIO. Tapi, ada baiknya kita lihat dahulu diagram komponen yang membangun RIO *cards* tersebut.



Gambar 7.18 Diagram masukan

Keterangan gambar 1:

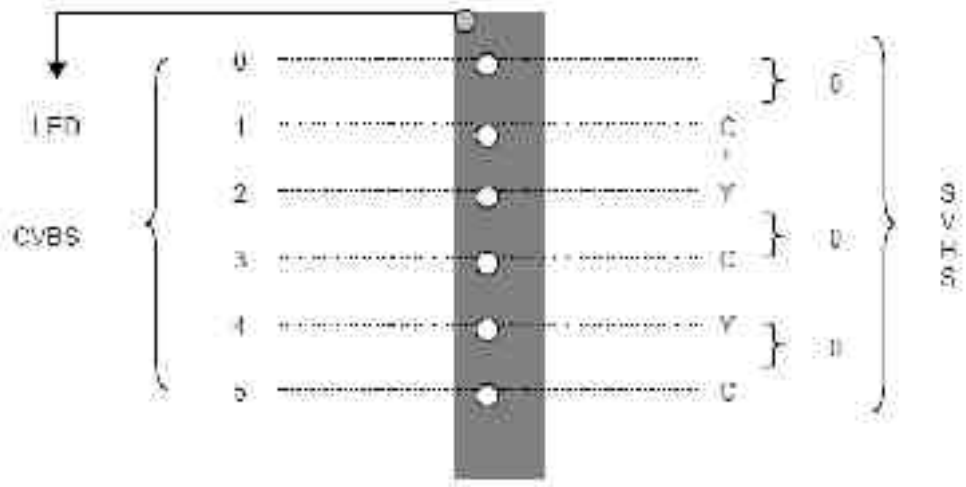
a. Masukan multiplexer berupa CVBS ataupun Y/C
b. Masukan bagian modul 0 terbagi 2, yakni:
   1 CVBS/Y yang masuk ke masing-masing bagian analog control dan offset
   2 Y/C yang hanya masuk ke bagian analog control
c. Input analog control bagian modul 1 berupa CVBS/Y dan Y/C



Gambar 7.19 Diagram bagian video ke PCI

Keterangan gambar 2:
a. Input ke Video to PCI berupa Y/YUV yang masuk dari input modul 0 menuju down scaler
b. Register Program Sequencer memberikan instruksi ke masing-masing blok di Video to PCI.

Dari **Gambar 7.18**, kita lihat bahwa masukan utama terdiri atas 6 buah penghubung SMB. RIO card yang merupakan pengambil gambar utama dapat menerima masukan dari segala jenis sinyal video standar,
termasuk dari perekam video.
Sinyal-sinyal input yang mendukung adalah:
- CVBS (CCIR dan EIA)
- Y/C (S-video, S-VHS, ataupun Hi8 untuk PAL-B/G, NTSC-M, danSECAM standar)
- RGBA (CCIR dan EIA).



Gambar 7.20 Diagram penghubung ke masukan *multiplexer*

Penghubung-penghubung itu dapat menerima banyak jenis masukan seperti :

- *Composite video sources* (CVBS) sebanyak 6 buah.
- Sambungan Y/C, maupun sumber S-video (S-VHS atau Hi8) sebanyak 3 buah.
- *Progressive scan* sebanyak tiga buah. Masing-masing kamera harus mempunyai keluaran video 1 yang tersambung ke Y, dan keluaran video 2 yang tersambung ke penghubung C.
- Kamera RGBA sebanyak satu buah, dengan penghubung R, G, B, danA ke bagian seperti CVBS dari masukan 0 sampai dengan 3.

Kombinasi dari masing-masing masukan di atas. Misalnya, dua buah kamera CVBS dan dua buah kamera Y/C dapat dihubungkan bersamaan, dan lain-lain.

7.3.3. SPESIFIKASI DAN PERANGKAT LUNAK PENDUKUNG

- Interface komputer utama
    a. Transfer DMA berkecepatan tinggi (sampai dengan 132 MB per
    b. detik)
    c. Pengendali transfer yang dapat diprogram

d. Sistem yang mendukung banyak *boards*.

- Perangkat lunak pendukung pemrograman RIO card ini adalah:
  a. Untuk DOS (32-bit) : Watcom C/C++(11.0)
  b. Untuk Windows 9x/NT (32-bit) : Microsoft Visual C 4.x/5.x
  c. Vx Works, pendukung sistem operasi *real-time*
- Tambahan
  a. RS-170, CCIR, NTSC, PAL, SECAM, Y/C ataupun masukan
  b. monokrom
  c. Resolusi masing-masing video:
- PAL, SECAM, CCIR berukuran 768x576
- NTSCdan berukuran EIA 640x480
- Format keluaran:
- RGB32 (αRGB)
- RGB24 (RGBR)
- RGB16 (5:6:5)
- RGB8 (3:3:2)
- YUV16 (CCIR)
- Y8 (256 grayscale)
- Y2 (2 bit grayscale)
- Y1 (B/W)
- Frekuensi rata-rata untuk video sumber: 14,75/7,38 MHz untuk PAL/SECAM dan 12,27/6,13 MHz untuk NTSC.
- PLL digital menyediakan sinkronisasi seimbang ketika menggunakankamera video dan perekam video .
- Mendukung kombinasi video sampai dengan tiga buah sumber video Y/C dan enam buah sumber video.
- Enam buah penghubung SMB untuk 6 kamera monokrom, tiga pasangkamera Y/C (S-Video), atau satu buah kamera RGBA.

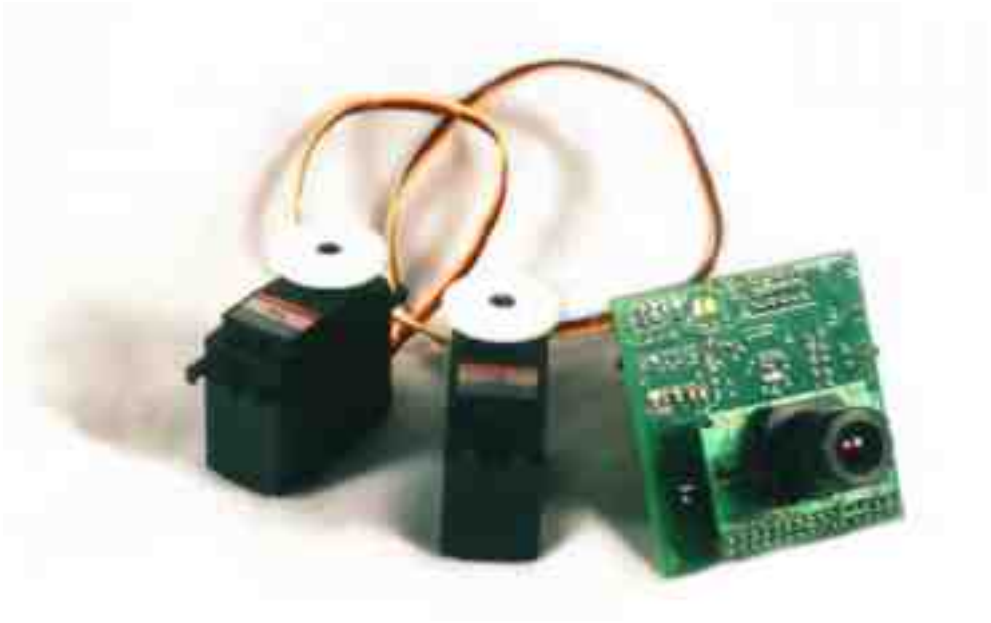
7.3.4 CMUCam2

CMUcam2 ini terdiri dari sebuah mikrokontroler SX52 (http://www.ubicom.com/ products/sx/sx.html) dihubungkan dengan OV6620 atau OV7620 Omnivision kamera CMOS (http://www.ovt.com) pada sebuah

chip yang memungkinkan data sederhana tingkat tinggi harus diekstrak dari kamera video streaming.

Papan berkomunikasi melalui RS-232 atau port serial TTL dan memiliki fungsi sebagai berikut:

- Lacak pengguna didefinisikan gumpalan warna sampai dengan 50 Frames Per Second *
- Track frame gerak dengan menggunakan pembedaan pada 26 Frames Per Second
- Carilah centroid pelacakan data
- Kumpulkan data berarti warna dan varians
- Kumpulkan 28 bin histogram dari tiap kanal warna
- Dapat Memanipulasi Horisontal Pixel Foto
- Dapat mentransfer bitmap biner secara real-time.
- Sewenang-wenang gambar windowing
- DApat mengatur properti gambar kamera
- Tingkat Resolusi sampai dengan 160 x 255 Resolusi **
- Mendukung Multiple Baudrates: 115.200, 57.600, 38.400, 19.200, 9.600,4.800, 2.400, 1.200
- Dapat mengontrol 5 servo output
- Dapat secara otomatis menggunakan servos untuk melakukan pelacakan dua sumbu warna.
- B / W Analog video output (PAL atau NTSC) **
- Bekerja dengan OV7620 atau modul OV6620

