LAMPIRAN

- **Program CodeVision**

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.3 Standard
Automatic Program Generator
© Copyright 1998-2011 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 09/07/2019
Author  : Windows User
Company : adfsf
Comments:
Chip type           : ATmega16
Program type        : Application
AVR Core Clock frequency: 16,000000 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 256
*****************************************************/

#include <mega16.h>

// Alphanumeric LCD functions
#include <alcd.h>

#ifndef RXB8
#define RXB8 1
#endif

#ifndef TXB8
#define TXB8 0
#endif

#ifndef UPE
#define UPE 2
#endif

#ifndef DOR
```

```
#define DOR 3
#endif

#ifndef FE
#define FE 4
#endif

#ifndef UDRE
#define UDRE 5
#endif

#ifndef RXC
#define RXC 7
#endif

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<DOR)
#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_BUFFER_SIZE == 256
   // special case for receiver buffer size=256
   if (++rx_counter == 0) rx_buffer_overflow=1;
#else
   if (rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
   if (++rx_counter == RX_BUFFER_SIZE)
      {
      rx_counter=0;
      rx_buffer_overflow=1;
      }
#endif
   }
}

#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_BUFFER_SIZE != 256
if (rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#endif
#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_BUFFER_SIZE != 256
   if (tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
#endif
   }
}

#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_BUFFER_SIZE != 256
   if (tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
#endif
   ++tx_counter;
   }
else
   UDR=c;
#asm("sei")
}
#pragma used-
#endif

