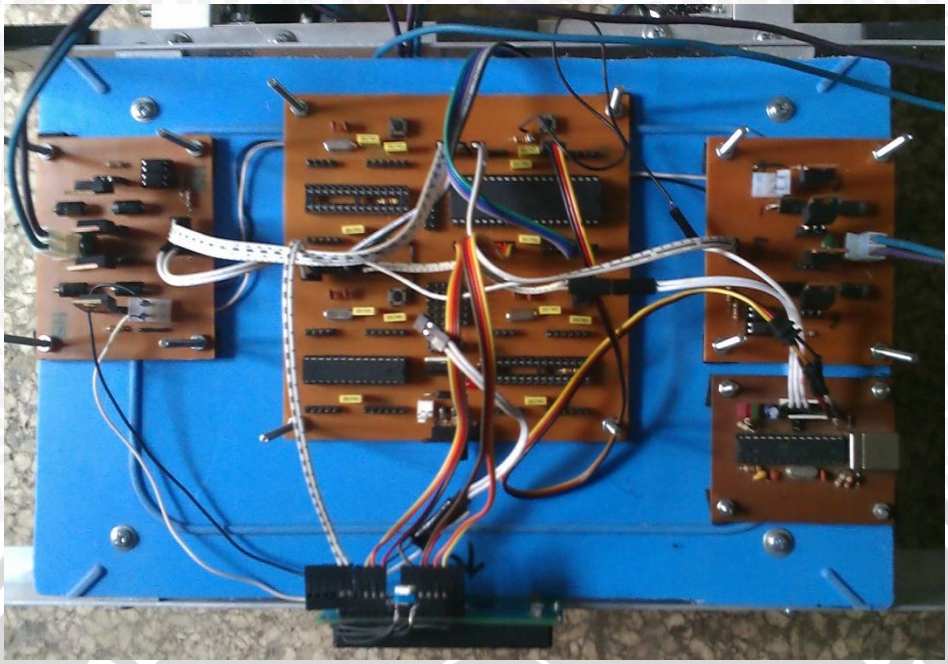
LAMPIRAN

LAMPIRAN 1

FOTO ALAT

Chassis dan Kerangka Robot

Motor DC dan *Gearbox*



Rangkaian Elektrik sistem

LAMPIRAN 2

LISTING PROGRAM

Mikrokontroler *Master*

```
//0x0f=maju
//oxf0=mundur
#definewaktu_sampling      40
#define time_sampling      (65535-43200*waktu_sampling/1000)
//#define time_sampling      60351;//120ms
//#define time_sampling      56895//200ms
//#define time_sampling      48255//400ms
//#define time_sampling      39615//600ms
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdlib.h>
#include <math.h>
#include "SPI.h"
#include "lcd.h"

//definisi fuzzy
#define LH    0
#define LM    1
#define Z     2
#define RM    3
#define RH    4
#define Vkac        0
#define Vkas        1
#define Vt          2
#define Vkis        3
#define Vkic        4
```

```
volatile float output;
/********
alpha = -15 sd 15
l=0
30 28 22 19,15 11  8   2   0
15 13 7  4  0  -4 -7 -13  -15
********/
//Aturan Fuzzy
volatile unsigned char rule[5][5] =
        {
                {RH,RH,RH,RM,Z},    //posisi kiri
                {RH,RH,RM,Z,LM},    //posisi tengah
                {RH,RM,Z,LM,LH},    //posisi kanan
                {RM,Z,LM,LH,LH},
                {Z,LM,LH,LH,LH}
        };

volatile float Error_sudut[5];
volatile float Del_Error_sudut[5];
volatile float Rule_out[25];

int var_Kp=1, var_Ki=1, var_Kd=1;
volatile int lpwm, rpwm,
MAXPWM=255,MAXPWMKIRI=192,MAXPWMKANAN=192,
MINPWM=0,bobot=0;
int proportional,derivative,integral,last_proportional,pangkat=0;
volatile float delta_error1,delta_error2,delta_error3,delta_error4,delta_error5;
volatile int pulsa=0,pulsamsb=0,rpm,speed_temp,speed,PWM;
volatile float
PV,P,I,D,error,last_error,MV,Kp=15,Ki=21.429,Kd=2.625,delta_error;//3,10,0.22
5
volatile uint8_t data_spi_rx;
```

```
volatile uint8_t data_spi_tx,bel=0;
volatile uint8_t
data[8],i=0,a[8],tengah[8]={94,76,101,152,143,95,100,153},sensor,sudut,hasil_ou
t;
volatile int dirka,dirki,pwmka,pwmki;
volatile int data_PV;
volatile int rpm_kanan,rpm_kiri;
volatile int temp0_kiri,temp1_kiri,temp0_kanan,temp1_kanan;
uint8_t siki,sika;
volatile int EN_send=1;
char str[6];
volatile unsigned int count;
volatile unsigned long jarak,waktu;

void setMLeftSpeed(volatile int speed);
void setMRightSpeed(volatile int speed);
void set_motors(volatile int m_left,volatile int m_right);
void scanBlackLine2();
void InitUART(void);
void TransmitByte( unsigned char data );
void baca_sudut();

volatile int puluhribu,ratusan,puluhan,satuan,temp_TCNT1,temp,temp1,temp2;
void hex2ascii(volatile uint8_t data);
ISR(TIMER1_OVF_vect){
        TCNT1 = time_sampling;
        TCNT0 = 0;
        //SENSOR
        PORTA=0b10100000;
        SPI_MasterTransmit_int(0xAA);
        SPI_MasterTransmit_int(0xff);
        sensor  = SPDR;
```

```
baca_sudut();
temp = TCNT0;
scanBlackLine2();
//MOTOR KANAN
PORTA=0b01100000;
SPI_MasterTransmit_int(255);
SPI_MasterTransmit_int(255);
if(dirka)
        SPI_MasterTransmit_int(0x0f);//maju
else
        SPI_MasterTransmit_int(0xf0);//mundur
temp0_kanan=SPDR;//lsb

SPI_MasterTransmit_int(pwmka%255);//lsb
temp1_kanan=SPDR;//msb

SPI_MasterTransmit_int(pwmka/255);//msb

//temp1 = TCNT0;
//MOTOR KIRI
PORTA=0b11000000;
SPI_MasterTransmit_int(255);
SPI_MasterTransmit_int(255);
if(dirki)