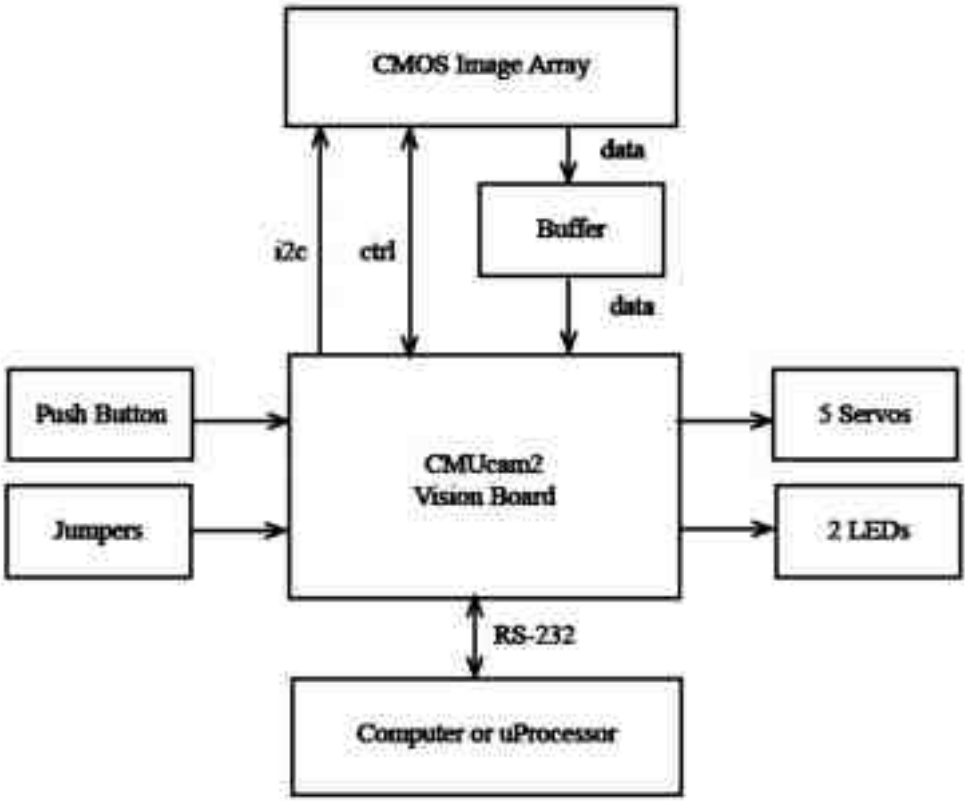


Gambar 7.21 CMUcam dan servo

7.3.5 Penggunaan

Salah satu kegunaan utama CMUcam2 ini adalah untuk melacak atau monitoring. Kinerja terbaik dapat dicapai bila ada yang sangat kontras dan warna intens. Misalnya, melacak bola merah di latar belakang putih, tetapi akan sulit untuk membedakan antara berbagai warna cokelat dalam mengubah cahaya. Pelacakan obyek berwarna-warni dapat digunakan untuk melokalisasi landmark, mengikuti garis, atau mengejar sebuah benda yang bergerak. Dengan menggunakan statistik warna, memungkinkan CMUCam berfungsi untuk memantau sebuah objek, mendeteksi warna tertentu atau melakukan deteksi gerak. Jika kamera mendeteksi perubahan warna drastis, maka kemungkinan

sesuatu di tempat berubah. Menggunakan "mode line," CMUcam2 berfungsi mendapatkan gambar resolusi rendah biner obyek yang berwarna-warni.



Gambar 7.22 Blog diagram CMUcam2

Latihan

1. Apa yang dimaksud dengan robot vision? Jelaskan fungsi sensor visual pada robot vision!
2. Dalam sistem representasi dan pemodelan citra dikenal tiga pendekatan yaitu: model persepsi model local dan model global. Jelaskan perbedaan dari ketiga model tersebut!
3. Sebutkan dan jelaskan komponen-komponen dalam sebuah sistem pengolahan citra digital!
4. Sebutkan dan jelaskan tahapan pengolahan sinyal digital!
5. Jelaskan dan berikan contoh empat teknik peningkatan kualitas citra: operasi titik, operasi spasial, operasi transformasi dan pewarnaan semu!
6. Jelaskan dan berikan contoh proses restorasi dan penyaringan citra!
7. Dalam teknik analisa citra dikenal tiga pendekatan: ekstraksi fitur, segmentasi dan klasifikasi, jelaskan dan berikan contohnya masing-masing!
8. Jelaskan arsitektur dan cara kerja penangkap gambar (frame grabber) Ellips RIO card!

REFERENSI

TharomTabratas, (2003), http://www.ilmukomputer.com

http://www.cs.cmu.edu/~cmucam/cmucam2/CMUcam2_manual.pdf

http://kliktedy.wordpress.com/2010/05/18/konsep-sensor-warna-1/

http://student.eepis-its.edu/~panca/buku%20ta/materi/line-tracker-robot-avr-c.pdf

BAB VIII

PROYEK ROBOTIKA

8.1 Perancangan Dan Pembuatan Mekanik Robot Line Follower

Dalam merancang mekanik robot line follower harus terlebih dahulu kita tahu, apa saja yang dibutuhkan untuk membangun sebuah robot line follower.

Robot line follower terdiri dari :

1. Sensor line,
2. Roda bebas/freewheel,
3. Motor DC,
4. Ban,
5. Cassis
6. Baterai 12 V, dan
7. Rangkaian robotika.

Sedangkan bahan-bahan yang dibutuhkan dalam pembuatan sebuah robot line follower.

Bahan dasar chassis yang biasa digunakan untuk robot line follower:

1. Akrilik
2. Almunium
3. Spacer

Setelah kita tahu apa saja yang dibutuhkan, selanjutnya kita membuat suatu bentuk rancangan kasarnya, fungsinya agar kita mempunyai gambaran bentuk robot sebelum robot tersebut telah berwujud aslinya. Untuk merancangnya dapat menggunakan software 3dmax, google sketch up, dll.

- **Google sketch up**

Google sketch up merupakan software yang open source (gratis). Pada dasarnya software ini berfungsi untuk pembuatan animasi, pembuatan desain alat, gedung, rumah ataupun robot.

Keunggulan software ini ketimbang yang lain, yaitu :

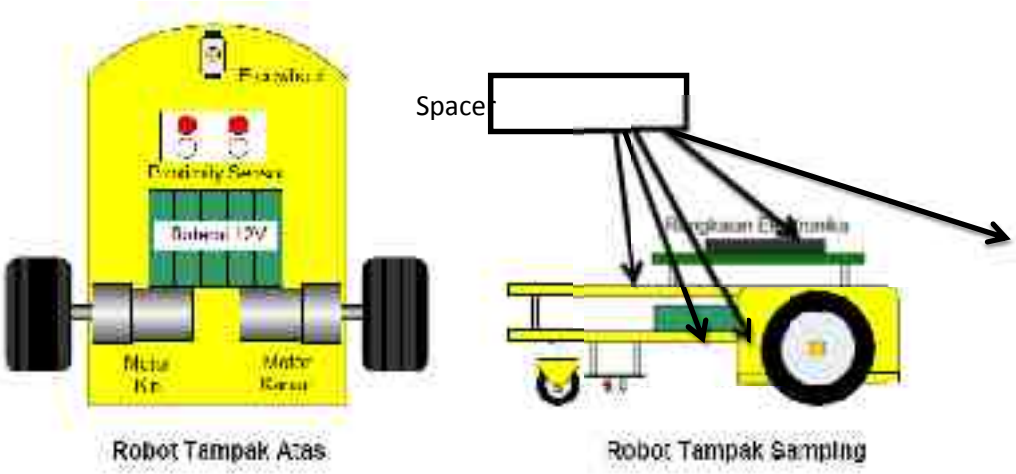
1. Mudah digunakan
2. Cepat dalam pembuatan suatu desain
3. Tidak membutuhkan spesifikasi PC/laptop yang tinggi.
4. Gratis
5. Terdapat banyak model-model bentuk yang sudah jadi, sehingga memudahkan dalam mendisain suatu alat.

Software ini dapat diunduh pada alamat http://sketchup.google.com/download/.

Berikut ini merupakan interface google Sketch Up 8.

Gambar 8.1 Interface dari google sketch up

Berikut contoh desain mekanik pada robot line follower.



Gambar 8.2 Desain mekanik dari pada robot line follower

Setelah robot telah didisain, maka tahap berikutnya adalah tahap pembuatan.

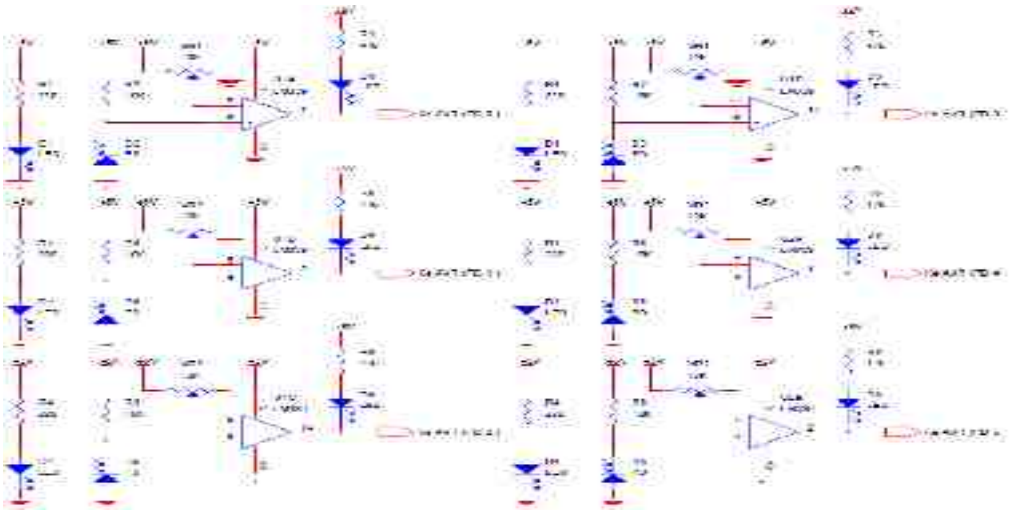
Langkah-langkah pembuatan adalah sebagai berikut :

- Buat terlebih dahulu yang mendasar.
  Contohnya seperti chassis.
- Kemudian tempatkan posisi baterai , Proximity Sensor, Rangkaian elektronika, motor kanan dan motor kiri, dan freewheel. Sesuai dengan desain yang telah dibuat sebelumnya.
- Lalu beri tanda dengan menggunakan spidol. Tujuannya agar memudahkan pada saat pengeboran nanti.
- Bor satu persatu untuk membuat suatu dudukan pada chassis.
- Pasangkan spacer satu-persatu pada lubang yang telah dibuat.
- Pasang semua komponen sesuai dengan gambar disain yang telah dibuat.

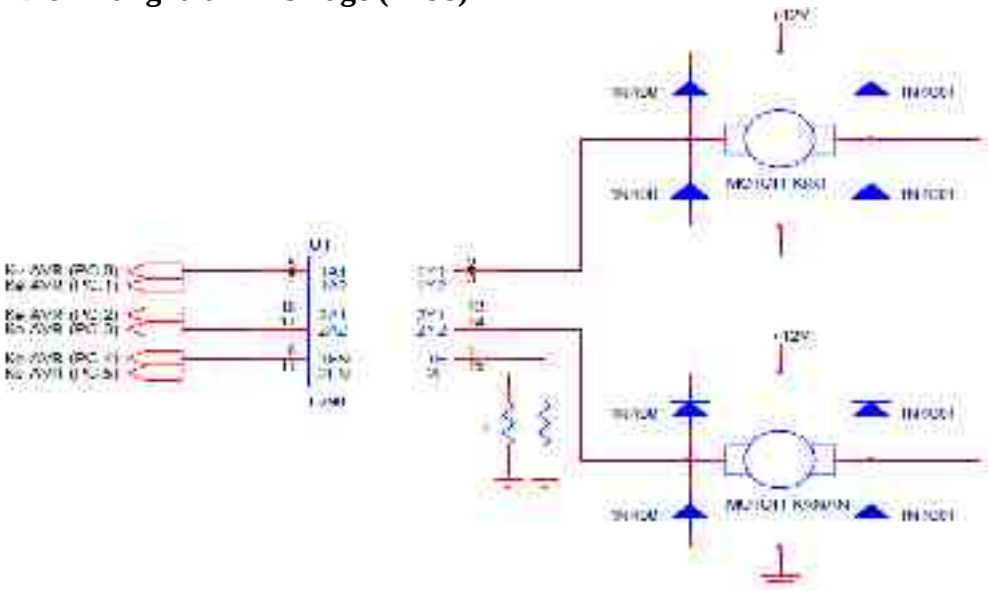
8.2 Perancangan Dan Pembuatan Sistem Elektronik Robot

Dalam merancang suatu sistem elektronik pada robot line follower. Sebelumnya kita harus tahu blok rangkaian robot line follower. Berikut blok rangkaian pada robot line follower.