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

```
#include <stdlib.h>
#include <delay.h>
// SPI functions
#include <spi.h>
unsigned char data[20];
int s,sp;
char lcd_buffer[33];
unsigned long value,value1;
float suhu,suhu1,total;
float kesalahanSuhu,selisihSuhu,pwm,kesalahanSuhu1;
float negatifBesar,negatifSedang,nol,positifSedang,positifBesar;
float negatif,nol1,positif;
float tidakBergerak,lambat,sedang,cepat,sangatCepat;
float
rule1,rule2,rule3,rule4a,rule4b,rule5a,rule5b,rule6a,rule6b,rule7a,rule7b,rule8,rule
9,rule10,rule11,rule12,rule13,rule14,rule15;
float x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15;
float def;
void kesalahanSuhu_NB(){
   if(kesalahanSuhu<=-8){
      negatifBesar=1;}
   else if (kesalahanSuhu>=-8&&kesalahanSuhu<=-4){
      negatifBesar=((-4 - kesalahanSuhu)/4);
   }
   else if (kesalahanSuhu>=-4){
      negatifBesar=0;}
   }

void kesalahanSuhu_NS(){
   if(kesalahanSuhu<=-8){
      negatifSedang=0;}
   else if (kesalahanSuhu>=-8&&kesalahanSuhu<=-4){
      negatifSedang=((kesalahanSuhu+8)/4);
   }
   else if (kesalahanSuhu>=-4&&kesalahanSuhu<=0){
      negatifSedang=((0-kesalahanSuhu)/4);}

   }

void kesalahanSuhu_Nol(){
   if(kesalahanSuhu<=-4){
      nol=0;}
```

```
    else if (kesalahanSuhu>=-4&&kesalahanSuhu<=0){
      nol=((kesalahanSuhu+4)/4);
    }
    else if (kesalahanSuhu>=0&&kesalahanSuhu<=4){
      nol=((4-kesalahanSuhu)/4);}
    }

void kesalahanSuhu_PS(){
    if(kesalahanSuhu<=0){
      positifSedang=0;}
    else if (kesalahanSuhu>=0&&kesalahanSuhu<=4){
      positifSedang=((kesalahanSuhu-0)/4);
    }
    else if (kesalahanSuhu>=4&&kesalahanSuhu<=8){
      positifSedang=((8-kesalahanSuhu)/4);}
    }


void kesalahanSuhu_PB(){
    if(kesalahanSuhu<=4){
      positifBesar=0;}
    else if (kesalahanSuhu>=4&&kesalahanSuhu<=8){
      positifBesar=((kesalahanSuhu-4)/4);
    }
    else if (kesalahanSuhu>=8){
      positifBesar=1;}
}


void selisihSuhu_N(){
    if(selisihSuhu<=-0.6){
      negatif=1;}
    else if (selisihSuhu>=-0.6&&selisihSuhu<=0){
      negatif=(( 0 -selisihSuhu)/0.6);
    }
    else if (selisihSuhu>=0){
      negatif=0;}
}

void selisihSuhu_Nol(){
    if(selisihSuhu<=-0.6){
      nol1=0;}
```

```
    else if (selisihSuhu>=-0.6&&selisihSuhu<=0){
       nol1=((selisihSuhu +0.6)/0.6);
    }
    else if(selisihSuhu>=0&&selisihSuhu<=0.6){
       nol1=((0.6- selisihSuhu)/0.6);}
}

void selisihSuhu_P(){
    if(selisihSuhu<=0){
       positif=0;}
    else if (selisihSuhu>=0&&selisihSuhu<=0.6){
       positif=( (selisihSuhu -0)/0.6);
    }
    else if (selisihSuhu>=0.6){
       positif=1;}
    }
 void fuzzyfikasi(){
    kesalahanSuhu_NB();
    kesalahanSuhu_NS();
    kesalahanSuhu_Nol();
    kesalahanSuhu_PS();
    kesalahanSuhu_PB();
    selisihSuhu_N();
    selisihSuhu_Nol();
    selisihSuhu_P();
    }

float Min (float a, float b){
    return (a<b?a:b);
   /// else if (b > a){return b;}
  //  else return a;
}

void rulebase(){
    fuzzyfikasi();
//if(kesalahan_suhu is negatif besar) and (selisih suhu is negatif) then (pwm is
sangat cepat)
    x1=Min(negatifBesar,negatif);
    rule1=204+(25.5*x1);
//if(kesalahan suhu is negatif besar) and (sselisih suhu nol) then (pwm is sangat
cepat)
    x2=Min(negatifBesar,nol1);
```

```
   rule2=204+(25.5*x2);
//if(kesalahan suhu is negatif besar) and (sselisih suhu positif) then (pwm is sangat
cepat)
   x3=Min(negatifBesar,positif);
   rule3=204+(25.5*x3);
//if(kesalahan suhu is negatif sedang) and (sselisih suhu negatif) then (pwm is
cepat)
    x4=Min(negatifSedang,negatif);
   rule4a=127.5+(51*x4);
   rule4b=229.5-(51*x4);
//if(kesalahan suhu is negatif sedang) and (sselisih suhu nol) then (pwm is sedang)
  x5=Min(negatifSedang,nol1);
  rule5a=76.5+(51*x5);
  rule5b=178.5-(51*x5);
//if(kesalahan suhu is negatif sedang) and (sselisih suhu positif) then (pwm is
lambat)
  x6=Min(negatifSedang,positif);
  rule6a=25.5+(51*x6);
  rule6b=127.5-(51*x6);
  //if(kesalahan suhu is   nol) and (sselisih suhu negatif) then (pwm is lambat)
  x7=Min(nol,negatif);
  rule7a=25.5+(51*x7);
  rule7b=127.5-(51*x7);
    //if(kesalahan suhu is   nol) and (sselisih suhu nol) then (pwm is tidak
bergerak)
  x8=Min(nol,nol1);
  rule8 =51-(25.5*x8);
   //if(kesalahan suhu is   nol) and (sselisih suhu positif) then (pwm is tidak
bergerak)
  x9=Min(nol,positif);
