LAMPIRAN I

PROGRAM LENGKAP
/******************************************************************************

This program was produced by the
CodeWizardAVR V1.25.8 Standard
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 09/07/2013
Author : F4CG
Company : F4CG
Comments:

Chip type : ATmega8
Program type : Application
Clock frequency : 16,000000 MHz
Memory model : Small
External SRAM size : 0
Data Stack size : 256
******************************************************************************

#include <mega8.h>
// Alphanumeric LCD Module functions

#asm

.equ __lcd_port=0x12 ;PORTD

#endasm

#include <lcd.h>

#include <stdio.h>

int kipas1, kipas2, kipas3, pwm, pwm1, pwm2, pwm3, sensor1, sensor2, sensor3;
int hit1, hit2, hit3, st1, st2, st3, ht1, ht2, ht3;

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Place your code here
pwm++;

if (pwm==2000)
{
pwm=0;
}

if(pwm<kipas1) {PORTC.3=1;}
else        {PORTC.3=0;}

if(pwm<kipas2) {PORTC.4=1;}
else        {PORTC.4=0;}

if(pwm<kipas3) {PORTC.5=1;}
else        {PORTC.5=0;}
}

#include <delay.h>
#define FIRST_ADC_INPUT 0
#define LAST_ADC_INPUT 2

unsigned int adc_data[LAST_ADC_INPUT-FIRST_ADC_INPUT+1];
#define ADC_VREF_TYPE 0x40

// ADC interrupt service routine
// with auto input scanning
interrupt [ADC_INT] void adc_isr(void)
{
    register static unsigned char input_index=0;
    // Read the AD conversion result
    adc_data[input_index]=ADCW;
    // Select next ADC input
    if (++input_index > (LAST_ADC_INPUT-FIRST_ADC_INPUT))
        input_index=0;
    ADMUX=(FIRST_ADC_INPUT | (ADC_VREF_TYPE & 0xff))+input_index;
    // Delay needed for the stabilization of the ADC input voltage
    delay_us(10);
    // Start the AD conversion
    ADCSRA|=0x40;
}

// Declare your global variables here
char data[16];
int data_adc1, data_adc2, data_adc3;

void c3tutup()
{
  PORTB.5=1;
  delay_ms(300);
  PORTB.5=0;
  delay_ms(300);
}

void c3buka()
{
  PORTB.4=1;
  delay_ms(300);
  PORTB.4=0;
  delay_ms(300);
  PORTD.3=1;
}

void c2tutup()
{
  PORTB.3=1;
  delay_ms(300);
  PORTB.3=0;
  delay_ms(300);
void c2buka()
{
  PORTB.2=1;
delay_ms(300);
  PORTB.2=0;
delay_ms(300);
  PORTD.3=1;
}

void c1tutup()
{
  PORTB.1=1;
delay_ms(300);
  PORTB.1=0;
delay_ms(300);
}

void c1buka()
{
  PORTB.0=1;
delay_ms(300);
  PORTB.0=0;
delay_ms(300);
PORTD.3=1;
}

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

    // Input/Output Ports initialization
    // Port B 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
    PORTB=0x00;
    DDRB=0xFF;

    // Port C initialization
    // Func6=In Func5=Out Func4=Out Func3=Out Func2=In Func1=In Func0=In
    // State6=T State5=0 State4=0 State3=0 State2=T State1=T State0=T
    PORTC=0x00;
    DDRC=0x38;

    // Port D initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=Out Func2=In Func1=In Func0=In
    // State7=T State6=T State5=T State4=T State3=0 State2=T State1=T State0=T
    PORTD=0x00;
DDRD=0x08;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 2000,000 kHz
TCCR0=0x02;
TCNT0=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
MCUCR=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;

// ADC initialization
// ADC Clock frequency: 125,000 kHz
// ADC Voltage Reference: AVCC pin
ADMUX=FIRST_ADC_INPUT | (ADC_VREF_TYPE & 0xff);
ADCSRA=0xCF;

// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")
lcd_gotoxy(0,0);
lcd_putsf("My Project");
lcd_gotoxy(0,1);
lcd_putsf("Sensor Asap");
delay_ms(1000);
lcd_clear();
hit1=0;
hit2=0;
hit3=0;
st1=0;
st2=0;
st3=0;
ht1=0;
ht2=0;
ht3=0;
c1buka();
while (1)
{
    // Place your code here
    data_adc1=adc_data[2];
    data_adc2=adc_data[0];
    data_adc3=adc_data[1];

    sensor1=data_adc1-200;
    sensor3=data_adc2-200;
    sensor2=data_adc3-200;

    if(sensor1<=0){sensor1=0;}
    if(sensor2<=0){sensor2=0;}
    if(sensor3<=0){sensor3=0;}

    sprintf(data, "%d", sensor1);
    lcd_gotoxy(0,0);
    lcd_puts(data);
}
sprintf(data, "%d", sensor2);
lcd_gotoxy(5, 0);
lcd_puts(data);
sprintf(data, "%d", sensor3);
lcd_gotoxy(10, 0);
lcd_puts(data);
if (ht1 >= 10 && st1 == 0) {c1buka(); ht1 = 0; st1 = 1;}
pwm1 = sensor1 * 3;
if (pwm1 == 0) {kipas1 = 0;}
else if (pwm1 <= 800) {kipas1 = 800;}
else {kipas1 = pwm1;}
sprintf(data, "%d", kipas1);
lcd_gotoxy(0, 1);
lcd_puts(data);

if (ht2 >= 10 && st2 == 0) {c2buka(); ht2 = 0; st2 = 1;}
pwm2 = sensor2 * 3;
if (pwm2 == 0) {kipas2 = 0;}
else if (pwm2 <= 800) {kipas2 = 800;}
else {kipas2 = pwm2;}
sprintf(data, "%d", kipas2);
lcd_gotoxy(5, 1);
lcd_puts(data);

if (ht3 >= 10 && st3 == 0) {c3buka(); ht3 = 0; st3 = 1;}
pwm3 = sensor3 * 3;
if (pwm3 == 0) {kipas3 = 0;}
else {kipas3 = pwm3;}
sprintf(data, "%d", kipas3);
lcd_gotoxy(10, 1);
lcd_puts(data);
else {if(pwm3<=800){kipas3=800;} else {kipas3=pwm3;}}

sprintf(data,"%d",kipas3);
lcd_gotoxy(10,1);
lcd_puts(data);

if(sensor1==0){hit1=hit1+1;} else{hit1=0;}
if(sensor2==0){hit2=hit2+1;} else{hit2=0;}
if(sensor3==0){hit3=hit3+1;} else{hit3=0;}

if(hit1>=10 && st1==1) {c1tutup(); hit1=0; st1=0;}
if(hit2>=10 && st2==1) {c2tutup(); hit2=0; st2=0;}
if(hit3>=10 && st3==1) {c3tutup(); hit3=0; st3=0;}

if(sensor1>=1){ht1=ht1+1;} else{ht1=0;}
if(sensor2>=1){ht2=ht2+1;} else{ht2=0;}
if(sensor3>=1){ht3=ht3+1;} else{ht3=0;}

if(sensor1==0 && sensor2==0 && sensor3==0) {PORTD.3=0;}

delay_ms(300);
lcd_clear();
};
}
LAMPIRAN II

DATASHEET
MQ-2 Semiconductor Sensor for Combustible Gas

Sensitive material of MQ-2 gas sensor is SnO2, which with lower conductivity in clean air. When the target combustible gas exist, The sensor’s conductivity is more higher along with the gas concentration rising. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.

MQ-2 gas sensor has high sensitivity to LPG, Propane and Hydrogen, also could be used to Methane and other combustible steam, it is with low cost and suitable for different application.

**Character**
* Good sensitivity to Combustible gas in wide range
* High sensitivity to LPG, Propane and Hydrogen
* Long life and low cost
* Simple drive circuit

**Application**
* Domestic gas leakage detector
* Industrial Combustible gas detector
* Portable gas detector

**Technical Data**

<table>
<thead>
<tr>
<th>Character</th>
<th>Sensitivity Rs(in air)/Rs(1000ppm isobutane)≥5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensing Resistance Rs</td>
<td>2KΩ-20KΩ (in 2000ppm C3H8)</td>
</tr>
<tr>
<td>Sensitivity S</td>
<td>Rs(in air)/Rs(1000ppm isobutane)≥5</td>
</tr>
<tr>
<td>Slope α</td>
<td>≤0.6(Rs2000ppm/Rs2000ppm C3H8)</td>
</tr>
<tr>
<td>Condition</td>
<td>Tem. Humidity 20℃±2℃; 65%±5%RH</td>
</tr>
<tr>
<td>Standard test circuit</td>
<td>Vc:5.0V±0.1V; Vh: 5.0V±0.1V</td>
</tr>
<tr>
<td>Preheat time</td>
<td>Over 48 hours</td>
</tr>
</tbody>
</table>

**Configuration**

*Good sensitivity to Combustible gas in wide range
* High sensitivity to LPG, Propane and Hydrogen
* Long life and low cost
* Simple drive circuit

