Coursework 2

Hassan Megahed CU1900237

Intro:

Human machine interface (HMI) is crucial, providing access and intake on the process for the user. The purpose of the following code is to take input from the user and report the temp over car display ideally.

System design:

A MICRO controller model PIC18F45K50 was used to implement this coding for simulation and model PIC18F47K42. The sensor utilized to read the temperature was LM35 displaying its readings through code onto the LCD, connected to Port A RA2 (analog input to digital). The LCD in use was a hd44780 16x2 sized LCD screen, connected to port D. for user input the buttons present on the easy PIC was optimized connected to port B. RB1 RB2 RB3 RB4 were the buttons in use. A heater was also added as part of the hardware to control the environment of the sensor for experimental flexibility, connected to port C RC5. Finally, a buzzer was connected at Port C RC1 as an indicator of overstepped boundaries.

BOM:

The following bill of materials includes a list of all materials used along with a general description, price and online purchasing link.

Hardware: EasyPIC V8:

Development board used by engineers to program compatible microcontrollers for embedded systems

Price: 199.20$ per unit

Purchasing link: https://www.mikroe.com/easypic

Hardware: LM35:

Precision Centigrade Temperature Sensor (reads temperature) from −55°C to 150°C, contains Vs Vout and gnd, with a linearity of + 10-mV/°C.

Price: 35.70EGP per unit

Purchasing link: https://www.noon.com/egypt-en/temperature-sensor-lm35dz-black/N50978788A/p/?gclid=EAIaIQobChMI3MzdhO7h9AIVQ7TVCh1KzQ_sEAQYASABEgKlD_D_BwE

Hardware: LCD:

hd44780 16x2, LCD controller connected to the Easy PIC displaying 2 columns and 16 rows of characters.

Price: 72.95EGP per unit

Purchasing link: https://www.noon.com/egypt-en/1602-16x-2-character-lcd-display-module-hd44780-controller-green-15-9-x-9-x-3-3cm/N50978662A/p/?gclid=EAIaIQobChMIyM3a2-7h9AIVAdxRCh2pXwy3EAQYASABEgLJvvD_BwE

Hardware: Heater:

Used to control the environment of the sensor, heats surrounding at a high rate.

Price: 75EGP per unit

Purchasing link: https://www.amazon.eg/-/en/XH-W1209-Thermostat-Temperature-Control-raspberry/dp/B091KS7JFQ/ref=asc_df_B091KS7JFQ/?tag=egoshpadde-21&linkCode=df0&hvadid=545135802409&hvpos=&hvnetw=g&hvrand=11817672940582975432&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=21468&hvtargid=pla-1438475353649&psc=1

Hardware: Buzzer:

Turns voltage into sound as v input increases sound volume increases

Price: 40EGP per unit

Purchasing link: https://www.noon.com/egypt-en/small-buzzer/Z6D15C01D594CA11B99D7Z/p/?o=z6d15c01d594ca11b99d7z-1&gclid=EAIaIQobChMIyM3a2-7h9AIVAdxRCh2pXwy3EAQYASABEgLJvvD_BwE

Software: MikroC Pro for PIC:

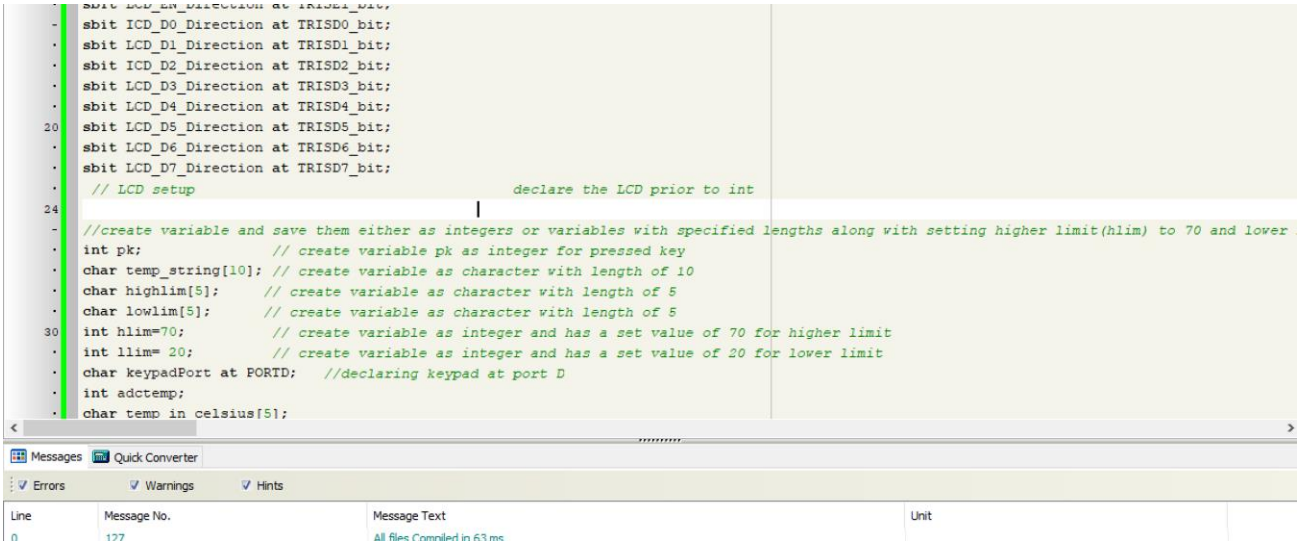
C compiler for PIC devices, offers access to libraries for the code at hand