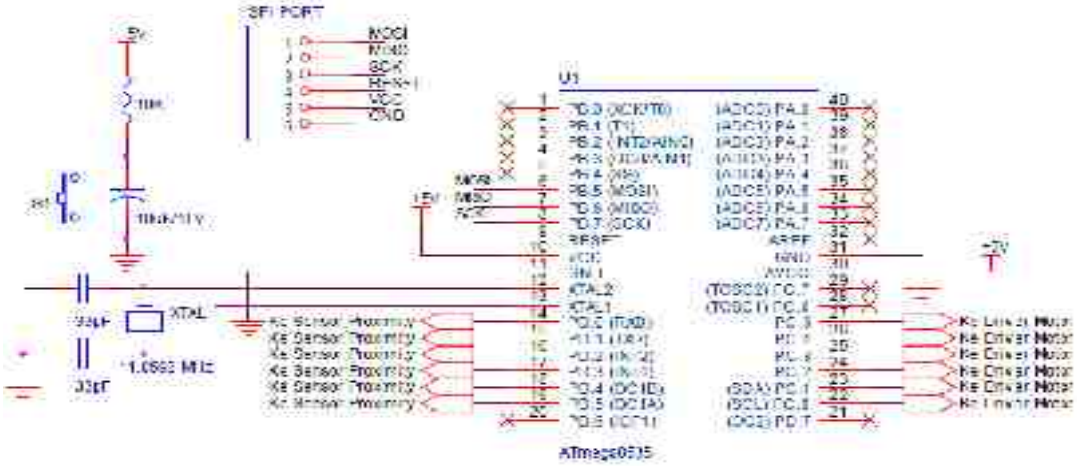
a. Blok Rangkaian Sensor



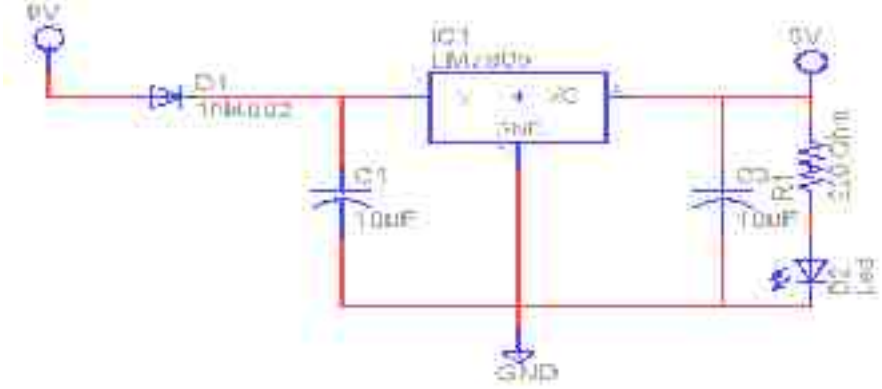
b. Blok Rangkaian H-bridge (L298)



c. Blok Rangkaian MinSys AVR ATmega8535



d. Rangkaian Regulator 5 Volt



Setelah mengetahui blog rangkaian robot line follower, kemudian baru kita gabungan semuanya dalam bentuk jalur PCB. Untuk membuat jalur PCB dapat menggunakan software komputer. Banyak software yang dapat digunakan dalam membuat jalur PCB contohnya seperti eagle, rimu PCB, protel DXP dll. Namun disini penulis menggunakan software eagle dalam membuat jalur PCB.

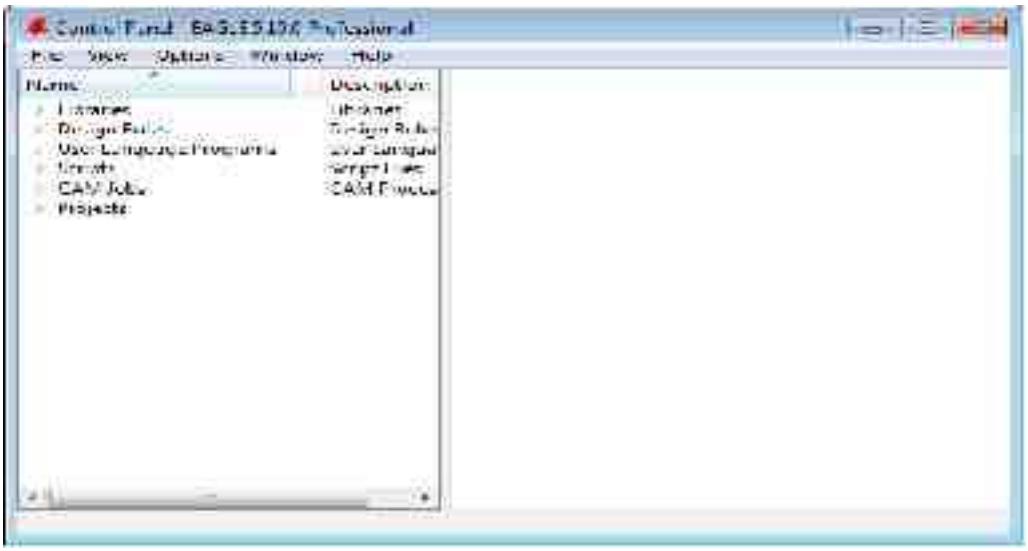
8.2.1 Cadsoft Eagle

Eagle merupakan software yang dapat membuat rangkaian skematic dan jalur PCB. Software ini membutuhkan lisensi, namun dapat digunakan sebagai freeware. Yang membedakannya adalah pada ukuran lembar kerjanya saja yang terbatas jika menggunakan freeware.

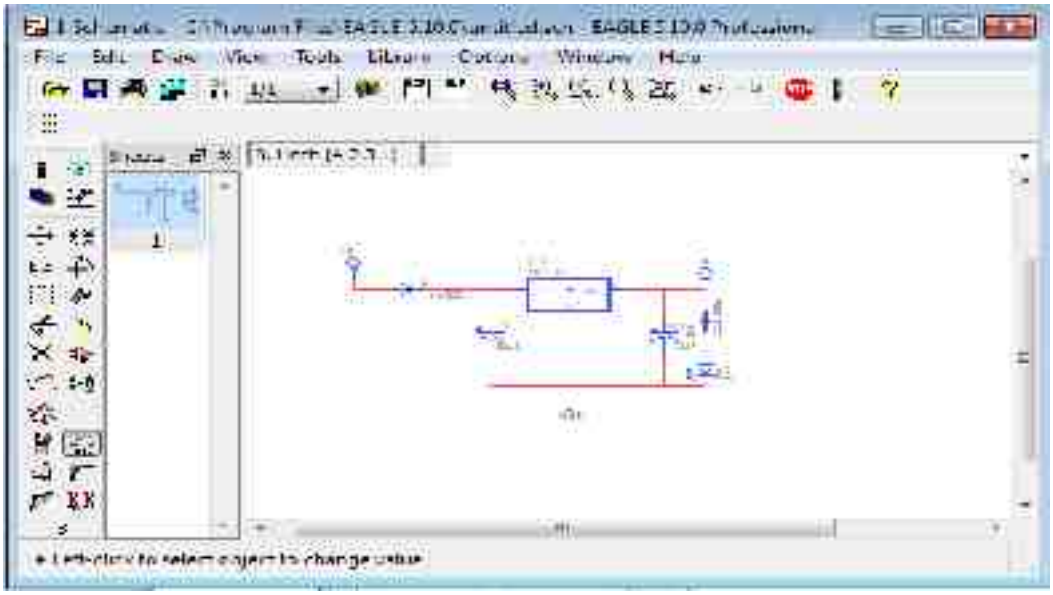
Keunggulan software ini, yaitu :

1. Library yang tersedia banyak.
2. Dapat menggunakan 2 cara dalam pembuatan jalur PCB yaitu manual dan otomatis
3. Tidak membutuhkan spesifikasi PC/laptop yang tinggi.
4. Interfacenya simple dan user friendly
5. Mudah dalam membuat rangkaian atau jalur pcb yang sederhana, hingga yang kompleks.

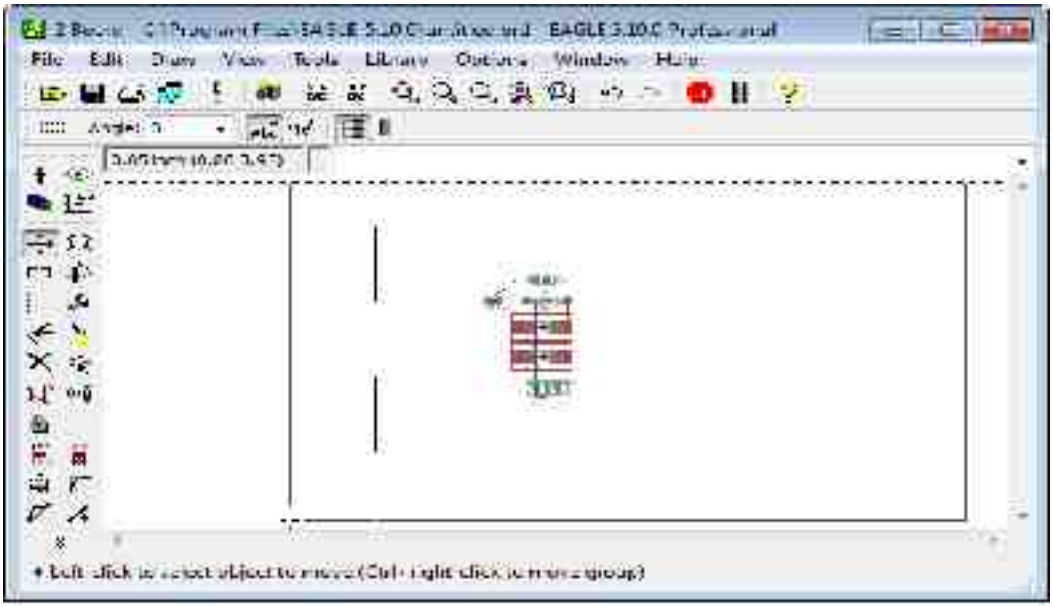
Untuk dapat menggunakan software ini dapat di unduh pada alamat http://www.cadsoft.de/download.htm berikut interface pada eagle.



