```
#include <xc.h>

#include <stdint.h>

#include <math.h>

#include "supporing_cfile/lcd.h"

#include "supporing_cfile/adc.h"

#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)

#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = ON       // Power-up Timer Enable bit (PWRT enabled)

#pragma config BOREN = ON       // Brown-out Reset Enable bit (BOR enabled)

#pragma config LVP = OFF        // Low-Voltage (Single-Supply) In-Circuit Serial
Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for
programming)

#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data
EEPROM code protection off)

#pragma config WRT = OFF        // Flash Program Memory Write Enable bits (Write
protection off; all program memory may be written to by EECON control)

#pragma config CP = OFF         // Flash Program Memory Code Protection bit
(Code protection off)

/*

 Program Flow related definition

 */

#define gas_detect   PORTDbits.RD5

#define gas_Detect_Pin_Direction  TRISDbits.TRISD5

#define FIRST_LINE 0x80

#define SECOND_LINE 0xC0

#define RL_VALUE (10)     //define the load resistance on the board, in kilo
ohms

#define RO_VALUE_CLEAN_AIR (9.83)  //(Sensor resistance in clean air)/RO,

                                    //which is derived from the chart in
datasheet

float MQ6_curve[3]  = {2.3,0.30,-0.41};   //two points from LPG curve are taken
```

```
point1:(200,1.6) point2(10000,0.26)

                                        //take log of each point (lg200, lg
1.6)=(2.3,0.20)  (lg10000,lg0.26)=(4,-0.58)

                                        //find the slope using these points.
take point1 as reference

                                        //data format:{ x, y, slope};

float Ro =  0;                  //Ro is initialized to 10 kilo ohms

#define _XTAL_FREQ 20000000 //20 Mhz

// System related definitions

void system_init(void);

void introduction_screen(void);

void clear_screen(void);

//int   GetPercentage(float rs_ro_ratio, float *pcurve);

int   gas_plot_log_scale(float rs_ro_ratio, float *curve);

float read_mq();

float calculate_resistance(int raw_adc);

float SensorCalibration();

void main() {

     system_init();

     clear_screen();

     lcd_com(FIRST_LINE);

     lcd_puts("Calibrating....");

     Ro = SensorCalibration();

     //clear_screen();

     lcd_com(FIRST_LINE);

     lcd_puts("Done!          ");

     //clear_screen();

     lcd_com(FIRST_LINE);
```

```
     lcd_print_number(Ro);
     lcd_puts(" K Ohms");
     __delay_ms(1500);
     gas_detect = 0;
 while(1){
     if(gas_detect == 0){
         lcd_com(FIRST_LINE);
         lcd_puts("Gas is present  ");
         lcd_com(SECOND_LINE);
         lcd_puts ("Gas ppm = ");
         float rs = read_mq();
         float ratio = rs/Ro;
        lcd_print_number(gas_plot_log_scale(ratio, MQ6_curve));
         __delay_ms(1500);
        clear_screen();
     }
     else{
         lcd_com(FIRST_LINE);
         lcd_puts("Gas not present  ");
         //lcd_com(SECOND_LINE);
        // lcd_print_number(gas_plot_log_scale(read_mq()/Ro, MQ6_curve));
     }
    }
}
 void system_init(){
     TRISB = 0; // LCD pins set to out.
     gas_Detect_Pin_Direction = 1; //Configure RD0 as input
```

```
     lcd_init();
     ADC_Init();
     introduction_screen();
     //dht11_init();
 }
 /*
 This Function is for Clear screen without command.
 */
void clear_screen(void){
    lcd_com(FIRST_LINE);
    lcd_puts("                ");
    lcd_com(SECOND_LINE);
    lcd_puts("                ");
}
 /*
 This Function is for playing introduction.
 */
void introduction_screen(void){
    lcd_com(FIRST_LINE);
    lcd_puts("Welcome to");
    lcd_com(SECOND_LINE);
    lcd_puts("circuit Digest");
    __delay_ms(1000);
    __delay_ms(1000);
    clear_screen();
    lcd_com(FIRST_LINE);
    lcd_puts("MQ6 Sensor");
```

```
    lcd_com(SECOND_LINE);

    lcd_puts("with PIC16F877A");

    __delay_ms(1000);

    __delay_ms(1000);

}

/*

 * Sensor Related Functions

 */

float SensorCalibration(){

  int count;                                    // This function assumes that
sensor is in clean air.

  float val=0;

  for (count=0;count<50;count++) {                    //take multiple samples and
calculate the average value

    val += calculate_resistance(ADC_Read(0));

    __delay_ms(500);

  }

  val = val/50;

  val = val/RO_VALUE_CLEAN_AIR;                         //divided by
RO_CLEAN_AIR_FACTOR yields the Ro

                                                         //according to the chart
in the datasheet

  return val;

}

float read_mq()

{

  int count;

  float rs=0;

  for (count=0;count<5;count++) {                                  // take
multiple readings and average it.
```

```
    rs += calculate_resistance(ADC_Read(0));   // rs changes according to gas
concentration.

    __delay_ms(50);

  }

  rs = rs/5;

  return rs;

}

float calculate_resistance(int adc_channel)

{                                                          // sensor and load
resistor forms a voltage divider. so using analog value and load value

  return ( ((float)RL_VALUE*(1023-adc_channel)/adc_channel));     // we will
find sensor resistor.

}

int gas_plot_log_scale(float rs_ro_ratio, float *curve)

{

    return pow(10,(((log(rs_ro_ratio)-curve[1])/curve[2]) + curve[0]));

}
```