Price:215.20$ per unit

Purchasing link: https://www.mikroe.com/mikroc-pic

SW implementation:

The code starts off with the LCD declaration along with LCD module connections, declaring the LCD prior to its initialization. The second stage was creating variables for the signal value, value of temperature in degrees and value to store it as a string. same goes for the value of the limits.



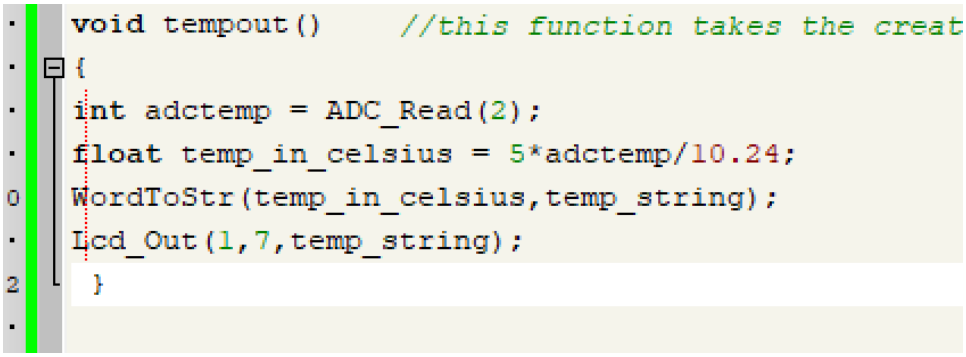
```
sbit ICD_D0_Direction at TRISD0_bit;
sbit LCD_D1_Direction at TRISD1_bit;
sbit ICD_D2_Direction at TRISD2_bit;
sbit LCD_D3_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;
 // LCD setup                                  declare the LCD prior to int

//create variable and save them either as integers or variables with specified lengths along with setting higher limit(hlim) to 70 and lower
int pk;               // create variable pk as integer for pressed key
char temp_string[10]; // create variable as character with length of 10
char highlim[5];     // create variable as character with length of 5
char lowlim[5];      // create variable as character with length of 5
int hlim=70;          // create variable as integer and has a set value of 70 for higher limit
int llim= 20;         // create variable as integer and has a set value of 20 for lower limit
char keypadPort at PORTD;   //declaring keypad at port D
int adctemp;
char temp_in_celsius[5];
```

Messages | Quick Converter

Errors | Warnings | Hints

Line	Message No.	Message Text	Unit
0	127	All files Compiled in 63 ms	

The next step in the process was generating functions, starting off with a function for temp calculation and output on the LCD. This function takes the created variable adctemp and saves the signal taken from the sensor to it. The signal received from the sensor then goes through an equation necessary to retrieving a value for the temperature in degrees. LCD cannot display integers, floats etc. and therefore a word to string was used converting the value to a string saved in the created char variable with length of 10. finally, it was displayed in row 1 column 7 beside" Temp:".



```
void tempout()     //this function takes the creat
{
int adctemp = ADC_Read(2);
float temp_in_celsius = 5*adctemp/10.24;
WordToStr(temp_in_celsius,temp_string);
Lcd_Out(1,7,temp_string);
 }
```

The following function was one for the higher limit. This function's first purpose is setting the picked buttons once to increase the set higher limit by one using button RB2 and decreasing the higher limit by 1 using the RB1 button, overwriting the original set variable. The value in this variable is then converted to string whixh is no compatible for displaying on the LCD and is outputted along with "HL" that represents higher limit.

```
void higherlimout()   //this function contains
{
if (RB2_bit ==1)
{
hlim = hlim+1;
}
if (RB1_bit ==1)
{
hlim = hlim-1;
}

shortToStr(hlim,highlim);
Lcd_Out(2,2,highlim);
Lcd_Out(2, 1, "HL:");

 }
```

The function after was for the lower limit. This function sets the picked buttons once to increase the set lower limit (llim) by one using button RB3 and decreasing the lower limit by 1 using the RB4 button, overwriting the original set variable. The value in this variable is then converted to string which is now compatible for displaying on the LCD and is outputted along with "LL" that represents lower limit.

```
void lowerlimout() // mirroring the
{
if (RB3_bit ==1)
{
llim = llim+1;
}
if (RB4_bit ==1)
{
llim = llim-1;
}

shortToStr(llim,lowlim);
Lcd_Out(2,10,lowlim);
Lcd_Out(2, 8, "LL:");


}
```

Moving on to the main function, it starts with assignment of the PORTS ANSEL's with 0 are assigned to become digital and 1 for analog. TRIS's with 0 are output while 1 are for inputs.