  rule9 =51-(25.5*x9);
    //if(kesalahan suhu is   positif sedang) and (sselisih suhu negatif) then (pwm is
tidak bergerak)
  x10=Min(positifSedang,negatif);
  rule10=51-(25.5*x10);
      //if(kesalahan suhu is   positif sedang) and (sselisih suhu nol) then (pwm is
tidak bergerak)
  x11=Min(positifSedang,nol1);
  rule11=51-(25.5*x11);
      //if(kesalahan suhu is   positif sedang) and (sselisih suhu positif) then (pwm
is tidak bergerak)
  x12=Min(positifSedang,positif);
```

```
  rule12=51-(25.5*x12);
      //if(kesalahan suhu is   positif besar) and (sselisih suhu negatif) then (pwm is
tidak bergerak)
  x13=Min(positifBesar,negatif);
  rule13=51-(25.5*x13);
      //if(kesalahan suhu is   positif besar) and (sselisih suhu nol) then (pwm is
tidak bergerak)
  x14=Min(positifBesar,nol1);
  rule14=51-(25.5*x14);

      //if(kesalahan suhu is   positif besar) and (sselisih suhu positif) then (pwm is
tidak bergerak)
  x15=Min(positifBesar,positif);
  rule15=51-(25.5*x15);

  def =
((x1*rule1)+(x2*rule2)+(x3*rule3)+(x4*rule4a)+(x4*rule4b)+(x5*rule5a)+(x5*ru
le5b)+(x6*rule6a)+(x6*rule6b)+(x7*rule7a)+(x7*rule7b)+(x8*rule8)+(x9*rule9)
+(x10*rule10)+(x11*rule11)+(x12*rule12)+(x13*rule13)+(x14*rule14)+(x15*rul
e15))/(x1+x2+x3+x4+x4+x5+x5+x6+x6+x7+x7+x8+x9+x10+x11+x12+x13+x14
+x15);
 }

 void bacaSuhu(){
  //cs0
  PORTB.4=0;
  value=(unsigned long)spi(0)<<8;
  value |=spi(0);
  PORTB.4=1;
       //cs1
  PORTB.3=0;
  value1=(unsigned long)spi(0)<<8;
  value1 |=spi(0);
  PORTB.3=1;
  suhu=  (value/40.0)+((value%40)/100.0) ;
  suhu1=  (value1/40.0)+((value1%40)/100.0);
   total= (suhu+suhu1)/2.0;
 // sprintf(lcd_buffer,"SUHU=%4u.%u %cC
",suhu,((value%40)+(value1%40)/2),0xdf);
  // lcd_clear();
  // lcd_gotoxy(0,0);
 //  lcd_puts(lcd_buffer);
```

```
  }
  void runfuzzy(){
    bacaSuhu();
    kesalahanSuhu1= kesalahanSuhu;
    kesalahanSuhu=total-sp;
    selisihSuhu=kesalahanSuhu-kesalahanSuhu1;
    rulebase();
    }
// 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=Out Func6=In Func5=Out Func4=Out Func3=Out Func2=In Func1=In
Func0=In
// State7=0 State6=T State5=0 State4=0 State3=0 State2=T State1=T State0=T
PORTB=0x00;
DDRB=0xB8;

// 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=Out Func4=Out Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=0 State4=0 State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x30;
```

```
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 250,000 kHz
// Mode: Fast PWM top=0x00FF
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0xA1;
TCCR1B=0x0B;
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: Timer2 Stopped
// Mode: Normal top=0xFF
// 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: 9600
UCSRA=0x00;
UCSRB=0xD8;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x67;

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

// ADC initialization
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI Type: Master
// SPI Clock Rate: 4000,000 kHz
// SPI Clock Phase: Cycle Start
// SPI Clock Polarity: Low
// SPI Data Order: MSB First
SPCR=0x50;
SPSR=0x00;
```

```
// TWI initialization
// TWI disabled
TWCR=0x00;

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTC Bit 7
// RD - PORTC Bit 6
// EN - PORTC Bit 5
// D4 - PORTC Bit 4
// D5 - PORTC Bit 3
// D6 - PORTC Bit 2
// D7 - PORTC Bit 1
// Characters/line: 16
lcd_init(20);

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

while (1)
      {
    if(rx_counter >=3){
      if(getchar()=='s'){

      int s;
      scanf("%d",&s);
      sp=s;}}
      sp=sp;
      lcd_clear();
     lcd_gotoxy(0,0);
     sprintf(data,"SP:%d",sp);
     lcd_puts(data);
     runfuzzy();
     OCR1A = def;
       ftoa(total,2,lcd_buffer);
        lcd_gotoxy(6,0);
        lcd_puts("Suhu:");
        lcd_puts(lcd_buffer);
        ftoa(kesalahanSuhu,2,lcd_buffer);
        lcd_gotoxy(0,1);
        lcd_puts("K_Suhu:");
```

```
        lcd_puts(lcd_buffer);
        ftoa(selisihSuhu,2,lcd_buffer);
        lcd_gotoxy(0,2);
        lcd_puts("S_Suhu:");
        lcd_puts(lcd_buffer);
         ftoa(def,2,lcd_buffer);
        lcd_gotoxy(0,3);
         lcd_puts("Defuz:");
        lcd_puts(lcd_buffer);
        if(def<98 ){OCR1A=0;}
      printf("%d",((value+value1)/2));
       printf(",");
      printf("%d\n",sp);
      delay_ms(500);
      }
}
```

