

# **Application Note**

## **LM34 Temperature Sensor Implementation with PIC**

Muhammad Jawad Zaheer  
November 13th, 2009

### **Executive Summary**

This paper discusses the various applications of temperature sensors and their implementation in our project. Many consumers use digital thermometers every day and due to their usefulness they have become cheap and readily available. In our project, temperature sensors are used to detect the ambient temperature. Precision Fahrenheit Temperature Sensor LM34, created by National Semiconductor, with the PIC18F4520 microprocessor provided by Microchip, is used to implement this device. Physical connection between the two devices can be accomplished using a simple breadboard. Different setup of hardware and software implementation will be discussed, which is required by the system to read the desired temperatures.

### **Keywords**

Temperature Sensor, LM34, PIC18F4520 Microprocessor, LCD.

## Table of Contents

<b>1. Objective.....</b>	<b>3</b>
<b>2. Introduction .....</b>	<b>3</b>
<b>3. Implementation.....</b>	<b>4</b>
3.1 Hardware Interface.....	4
3.2 Software Interface.....	8
<b>4. Conclusion.....</b>	<b>8</b>

## 1. Objective

Our group is designing a body temperature regulation jacket for quadriplegic patients. The need for this interface is to measure the ambient temperature. The system, consisting of a Peltier junction, does heating and cooling based on the ambient temperature. The actual ambient temperature will also be sent to the LCD screen for the user to know the surrounding temperature. The description of Peltier junction and LCD are not discussed in this application note.

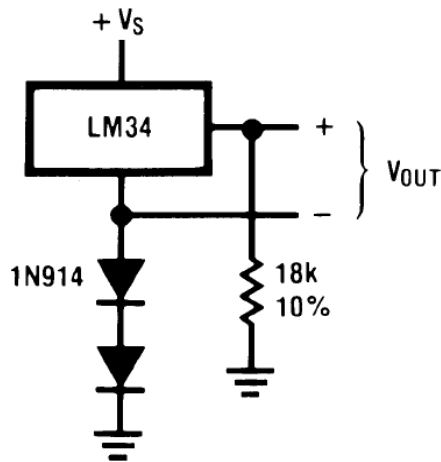
## 2. Introduction

There are many commonly used electrical temperature sensors that can be consider for this system but they are mostly difficult to apply. Thermistors are nonlinear to any temperature scale and needs calibration while thermocouples have low output levels and require cold junction compensation. They also requires extra hardware interface that can change this resistance value into a voltage signal to transmit the signal.

The National Semiconductor designed the precision integrated-circuit temperature sensors, LM34, to overcome these limitations. The output voltage of LM34 is linearly proportional to the Fahrenheit temperature and does not require any external calibration. It outputs approximately  $+10\text{mV}/^{\circ}\text{F}$  over its wide operating temperature range of  $-50^{\circ}\text{F}$  to  $+300^{\circ}\text{F}$ . As it is calibrated in Fahrenheit temperature compared to other linear sensors calibrated in degrees Celsius, the user is not required to do extra calculations to obtain convenient Fahrenheit scaling. The device can be used with

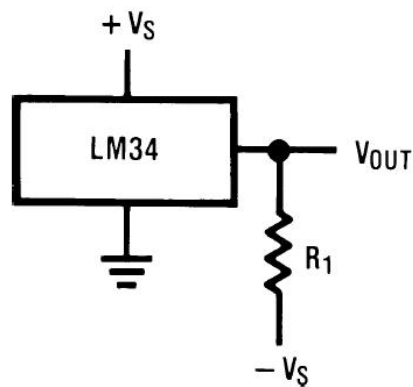


Using single supply, another resistor is added from the output pin to ground, connecting two diodes in series between the GND pin and the circuit ground, and taking a differential reading. The diodes are used to draw the essential current required for negative temperatures.



**Figure 2: Temperature Sensor, Single Supply,  $-50^{\circ}$  to  $+300^{\circ}\text{F}$**

If dual supplies are available, as in our project, the sensor is used over its full temperature range by just adding a pull down resistor from the output to the negative supply as shown in **Figure 3**.

