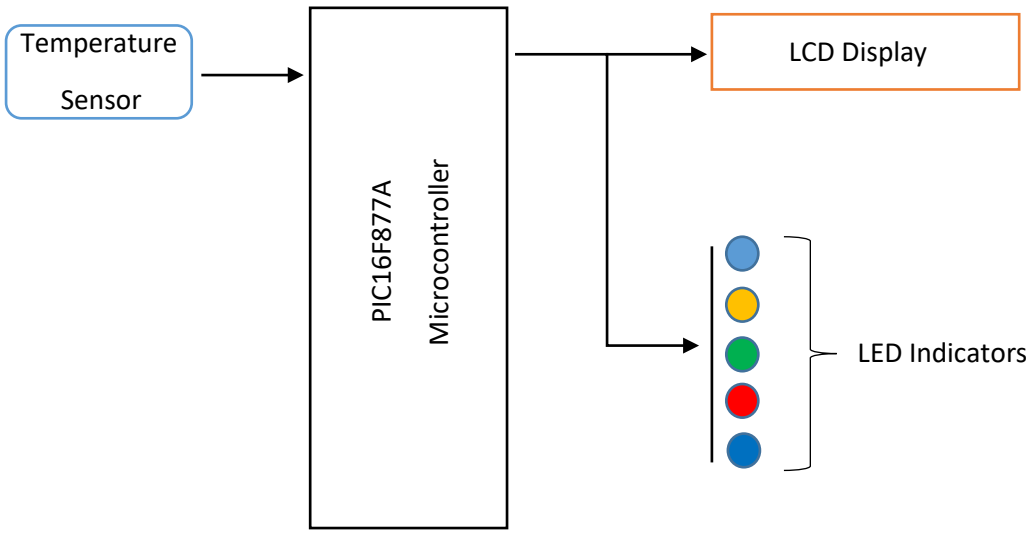
Objective

Our objective was to build a temperature monitor using a microcontroller which can show temperature in real time and indicate range of current temperature with the knowledge we gained from the Microprocessor Interfacing and Embedded System course.

Applications

Temperatures Monitor shows the current temperature of the surroundings and also indicates the range of the temperature. It can be used to get information of the current temperature and whether it's being too hot or too cold. It also gives instant reading of a place when going to a new place. From the reading and indicator, it can be easily known if the temperature has fallen or risen suddenly. It can also be used near machines that produces heat to know if the machine is heating up itself and it's surroundings.

Block Diagram



Required Components

- PIC16F877A Microcontroller
- LM35 Temperature Sensor
- 16x2 LCD Display
- Different Color LEDs
- 10k Potentiometer
- Jumper Wire
- 9V Battery Connector
- 5V Power Supply kit
- 22pF Ceramic Capacitor
- 20MHz Crystal Oscillator
- 10K Ohm Resistor
- 220 Ohm Resistors
- Breadboard
- PICkit 3
- PIC Programming Adapter

