MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

V Semester B.Tech. (IT)

EMBEDDED SYSTEMS

LAB PROJECT IMPLEMENTATION DOCUMENT

GROUP 11

TEMPERATURE SENSOR INTERFACING ON LIQUID CRYSTAL DISPLAY (LCD)

submitted by

RAVI SHARMA - 200911268
SHREEYANKA DAS- 200911270

PROBLEM STATEMENT

Interface a temperature sensor to LPC1768 board and write a programme to display the result on the liquid crystal display (LCD).

ABSTRACT

Temperature is the most often-measured environmental quantity. This might be expected since most physical, electronic, chemical, mechanical, and biological systems are affected by temperature. A temperature sensor is an electronic device that measures the temperature of its environment and converts the input data into electronic data to record, monitor, or signal temperature changes. A temperature sensor is a key component of any process heating application as it provides temperature feedback about the process, which can be used to monitor or control the process. Whether the purpose is process maintenance or freeze protection, heat trace is a common process heating application where sensor placement is critical.

HARDWARE USED

- LPC1768 Training Kit:
  The NXP (founded by Philips) LPC1768 is an ARM 32-bit CortexM3 Microcontroller. The code has been written in Embedded C and tested out on an ALS evaluation board. A 12-bit internal Analog-to-Digital Converter has been used to convert the voltage value to its corresponding value in degrees Celsius.

- LM35 temperature sensor:
  It is a 3-terminal sensor used to measure surrounding temperature ranging from -55 °C to 150 °C. LM35 gives temperature output which is more precise than thermistor output. The sensitivity of LM35 is 10 mV/degree Celsius. As temperature increases, output voltage also increases. The output analog voltage can be converted to digital form using ADC so that a microcontroller can process it.

