EE 347

Microcontroller System Design

Project 1 Report

# The Tea Kettle Temperature Control Project

Written by: Nathan Genetzky and Jerome Charles

Due date: March 23, 2015

Instructor: Dr. Robert Fourney

## Introduction

The aim of this project was to use an MCP7900 thermistor and a PIC18F8722 microcontroller to control the temperature of a tea-kettle using a “bang-bang” approach. What this means is that if the kettle went over a desired temperature the system will turn it off, and if the kettle temperature goes under the desired temperature, the system turns it back on until it raises to the desired temperature.

This project was coded using both the PIC assembly and C programming languages. The C program was more intricate and required more lines of code. This was because the user in C could enter the temperature in degrees Celsius; whereas in the Assembly program, the user was restricted to three temperature settings that would be detected  by using the keypad.

## Procedure and Theory

As part of the preparation for the project, Dr. Fourney had assigned several labs that included code that would prove useful for this project. During these labs the 4x4 keypad, the LCD lab, and the A/D converter were used. The thermistor voltage for several different temperatures was observed in order to determine a close relation between temperature and voltage. Code was extracted from previous programs that used the keypad, LCD, and ADC; the challenge was then bring all the code together to ensure they worked correctly.

The Linear Active Thermistor™ Integrated Circuit, MCP9700A, will be referred to as simply the thermistor. The thermistor accuracy is +/- 1 degree Celsius according to the data sheet. This could be improved by performing calibration at +25$°C$. The temperature can be found using Equation 2 which is derived from equation 1. The variables are described as follows: $T_a$ ambient Temperature, $V_{out}$ Sensor Output Voltage, $V_{0°C}$ Sensor Output Voltage at $0°C$, $T_C$ Temperature Coefficient. According to the data sheet the following values can be expected: $T_C = 10\frac{mV}{°C}$ & $V_{0°C} = 500\ mV$.

$$V_{out} = T_C T_a + V_{0°C} \qquad \text{Equation 1}$$

$$V_A = \frac{V_{out} - V_{0°C}}{T_C} \qquad \text{Equation 2}$$

The assembly code was required more time to write because the language was closer to the hardware. The assembly code was programmed second since there was an increased probability that there was an error in the code. The programs were different enough that the logic could not simply be ported over from C to assembly.

Our flowchart for the C program is shown in Figure 1. The definition of HighTemp and LowTemp varied between C and ASM. In C the temperature variable is defined in degrees Celsius, however in ASM it is defined in A2D digital output. Any box involving LCD should be disregarded for the ASM flowchart but otherwise the logical flow is the same between C and ASM.