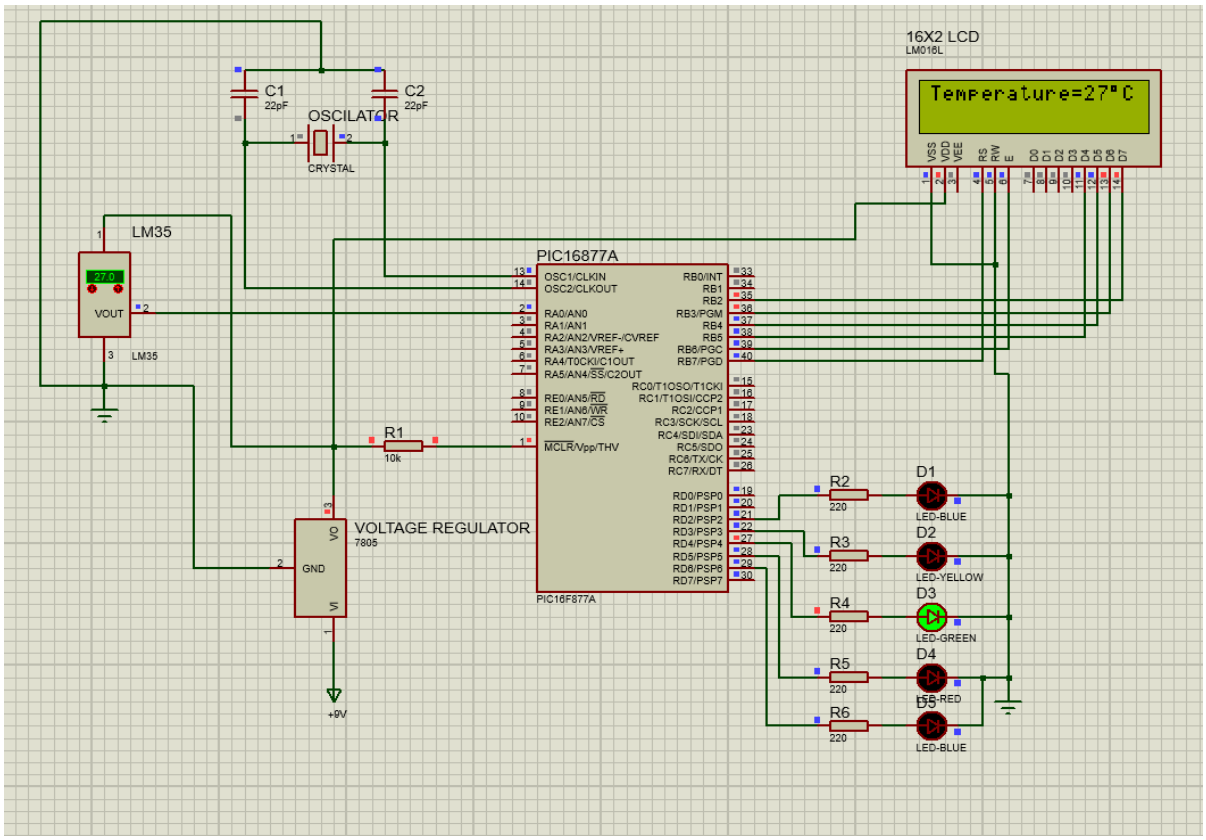
Estimated Cost

Component	Quantity	Cost (BDT)
PIC16F877A	1	200
LM35	1	70
16x2 LCD	1	135
LEDs	5	10
Potentiometer	1	15
Jumper Wire	-	60
Battery Connector	1	30
Power Supply Kit	1	286
Capacitor	2	8
Crystal Oscillator	1	9
Resistors	6	10
Breadboard	1	70
PICkit 3	1	5755
PIC Programming Adapter	1	270
Battery	1	40
	Total	6958

Working Procedure

- First, we picked PIC16F877A microcontroller to use in our project.
- Then we wrote a program using C language in MikroC IDE for the project.
- In the program we defined the port and pin numbers to be connected with LCD display, Temperature Sensor and LED outputs.
- During the programing we include lcd, conversion and string library to the project. Without the library the program won't compile.
- We wrote a program that uses first pin of Port A of PIC16F877A to take input from sensor and Port B to display LCD and Port D to output to LEDs.
- Then we build the program and generated a hex file for the project.
- After we used proteus to build a circuit and simulate for our project.
- In proteus, we connected the pin 1 of PIC16F877A with a 10k resistor to the power which works as a pull up resistor.
- We added pin 1 to power, 3 to ground and 2 to PIC16F877A pin 2 of the LM35 sensor following the pin diagram of LM35 sensor.
- Then we added a 20MHz oscillator with pin 13 and 14 of PIC16F877A and two 22pF capacitors with the oscillator that connect it to ground
- Then we connected the LCD display with the PIC16F877A using Port B and following the pin diagram of PIC16F877A and 16x2 LCD display.
- Finally, we connected the LEDs with the pin of Port D as we defined in the program.
- We used a voltage regulator that regulate voltage from 9v to 5v to supply power to the all components.
- Then, we loaded the hex file into the PIC16F877A and simulated the circuit. After few adjustments in the codes it worked as expected.
- After that, we implemented the circuit following the simulation and pin diagram of PIC16F877A, LCD display, and sensor and used a 9v battery with 5v power supply kit to power up the circuit.
- We used PICkit3 and connected it with PIC16F877A through PIC programming adapter and burned our hex file into the PIC16F877A and then implemented it into the circuit.

Circuit from the Proteus Simulation



Code for the Project written in C language

```
sbit LCD_D7 at RB2_bit;

sbit LCD_D6 at RB3_bit;

sbit LCD_D5 at RB4_bit;

sbit LCD_D4 at RB5_bit;

sbit LCD_EN at RB6_bit;

sbit LCD_RS at RB7_bit;


sbit LCD_D7_Direction at TRISB2_bit;

sbit LCD_D6_Direction at TRISB3_bit;

sbit LCD_D5_Direction at TRISB4_bit;
```

```
sbit LCD_D4_Direction at TRISB5_bit;
sbit LCD_EN_Direction at TRISB6_bit;
sbit LCD_RS_Direction at TRISB7_bit;

int temp;
char temper[7];

void readTenperature(void)
{
    temp = ADC_Read(0);
    temp = temp * 0.4887;

    if(temp<=10)
    {
     PORTD=0b00000100;
    }
    else if(temp>10 && temp<=22 )
    {
     PORTD=0b00001000;
    }
    else if(temp>22 && temp<=32 )
    {
     PORTD=0b00010000;
    }
    else if(temp>32 && temp<=48 )
    {
     PORTD=0b00100000;
    }
```

```
    else if(temp>48)
    {
     PORTD=0b01000000;
    }
}

void conversion(void)
{
    inttostr(temp,temper);
}

void display(void)
{
    lcd_out(1,1,"Temperature=");
    lcd_out(1,13,Ltrim(temper));
    Lcd_Chr_Cp(0xdf);
    Lcd_Chr_Cp('C');
    Lcd_Chr_Cp(' ');
}

void main()
{
    ADC_Init();
    Lcd_Init();
    Lcd_Cmd(_LCD_TURN_ON);
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Cmd(_LCD_CURSOR_OFF);
    Lcd_out(1, 3, "Temperature");
```

```
    lcd_out(2, 3, "Monitor");
    delay_ms(1000);
    Lcd_Cmd(_LCD_CLEAR);
    TRISD=0;

    while(1)
    {
     readTenperature();
     conversion();
     display();
    }
}
```