        SPI_MasterTransmit_int(0x0f);
else
        SPI_MasterTransmit_int(0xf0);
temp0_kiri=SPDR;//lsb

SPI_MasterTransmit_int(pwmki%255);//lsb
temp1_kiri=SPDR;//msb
```

```
        SPI_MasterTransmit_int(pwmki/255);//msb
        rpm_kanan = temp1_kanan*255 + temp0_kanan;
        rpm_kiri  = temp1_kiri*255  + temp0_kiri;

        temp = sudut;
        puluhribu = temp/1000 +48;
        ratusan   = (temp%1000)/100 +48;
        puluhan   = (temp%100)/10 +48;
        satuan    = (temp%10) +48;
        TransmitByte(puluhribu);
        TransmitByte(ratusan);
        TransmitByte(puluhan);
        TransmitByte(satuan);
        TransmitByte(0x0D);

        //PORTA = ~PORTA;
}
void tampil(int dat);

/*************************************
=====================================
        Sub Fungsi Fuzzy
=====================================
*************************************/
//fungsi keanggotaan
float triangle(float value, float x0, float x1, float x2)
{
        float result=0;
        if((value <= x0) || (value >= x2))result=0;
        else if(value == x1)result=1;
        else if((value>x0) && (value<x1))result= ((value-x0)/(x1-x0));
        else result = (((-value)+x2)/(x2-x1));
```

```
        return result;
}
float grade(float value, float x0, float x1)
{
        float result=0;
        if(value <= x0) result = 0;
        else if (value >= x1) result =1;
        else result=((value-x0)/(x1-x0));
        return result;
}
float reverse_grade(float value, float x0, float x1)
{
        float result=0;
        if (value >= x1) result =0;
        else if(value<=x0) result = 1;
        else result = (((-value)+x1)/(x1-x0));
        return result;
}
float trapesium(float value,float x0,float x1,float x2,float x3)
{
        float result=0;
        if((value<=x0) || (value>=x3)) result=0;
        else if((value>=x1)&&(value<=x2))result=1;
        else if((value>x0)&&(value<x1))result=((value-x0)/(x1-x0));
        else result = ((-value+x3)/(x3-x2));
        return result;
}

void fuzzifikasi()
{
        Error_sudut[LH]         = reverse_grade(error,-15,-7.5);
        Error_sudut[LM]        = triangle(error,-15,-7.5,0);
```

```
        Error_sudut[Z]          = triangle(error,-7.5,0,7.5);
        Error_sudut[RM]         = triangle(error,0,7.5,15);
        Error_sudut[RH]         = grade(error,7.5,15);

        Del_Error_sudut[LH] = reverse_grade(delta_error1,-7.5,-3.75);
        Del_Error_sudut[LM] = triangle(delta_error2,-7.5,-3.75,0);
        Del_Error_sudut[Z]  = triangle(delta_error3,-3.75,0,3.75);
        Del_Error_sudut[RM]= triangle(delta_error4,0,3.75,7.5);
        Del_Error_sudut[RH] = grade(delta_error5,3.75,7.5);
}
//pengecekan aturan fuzzy yang dipakai
void check_rule()
{
        float temp=0;
        uint8_t x,y;
        for(x=0;x<5;x++)Rule_out[x]=0;
        for(x=0;x<5;x++)
        {
                for(y=0;y<5;y++)
                {
                        if((Error_sudut[y]>0) && (Del_Error_sudut[x]>0))
                        {
                                if(Error_sudut[y]>Del_Error_sudut[x]) temp =
Del_Error_sudut[x];
                                else temp = Error_sudut[y];
                                if (temp > Rule_out[rule[x][y]])
Rule_out[rule[x][y]] = temp;
                        }
                }
        }
}
```

