# Temperature Alert System with PIC MICROCONTROLLER (PIC16F877A)

CSE331.1

## Group Members:

1. Name : Monsur Hillas ID : 172 1911 042
2. Name : Nayeem Mohammad ID : 161 1553 042
3. Name : Imran Hasan Bhuiyan  ID : 161 2412 042
4. Name : Shanzida Akter ID : 152 0533 642

## Objective:

Our main objective is to build a system using a PIC MICROCONTROLLER (PIC16F877A) where the system will sense the current temperature of the environment it is placed in and will display it in the LCD screen attached to the microcontroller. Also, there will be a buzzer system in the system which will warn the user if the temperature increases more than a specified level. Our main goal is to create an interface between hardware and software which we have learned in our class.

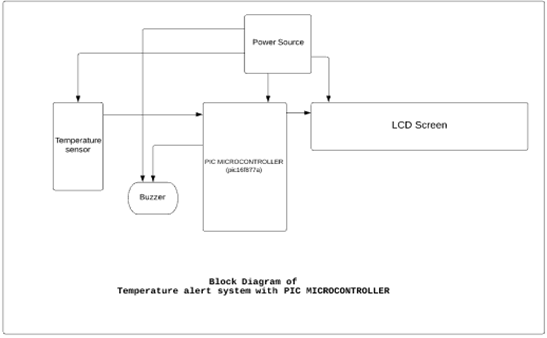
## Potential Application:

The main application of this project is to sense the temperature and notifying the user if the temperature crosses a specific limit. This system can be potentially used in a few places. Below we have listed a few potential applications of our project.

1. Fire Detection: In case of a fire, the temperature will increase. If the temperature increases more than a certain threshold (say 65-degree Celsius) which can’t be reached due to natural causes, we can assume it happened due to fire. If used alongside a smoke detector system, our system can be used to create a pretty accurate fire alarm system.
2. Air Conditioner: The compressor in Air Conditioner doesn’t need to work all the time. It only needs to work when the temperature inside the room has risen more than the specified limit set by the user. So our system can be used in Air Conditioners to restart the compressor again when the temperature inside the room increases.

1. To know the current Temperature: Our system can be used anywhere where the user just needs to know the current temperature of the environment.

## Block Diagram:

****

## Required component:

|  |  |
| --- | --- |
| **Serial no** | **Component Name** |
| **1** | Pic Microcontroller (PIC16F877A) |
| **2** | LCD Screen (16X2) |
| **3** | Potentiometer(10K) |
| **4** | Male-Male jumper wires |
| **5** | 16.000 MHz Low profile crystal |
| **6** | MC7805CT 1A voltage regulators |
| **7** | LM35 Celsius temperature sensor |
| **8** | 3.3-5V Continuous Tone Active Buzzer |
| **9** | 1 kΩ Resistor |
| **10** | 9V DC battery |
| **11** | 9V DC battery terminals |
| **12** | Breadboard (x2) |
| **13** | PICKIT 3 programmer offline programming |

## Estimated Cost:

|  |  |  |
| --- | --- | --- |
| **Serial no** | **Component Name** | **Cost** |
| **1** | Pic Microcontroller (PIC16F877A) | 250/= TK |
| **2** | LCD Screen (16X2) | 120/= TK |
| **3** | Potentiometer(10K) | 10/= TK |
| **4** | Male-Male jumper wires | 85/= TK |
| **5** | 16.000 MHz Low profile crystal | 7/= TK |
| **6** | MC7805CT 1A voltage regulators | 10/= TK |
| **7** | LM35 Celsius temperature sensor | 35/= TK |
| **8** | 3.3-5V Continuous Tone Active Buzzer | 25/= TK |
| **9** | 1 kΩ Resistor | 10/= TK |
| **10** | 9V DC battery | 50/= TK |
| **11** | 9V DC battery terminals | 10/= TK |
| **12** | BreadBoard (x2) | 100/= TK |
| **13** | PICKIT 3 programmer offline programming | 1400/= TK |
|  | **TOTAL** | **2112/= TK** |

## Working Procedure:

For the procedure part we divided our tasks in two major parts.

i)                    Software Simulation

ii)                   Hardware Simulation

Software Simulation:

For software simulation we used two software to create our projects simulation:

1.       Proteus

2.       Micro C Pro

In Proteus we created the circuit of our project. We placed the microcontroller (PIC16F877A) in the center of the board and then connected to other peripheral devices. After completing the circuit, we needed to simulate the circuit. But to simulate this we needed to burn the microcontroller of proteus in order to get the expected result.