Gambar 8.3 Interface Control Panel Eagle 5.10.0



Gambar 8.4 Interface perancangan schematic



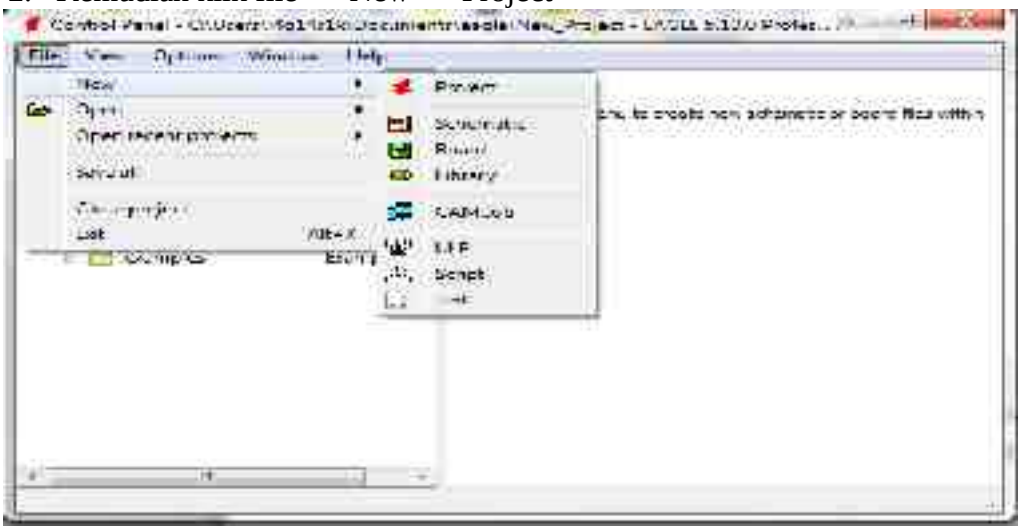
Gambar 8.5 Interface perancangan board /jalur PCB

A. Pembuatan project pada eagle

Langkah-langkah pembuatan project sebagai berikut.

1. Buka aplikasi eagle, klik start >> All Programs >> EAGLE Layout Editor 5.10.0 >> EAGLE 5.10.0

2. Kemudian klik file >> New >> Project

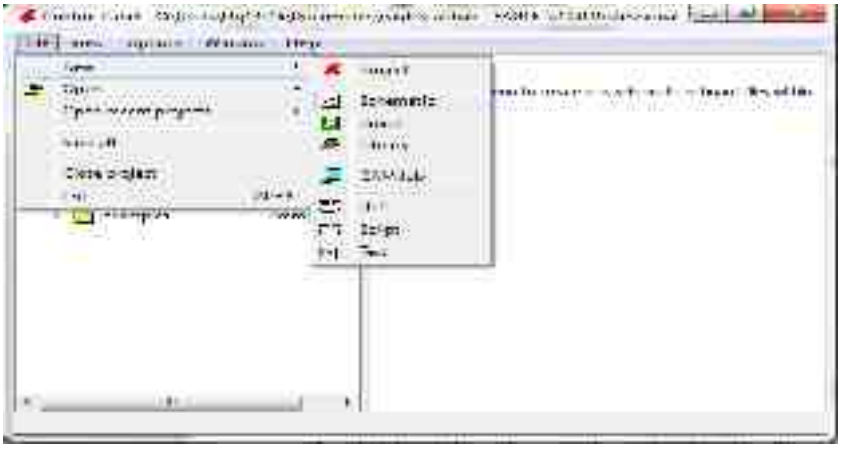


3. Beri nama project. Contohnya : Latihan

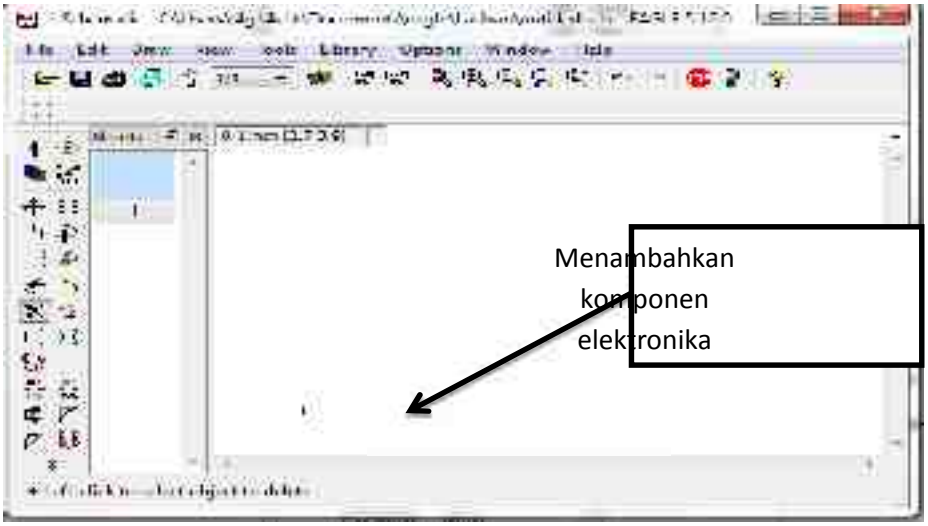
B. Pembuatan Schematic pada Eagle

Langkah-langkah pembuatan schematic sebagai berikut :

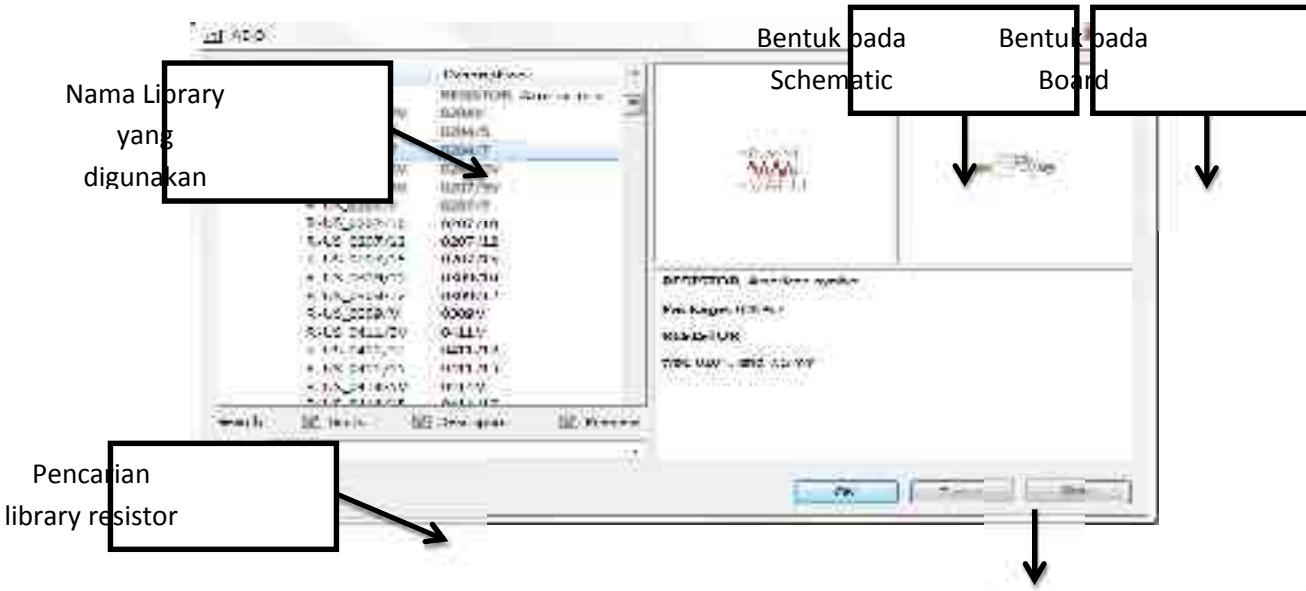
1. Pada control panel eagle. Klik File >> New >> Schematic. Seperti pada contoh gambar dibawah ini.