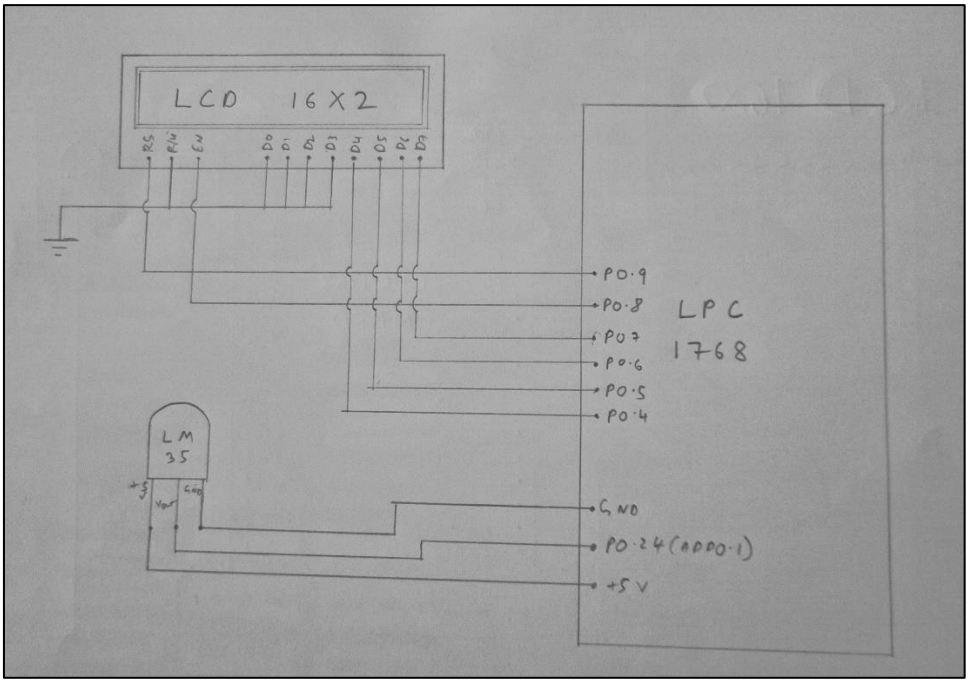
SOFTWARE USED

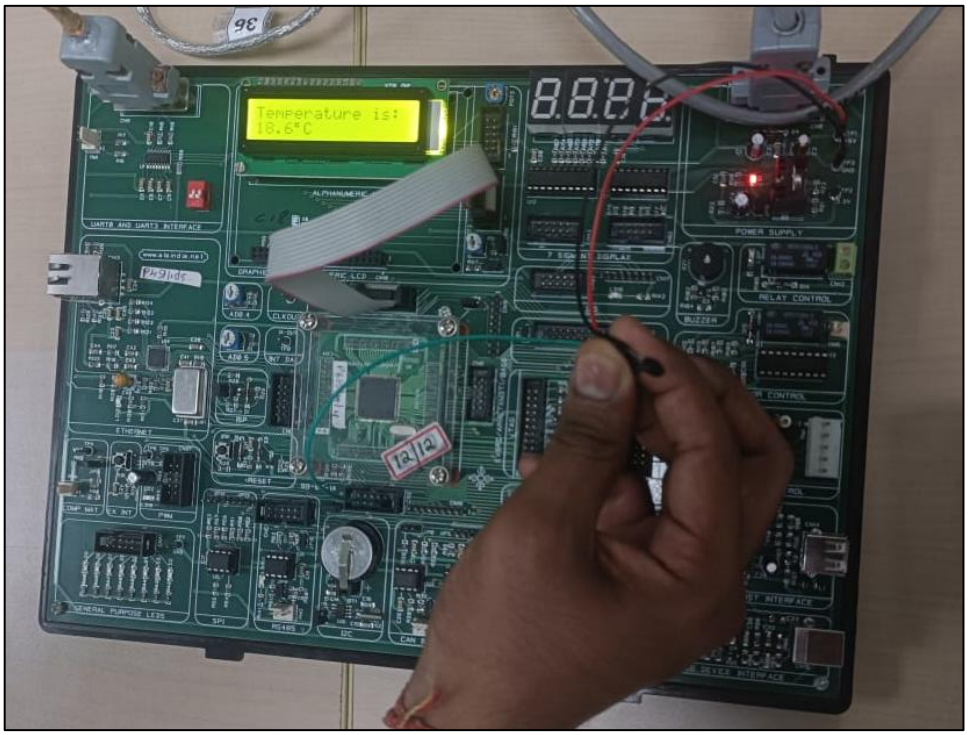
- Keil uVision4:
  Keil MDK is the complete software development environment for a range of Arm Cortex-M based microcontroller devices. MDK includes the µVision IDE and debugger, Arm C/C++ compiler, and
  essential middleware components.

- Flash Magic 7.20:
  Flash Magic is a PC tool for programming flash-based microcontrollers from NXP using a serial or Ethernet protocol while in the target hardware. Straightforward and intuitive user interface, with five simple steps to erasing and programming a device and setting key options.

INTERFACING

Circuit Diagram and connection:





SOURCE CODE

```
#include <LPC17xx.h>

#define RS  8 //P0.9 as Register Select

#define EN  9 //P0.8 as LCD Enable

#define DT  4  //P0.4 to P0.7 as Data Lines

float x, y, temp;

unsigned long int temp1,temp3;

unsigned char flag1 =0, flag2 =0;

unsigned long a, b, temp2, r1, i;

//LCD Initialization commands

unsigned long int init_command[] =
{0x30,0x30,0x30,0x20,0x28,0x0c,0x06,0x01,0x80};

unsigned int digits[] = {0, 0, 0, 0}; // will store the digit to be displayed

void display(void);

//To generate the intended delay

void delay_lcd(unsigned int r1)

{

   unsigned int r;

   for(r=0; r<r1; r++);//Empty loop working as delay
```

```
    return;
}
void port_write()
{
    LPC_GPIO0->FIOPIN = 0;
    LPC_GPIO0->FIOPIN = temp3;//Sending data to the pins
    if (flag1 == 0) //checking for command or data
        LPC_GPIO0->FIOCLR = 1<<RS;
    else
        LPC_GPIO0->FIOSET = 1<<RS;
    LPC_GPIO0->FIOSET = 1<<EN;//Enabling the LCD to write data
    delay_lcd(25);
    LPC_GPIO0->FIOCLR = 1<<EN;
    delay_lcd(1000000); //Generating delay
}
void lcd_init()
{
    LPC_GPIO0->FIODIR = 1<<RS|1<<EN|0XF<<DT;//P0.4-P0.9 as Output
    flag1 =0;
    //Sending commands for Initialization of LCD
```

```
    for (i=0; i<9; i++)
    {
        temp1 = init_command[i];
        flag2 =  (flag1 == 1) ? 0 :((temp1 == 0x30) || (temp1 == 0x20)) ? 1 : 0;
        temp3 = temp1 & 0xf0;
        port_write();
        if (!flag2)//If LCD is in 4-bit mode sending the data accordingly
        {
            temp3 = temp1 & 0x0f;
            temp3 = temp3 << DT;//Shifting the data to put it on the intended pins
            port_write();
        }
    }
}
//To display the initial message
void disp_data()
{
    unsigned char msg[16]="Temperature is:"; //Initial Message
    i=0;
    flag1=1;
```

```
    while(msg[i]!='\0') //Until the message ends
    {
       temp1=msg[i];
       temp3 = temp1 & 0xf0; //Extracting the most significant 4 bits
       port_write();

       temp3 = temp1 & 0x0f;//Extracting the remaining bits
       temp3 = temp3 << DT;
       port_write();
       i++;
    }
    flag1=0; //Command mode for lcd
    temp3=0xc; //Sending the command to put the cursor on the 2nd row
    temp3=temp3<<DT;
    port_write();

    temp3=0;
    temp3=temp3<<DT;
    port_write();
}
```

```
int main(void)
{
    SystemInit();
    SystemCoreClockUpdate();

    lcd_init();
    disp_data();
    LPC_PINCON->PINSEL0 &= 0X0000000; //P0.4 to P0.9 as GPIO
    LPC_PINCON->PINSEL1 |= 1 << 16; //P0.24 as ADC input (ADD0.1)
    LPC_SC->PCONP |= 1 << 12; //Power to the ADC by enabling the 12th pin of
    //PCONP (Power Control for Peripheral)
    LPC_ADC->ADCR = (1 << 1 | 1 << 16 | 1 << 21); //Enable channel 1
(ADD0.1) in burst mode and enable
    //power down(PDN)
    NVIC_EnableIRQ(ADC_IRQn); //Enable the NVIC
    LPC_ADC->ADINTEN = (1 << 1); //Enable interrupt on channel 1 (ADD0.1)
    while (1);
}
```

```
void ADC_IRQHandler()

{

   a = (LPC_ADC->ADSTAT) & 1 << 1; //Check if channel 1's DONE bit is high

   if (a)

   {

      //if DONE bit high, read the data in ADDR1 register (this also clears the DONE bit)

      b = (LPC_ADC->ADDR1);

   }

   b = b & 0xFFFF; //The data is present on 4th to 15th bit

   b >>= 4; //to get the digital value in lower bit positions

   y = ((float)b * (3.3/ 4096)*100); //Conversion of result in the register to temperature in C



   digits[3] = ((int)y / 10); //Extracting 1st digit

   digits[2] = ((int)(y) % 10); //Extracting 2nd digit

   digits[1] = ((int)(y * 10) % 10); //Extracting 1st digit after decimal

   //Display the temperature on the lcd

   display();

}
```