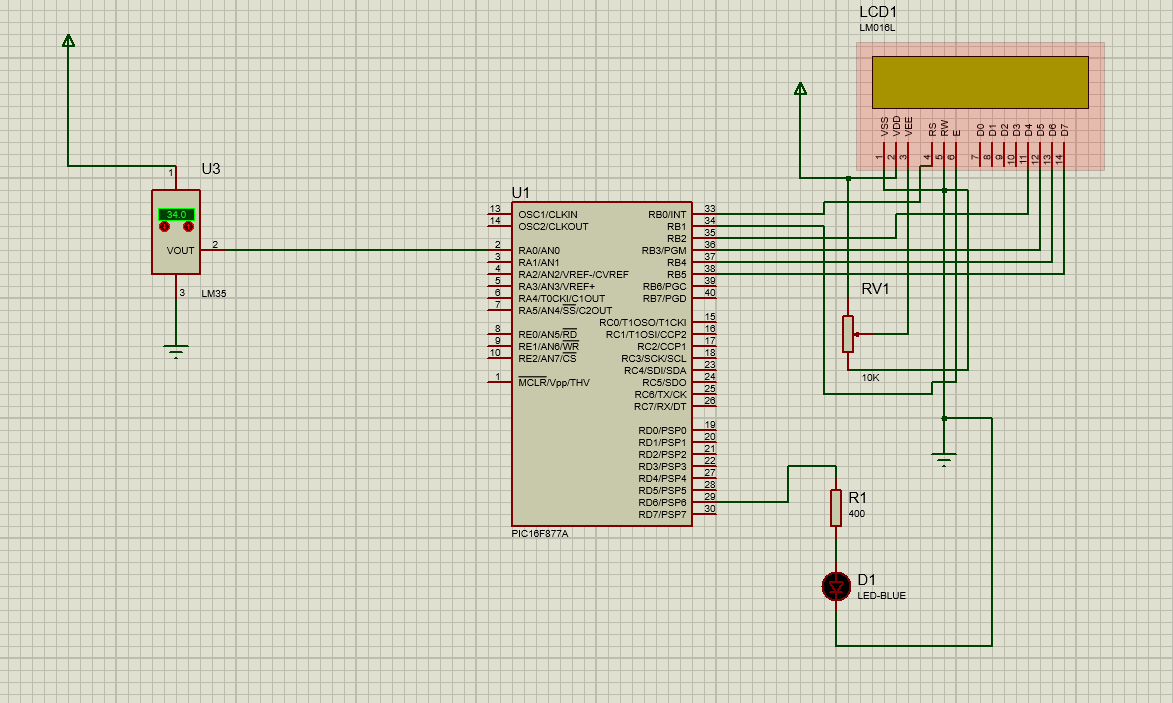
We started to code in an IDE called Micro C pro. We wrote a C code for the project and then compiled it. The IDE auto generated the hex file of that code. We used this hex file to burn the microcontroller of the proteus circuit. After burning the MCU, we simulated it and got our expected result. With this we completed our software simulation with the help of Proteus and Micro C pro.

Hardware Simulation:

After successfully completing the software simulation, we bought the necessary hardware parts that were needed to complete the hardware simulation from a local electronics shop. Then we started to assemble them. We used the circuit which we created in proteus and exactly built the same circuit on breadboard. After completing the circuit, we needed to burn the hex file in the microcontroller, but we have to burn the microcontroller only, we cannot burn the microcontroller when it is connected to other peripheral devices. So, we took the microcontroller from there and placed it in another empty portion of the breadboard and only burned the microcontroller with the help of pic microcontroller burner PICKIT 3 offline programmer. Then we again placed the microcontroller in its own position in breadboard and connected the total circuit with a power supply, and we got the same result that we got in software simulation.

With this we completed our hardware simulation.

## Figure from Simulation Tool:

****

## Interfacing Language from Simulation Tool:

For simulation we have used C as an Interfacing language. The code is given below:

// Lcd pinout settings

sbit LCD\_RS at RB0\_bit;

sbit LCD\_EN at RB1\_bit;

sbit LCD\_D7 at RB5\_bit;

sbit LCD\_D6 at RB4\_bit;

sbit LCD\_D5 at RB3\_bit;

sbit LCD\_D4 at RB2\_bit;

// Pin direction

sbit LCD\_RS\_Direction at TRISB0\_bit;

sbit LCD\_EN\_Direction at TRISB1\_bit;

sbit LCD\_D7\_Direction at TRISB5\_bit;

sbit LCD\_D6\_Direction at TRISB4\_bit;

sbit LCD\_D5\_Direction at TRISB3\_bit;

sbit LCD\_D4\_Direction at TRISB2\_bit;

int vref = 5 ; // its our applied voltage change if 3.3 or 3.7 etc

int tmp;   // intialzie variable int as tmp

int temp\_celcius;

char tmp1[3]; // intialize char to store a string of 3 chars

void main() {

LCD\_Init(); //Intializes the LCD modules

DElay\_ms(250);

ADC\_Init(); // Initializes the ADC Module for ADC Conversions

Delay\_ms(250);

TRISD = 0x00;

PORTD=0x00;

LCD\_OUT(1,1, " CSE331 ");// LCD Will display at row 1 column 1 Learning.

