Computer Engineering Dept.

COE344: Embedded Systems

Gas Detector

**System Requirements**

* Ti Tiva TM4C123GH6PM MCU
* Wires
* 1602A LCD
* Potentiometer
* MQ-2 Gas sensor
* Passive Buzzer
* 3x 1kΩ resistors

**System description:**

With the gas detection system, we are using the MQ-2 gas sensor in Detecting LPG gas which is used in home kitchens and used in lighters. An LCD will show the gas concentration (PPM). And in case there is gas leakage, a buzzer will make alarming sound and an alerting message will pop out on the LCD.

**Hardware Setup (HS):**

Ex: à

PE0: Analog. Connected to A0 | PE2: output. Connected to RS

PE3: output. Connected to E | PD0: output. Connected to D0

PD1: output. Connected to D1 | PD2: output. Connected to D2

PD3: output. Connected to D3 | PC4: output. Connected to D4

PC5: output. Connected to D5 | PC6: output. Connected to D6

PC7: output. Connected to D7 | PA2: output. Connected to Buzzer

Diagram, schematic

Description automatically generated

We took the sensor readings from A0 of MQ-2 and stored it in PE0 which we will use it to calculate the ppm through ppm function, then printing it on the LCD by separating the ppm value in a buffer then sending each character alone with a delay(wtimer3) to the LCD through port D and C to print them. We are using PE2 and PE3 for RS and E to chose between sending an instruction or data, and we are making a delay(wtimer3) when sending so we could fetch the data. In case there is gas leakage, we’ll print an alert message and use PA2 to activate the buzzer.

**Program Skelton:**

int main(void){

    //Function calls

portA2\_setup();

    portE0\_and\_ADC\_setup();

    portD0\_1\_2\_3\_setup();

    portC4\_5\_6\_7\_setup();

    portE2\_3\_setup();

    lcd\_setup();

    while(1){

     // Start of conversion

     // wait for conversion to finish

     // Clear end of conversion flag

     // read the calculated ppm of the sensor value

     // store formatted output on temp buffer

     // print ppm on LCD

// Check if PPM is more than 100

If True:

activate buzzer

print alert message

If False:

Deactivate buzzer

     // delay 1sec

     // clear display on LCD

    }

}

**Code**

#include "TM4C123GH6PM.h"

#include <math.h>

#include <stdio.h>

//Sensor PPM  Functions and variables

void portE0\_and\_ADC\_setup(void);

float ppm(float);

float r0 = 0.4388;

float b = 1.2506;

float m = -0.4548;

float ppm\_val;

char temp[15];

// LCD Functions and variables

void portD0\_1\_2\_3\_setup(void);

void portC4\_5\_6\_7\_setup(void);

void portE2\_3\_setup(void);

void lcd\_setup(void);

void lcd\_inst(unsigned char);

void lcd\_data\_char(unsigned char);

void lcd\_data\_string(char \*);

const unsigned char portD\_mask = 0x0F;

const unsigned char portC\_mask = 0xF0;

// Delay funtions

void wtimer3\_setup(int);

void delay(int);

// Passive buzzer function

void portA2\_setup(void);

int main(void){

    //Function calls

    portA2\_setup();

    portE0\_and\_ADC\_setup();

    portD0\_1\_2\_3\_setup();

    portC4\_5\_6\_7\_setup();

    portE2\_3\_setup();

    lcd\_setup();