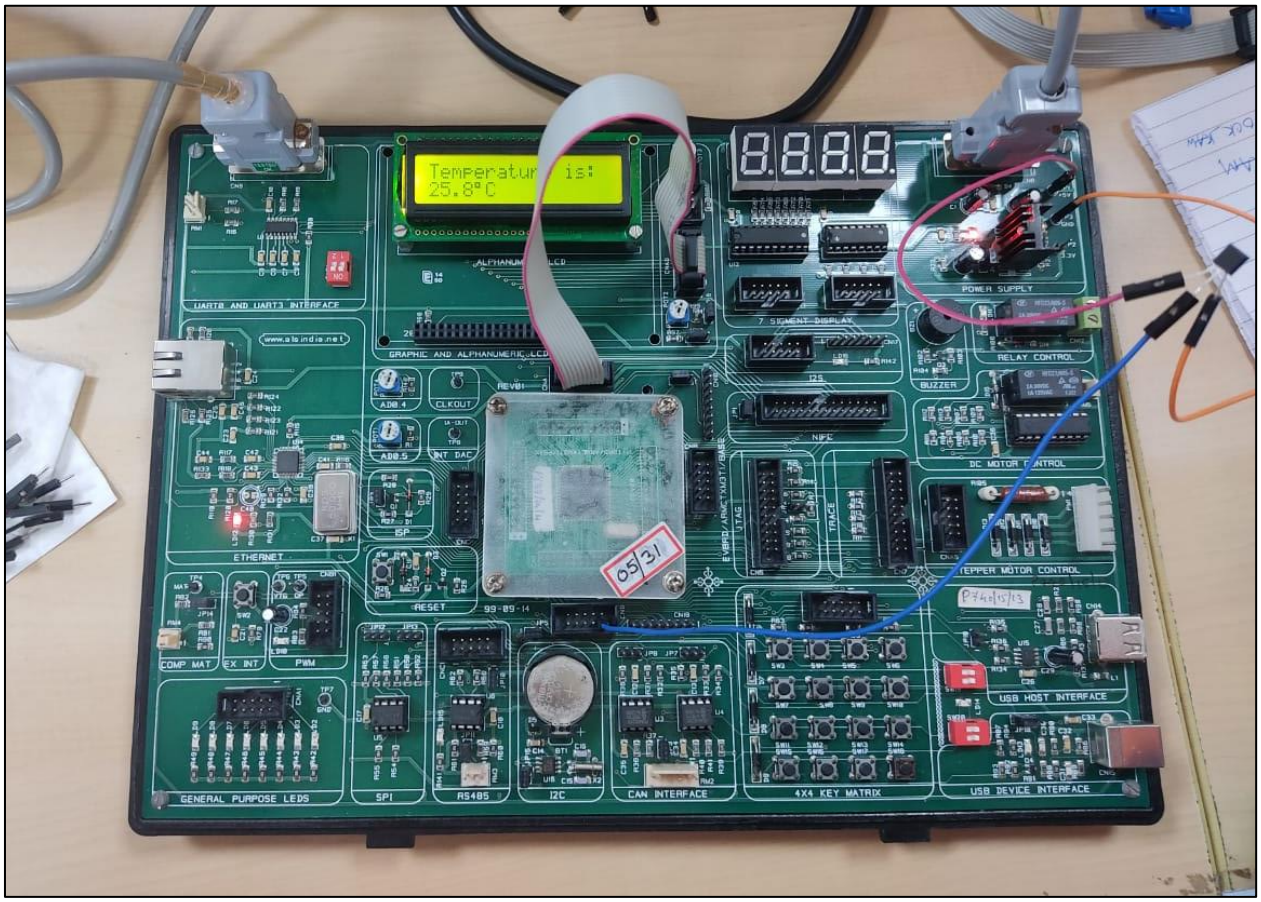
```
//To Display the extracted Temperature
void display(void)
{
   flag1=1;
   temp3=3; //Sending the ASCII value of the digits one by one
   temp3=temp3<<DT;
   port_write();

   temp3=digits[3];
   temp3=temp3<<DT;
   port_write();

   temp3=3;
   temp3=temp3<<DT;
   port_write();

   temp3=digits[2];
   temp3=temp3<<DT;
   port_write();
```

```
temp3=2;//Send the ASCII value of decimal point after 2nd digit
temp3=temp3<<DT;
port_write();

temp3=14;
temp3=temp3<<DT;
port_write();

temp3=3;
temp3=temp3<<DT;
port_write();

temp3=digits[1];
temp3=temp3<<DT;
port_write();

temp3=0xd;//Sending ASCII value of Degree symbol
temp3=temp3<<DT;
port_write();
temp3=0xf; //ASCII of degree symbol is 0XDF
```