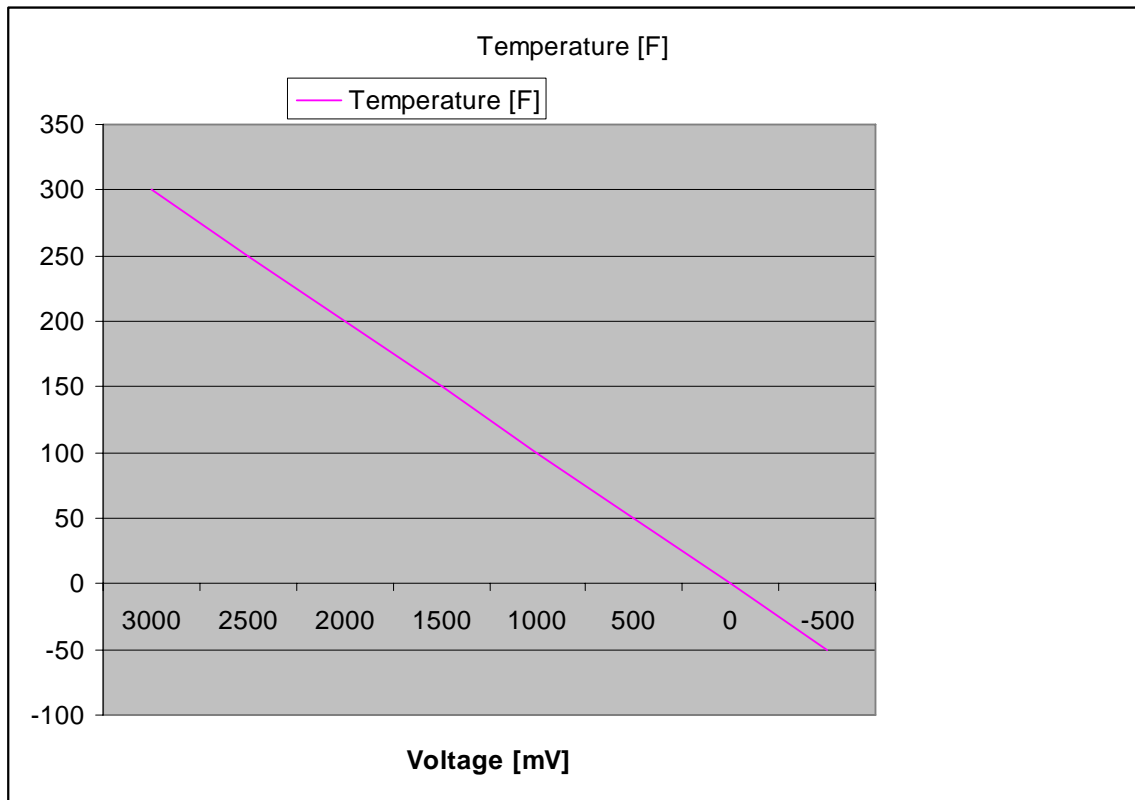


**Figure 3: Full-Range Fahrenheit Temperature Sensor, Dual Supply**

The value of this resistor is calculated using following formula

$$R_1 = \frac{(-V_s)}{50\mu A}$$

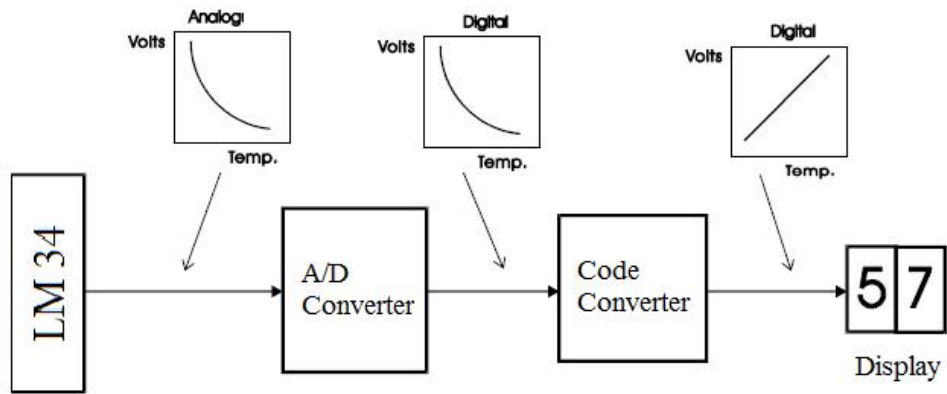
While using the Voltage source of 5V, it requires 100kΩ resistor to provide the linear scale. Using these values, our actual temperatures based on output voltages is shown in the following chart, which shows excellent linearity:



**Figure 4: Linear Relationship between Voltage and Temperature**

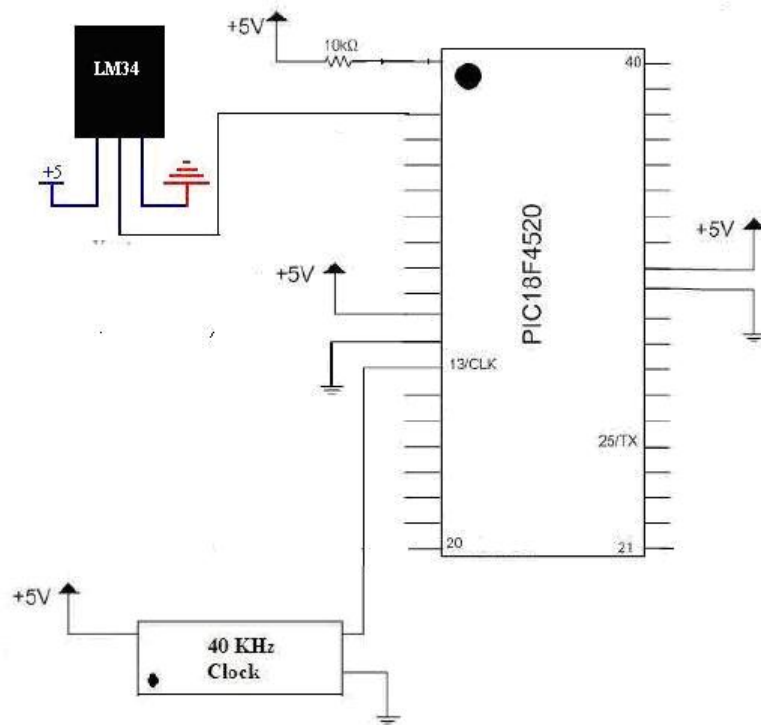
The output of the sensor is then connected to Pin 2 of the PIC microprocessor. This reading is sent to a built-in Analog to Digital converter on the PIC which stores the value in HEX. The output is then used to determine which mode of Peltier junction (heating or cooling) should start. The temperature is checked every minute to account for any changes. The timing of the circuitry is controlled by an external clock which is

set to 400 MHz. The output of the sensor reading is also output at the LCD screen using the ports D of the PIC. The logic follow starting from LM 34 till the display is shown below.



**Figure 5: Logical flow from LM34 to Display**

The complete sensor connections are shown in figure below.



**Figure 6: Wiring diagram of Temperature sensor with Microprocessor and Clock**

### ***3.2 Software Interface***

A C++ program is written using MPLAB. The following code is used to program the microprocessor which controls the I/O lines and measure the temperature. The program looks whether the system is in automatic or manual mode. Once it's in automatic mode, it checks the analog value coming in from the temperature sensor to Pin 2 of the PIC. It converts the value in HEX. Then it looks whether the surrounding is cold or hot. Once the state is known, the PIC will output a signal with variable PWM to perform heating or cooling. The software code is attached at the appendix of the application note. It also checks for the temperature reading every minute to counter the change in temperature.

## **4. Conclusion**

In this application note, the concept and usage of temperature sensors is discussed. As can be seen, the LM34 is user-friendly temperature sensors with excellent linearity. It gives a basic idea about the uses of temperature sensors, hardware and software implementation.. The result of this hardware configuration allows for a relatively simple and cheap method of collecting temperature data. Once the data is gathered and stored in the register of the microcontroller, many things can be done. For example, it can be written to an LCD. This design is chosen due to the fact it fulfills multiple needs for our project and with some modification could be used for different projects.