Discussion

In this project we sensed the temperature with LM35 temperature sensor and used PIC16F877A to display the temperature from sensor to LCD display and LED indicators. We choose PIC16F877A as it is widely used when working with the sensors and also a cheap microcontroller. LM35 temperature sensor can sense from -50 to 150 degree temperature. It gives output in mV varying on the temperature it reads from surroundings. It changes 10mV for each degree temperature change. The 16x2 LCD display can show 16 characters in two rows. It shows the reading from the temperature sensor converted into celsius through the program in the display. The led lights indicate the range of temperatures, whether it is cold or normal or too high temperature.

While programing for this project we first had to select a port to read the temperature from the sensor. We used port A to receive the reading from the temperature sensor. The AN0 pin of port A gives us the analog signal from the temperature sensor and converts it into the digital signal. Our whole system is digital but the temperature sensor gives us reading in analog and the ADC gives us the digital value for the reading. The reading is then stored in a variable in PIC16F877A and converted into character to show it in the lcd display. The C conversion and string library is used to convert the value to character in the program. This library also has pre-defined function to execute commands like showing character and clearing the display.

The lcd display is connected with the PIC16F877A by defining the pins of port B to the pins of the lcd display. The lcd library has the defined function to connect the lcd with the pin of microcontroller. We used that to connect our lcd display with the PIC16F877A. We chose port D to show the indicators LEDs. To use port D for output we use the TRIS function to set the Port D to 0 so that the pins of port D works as output pin. For different range of temperature, we used port programming and set the value of port D to lit up different LEDs connected to port D for different range of temperature. Lowest temperatures lit up blue led, low temperature yellow led, normal temperature green led, high temperature red led and very high temperature multicolor led. For different temperature range we set the bit for port D according to our need to lit up the exact led light. In the main function of program, we call an infinite loop that continuously reads and displays temperature reading with the led indicator until the power is turned off. In the circuit simulation and hardware implementation we followed the pin diagram to build the circuit. The pin 1 of PIC16F877A is master clear. We connected an 10k resistor to power with it so that it works as a pull up resistor. If we don't do that the pin 1 won't be always high and if it goes to low or ground it will reset the PIC16F877A. The pin 2 is an analog pin that we connected with the pin 2 of the LM35, through which the PIC16F877A gets the temperature reading from the sensor. The 13 and 14 pins of PIC16F877A is connected with a 20MHz oscillator. The PIC16F877A has no internal oscillator. So, we provided a 20MHz oscillator to give the clock signal to the PIC16F877A. We added two 22pF capacitors with the oscillator to make an oscillator circuit. Without the capacitor there won't be a 360-degree phase shift and it won't oscillate. We connected the LEDs with the port D through 220ohm resistors. Because if we don't add resistor the output from the port D pins are 5v which will damage the LEDs. We connected the LCD with the port D of PIC16F877A. We used 4-bit interfacing of LCD using the D4, D5, D6, D7 pin of lcd with the PIC16F877A. the RW pin of the LCD is connected to the ground to enable write to the LCD. The pin 15 of the lcd is connected to the power through a 220 ohm resistor and the pin 16 is connected to the ground for the backlight of the led. For the power, we used regular 9V battery. But as the PIC16F877A or any components in the circuit cannot take 9v directly we used initially though to use a voltage regulator to regulate 9v to 5v. Later we found there is power supply kit available to turn 9v to 5v. So, we used a battery connector to connect 9v battery to the power supply kit and get 5v across the rail of the breadboard to power the circuit.

When we simulated the circuit, we didn't face much trouble to build the circuit. But during the hardware implementation we faced few issues. First issue we faced was, in the proteus simulation tool all the pin like Vdd and Vcc or positive and negative pin of the lcd are not given. So, we had to follow the the pin diagram and online material to complete power connection of the component. We connected PIC16F877A to the power and ground from both side through pin 11,12 and 31,32 following the pin diagram of

the PIC16F877A. Though the rest of the component worked, we faced issues with LCD. First it didn't light up. We found that its 15 and 16 pins need to be connected to the power for backlit. Then we connected it with the power. Then there was nothing showing in the display. We connected a 10k potentiometer to the V0 pin of the lcd to control the contrast of the lcd display. There was power in the lcd but nothing was showing up in the display as expected. To find what is wrong, we checked every pin connection, deconstructed and reconstructed the circuit. We checked the soldering of the lcd and every pin connected to the lcd to see if the power and signal was going right. Everything seemed right to us but still the expected result was not showing in the display. As everything else we checked was ok we think the LCD we have is faulty.