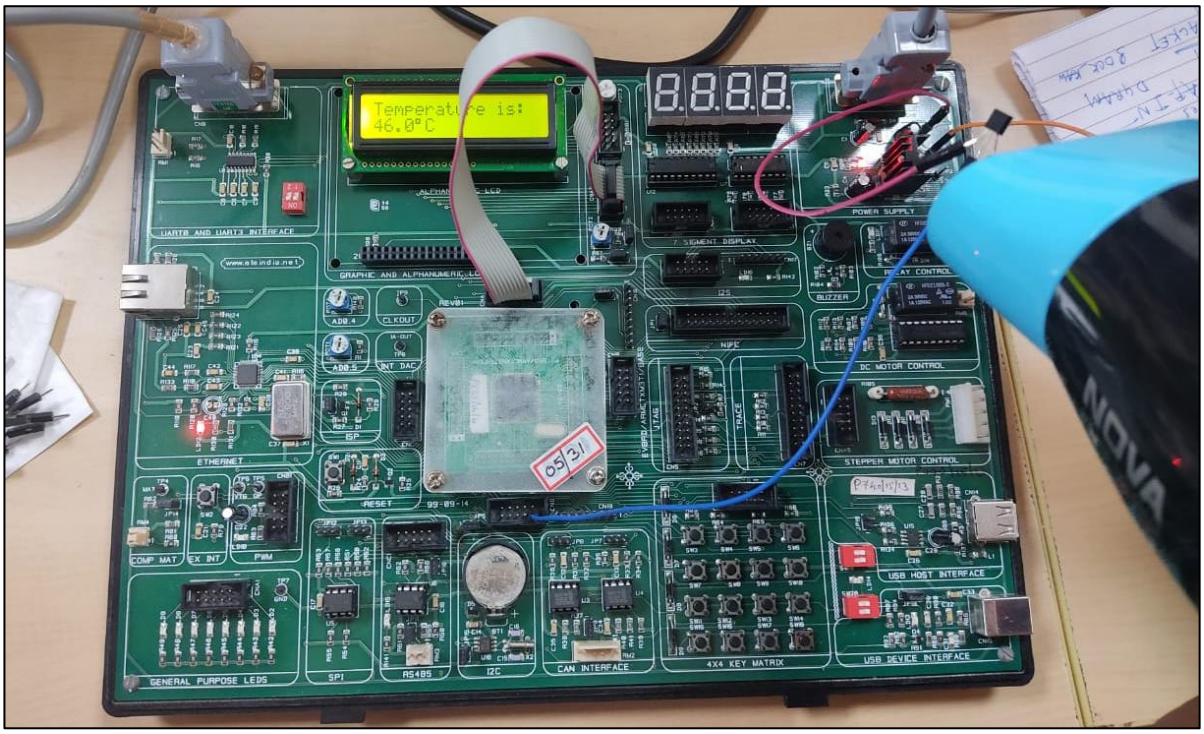
```
    temp3=temp3<<DT;
    port_write();
    temp3=4; //Sending the ASCII of 'C' for Celsius notation
    temp3=temp3<<DT;
    port_write();
    temp3=3;
    temp3=temp3<<DT;
    port_write();
    flag1=0;
    temp3=0xc; //Again sending the command to set cursor at
                        starting of the 2nd row
    temp3=temp3<<DT;
    port_write();
    temp3=0; //This helps to overwrite the new temperature over the old one
    temp3=temp3<<DT;
    port_write();

    delay_lcd(3000000);
}
```