```
void defuzzifikasi() //defuzzifikasi Centre of Area
{
        float temp_pembagi,temp_kanan,temp_kiri;
        temp_pembagi        = Rule_out[LH] + Rule_out[LM] + Rule_out[Z] +
Rule_out[RM] + Rule_out[RH];
        temp_kanan          = (Rule_out[LH]*0) +
(Rule_out[LM]*MAXPWMKANAN/4) + (Rule_out[Z]*MAXPWMKANAN/2)
                            + (Rule_out[RM]*MAXPWMKANAN*3/4) +
(Rule_out[RH]*MAXPWMKANAN);
        rpwm                = temp_kanan/temp_pembagi;
        temp_kiri           = (Rule_out[LH]*MAXPWMKIRI) +
(Rule_out[LM]*MAXPWMKIRI*3/4) + (Rule_out[Z]*MAXPWMKIRI/2)
                            + (Rule_out[RM]*MAXPWMKIRI/4) +
(Rule_out[RH]*0);
        lpwm                = temp_kiri/temp_pembagi;
}
int main (void)
{
        _delay_ms(2500);
  SPI_MasterInit_int();
        DDRC  =(1<<2)|(1<<3)|(1<<4)|(1<<5)|(1<<6)|(1<<7);
//      PORTC=0;
        DDRD=(1<<4)|(1<<5)|(1<<6)|(1<<7);
        DDRB |= _BV(0);
        //PORTD=0;
        lcd_init();
        lcd_clrscr();
        lcd_gotoxy(0,0);
  lcd_puts("lepas");

        DDRA = 255;
        _delay_ms(5000);
```

```
        InitUART();    //UART init
        //TCCR0 = ( (1<<CS01) | (1<<CS00) );
        TIMSK = 0x04;
        TCCR1B = ( (1<<CS12) | (0<<CS10) );
        TCNT1 = time_sampling;
        sei();
        lcd_clrscr();
    while (1)
    {
                data_PV = PV+7;
                pangkat=256;
                lcd_gotoxy(0,0);
                tampil(sudut);

                lcd_gotoxy(8,0);
                tampil(sensor);

                lcd_gotoxy(0,1);
                //lcd_puts("Kiri : ");
        tampil(lpwm);

                lcd_gotoxy(8,1);
                //lcd_puts("Kanan: ");
        tampil(rpwm);
                //set_motors(-1530,1530);
        }
        return 0;
}
void setMLeftSpeed(volatile int speed){
    unsigned char reverse = 0;
    if (speed < 0){
       speed = -speed;    // make speed a positive quantity
```

```
        reverse = 1;        // preserve the direction
    }
    if (speed > 0xFF)    // 0xFF = 255
        speed = 0xFF;
    if (reverse){
        //PORTD &= ~_BV(3);
        dirki=0;
                pwmki = speed;
    }
    else{    // forward
        dirki=1;
        pwmki = speed;
    }
}
void setMRightSpeed(volatile int speed){
    unsigned char reverse = 0;
    if (speed < 0){
        speed = -speed;   // make speed a positive quantity
                reverse = 1;     // preserve the direction
            }
            if (speed > 0xFF)        // 0xFF = 255
                speed = 0xFF;
            if (reverse){
                dirka=0;
        pwmka = speed;
            }
            else{    // forward
                dirka=1;
        pwmka = speed;
            }
}
void set_motors(volatile int m_left, volatile int m_right){
```

```
        setMLeftSpeed(m_left);
        setMRightSpeed(m_right);
}
void scanBlackLine2()
{
//    if((sensor&0b00000001)==0){siki=1;sika=0;}
//    else if((sensor&0b10000000)==0){siki=0;sika=1;}
   switch(sensor)
   {
      case 0b11111110:      // ujung kanan
           PV = -15;
           break;
      case 0b11111100:
           PV = -13;
           break;
      case 0b11111101:
           PV = -11;
           break;
      case 0b11111001:
           PV = -8;
           break;
      case 0b11111011:
           PV = -6;
           break;
      case 0b11110011:
           PV = -4;
           break;
      case 0b11110111:
           PV = -2;
           break;
      case 0b11100111:      // tengah
           PV = 0;
```

```
        break;
case 0b11101111:
        PV = 2;
        break;
case 0b11001111:
        PV = 4;
        break;
case 0b11011111:
        PV = 6;
        break;
case 0b10011111:
        PV = 8;
        break;
case 0b10111111:
        PV = 11;
        break;
case 0b00111111:
        PV = 13;
        break;
case 0b01111111:        // ujung kiri
        PV = 15;
        break;
case 0b11111111:        // loss

      //set_motors(MAXPWM,MAXPWM);
      if(siki){
       //set_motors(-MAXPWM,MAXPWM);
                              pwmki = 6*MAXPWM;
                              pwmka = 6*MAXPWM;
                              dirki=0;
                              dirka=1;
                        }
```

```
            else if(sika){
              //set_motors(MAXPWM,-MAXPWM);
                                    pwmki = 6*MAXPWM;
                                    pwmka = 6*MAXPWM;
                                    dirki=1;
                                    dirka=0;
                                }
            else
              goto lanjut;
            goto exit;
    }
    lanjut:
          error = PV;
    if( PV<0 ){
       siki=1;sika=0;
    }
          else{
                  siki=0;sika=1;
          }
          delta_error = error-last_error;
//        delta_error = error - last_error;
    fuzzifikasi();          //fuzy
          check_rule();
          defuzzifikasi();
          last_error = error;
          //hasil_out = output;
    if (lpwm < 0) lpwm = 0;
    if (lpwm > 255) lpwm = 255;
    if (rpwm < 0) rpwm = 0;
    if (rpwm > 255) rpwm = 255;

     if(lpwm<(MAXPWMKIRI/4))
```

```
                pwmki=0;
        else
        pwmki = 6*lpwm;
        if(rpwm<(MAXPWMKANAN/4))
                pwmka=0;
        else
                pwmka = 6*rpwm;
        dirki=1;
        dirka=1;
   exit: i=0;
}
void InitUART(void){
        // USART initialization
        // Communication Parameters: 8 Data, 1 Stop, No Parity
        // USART Receiver: On
        // USART Transmitter: On
        // USART Mode: Asynchronous
        // USART Baud Rate: 9600
        UCSRA=0x00;
        UCSRB=0x98;
        UCSRC=0x86;
        UBRRH=0x00;
        //UBRRL=0x33; // 8 Mhz Atmega 8535
        //UBRRL=0x4D; // 12 Mhz Atmega 32
        UBRRL=11; // 11,0592 Mhz Mhz Atmega 8535
}
void TransmitByte( volatile unsigned char data ){
        while ( !(UCSRA & (1<<UDRE)) );  // Wait for empty transmit buffer
        UDR = data;                                                // Start transmittion
        //_delay_ms(10);
}
void hex2ascii(volatile uint8_t data){
```

```
        ratusan = data/100 +48;
        puluhan = (data%100)/10 +48;
        satuan  = (data%10) +48;

        TransmitByte(ratusan);
        TransmitByte(puluhan);
        TransmitByte(satuan);
}

void tampil(int dat)
{
  int data;
  data = dat / 10000;
  data+=0x30;
  lcd_putch(data);

  dat%=10000;
  data = dat / 1000;
  data+=0x30;
  lcd_putch(data);

  dat%=1000;
  data = dat / 100;
  data+=0x30;
  lcd_putch(data);

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