2. Klik Add pada toolbar

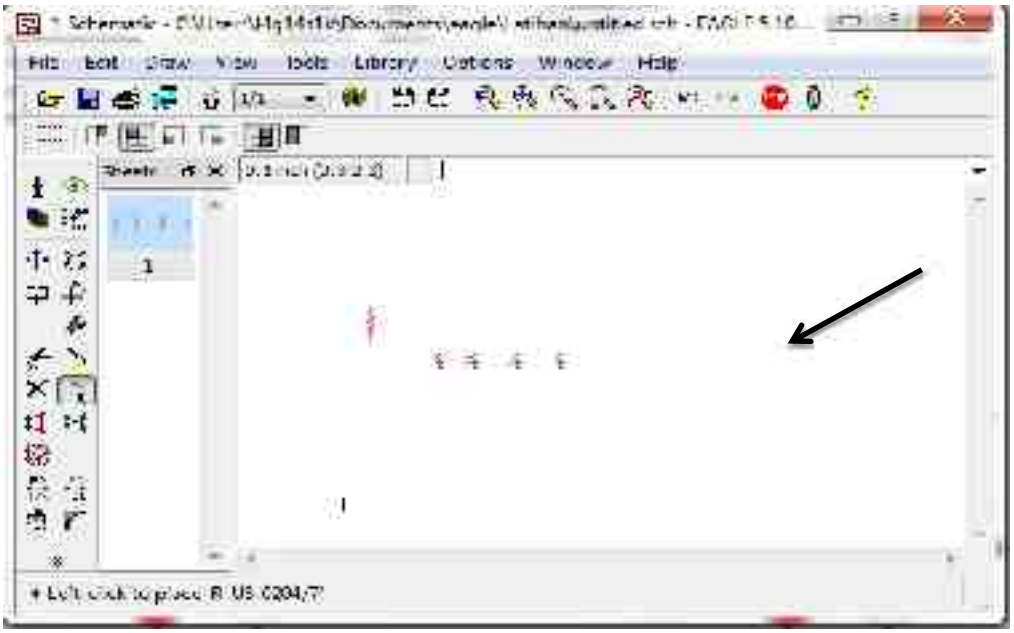


3. Ketik “resistor” pada search, kemudian cari library resistor bernama “R-US_0204/7”, klik Ok.

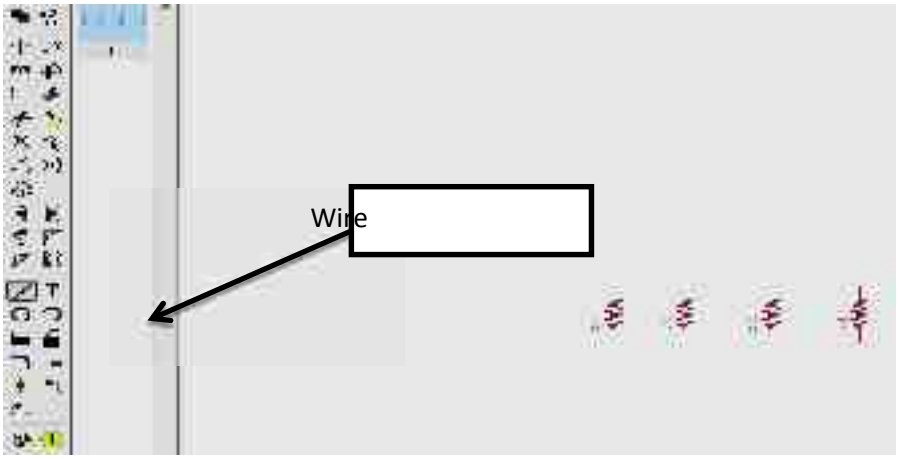


4. Masukkan library tersebut pada layar kerja.

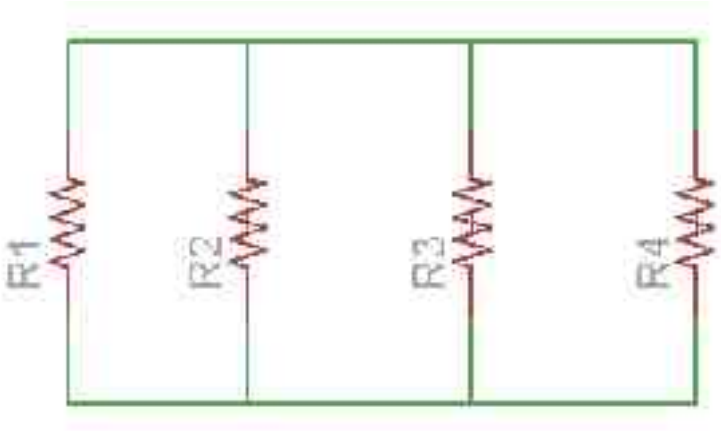
5. Seperti gambar dibawah ini



6. Klik wire / pada toolbar di sebelah kiri



7. Kemudian hubungkan pada tiap ujung resistor hingga membentuk rangkaian parallel.



8. Simpan schematic dengan menekan tombol **Ctrl +S**, beri nama *latihan.sch*

- **Pembuatan Jalur PCB pada eagle**

Setelah schematic dibuat maka tahap selanjutnya adalah pembuatan sebuah board atau jalur PCB.

Pembuatan board atau jalur PCB dapat dibuat dengan 2 cara.

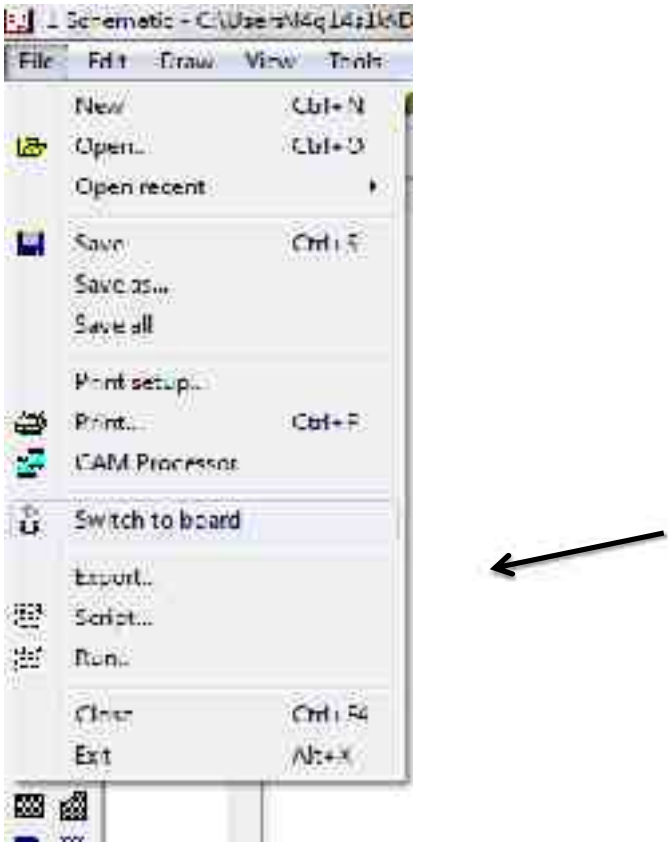
a. Otomatis
   Dibuat schematic terlebih dahulu baru di buat layout boardnya
b. Manual
   Tanpa membuat schematic dan langsung membuat jalur PCB pada layar kerja board.

a. Pembuatan secara otomatis

Berikut langkah-langkah pembuatan layout board secara otomatis.

1. Buka kembali *Latihan.sch* pada control panel eagle

2. Klik File >> Switch to board. Seperti pada gambar berikut :

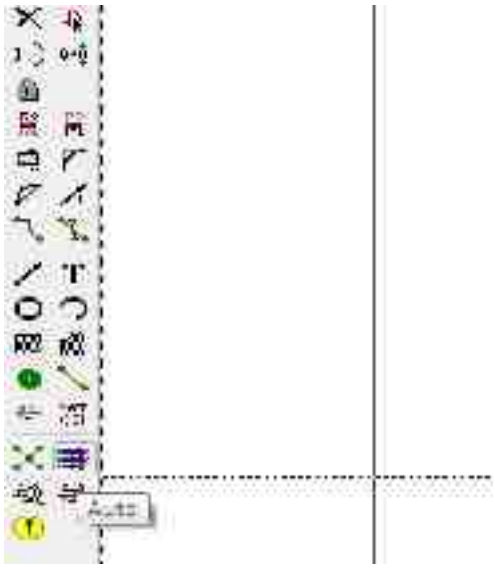


3. Muncul tanda peringatan, pilih Yes

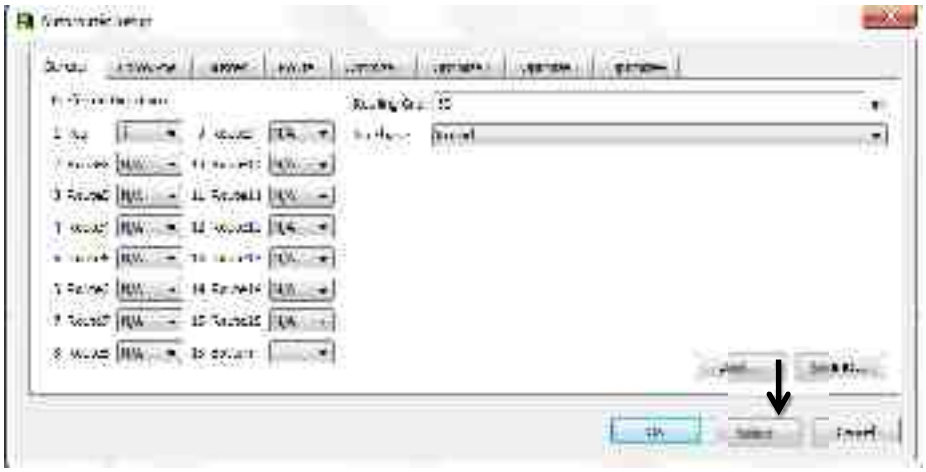
4. Sebuah form baru muncul dengan gambar jalur PCB yang masih tidak beraturan.

5. Pindahkan jalur PCB tersebut ke dalam kotak yang telah tersedia. Atur posisi komponen sesuai yang diinginkan

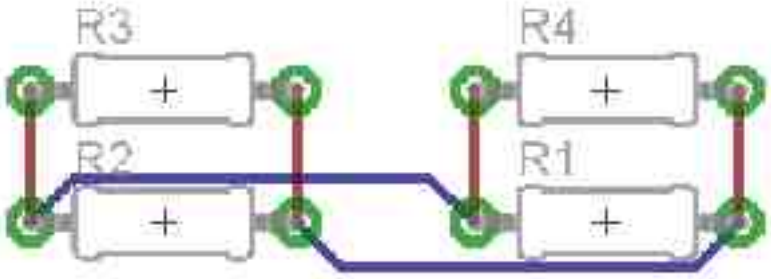
6. Klik Auto atau untuk merapihkan jalur secara otomatis.



7. Maka akan muncul Form Autorouter Setup seperti gambar di bawah, Klik OK



8. Maka hasil yang didapat seperti tampak pada gambar dibawah ini.



- **Pembuatan secara manual**

Untuk pembuatan jalur PCB secara manual dapat mengerjakannya langsung pada boar.

Adapun langkah-langkahnya adalah sebagai berikut :

1. Open eagle pada start menu
2. Kemudian muncul control panel eagle, klik File >> New >> Board
3. Kemudian masukkan library komponen yang dibutuhkan, dengan mengklik Add pada toolbar sebelah kiri.
4. Lalu atur posisi penempatan komponen
5. Klik Wire pada toolbar sebelah kiri. Lalu bentuk jalur sesuai dengan rangkaian

8.3 Perancangan Dan Pembuatan Sistem Kendali Robot

Sebelum kita membuat sebuah sistem kendali, terlebih dahulu kita mengetahui prinsip kerja dari setiap rangkaian elektronika yang terdapat didalam robot line follower. Mulai dari rangkaian sensor, rangkaian minsys, rangkaian penggerak motor (L298), dan rangkaian regulator. Berikut prinsip kerjanya.