- **Program GUI MATLAB**

```
function varargout = coba(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @coba_OpeningFcn,
...
                   'gui_OutputFcn',  @coba_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before coba is made visible.
function coba_OpeningFcn(hObject, eventdata, handles,
varargin)
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
function varargout = coba_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata,
handles)
global s i;
xlim([0 handles.time]);
ylim([0 100]);
l1 = line( nan,nan,'Color','r','LineWidth',2);
l2 = line( nan,nan,'Color','b','LineWidth',2);

xlabel('waktu (s)')
```

```
ylabel('Suhu  (oC)')
%title('Captura de voltaje en tiempo real con Arduino')
grid on
hold on

%% Bucle
% inicializar
v1 = zeros(1,handles.time*rate);
v2 = zeros(1,handles.time*rate);
i = 1;
t = 0;

% ejecutar bucle cronometrado
tic
while t<handles.time

    % leer del puerto serie
    a = fscanf(s,'%d,%d')';
    v1(i)=a(1)/40;
     v2(i)=a(2) ;
    % dibujar en la figura
     x = linspace(0,i/rate,i);
    set(l1,'YData',v1(1:i),'XData',x );
    set(l2,'YData',v2(1:i),'XData',x );
    drawnow
    % seguir
    i = i+1;
    legend('suhu','SetPoint');
    pause(.01);
end

fclose(s);
delete(s);
clear s;
function com_port_Callback(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

  function edit2_Callback(hObject, eventdata, handles)
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

 function edit4_Callback(hObject, eventdata, handles)
```

```
handles.data1=get(hObject,'string');
handles.time=str2double(handles.data1);
guidata(hObject,handles);

% --- Executes during object creation, after setting
all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in connect.
function connect_Callback(hObject, eventdata, handles)
global s;
guidata(hObject, handles);

port=get(handles.com_port,'string');

port=strcat('COM',port);
 s = serial(port );
fopen(s);


msgbox('Arduino successfully connected','Connected');
% --- Executes on button press in disconect.
function disconect_Callback(hObject, eventdata,
handles)
global s;
delete(s);
msgbox('Arduino successfully
Disonnected','Disconnected');

 function sp_Callback(hObject, eventdata, handles)
function sp_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function sandi_Callback(hObject, eventdata, handles)
function sandi_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sandi (see GCBO)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata,
handles)
global s;
data=get(handles.sandi,'string');
sp=get(handles.sp,'string');
fprintf(s,sp);
fprintf(s,data);
```

- **Dokumentasi**