```
    dat%=10;
    data = dat + 0x30;
    lcd_putch(data);
}

void baca_sudut(){
//30 28 22 19,15 11  8   2   0
        if(sensor==0b01111111)
                sudut = 30;
        else if(sensor==0b00111111)
                sudut = 28;
        else if(sensor==0b10011111)
                sudut = 22;
        else if(sensor==0b11001111)
                sudut = 19;
        else if(sensor==0b11100111)
                sudut = 15;
        else if(sensor==0b11110011)
                sudut = 11;
        else if(sensor==0b11111001)
                sudut = 8;
        else if(sensor==0b11111100)
                sudut = 2;
        else if(sensor==0b11111110)
                sudut = 0;
}
```

Mikrokontroler *Slave* (Sensor)

```
#define ADC_VREF_TYPE 0x60
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "SPI.h"
volatile uint8_t data_spi_rx;
volatile uint8_t data_spi_tx;
uint8_t i=0;
volatile uint8_t j=0;
volatile int
data_sensor[10],a[8],tengah[8]={94,76,101,152,143,95,100,153},sensor,data;
unsigned char read_sensor(unsigned char num);
unsigned char read_adc(unsigned char adc_input);
int main (void)
{
        _delay_ms(500);
        SPI_SlaveInit();
        ADMUX=ADC_VREF_TYPE & 0xff;
        ADCSRA=0x87;
        DDRC |= _BV(0); DDRC |= _BV(1); DDRC |= _BV(2);
        while (1)
        {
                for(i=0;i<8;i++){
                        data_sensor[i] = read_sensor(i);
                        if (data_sensor[i]>tengah[i]) a[i]=1;
                else a[i]=0;
                }
```

```
data=((a[7]*128)+(a[6]*64)+(a[5]*32)+(a[4]*16)+(a[3]*8)+(a[2]*4)+(a[1]
    *2)+(a[0]*1) );
     if(data==0b10000000)
          data = (1*15)/1;
     else if(data==0b11000000)
          data = (0.248469*7.5
               +0.741299*15)/(0.248469+0.741299);
     else if(data==0b01000000)
          data = (0.5*0.75 + 0.5*15)/(0.5+0.5);
     else if(data==0b01100000)
          data = (0.182134571*0 + 0.229907193*15 +
               0.752900232*7.5)/(0.182134571+0.229907193+0.7
               52900232);
     else if(data==0b00100000)
          data = (0*15 + 1*7.5 + 0.3306285*0)/(0+1+0.3306285);
     else if(data==0b00110000)
          data = (0.515081206*0 +
               0.752900232*7.5)/(0.515081206+0.752900232);
     else if(data==0b00010000)
          data = (0.5*7.5 + 0.7*0)/(0.5 + 0.7);
     else if(data==0b00011000)
          data = (1*0)/1;
     else if(data==0b00001000)
          data = (0.5*(-7.5) + 0.7*0)/(0.5 + 0.7);
     else if(data==0b00001100)
          data = (0.752900232*(-7.5) +
               0.515081206*0)/(0.752900232+0.515081206);
     else if(data==0b00000100)
          data = (0*(-15) + 0.3306285*0+1*(-7.5))/(0+1+0.3306285);
     else if(data==0b00000110)
```

```
                data = (0.752900232*(-7.5) + 0.229907193*(-15) +
                        0.182134571*0)/(0.752900232+0.229907193+0.182
                        134571);
            else if(data==0b00000010)
                data = (0*0 + 0.5*(-7.5) + 0.5*(-15))/(0 + 0.5 + 0.5);
            else if(data==0b00000011)
                data = (0.741299304*(-15) + 0.248468677*(-
                        7.5))/(0.741299304+0.248468677);
            else if(data==0b00000001)
                data = (1*-15)/1;
            else{
                data=0;
            }
            sensor = data+15;
        }
}
ISR(SPI_STC_vect){
        SPDR=sensor;
}
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
_delay_us(50);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
void selectora(int a,int b,int c)
```