8.3.1 Sensor Proximity

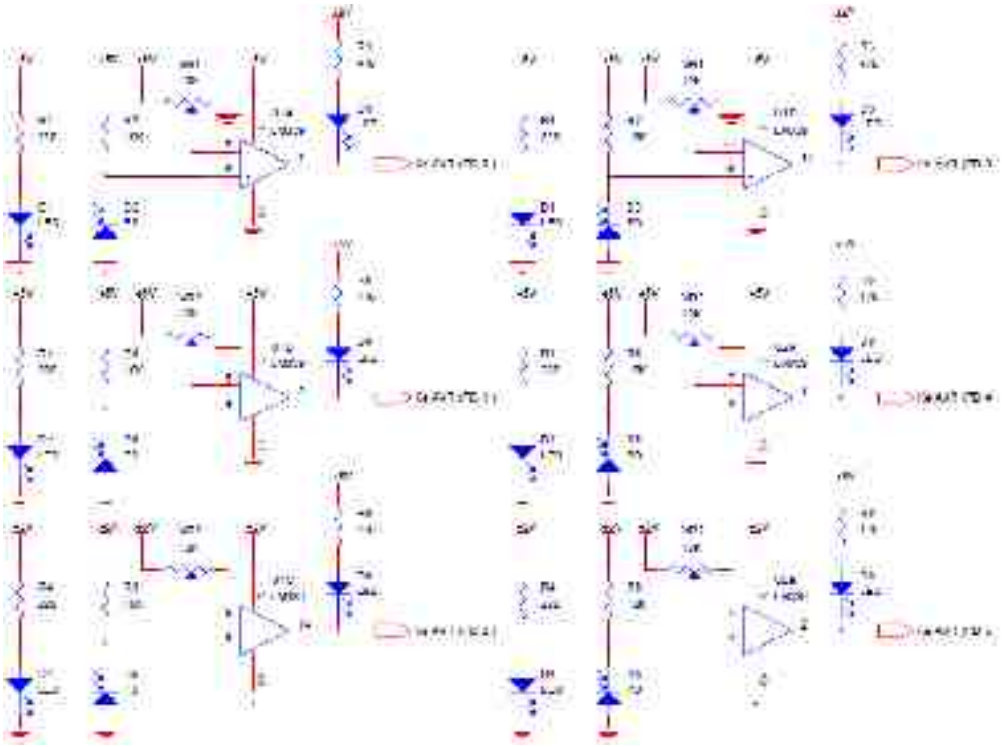
Sensor proximity bisa kita buat sendiri. Prinsip kerjanya sederhana, hanya memanfaatkan sifat cahaya yang akan dipantulkan jika mengenai benda berwarna terang dan akan diserap jika mengenai benda berwarna gelap. Sebagai sumber cahaya kita gunakan LED *(Light Emiting Diode)* yang akan memancarkan cahaya 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 8.6 Prinsip Kerja Sensor Proximity

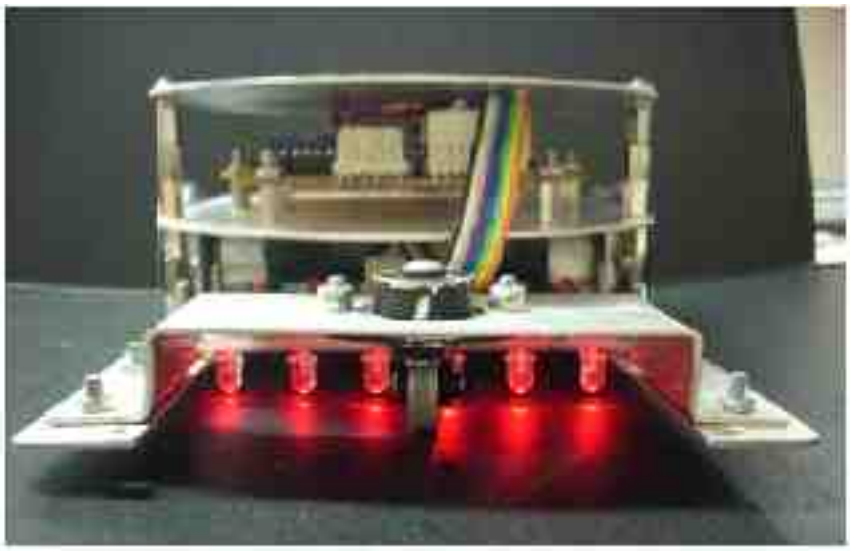
Sifat dari photodioda adalah jika semakin banyak cahaya yang diterima, maka nilai resistansi diodanya semakin kecil. Dengan melakukan sedikit modifikasi, maka besaran resistansi tersebut dapat diubah menjadi tegangan. Sehingga jika sensor berada diatas garis hitam, maka tegangan keluaran sensor akan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

Berikut adalah gambar rangkaian sensor proximity yang digunakan pada robot ini :



Gambar 8.7 Rangkaian Sensor Proximity

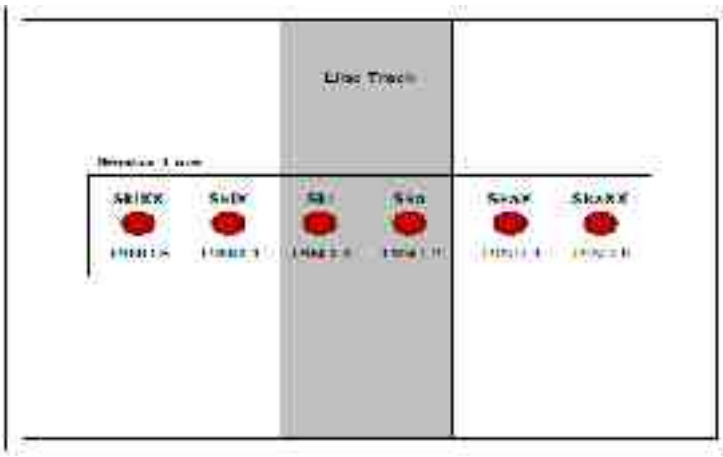
Agar dapat dibaca oleh mikrokontroler, maka tegangan sensor harus disesuaikan dengan level tegangan TTL yaitu 0 – 1 volt untuk logika 0 dan 3 – 5 volt untuk logika 1. Hal ini bisa dilakukan dengan memasang operational amplifier yang difungsikan sebagai komparator. Output dari photodiode yang masuk ke 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 8.8. 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.

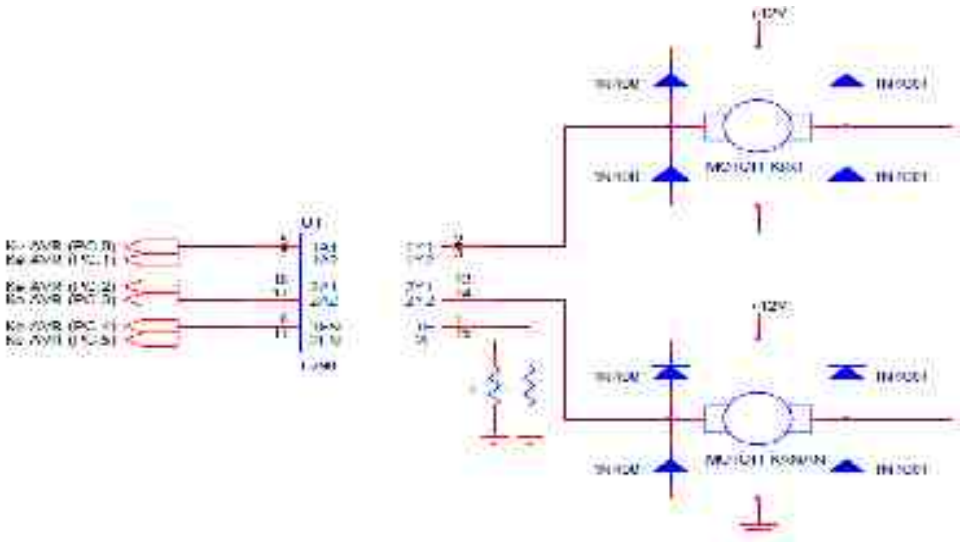
Perhatikan gambar berikut :



Gambar 8.9 Jarak Antar Sensor Proximity

8.3.2 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 8.10. Rangkaian Driver Motor DC

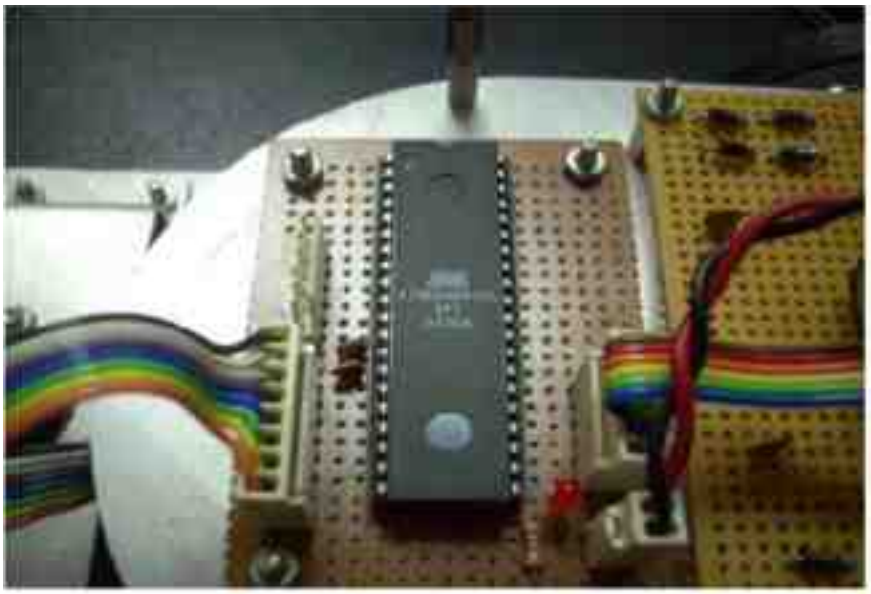
Untuk menentukan arah pergerakan motor maka pada input L298 harus diberikan kondisi sesuai dengan tabel berikut :

Tabel 8.1 Tabel Kebenaran Driver Motor

Motor 1						Motor 2					
Input			Output			Input			Output		
1A1	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

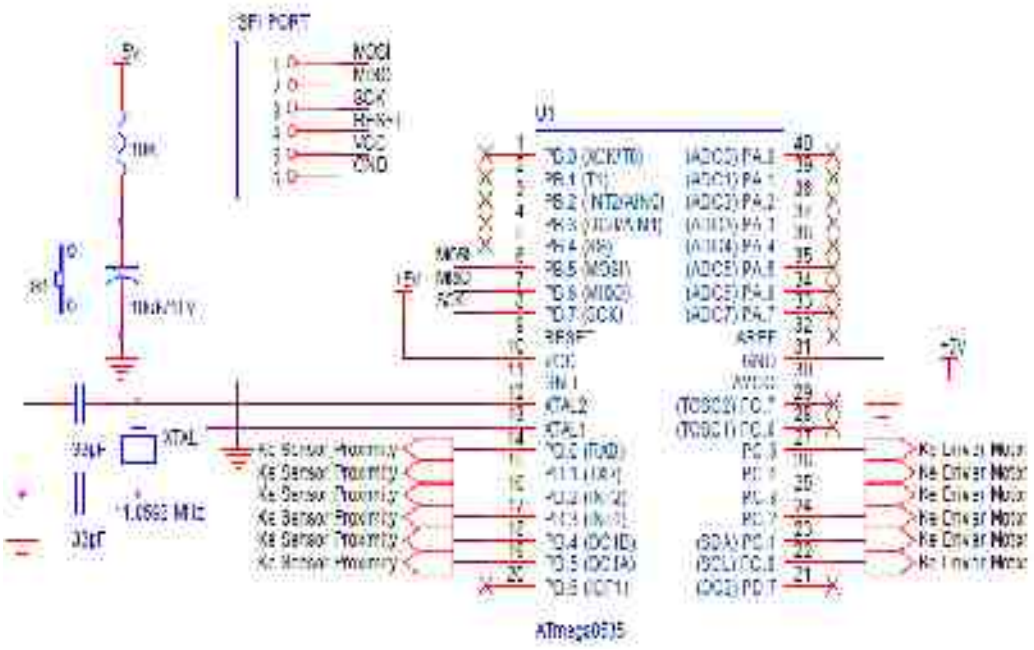
8.3.3 AVR Microcontroller

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



Pada robot line track ini, keluaran sensor proximity dihubungkan ke PortD.0 dan PortD.5 pada mikrokontroler.

