#define F\_CPU 16000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <i2c.h>

#include <LCD\_I2C.h>

#include <stdbool.h>

char\* Red\_b, Blue\_b, Green\_b, Red\_w, Blue\_w, Green\_w; //for calibration values

//defining pins of LED bulbs

#define Red PORTC0

#define Green PORTC1

#define Blue PORTC2

// defining keypad

#define KEY\_PRT PORTB

#define KEY\_DDR DDRB

#define KEY\_PIN PINB

unsigned char keypad[4][4] = {

{' ', '0', '=', '+'},

{'1', '2', '3', '-'},

{'4', '5', '6', '\*'},

{'7', '8', '9', '/'},

};

unsigned char colloc, rowloc;

char keyfind(){

while (1)

{

KEY\_DDR = 0xF0; /\* set port direction as input-output \*/

KEY\_PRT = 0xFF;

do

{

KEY\_PRT &= 0x0F; /\* mask PORT for column read only \*/

asm("NOP");

colloc = (KEY\_PIN & 0x0F); /\* read status of column \*/

} while (colloc != 0x0F);

do

{

do

{

*\_delay\_ms*(20); /\* 20ms key debounce time \*/

colloc = (KEY\_PIN & 0x0F); /\* read status of column \*/

} while (colloc == 0x0F); /\* check for any key press \*/

*\_delay\_ms* (40); /\* 20 ms key debounce time \*/

colloc = (KEY\_PIN & 0x0F);

} while (colloc == 0x0F);

/\* now check for rows \*/

KEY\_PRT = 0xEF; /\* check for pressed key in 1st row \*/

asm("NOP");

colloc = (KEY\_PIN & 0x0F);

if (colloc != 0x0F)

{

rowloc = 0;

break;

}

KEY\_PRT = 0xDF; /\* check for pressed key in 2nd row \*/

asm("NOP");

colloc = (KEY\_PIN & 0x0F);

if (colloc != 0x0F)

{

rowloc = 1;

break;

}

KEY\_PRT = 0xBF; /\* check for pressed key in 3rd row \*/

asm("NOP");

colloc = (KEY\_PIN & 0x0F);

if (colloc != 0x0F)

{

rowloc = 2;

break;

}

KEY\_PRT = 0x7F; /\* check for pressed key in 4th row \*/

asm("NOP");

colloc = (KEY\_PIN & 0x0F);

if (colloc != 0x0F)

{

rowloc = 3;

break;

}

}

if (colloc == 0x0E) {

return (keypad[rowloc][0]);

}

else if (colloc == 0x0D) {

return (keypad[rowloc][1]);

}

else if (colloc == 0x0B) {

return (keypad[rowloc][2]);

}

else {

return (keypad[rowloc][3]);

}

}

*uint8\_t* pin = 3;

*uint16\_t* Red\_val, Green\_val, Blue\_val;

void adc\_init()

{

ADMUX = (1 << REFS0) | (1 << MUX0) | (1 << MUX1);

// ADC Enable and prescaler of 128

// 16000000/128 = 125000

ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1);

}

*uint16\_t* adc\_read(*uint8\_t* ch)

{

// select the corresponding channel 0~7

// ANDing with ’7? will always keep the value

// of ‘ch’ between 0 and 7

ch &= 0b00000111; // AND operation with 7

ADMUX = (ADMUX & 0xF8) | ch; // clears the bottom 3 bits before ORing

// start single convertion

// write ’1? to ADSC

ADCSRA |= (1 << ADSC);

// wait for conversion to complete

// ADSC becomes ’0? again

// till then, run loop continuously

while (ADCSRA & (1 << ADSC));

return (ADC);

}

//function for get 3 digit from keypad

char\* get\_3\_digit() {

int count = 0;

char\* val;

while (count < 3) {

char digit = keyfind();

if (digit) {

lcd\_cmd(0xC9 + count);

//lcd\_msg("hey");

char\* str\_digit[2];

str\_digit[0] = digit;

str\_digit[1] = '\0';

lcd\_msg(str\_digit);

val += digit;

count += 1;

}

}

return val;

}

// function for display details on lcd display

char\* displayWrite(char\* clr\_mode, char\* clr ) {

lcd\_init();

lcd\_cmd(0xC0);

lcd\_msg(clr\_mode);

lcd\_cmd(0xC7);

lcd\_msg(clr);

lcd\_cmd(0x80);

lcd\_msg("- clear");

lcd\_cmd(0x89);

lcd\_msg("Enter +");

char\* val = get\_3\_digit();

if (val) {

\_Bool flag1 = true;

while (flag1) {

char stat = keyfind();

if (stat == '+') {

return val;

}

else if (stat == '-') {

displayWrite(clr\_mode, clr );

}

}

}

}

void RGB\_off(){

DDRC &= ~(1<<PORTC0 | 1<<PORTC1 | 1<<PORTC2);

}

void setPin(char port,int pin,int state){ //port=A,B,C,D. pin 0-7. state 0 or 1.

switch(port){

case 'B':

if(state==1)