    while(1){

      ADC0->PSSI |= 0x08; // Start of conversion

      while(~ADC0->RIS&0x08); // wait for conversion to finish

      ADC0->ISC |= 0x08; // Clear end of conversion flag

      ppm\_val = ppm(ADC0->SSFIFO3); // read the calculated ppm of the sensor value

      sprintf(temp, "%f", ppm\_val); // store formatted output on temp buffer

      lcd\_data\_string("LPG PPM: "); // print on LCD

      lcd\_data\_string(temp); // print ppm on LCD

      if(ppm\_val>100){

        GPIOA->DATA |= 0x04; // Activate buzzer

        lcd\_inst(0xC0); // Go to second line on LCD

        lcd\_data\_string("Gas detected!"); // print on LCD

      }

      else{

        GPIOA->DATA &= ~0x04; // Deavtivate buzzer

      }

      delay(1000); //delay 1sec

      lcd\_inst(0x01); // clear display on LCD

    }

}

void delay(int ms){

    wtimer3\_setup(ms); // call for delay

    while(~WTIMER3->RIS&0x01); //Block statment

}

// setting up Wide Timer 3

void wtimer3\_setup(int ms){

    SYSCTL->RCGCWTIMER |= 0x08; //connect clock to Wide Timer 3

    WTIMER3->CTL &= ~0x01; //Disable timer

    WTIMER3->CFG |= 0x04; // Single timer

    WTIMER3->TAMR |= 0x01; // one-shot mode

    WTIMER3->TAILR = (16000\*ms)-1; //number of counts

    WTIMER3->ICR |= 0x01; // clear bit0 of RIS register

    WTIMER3->CTL |= 0x01; // enable timer

}

// setting pin(0, 1, 2, 3) of port D for sending first half instruction to LCD

void portD0\_1\_2\_3\_setup(void){

    SYSCTL->RCGCGPIO |= 0x08; //Connect clock to port D

    GPIOD->DIR |= 0x0F; //Put pins 0, 1, 2, 3 of port D as an Output

    GPIOD->DEN |= 0x0F; //Digitally enable pins 0, 1, 2, 3 of port D

}

// setting pin(4, 5, 6, 7) of port C for sending second half instruction to LCD

void portC4\_5\_6\_7\_setup(void){

    SYSCTL->RCGCGPIO |= 0x04; //Connect clock to port C

    GPIOC->DIR |= 0xF0; //Put pins 4, 5, 6, 7 of port C as an Output

    GPIOC->DEN |= 0xF0; //Digitally enable pins 4, 5, 6, 7 of port C

}

// setting pin(2, 3) of port E for RS and E respectively for LCD

void portE2\_3\_setup(void){

    SYSCTL->RCGCGPIO |= 0x10; //Connect clock to port E

    GPIOE->DIR |= 0x0C; //Put pins 2, 3 of port E as an Output

    GPIOE->DEN |= 0x0C; //Digitally enable pins 2, 3 of port E

}

// setting LCD calls

void lcd\_setup(void){

    lcd\_inst(0x01); // Clear LCD display

    lcd\_inst(0x02); // Return home

    lcd\_inst(0x06); // Increment cursor mode enabled

    lcd\_inst(0x38); // 8-bit mode, two line mode

    lcd\_inst(0x0E); // Display on, Cursor on

}

// Sending instruction to LCD

void lcd\_inst(unsigned char com){

    delay(4); // delay 4msec

    GPIOE->DATA &= ~0x04; // RS = 0

    GPIOE->DATA &= ~0x08; // E = 0

    GPIOD->DATA = portD\_mask&com; // write the fisrt four command bits to LCD

    GPIOC->DATA = portC\_mask&com; // write the second four command bits to LCD

    GPIOE->DATA |= 0x08; // E = 1

    GPIOE->DATA &= ~0x08; // E = 0

}

// write a char to LCD

void lcd\_data\_char(unsigned char da){

    delay(4); // delay 4msec

    GPIOE->DATA |= 0x04; // RS = 1

    GPIOE->DATA &= ~0x08; // E = 0

    GPIOD->DATA = portD\_mask&da; // write the fisrt four data bits to LCD

    GPIOC->DATA = portC\_mask&da; // write the second four data bits to LCD

    GPIOE->DATA |= 0x08; // E = 1

    GPIOE->DATA &= ~0x08; // E = 0

}

// write a string to LCD

void lcd\_data\_string(char \*da){

    delay(4); // delay 4msec

    // while data not pointed to 0

    while(\*da != 0){

        lcd\_data\_char(\*da); // write the pointed char to LCD

        da++; // Increment data by 1

    }

}

// ppm calculation

float ppm(float sensor\_val){

    // declare variables

    float sensor\_volt;

    float rs;

    float rs\_ro\_ratio;

    float ppm;

    sensor\_volt = (sensor\_val\*5)/ 4096; // Sensor voltage formula

    rs = (5-sensor\_volt)/sensor\_volt; // resistance of air with change of gas concentration formula

    rs\_ro\_ratio = rs/r0; // resistance ratio of rs and r0(resistance of sensor in fresh air) formula

    ppm = pow(10, (((log10(rs\_ro\_ratio))-b)/m)); // gas concentration formula

    return ppm;

}

// setting pin 0 of port E for ADC, and setting ADC

void portE0\_and\_ADC\_setup(void){

    SYSCTL->RCGCADC |= 0x01;//Connect clock to ADC 0

    SYSCTL->RCGCGPIO |= 0x10; //Connect clock to port E

    GPIOE->AFSEL |= 0x01;  //Alternative function of pin 0 of port E

    GPIOE->DEN &= ~0x01; //Digitally disable pin 0 of port E

    GPIOE->AMSEL |= 0x01; // Analog mode(function)

    ADC0->ACTSS &= ~0x08; // Deactivate sample sequencer 3

    ADC0->EMUX &= ~0xF000; // Select processor event for sample sequencer 3

    ADC0->SSMUX3 = 3; // Select AIN3(Channel 3) for sample sequencer 3

    ADC0->SSCTL3 |= 0x06; // Interrupt event(2nd bit), End of sequence(1th bit)

    ADC0->ACTSS |= 0x08; // Activate sample sequencer 3

}

void portA2\_setup(void){

    SYSCTL->RCGCGPIO |= 0x01; //Connect clock to port A

    GPIOA->DIR |= 0x04; //Put pin 2 of port A as an Output

    GPIOA->DEN |= 0x04; //Digitally enable pin 2 of port A

}