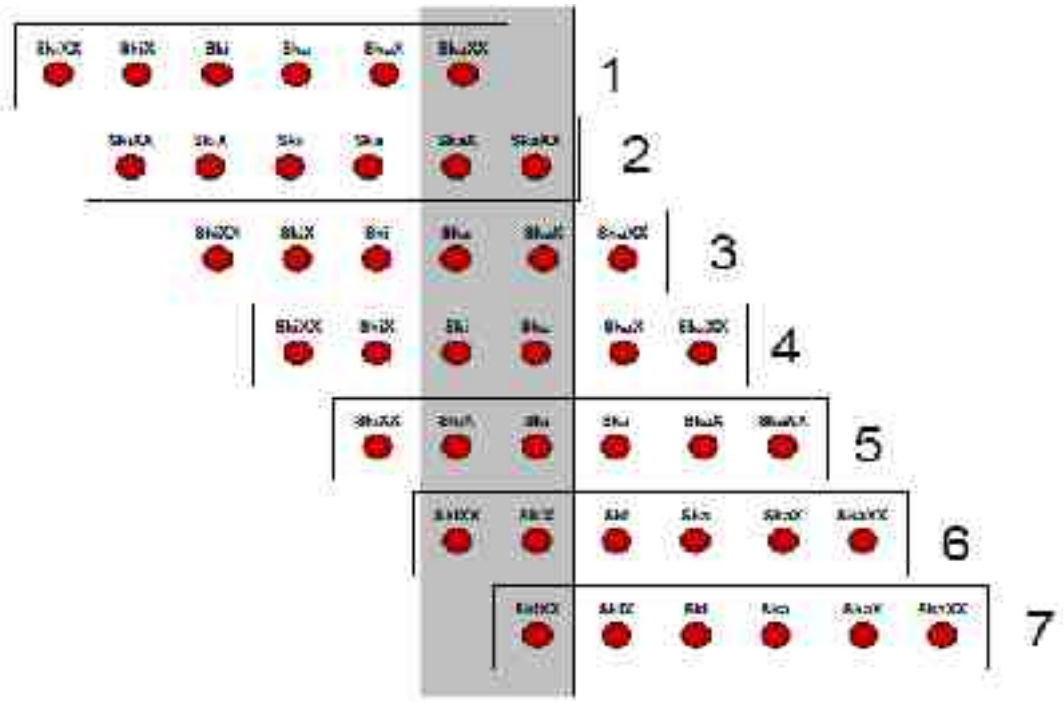
Sedangkan driver motor dihubungkan ke PortC.0 s/d PortC.5 seperti terlihat pada gambar berikut :



8.3.4 Algoritma Pergerakan Robot Line Follower

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 8.11 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 8.2 Aksi Pergerakan Robot

Posisi Sensor	Aksi Robot	Roda Kiri	Roda Kanan
	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

Berikut merupakan contoh program line follower menggunakan CodeVisionAVR :

```
#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();
};
}
```

8.3.5 Algoritma Pergerakan Robot Avoider dengan Fuzzy Logic

Fuzzy logic merupakan logika acak yang menggunakan perbandingan data-data yang didapat oleh sensor yang kemudian dibandingkan, hasil dari perbandingan tersebut akan menentukan jalan dari suatu robot. Apakah robot dalam kondisi miring atau dalam kondisi yang salah. Dalam sistem Fuzzy untuk mendapatkan suatu hasil yang kita inginkan,dapat menggunakan persamaan matematika. Tetapi kita menerapkan suatu sistem kemampuan manusia untuk mengendalikan sesuatu , yaitu dalam bentuk aturan-aturan Jika – maka (*If – Then Rules*), sehingga proses pengendalian akan mengikuti pendekatan secara linguistik, sistem ini disebut dengan sistem kendali logika fuzzy, yang mana sistemkendali logika fuzzy ini tidak memiliki ketergantungan pada variabel – variabel proses kendali. Sistemini dikembangkan dalam bidang teknik kontrol, terutama untuk sistem nonlinier dan dinamis. Dalam pemanfaatan fuzzy logic disini digunakan untuk robot avoider atau robot wallfollower yang pengendaliannya menggunakan beberapa nilai yang didapat dari beberapa sensor pada robot. Sebagai contoh dalam kasus berikut:

Robot maze dengan 5 sensor Ultrasonic (sensor PING) dan 2 sensor infra merah sharp GP2D15 aktif low.

Perbandingan sebagai berikut:

Pingki = ping1 + ping2+ ping3

Pingka = ping3+ ping4+ping5

Jika keadaan Pingki = Pingka, dan S1=S2=0, maka robot dalam keadaan sejajar dengan atau dengan kata lain robot dalam posisi lurus, jadi untuk keadaan ini robot akan bergerak maju lurus.

Pada saat seperti ini keadaan data ping akan menjadi :

Pingki > PIngka, dan S1 != S2,

Maka dapat diketahui bahwa posisi robot dalam keadaan miring atau tidak lurus, dalam keadaan ini, robot akan bergerak serong kekiri untuk meluruskan pergerakannya, untuk kondisi sebaliknya yaitu Pingki < PIngka dan S1 != S2, maka robot akan bergerak sebalinya.

Tetapi jika Pingki == PIngka, maka robot akan bergerak lurus kembali, karena kondisi ini jarang terjadi maka biasanya robot akan terus berjalan secara bergelombang. Jadi pada kasus ini kondisi untuk ping1 dan ping5 ditambahkan kondisi bahwa ping1 atau ping5 harus dalam keadaan >= 3cm dari jarak dinding. Hal ini untuk menghindari robot menabrak dinding.

Pada dasarnya sistem Fuzzy Logic bukan merupakan system yang kompleks, tetapi merupakan system yang bekerja cepat dengan langsung mengerjakan perintah yang ada pada kondisi yang terjadi. Sistem Fuzzy seperti ini sering digunakan untuk algoritma WallFollower dan Avoider yang memerlukan penanganan system yang cepat dan untuk arena yang luas.

Berikut merupakan robot Avoider

```
#include <mega8535.h>
#include <delay.h>

// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>

void tampil(unsigned char dat)                  //rutin tampil pada LCD
{
```

```
        unsigned char data;

        data = dat / 100;
        data+=0x30;
        lcd_putchar(data);

        dat%=100;
        data = dat / 10;
        data+=0x30;
        lcd_putchar(data);

        dat%=10;
        data = dat + 0x30;
        lcd_putchar(data);
        lcd_putsf(" ");
}

// ********** PING Function **********

#define PulseOutP1 PORTC.0
#define PulseInP1 PINC.0
#define PulseDirP1 DDRC.0

unsigned int counter, distance;
//unsigned char Msg1[16], Msg2[16];

unsigned char sensePING1(void)
{
  counter=0;                // Counter initial value
  PulseDirP1=1;              // Set port as output
  PulseOutP1=1;               // Send HIGH pulse to output port
  delay_us(5);              // Delay for 5 us
  PulseOutP1=0;               // Send LOW pulse to output port
  delay_us(750);             // HOLDOFF time 750 us
  PulseDirP1=0;              // Set port as input
```

```
   PulseOutP1=1;              // Pull-up signal is activated
   while (PulseInP1==0) {}      // If input port level is LOW
   while (PulseInP1==1)         // If input port level is HIGH
   {
counter++;                       // Accumulate counter until reflected
                                 // signal arrived
   }
```

```
distance=(counter*0.034442)/0.8;      // Multiply counter value with
                                      // V=344.424m/s to get real
                                      // distance in cm

return distance;
}


#define dirA   PORTD.0
#define dirB   PORTD.1
#define enka   PORTD.2
#define dirC   PORTD.3
#define dirD   PORTD.4
#define enki   PORTD.5

void maju()
{
     dirC=1;dirD=0;
     dirA=0;dirB=1;
     enka=1;enki=1;
}

void belok_kanan()
{
     dirC=0;dirD=1;
     dirA=0;dirB=1;
     enka=1;enki=1;
}
void mundur()
{
     dirC=0;dirD=1;
     dirA=1;dirB=0;
     enka=1;enki=1;
}
```

```
void stop()
{
      dirC=1;dirD=1;
      dirA=1;dirB=1;
      enka=0;enki=0;
}
void main(void)
{
      unsigned int sensorP1;

PORTA=0x00;
DDRA=0x00;

PORTB=0x00;
DDRB=0x00;

PORTC=0x00;
DDRC=0x00;

PORTD=0x00;
DDRD=0xFF;

// LCD module initialization
lcd_init(16);
while (1)
    {
    lcd_gotoxy(0,0);
     sensorP1=sensePING1();
     tampil(sensorP1);delay_ms(50);

     if (sensorP1<=10)
     {
           stop();
           mundur();delay_ms(200);
           while(sensorP1<=10)
```

```
            {
                belok_kanan();
            }
        }
       else maju();
      };
}
```

LATIHAN

1. Buat rancangan (desain) mobile robot Pemadam Api yang menggunakan mikrokontroller AVR ATmega16 dengan sensor ultrasonic, sensor infra merah, sensor api maupun sensor kompas, serta dilengkapi dengan tangki air maupun pompanya!

2. Buat program kontrolnya untuk mengendalikan mobile robot Pemadam Api hasil rancangan pada soal no 1 di atas!

REFERENSI

http://robotika.blog.gunadarma.ac.id/?p=44

Bahasa C UntukMikrokontroler ATMEGA8535, M.aryheryanto, ST & Ir. WisnuAdi P.