```
{
        if(a)
                PORTC |= _BV(0);
        else
                PORTC &= ~_BV(0);
        if(b)
                PORTC |= _BV(1);
        else
                PORTC &= ~_BV(1);
        if(c)
                PORTC |= _BV(2);
        else
                PORTC &= ~_BV(2);
}
unsigned char read_sensor(unsigned char num){
        unsigned char temp=0;
        switch(num)
  {
                case 6: selectora(1,1,1);       break;
     case 5: selectora(1,1,0);       break;
     case 7: selectora(1,0,1);       break;
     case 4: selectora(1,0,0);       break;
     case 0: selectora(0,1,1);       break;
     case 3: selectora(0,1,0);       break;
     case 2: selectora(0,0,1);       break;
     case 1: selectora(0,0,0);       break;
        }
        temp = read_adc(3);
        return temp;
}
```

Mikrokontroler *Slave* (Motor)

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "SPI.h"
volatile int data_spi_rx;
volatile int data_spi_tx;
//uint8_t i=0;
volatile uint8_t i, j=0,direction,k=0,terima=0,first=0;
volatile int rpm_actual[10],rpm_set_point[10],header[2];
volatile int pulsa=0,pulsamsb=0,rpm,speed_temp,speed,PWM;
volatile float putaran1;
volatile unsigned char data_sensor[10];
volatile float
PV,P,I,D,error,last_error,MV,Kp=2.94,Ki=9.8,Kd=0.2205;//3,10,0.225
void setMLeftSpeed(int speed);
int main (void)
{
        _delay_ms(5000);
        SPI_SlaveInit();
        //_delay_ms(5000);
        TCCR1A=0xA1;
        TCCR1B=0x0C;//C/B
        _delay_ms(5000);
        DDRB |= _BV(0);
        DDRB |= _BV(1);
        //inisialisi timer2
        //clock = 11179 Hz
        //untuk sampling pulsa rotary
```

```
        ASSR=0x00;
        TCCR2=0x07;
        TCNT2=39;
        OCR2=0x00;
        // Timer/Counter 0 initialization
        // Clock source: T0 pin Rising Edge
        TCCR0=0x07;
        TCNT0=0x00;
        // Timer2 dan Timer0 mode interupt overflow
        TIMSK=0x41;
        //TIMSK=0x01;//timer0 saja
        while (1)
        {
                //rpm_set_point[2]=1530/255;
                //rpm_set_point[1]=1530%255;
                rpm_actual[0] = rpm%255;//LSB
                rpm_actual[1] = rpm/255;//MSB
                SP=600;
                setMLeftSpeed(speed);
        }
}
ISR(SPI_STC_vect){
        data_spi_rx=SPDR;
        if(data_spi_rx==255)
                k++;
        else
                k=0;
        if(k>=2){
                j=0;
                terima=1;
        }
        else if(terima){
```

```
                rpm_set_point[j] = data_spi_rx;
                j++;
                if(j>=3)
                        terima=0;
        }
        SPDR=rpm_actual[j];//data_sensor[j];
}
ISR(TIMER2_OVF_vect){
        TCNT2 = 39;//20ms
        pulsa= pulsa + (pulsamsb*256+TCNT0);
        TCNT0=0;
        static int tim2; //clock = 11719
        tim2++;
        if(tim2>=10)//sampling setiap 100ms (butuh 5X interupsi overflow
        {
                //rpm = (float)pulsa*6/5; //jumlah putaran dalam sekali sampling
                  rpm = (float)pulsa*3/5;
                //rpm = putaran1*6; //dikali 60(1menit), dikali 10 (sampling
                        100ms) menjadi rpm
                PV = rpm;
                error = (float)(SP - PV)/10;
           P = Kp * error;
           I = (Ki * (error + last_error)) * 0.2;
           D = (Kd * (error - last_error)) / 0.2;
           last_error = error;
           MV = P ;//+ D;
           speed_temp = speed + MV;
           if(speed_temp<20)  //speed_temp=speed sementara
                speed=20;
                else if(speed_temp>255)
                        speed=255;
           else
```

```
            speed=speed_temp;
            pulsa=0;
            tim2=0;
    }
        pulsamsb=0;
}
ISR(TIMER0_OVF_vect){
        pulsamsb++;
}
void setMLeftSpeed(int speed){
   unsigned char reverse = 0;
   if (speed < 0){
      speed = -speed;    // make speed a positive quantity
      reverse = 1;          // preserve the direction
   }
   if(speed>10)
   speed-=10;
   else
   speed=0;
   if (speed > 0xFF)    // 0xFF = 255
      speed = 0xFF;
   if (reverse){
      PORTB &= ~_BV(0);
      OCR1A = speed;
   }
   else{    // forward
      PORTB |= _BV(0);
      OCR1A = 255 - speed;
   }
}
```