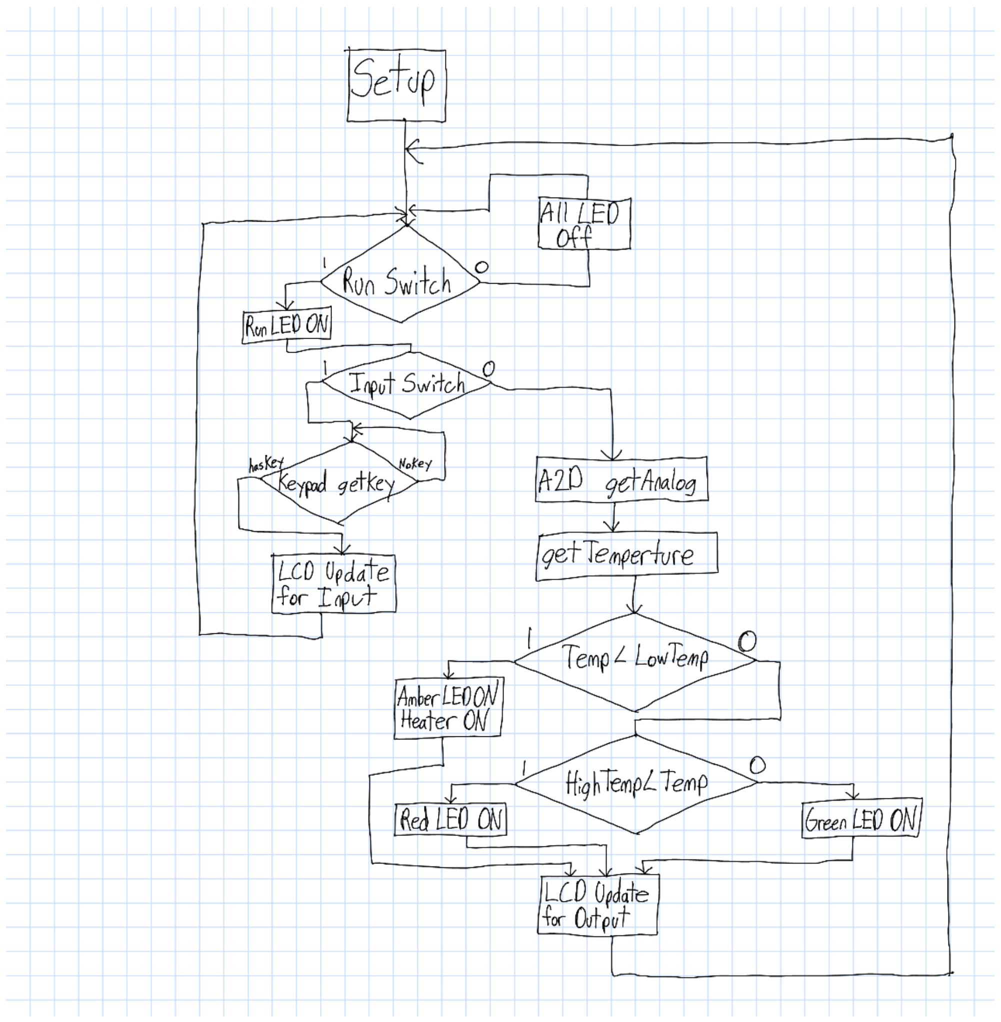

*Figure 1 Program Flowchart*

## Results and Analysis

The most common issues were introduced by hardware faults. Initially there were some issues with boards not having a proper working LCD display. Sometimes the PICs boards were not being recognized by the software when trying to download the program to the board. This was often resolved by trying a different board; in the future an effort was made to use the same board whenever working on the project. The Keypad did not work initially after it was combined with the LCD in the system because they both were trying to use the same pin.

There were times when noise in the input switch signal would cause the system to exit the feedback loop and wait for user input. In order to fix this interrupts were used for user input and redesign the algorithm to not wait for user input but instead just obtain it if it is present. Entry on the keypad would sometimes be registered as multiple entries. This could be fixed by increasing the time delay in the keypad function.

The C code was very similar, but the user was able to enter the temperature in degrees Celsius, meaning that more temperatures must be supported than in the assembly code. In order to provide this functionality, a function was created that determined the temperature based on the eight more significant bits of the ADC result. The temperature is then displayed on the LCD screen and is used for the feedback control of the teakettle. The function used to calculate the temperature was obtained from the thermistor and is shown in equation.

Unlike the C program, which calculated the temperature based on output of the ADC, the assembly program looks directly at the ADC output for the feedback regulation. This approach was chosen because calculations in ASM are much more difficult. This portion of the project required that the expected eight-bit result of ADC for each of the six temperatures required for feedback control were coded into the system. The current eight-bit result of ADC would then be directly compared to the values coded for the temperatures to determine the output of the system.

*Table 1 Calibration literal defines used in ASM project*

| Temperature | Values |
|---|---|
| TEMP 25 | 116 |
| TEMP 55 | 165 |
| TEMP 78 | 205 |
| TEMP 80 | 212 |
| TEMP 85 | 213 |
| TEMP 92 | 223 |
| TEMP 95 | 234 |

The values that were coded into the system for the 6 temperatures are shown in Table 3 below. This was referenced during the demo to verify the program executes correctly.

*Table 2 Temperatures and Decimal and Binary Inputs*

| Temperature | Decimal | Binary |
|---|---|---|
| Full boil High | D240 | 11110000 |
| Full boil Low | D238 | 11101110 |
| Simmer High | D236 | 11101100 |
| Simmer Low | D235 | 11101011 |
| Soup High | D215 | 11010111 |
| Soup Low | D232 | 11101000 |

After finishing the C and ASM versions of the project, it was seen that there was significant error each time the system was used with a new thermistor. A lot of time was then spent recalibrating the teakettle regulation system in order to get the actual and measured temperature as close to each other as possible. For C this was done by changing the parameters of the function used to calculate temperature. The value 9.93 was used for $T_a$ and the value 529 was used for $T_{0°C}$. For ASM this was done by changing expected ADC output for the six temperatures coded into the system

The response of the thermistors were very inconsistent. It was important to use the same thermistor throughout the project so there were fewer adjustments to the project; however it was found that even using the same thermistor did not guarantee consistent output. It was realized that some thermistors were MCP9700 while some were MCP9701. This caused issues because the graphs for the output voltage are quite different. Values had to be constantly tweaked to obtain smaller error in temperatures, between the set temperature and the actual temperature of the teakettle.