**Application**
* Domestic gas leakage detector
* Industrial Combustible gas detector
* Portable gas detector

**Basic test loop**

The above is basic test circuit of the sensor. The sensor need to be put 2 voltage, heater voltage(VH) and test voltage (VC). VH used to supply certified working temperature to the sensor, while VC used to detect voltage (VRL) on load resistance (RL) whom is in series with sensor. The sensor has light polarity, Vc need DC power. VC and VH could use same power circuit with precondition to assure performance of sensor. In order to make the sensor with better performance, suitable RL value is needed:

Power of Sensitivity body(Ps):

$$Ps = Vc^2 \times Rs/(Rs+RL)^2$$
Resistance of sensor (Rs): \[ Rs = \frac{V_c}{V_{RL}} - 1 \times RL \]

**Sensitivity Characteristics**

Fig. 1 shows the typical sensitivity characteristics of the MQ-2, ordinate means resistance ratio of the sensor \( \frac{Rs}{Ro} \), abscissa is concentration of gases. Rs means resistance in different gases, Ro means resistance of sensor in 1000ppm Hydrogen. All test are under standard test conditions.

**Influence of Temperature/Humidity**

Fig. 2 shows the typical temperature and humidity characteristics. Ordinate means resistance ratio of the sensor \( \frac{Rs}{Ro} \), Rs means resistance of sensor in 1000ppm Butane under different tem. and humidity. Ro means resistance of the sensor in environment of 1000ppm Methane, 20°C/65%RH

**Structure and configuration**

Structure and configuration of MQ-2 gas sensor is shown as Fig. 3, sensor composed by micro AL2O3 ceramic tube, Tin Dioxide (SnO2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-2 have 6 pin, 4 of them are used to fetch signals, and other 2 are used for providing heating current.
Notification

1 Following conditions must be prohibited

1.1 Exposed to organic silicon steam

Organic silicon steam cause sensors invalid, sensors must be avoid exposing to silicon bond, fixature, silicon latex, putty or plastic contain silicon environment

1.2 High Corrosive gas

If the sensors exposed to high concentration corrosive gas (such as H₂Sz, SOₓ, Cl₂, HCl etc), it will not only result in corrosion of sensors structure, also it cause sincere sensitivity attenuation.

1.3 Alkali, Alkali metals salt, halogen pollution

The sensors performance will be changed badly if sensors be sprayed polluted by alkali metals salt especially brine, or be exposed to halogen such as fluorin.

1.4 Touch water

Sensitivity of the sensors will be reduced when spattered or dipped in water.

1.5 Freezing

Do avoid icing on sensor’s surface, otherwise sensor would lose sensitivity.

1.6 Applied voltage higher

Applied voltage on sensor should not be higher than stipulated value, otherwise it cause down-line or heater damaged, and bring on sensors’ sensitivity characteristic changed badly.

1.7 Voltage on wrong pins

For 6 pins sensor, if apply voltage on 1、3 pins or 4、6 pins, it will make lead broken, and without signal when apply on 2、4 pins

2 Following conditions must be avoided

2.1 Water Condensation

Indoor conditions, slight water condensation will effect sensors performance lightly. However, if water condensation on sensors surface and keep a certain period, sensor’ sensitivity will be decreased.

2.2 Used in high gas concentration

No matter the sensor is electrified or not, if long time placed in high gas concentration, if will affect sensors characteristic.

2.3 Long time storage

The sensors resistance produce reversible drift if it’s stored for long time without electrify, this drift is related with storage conditions. Sensors should be stored in airproof without silicon gel bag with clean air. For the sensors with long time storage but no electrify, they need long aging time for stability before using.

2.4 Long time exposed to adverse environment

No matter the sensors electrified or not, if exposed to adverse environment for long time, such as high humidity, high temperature, or high pollution etc, it will effect the sensors performance badly.

2.5 Vibration

Continual vibration will result in sensors down-lead response then rupture. In transportation or assembling line, pneumatic screwdriver/ultrasonic welding machine can lead this vibration.

2.6 Concussion

If sensors meet strong concussion, it may lead its lead wire disconnected.

2.7 Usage

For sensor, handmade welding is optimal way. If use wave crest welding should meet the following conditions:

2.7.1  Soldering flux: Rosin soldering flux contains least chlorine

2.7.2  Speed: 1-2 Meter/ Minute

2.7.3  Warm-up temperature: 100±20℃

2.7.4  Welding temperature: 250±10℃

2.7.5  1 time pass wave crest welding machine

If disobey the above using terms, sensors sensitivity will be reduced.