The LCD cursor is removed and the screen initially displays good afternoon in row 1 assignment 2 in row 2 and stays that way for 1 sec before it clearing.

```
void main(){

//LCD Setup
 //ANSEL represents wther its analog or digital
ANSELD=0;
TRISD=0;
PORTD=0;
TRISE=0;
PORTE=0;
ANSELE=0xff;
Lcd_Init() ;
Lcd_Cmd(_LCD_CLEAR) ;   //clears LCD screen
Lcd_Cmd(_LCD_CURSOR_OFF) ;   // removes cursor
    //displays good afternoon assignment 2  leav
Lcd_Out(1,1, "Good Afternoon") ;
Lcd_Out(2,1, "Assignment 2") ;
Delay_ms(1000) ;
Lcd_Cmd(_LCD_CLEAR) ;
```

Further along the code ADC input, Button input and heater ports are assigned and turned on.

```
//Setup Adc input

ANSELA=0xff;
TRISA=0xff;
//RA2_bit=1;
ADC_Init();

//setup button input for limits

ANSELB=0;
TRISB=0xff;




//Setup Heater

ANSELC=0;
TRISC = 0;
RC5_bit=1;      // turn heater on
```

A while loop was then necessary to place the conditions for the buzzer in a place which was constantly run to continuously check if the conditions are met. first if condition states that if the temperature read exceeds the higher limit than RC1 where the buzzer is connected turns on, if 2 mentions that if the temperature read is below the lower limit RC1 will equal 1 turning the buzzer on and if none are met (else) the buzzer (RC1) remains off. And then all functions mentioned above are called displaying every output on the LCD.

```
  while(1)    // this conditions allow the buzzer to go off when temperature read
  {

if (((char)temp_in_celsius >= hlim))
  {
  RC1_bit=1;

  }
else if (((char)temp_in_celsius <= llim))
  {
  RC1_bit=1;
  }



  else
  {
  RC1_bit=0;
  }
  tempout();           // call function for temperature calculation and display
  higherlimout();     // call function for higher limit input and display
  lowerlimout();     // call function for lower limit and display
```

Test Cases:

Case 1: Accurate Temperature reading increasing

For the first Case, it was necessary to ensure that both the temperature reading through the sensor and the code displaying the value on the LCD was working accurately. Therefore, the heater was used, monitoring whether the temp value on display was correspondingly going up.

Case 2: Accurate Temperature reading decreasing

For the second case, a second test was conducted to make sure the reading would also go down as the temperature decreased and therefore after the heater increased the temperature of the surrounding it was turned off and as the temperature went down the LCD display was monitored to verify that it matched the actual temperature.

Case 3: higher limit input increasing

The third case was conducted to test the input (upper limit) was precise. The button reserved for increasing the higher limit (RB2) by 1 was pressed while the LCD display was monitored making sure the value was increasing correspondingly with the number of times the button was pushed. This in turn makes sure the code for LCD display is correct along with the assignment of the button.

Case 4: higher limit input decreasing

In case four the opposite was tested. RB1 was pressed and the output on the LCD was monitored. RB1 was reserved for decreasing the value of the higher limit by one every time it is pressed. If the value on the LCD changes correspondingly with the number of times the button is pressed.

Case 5: lower limit input increasing

Mirroring the case regarding the higher limit, case number 5 was conducted to test the input (lower limit) was precise. The button reserved for increasing the lower limit (RB3) by 1 was pressed while the LCD display was monitored making sure the value was increasing correspondingly with the number of times the button was pushed. This in turn makes sure the code for LCD display is correct along with the assignment of the button.

Case 6: lower limit input decreasing

In alignment with case four, RB4 was pressed and the output on the LCD was monitored. RB4 was reserved for decreasing the value of the higher limit by one every time it is pressed. If the value on the LCD changes correspondingly with the number of times the button is pressed.

Case 7: testing buzzer when upper limit is exceeded

Case 7 was conducted to ensure the buzzer goes off as soon as the upper limit is exceeded. The heater is used along with lowering the set upper limit and monitoring the output of the buzzer.

Case 8: testing buzzer when lower limit is exceeded

Case 8 was conducted to ensure the buzzer goes off as soon as the lower limit is exceeded. The heater was turned off to decrease the environmental temp and the lower limit was increased while monitoring the output of the buzzer.

Due to technical difficulties regarding the hardware in the lab, a new updated version of the code was created and ran on simulation for verification of its authenticity and precision. Many trials were conducted on the hardware all leading to unsuccessfulness, while the simulation ran the code as desired.