## References

**LM34 Data Sheet, National Semiconductors**

<http://www.national.com/ds/LM/LM34.pdf>

**PIC18F4520 Data Sheet, Microchip**

<http://ww1.microchip.com/downloads/en/DeviceDoc/39631B.pdf>

## APPENDIX

```
#include <p18cxxx.h>
#include <ADC.h>
#pragma config WDT=OFF

// Constant variables //////////////////////////////////////
const int coldAmbient = 0x0114; // Cold ambient temp => 0.7V
const int hotAmbient = 0x013A; // Hot ambient temp => 0.8V
////////////////////////////////////

void AutomaticControl();
void Manual();

void main()
{
    while(1)
    {
        if(1) // Check if connected to automatic
        {
            AutomaticControl();
        }
        else // Manual
        {
            Manual();
        }
    }
}

void AutomaticControl()
{
    long int loop, loop2, time;
    int adc_result[5];
    int adc_result2;

    TRISD = 0x04;

    PORTDbits.RD0 = 0; // RED
    PORTDbits.RD1 = 0; // GREEN
    PORTDbits.RD3 = 0; // YELLOW

    OpenADC(ADC_FOSC_32 & ADC_RIGHT_JUST & ADC_12_TAD,
ADC_CH0 & ADC_INT_OFF, 0); //open adc port for reading

    ADCON1 = 0x10; //set VREF+ to CH3 and VREF- to GND (VSS)
```

```

SetChanADC(ADC_CH0); // Setting ADC value at Pin 2
ConvertADC(); //perform ADC conversion
while(BusyADC()); //wait for result
adc_result2 = ReadADC(); //get ADC result

while(1)
{
    PORTDbits.RD0 = 0; // RED
    PORTDbits.RD1 = 0; // GREEN
    PORTDbits.RD3 = 0; // YELLOW

    if(adc_result2 <= coldAmbient) // Cold... 0.70V
    {
        for(loop2=0; loop2<numOfLoops; loop2++)
        {
            for(time=0; time<pwmPeriod; time++)
            {
                if (time < highDutyCycle)
                {
                    for(loop=1; loop<50000; loop++);
                    PORTDbits.RD3 = 1;
                }
                if (time >= highDutyCycle)
                {
                    for(loop=1; loop<50000; loop++);
                    PORTDbits.RD3 = 0;
                }
            }

            ConvertADC(); //perform ADC conversion
            while(BusyADC()); //wait for result
            adc_result[loop2] = ReadADC(); //get ADC result
        }
    }
    else if(adc_result2 < hotAmbient && adc_result2 > coldAmbient)
// Middle.. b/w 0.80 n 0.7V
    {
        for(loop2=0; loop2<numOfLoops; loop2++) // TODO:
        SHOULD DO NOTHING HERE - maybe Idle
        {
            for(time=0; time<pwmPeriod; time++)
            {
                if (time < highDutyCycle)
                {
                    for(loop=1; loop<50000; loop++);
                    //PORTDbits.RD3 = 1;
                }
            }
        }
    }
}

```

```

        }
        if (time >= highDutyCycle)
        {
            for(loop=1; loop<50000; loop++);
            //PORTDbits.RD3 = 0;
        }
    }

    ConvertADC(); //perform ADC conversion
    while(BusyADC()); //wait for result
    adc_result[loop2] = ReadADC(); //get ADC result
}
}
else if (adc_result2 >= hotAmbient) // Hot ...> 0.80V
{
    for(loop2=0; loop2<numOfLoops; loop2++)
    {
        for(time=0; time<pwmPeriod; time++)
        {
            if (time < highDutyCycle)
            {
                for(loop=1; loop<50000; loop++);
                PORTDbits.RD0 = 1;
            }
            if (time >= highDutyCycle)
            {
                for(loop=1; loop<50000; loop++);
                PORTDbits.RD0 = 0;
            }
        }

        ConvertADC(); //perform ADC conversion
        while(BusyADC()); //wait for result
        adc_result[loop2] = ReadADC(); //get ADC result
    }

    adc_result2 = (adc_result[0] + adc_result[1]+
adc_result[2]+ adc_result[3]+ adc_result[4])/5;
}
}
}

```