DDRB |= (1<<pin);

else

DDRB &= ~(1<<pin);

break;

case 'C':

if(state==1)

DDRC |= (1<<pin);

else

DDRC &= ~(1<<pin);

break;

case 'D':

if(state==1)

DDRD |= (1<<pin);

else

DDRD &= ~(1<<pin);

break;

default:

DDRB=0xFF;

PORTB=0xFF;

}

}

void pwm(int pin,int num){

TCCR0B |= (1<<CS00)|(1<<CS01);//prescalar /64

TCCR2B |= (1<<CS20)|(1<<CS21);//prescalar /64

switch(pin){

case 0:

TCCR0A |= (1<<WGM01)|(1<<WGM00)|(1<<COM0A1);//fast pwm, non inverted

setPin('D',6,1); // direction

OCR0A=num;

break;

case 1:

TCCR0A |= (1<<WGM01)|(1<<WGM00)|(1<<COM0B1);//fast pwm, non inverted

setPin('D',5,1);

OCR0B=num;

break;

case 2:

TCCR2A |= (1<<WGM21)|(1<<WGM20)|(1<<COM2A1);//fast pwm, non inverted

setPin('B',3,1);

OCR2A=num;

break;

case 3:

TCCR2A |= (1<<WGM21)|(1<<WGM20)|(1<<COM2B1);//fast pwm, non inverted

setPin('D',3,1);

OCR2B=num;

break;

}

}

int converter(char\* a){

//char str[30] =a;

char \*ptr;

int ret;

ret = *strtol*(a, &ptr, 10);

return ret;

}

int main(void){

i2c\_init();

i2c\_start();

i2c\_write(0x70);

lcd\_init();

lcd\_cmd(0x80); lcd\_msg("mode 1");

lcd\_cmd(0x88); lcd\_msg("mode 2");

lcd\_cmd(0xc5); lcd\_msg("mode 3");

DDRC = 0b00000111;

adc\_init();

while (1)

{

//mode selection

char mode = keyfind();

if (mode == '1') { //mode 1 - calibration mode

RGB\_off();

Red\_b = displayWrite("Min", "R-");

if (Red\_b) {

Green\_b = displayWrite("Min", "G-");

if (Green\_b) {

Blue\_b = displayWrite("Min", "B-");

if (Blue\_b) {

Red\_w = displayWrite("Max", "R-");

if (Red\_w) {

Green\_w = displayWrite("Max", "G-");

if (Green\_w) {

Blue\_w = displayWrite("Max", "B-");

if (Blue\_w) {

\_Bool flag = false;

\_Bool calibrated = true;

lcd\_init();

lcd\_cmd(0x82);

lcd\_msg("Calibration");

lcd\_cmd(0xC6);

lcd\_msg("Done!");

*\_delay\_ms*(1000);

main();

}

}

}

}

}

}

}

if (mode == '2') { //mode 2 - sensoring mode

lcd\_init();

while (1) {

PORTC ^= (1 << Red); *\_delay\_ms*(500); Red\_val = adc\_read(pin); PORTC ^= (1 << Red); //light up red bulb

PORTC ^= (1 << Green); *\_delay\_ms*(500); Green\_val = adc\_read(pin); PORTC ^= (1 << Green); //light up green bulb

PORTC ^= (1 << Blue); *\_delay\_ms*(500); Blue\_val = adc\_read(pin); PORTC ^= (1 << Blue); //light up blue bulb

// convert uint16 to string

char str\_red [sizeof(Red\_val) \* 8 + 1];

char str\_green [sizeof(Green\_val) \* 8 + 1];

char str\_Blue [sizeof(Blue\_val) \* 8 + 1];

lcd\_cmd(0x80);

*utoa*(Red\_val, str\_red, 10);

lcd\_msg("R:"); lcd\_cmd(0x83); lcd\_msg(str\_red);

*utoa*(Green\_val, str\_green, 10);

lcd\_cmd(0x88); lcd\_msg("G:"); lcd\_cmd(0x8B); lcd\_msg(str\_green);

*utoa*(Blue\_val, str\_Blue, 10);

lcd\_cmd(0xC6); lcd\_msg("B:"); lcd\_cmd(0xCA); lcd\_msg(str\_Blue);

}

}

if (mode == '3') { //mode 3 - light up RGB led for given R,G,B values

lcd\_init();

lcd\_cmd(0x81);

lcd\_msg("mode 3 selected");

*\_delay\_ms*(1000);

lcd\_init();

lcd\_cmd(0x81);

lcd\_msg("Enter RED value");

int Red\_Val= converter(get\_3\_digit());

*\_delay\_ms*(1000);

lcd\_init();

lcd\_cmd(0x81);

lcd\_msg("Enter GREEN value");

int Green\_Val= converter(get\_3\_digit());

*\_delay\_ms*(1000);

lcd\_init();

lcd\_cmd(0x81);

lcd\_msg("Enter BLUE value");

int Blue\_Val= converter(get\_3\_digit());

pwm(0,Red\_Val);

pwm(1,Green\_Val);

pwm(3,Blue\_Val);

}

}

}