LCD\_OUT(2,2, "PROJECT"); // LCD Will display at row column 2 Microcontrollers.

Delay\_ms(2000); // Will display this for 2 seconds

Lcd\_Cmd(\_LCD\_CLEAR); // Will clear LCD for new valuse to be displayed

LCD\_OUT(1,1, " TEMPERATURE ");// LCD Will display at row 1 column 1 Learning.

LCD\_OUT(2,2, " CONTROL "); // LCD Will display at row column 2 Microcontrollers.

Delay\_ms(2000); // Will display this for 2 seconds

Lcd\_Cmd(\_LCD\_CLEAR); // Will clear LCD for new valuse to be displayed

while(1) // Needed for operations its an internal loop which keeps on executing

{

tmp = ADC\_Read(0); // ADC read from pin AN 0

Delay\_ms(50);

temp\_celcius = ( tmp \* vref )/ 10 ;

delay\_ms(20);

if(temp\_celcius>35)

{

 LCD\_OUT(1,1, " TEMPERATURE ");// LCD Will display at row 1 column 1 Learning.

 LCD\_OUT(2,2, " HIGH "); // LCD Will display at row column 2 Microcontrollers.

 Delay\_ms(500); // Will display this for 2 seconds

 Lcd\_Cmd(\_LCD\_CLEAR); // Will clear LCD for new valuse to be displayed

 PORTD.f6=1;

}

else PORTD.f6=0;

IntToStr(temp\_celcius, tmp1);// will convert values of tmp to tmp1

delay\_ms(20);

// Now we can display this string on LCD

Lcd\_Cmd(\_LCD\_CLEAR); // Precaution clear for readings to be displayed

LCD\_OUT(1,1, "Temp = ");

LCD\_OUT(1,10,tmp1); // I changed column number to display result after above helping string

Delay\_ms(800); // Keep displaying same value for 0.5 sec

Lcd\_Cmd(\_LCD\_CLEAR); // Then clear LCD for new values

}

}

## Discussion:

From this project we have learned how to work with a microcontroller, how to create software simulation in proteus, how to generate hex files using specific IDE, and how to burn the hex file into microcontroller of the proteus circuit.

We have also learned about how the basic circuit of a microcontroller works, how to work with other devices along with microcontroller, how to interface them. By doing this project we have gained practical knowledge of creating application-based projects using microcontrollers.

We completed the project properly, but this completion was not so easy, we faced some difficulties in some steps.

At first, we were stuck for searching the IDE. We were confused about which IDE to use for generating hex files, as some IDEs didn’t allow compiling code in demo version and to compile our code, we needed the pro version of it. So, it took some time for us to find the perfect IDE to compile our C++ code and generate the hex file.

Secondly, in the beginning we thought we didn’t need to buy the PICKIT3 offline programmer as it was so costly and could cost us around 1400/= taka. We hoped to borrow it from someone else. But being unable to find it from anyone we know, we had to buy it ourselves and then we could burn the code in our microcontroller.

Thirdly, we had a misconception about powering the microcontroller. We thought MCLR (Master Clear Pin Reset) was the pin by which we can power the microcontroller, so we did accordingly (as in the software simulation there is no external connection to power the microcontroller). But the Microcontroller was not activated by that. And we were not getting the expected output from the hardware simulation and thought that there is some problem with our microcontroller. But after some time, we found that we need to use VDD and VSS pin of microcontroller to power up the microcontroller, and after that connection our project was running successfully.

Finally, we faced another problem in the end. The sensor (LM35) was giving a random value, like once it shows 25 Celsius, the next moment it shows 65 Celsius. We found out that there was some problem in that section of breadboard for which this problem was created. We placed the LM35 to another part of breadboard and our problem was solved.

We can say that we didn’t have to do a lot to get this project up and running. It was not a very difficult or tough project. Nonetheless, it gave us some basic ideas about how to work with microcontroller or to be specific, PIC microcontrollers.