OUTPUT

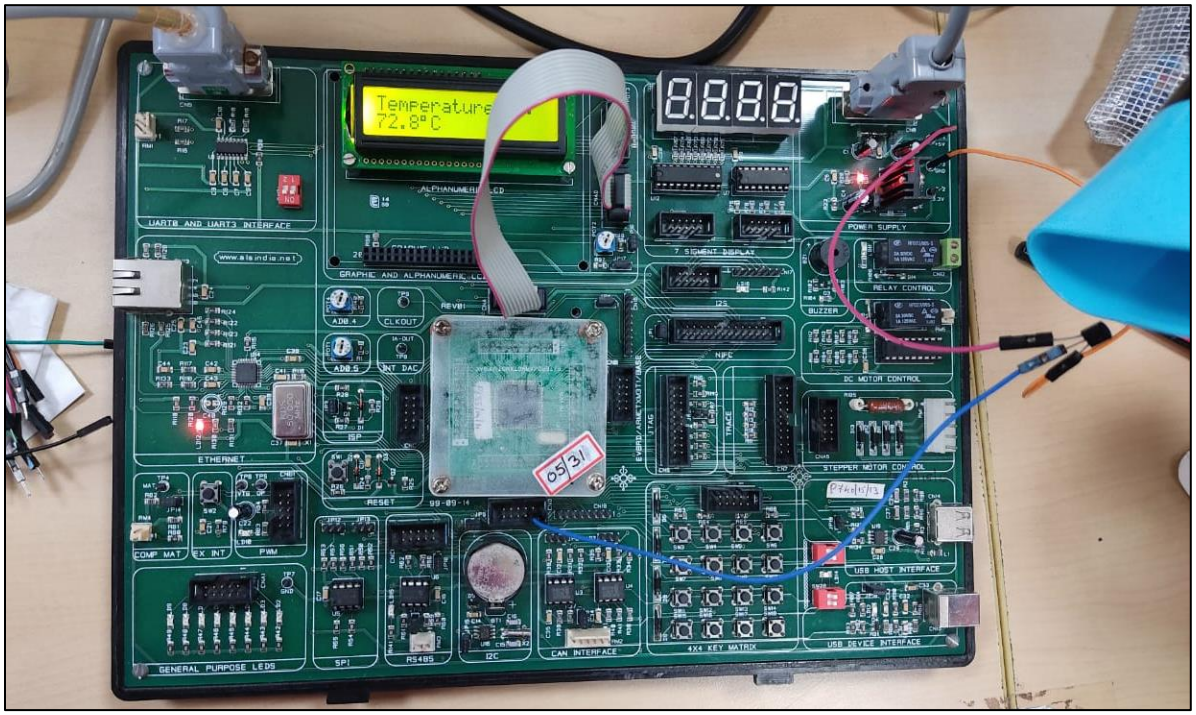
On initial connection, the temperature displayed by the sensor on the LCD was ranging from 24°C to 26°C which was the room temperature at that point.



To check the correctness of the sensor and the code we then blew hot air on the sensor using a hairdryer. The experiment resulted in gradual rise in temperature that could be seen on the LCD display clearly.



As it is visible that the temperature got high and gave a reading of 46.0°C. On further blow of air on the sensor the temperature kept on increasing and it gave a reading as high as 72.8°C.



CONCLUSION

The LPC1768 Training Kit project was created as a part of our ES LAB PROJECT Vth sem. The objective of building this project was to check the connectivity and effectiveness of the LM35 (Temperature sensor) with the LPC1768 Training Kit and its connection with the Liquid Crystal Display (LCD) to display the sensed temperature accurately. In this project we learnt to write an effective code for the LCD configuration of LPC 1768 and understood the functioning and working of both the LCD and the sensor. We got an opportunity to learn the basic concepts of Embedded C programming and voltage manipulating. During this project, I also learnt the implementation of the temperature sensor and its working and also got acquainted with the working of the Analog to Digital Convertor (ADC) module of the Kit. Developing a working code for the LPC1768 kit made me understand the functioning of the components more effectively.