REFERENSI

http://ekstrarobotik.tripod.com/id3.html

http://en.wikipedia.org/wiki/Humanoid_robot

http://newstekno.blogspot.com/2009/02/fungsi-robot.html

Agilent.(1999). *Quadrature Decoder/Counter Interface ICs. Technical Data*, Agilent Technologies, Inc., http://www.semiconductor.agilent.com

Analog Devices. (1999). *ADXL105 Datasheet*, Analog Devices, Inc., http://www.analog.com

Applied Measurement. (1998). *Miniature Series LVDT: Displacement Transducer. AML/M Series Data sheet*, Applied Measurement, Ltd., http://www.appmeas.co.uk

Braunl, T. (2003.*Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems.*BukuTeks. Berlin: Springer Verlag Berlin Heidelberg, Inc.

Dinsmore. (1999). *Datasheet Dinsmore Analog Sensor No. 1525*, Dinsmore Instrument Co., http://dinsmoregroup.com/dico

Honeywell.(2005). *Digital Compass Solution.* Sensor Product Datasheet, Honewell, Inc., http://www.magneticsensors.com

Precision Navigation. (1998). *Vector Electronic Module. Application Notes*, Precision Navigation, Inc., http://www.precisionnav.com

http://www.societyofrobots.com/actuators.shtml

http://ocw.gunadarma.ac.id/course/diploma-three-program/study-program-of-computer-engineering-d3/robotika/mekanika-robotika

http://id.wikipedia.org/wiki/debugging

Agilent.(1999). *Quadrature Decoder/Counter Interface ICs. Technical Data*, Agilent Technologies, Inc., http://www.semiconductor.agilent.com

Analog Devices. (1999). *ADXL105 Datasheet*, Analog Devices, Inc., http://www.analog.com

Applied Measurement. (1998). *Miniature Series LVDT: Displacement Transducer. AML/M Series Data sheet*, Applied Measurement, Ltd., http://www.appmeas.co.uk

Braunl, T. (2003.*Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems.*BukuTeks. Berlin: Springer Verlag Berlin Heidelberg, Inc.

Dinsmore. (1999). *Datasheet Dinsmore Analog Sensor No. 1525*, Dinsmore Instrument Co., http://dinsmoregroup.com/dico

Honeywell.(2005). *Digital Compass Solution.* Sensor Product Datasheet, Honewell, Inc., http://www.magneticsensors.com

Precision Navigation. (1998). *Vector Electronic Module. Application Notes*, Precision Navigation, Inc., http://www.precisionnav.com

http://www.societyofrobots.com/actuators.shtml

http://ocw.gunadarma.ac.id/course/diploma-three-program/study-program-of-computer-engineering-d3/robotika/mekanika-robotika

http://en.wikibooks.org/wiki/Robotics/Design_Basics/Building_Materials

http://www.societyofrobots.com/microcontroller_tutorial.shtml

http://wapedia.mobi/id/Mikrokontroler

http://prime.jsc.nasa.gov/ROV/systems.html

http://en.wikipedia.org/wiki/Suspension_(vehicle)

http://www.societyofrobots.com/mechanics_statics.shtml

Hewit, J. R danMarouf, K. B. (1996).*Practical Control Enhancement via Mechatronics Design*.IEEE Trans. Industrial Electronics.43(1). 16-22

Hewit, J.R. and Burdess (1981).*Fast Dynamic Decoupled Control for Robotics Using Active Force Control.*Trans. Mechanism and Machine Theory.**16**(5).535-542.

Hewit, J.R. danBurdess, J.S. (1986).*An Active Method for the Control of Mechanical System in The Presence of Unmeasurable Forcing*.Trans. Mechanism and Machine Theory.**21**(3).393-400.

Kwek, L. C., Wong, E. K., Loo, C.K. danRao, M.V.V. (2003).*Application of Active Control and Iterative Learning in a 5-Link Biped Robot*.Jurnal of Intelligent and Robotic Systems, 37, 143-162.

Microchip. (2001). PIC16F87X *Data Sheet, 28/40-pin 8 bit CMOS Flash Microcontrollers*. Data Sheet. Microchip Technology, Inc.

Ogata, K. (2002). *Modern Control Engineering: Fourth Edition*. BukuTeks. New Jersey: Prentice Hall-Pearson Education International.

Pitowarno, E. dan Musa Mailah. (2005). Motion Resolved Acceleration and Knowledge *Based Fuzzy Active Force Control for MobileManipulator*. Proc. Int'1 Conf. on Robotics, Vision, Informations and signal Processing (ROVISP 2005), Penang, Malaysia.

Pitowarno, E., Musa MailahdanHishamuddinJamaluddin. (2001). *Trajectory Error Pattern Refinement of A Robot Control Scheme Using A Knowledge-Based Method*.Proc. IEEE Int'1 Conf. on Information, Communications & Signal Processing (ICICS 2001), Singapore, P0301.

Pitowarno, E Musa MailahdanHishamuddinJamaluddin.(2005). *Motion Control for Mobile Manipulator Using Resolved Acceleration and Iterative-Learning Active Force Control*.Proc. Int'1 Conf. on Mechatronics (ICOM 2005) Kuala Lumpur, p542-549.

Uchiyama, M. (1989).*Contrrol of Robot Arms*.Trans. Japan Society of Mechanical Engineers. III. **32**(1). 1-9.

Hewit, J. R danMarouf, K. B. (1996).*Practical Control Enhancement via Mechatronics Design*.IEEE Trans. Industrial Electronics.43(1). 16-22

Hewit, J.R. and Burdess (1981).*Fast Dynamic Decoupled Control for Robotics Using Active Force Control.*Trans. Mechanism and Machine Theory.**16**(5).535-542.

Hewit, J.R. danBurdess, J.S. (1986).*An Active Method for the Control of Mechanical System in The Presence of Unmeasurable Forcing*.Trans. Mechanism and Machine Theory.**21**(3).393-400.

Kwek, L. C., Wong, E. K., Loo, C.K. danRao, M.V.V. (2003).*Application of Active Control and Iterative Learning in a 5-Link Biped Robot*.Jurnal of Intelligent and Robotic Systems, 37, 143-162.

Microchip. (2001). PIC16F87X *Data Sheet, 28/40-pin 8 bit CMOS Flash Microcontrollers*. Data Sheet. Microchip Technology, Inc.

Ogata, K. (2002). *Modern Control Engineering: Fourth Edition*. BukuTeks. New Jersey: Prentice Hall-Pearson Education International.

Pitowarno, E. dan Musa Mailah. (2005). Motion Resolved Acceleration and Knowledge *Based Fuzzy Active Force Control for MobileManipulator*. Proc. Int'1 Conf. on Robotics, Vision, Informations and signal Processing (ROVISP 2005), Penang, Malaysia.

Pitowarno, E., Musa MailahdanHishamuddinJamaluddin. (2001). *Trajectory Error Pattern Refinement of A Robot Control Scheme Using A Knowledge-Based Method*.Proc. IEEE Int'1 Conf. on Information, Communications & Signal Processing (ICICS 2001), Singapore, P0301.

Pitowarno, E Musa MailahdanHishamuddinJamaluddin.(2005). *Motion Control for Mobile Manipulator Using Resolved Acceleration and Iterative-Learning Active Force Control*.Proc. Int'1 Conf. on Mechatronics (ICOM 2005) Kuala Lumpur, p542-549.

Uchiyama, M. (1989).*Control of Robot Arms*.Trans. Japan Society of Mechanical Engineers. III. **32**(1). 1-9.

http://teundiksha.files.wordpress.com/2010/04/sekilas20codevisionavr.pdf

Bahasa C UntukMikrokontroler ATMEGA8535, M.aryheryanto, ST & Ir. WisnuAdi P.

SoebhaktiHendawan, (2007). STBasic AVR Microcontroller Tutorial, http://www.polibatam.ac.id

http://id.wikipedia.org/wiki/debugging

Bayle, B., Fourqet, J. Y., Lamiraux. F. and Renaud, M. (2002).*Kinematic Control of Wheeled Mobile Manipulators* , Proc. IEEE/RSJ Int'l Conf on Intellegent Robots and System. 1572-1577.

Braunl, T. (2003). *Embedded Robotics : Mobile Robot Design and Apllication with Embedded System.* Buku Teks , Berlin: Springer-Verlag Berlin Heidelberg, Inc.

Craig . J. J (2005).*Introduction to Robotic : Mechanical and control* . Buku

Teks. Sydney: Pearson Education, Inc.

D'souza, A., Vijayakumar, S., and Schaal, S. (2001). *Learning inverse Kinematics.* Proc. IEEE/RSJ Int'l Conf. Intelligent Robot and System. 298-303.

Fu, K. S., Gonzales, R. C., and Lee, C. S. G. (1987). Robotics: *Vision, Sensing, Vision, and Intelligence.* Buku Teks. New York: McGrawHill, Inc.

Hewit, J.R. and Burdess (1981). *Fast Dynamic Decoupled Control for Robotic Using Active Force Control*. *Trans. Mechanism and Machine Theory.* 16(5). 535-542.

Hibbeler, R. C. dan Fan, S. C. (1997). *Enginering Mechanics: Statics.* Buku Teks. Singapore: Simon & Schuster (Asia) Pte. Ltd.

http://id.wikipedia.org/wiki/Robot_Mobile

http://id.wikipedia.org/wiki/Aktuator

Sigit, Riyanto, "Robotika, Sensor & Aktuator", Graha Ilmu, Yogyakarta, 2007

http://id.wikipedia.org/wiki/Robot_Mobile

www.innovativeelectronics.com, Februari 2008

www.acroname.com, April 2008

Sigit, Riyanto, "Robotika, Sensor &Aktuator", GrahaIlmu, Yogyakarta, 2007

www.alldatasheet.com, Februari 2008

Eko Putra, Agfianto, *TeknikAntarmukaKomputer :KonsepdanAplikasi,*GrahaIlmu, Yogyakarta, 2002.

TharomTabratas, (2003), http://www.ilmukomputer.com

http://www.cs.cmu.edu/~cmucam/cmucam2/CMUcam2_manual.pdf

http://kliktedy.wordpress.com/2010/05/18/konsep-sensor-warna-1/

http://student.eepis-its.edu/~panca/buku%20ta/materi/line-tracker-robot-avr-c.pdf

http://robotika.blog.gunadarma.ac.id/?p=44

Bahasa C UntukMikrokontroler ATMEGA8535, M.aryheryanto, ST & Ir. WisnuAdi P.