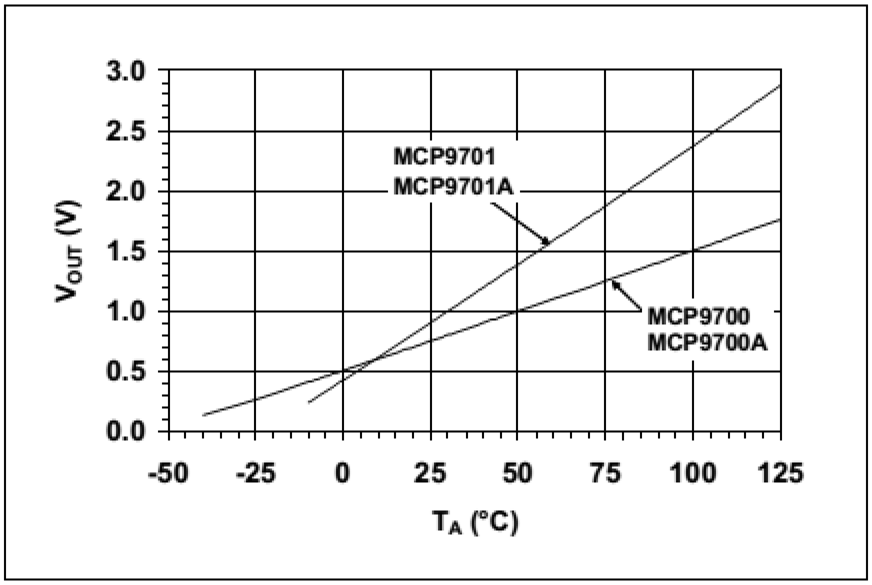

*Figure 2 Thermistor Output Voltage vs Ambient Temperature*

Results were quite satisfactory for the performances of both the C version and the ASM. Both turned the tea kettle on and heated it until it reached the user-entered desired temperature, then switched the kettle off again until the temperature again dropped below the desired temperature. The LCD properly displayed the target temperatures and users were able to monitor the actual temperature on the digital thermometers from lab.

In order to increase the resolution of the measurement the voltage reference capability of the A2D was implemented. A voltage divider was created using a 1.1 $K\Omega$ and 2.2$K\Omega$ resistor to create a voltage of 1.6V on the Vref+ pin (AN3). This decreased the range that the A2D expects from [0:5]V to [0:1.6]V which increases the resolution.

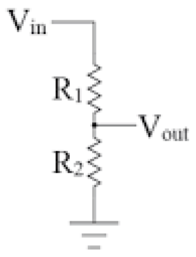

*Figure 3 Voltage divider for A2D. Vout is RA3 and Vin is +5V.*

By increasing the resolution of the measurement the system could measure accurately within $\pm 0.5°C$ over the critical range, $70°C$ to $95°C$, when properly calibrated. This behavior was only observed after the system has been fine-tuned by modifying the parameters used to calculate the temperature. Due to nonlinear behavior of the thermistor the temperature error increased as the measured temperature gets farther from the temperature the system was calibrated at.

## Conclusions

This project gave students the opportunity to learn more about and become more familiar with the PIC microcontroller. The Keypad code, LCD code and the A to D code from previous laboratories and projects in this class were integrated with the tea kettle, thermistor and switches to regulate temperatures of the kettle. The LCD, keypad and A to D code were used to display the inputted temperatures that were set manually .Once there were usable values after calibration of the thermistor e able to regulate the temperatures were able to be regulated according to the specs given by the project.

Overall it was good to see an error usually under $1.5°C$ in our programs, especially considering that when the thermistor was first calibrated, there was originally about 3°C of error from the actual temperature. Execution of the project was by no means perfect however our errors were negligible in the end. The overall functionality of the project worked well and to our expectations. The error for the temperature was a bit further larger for our ASM than for our C. Mainly because calculations could not be performed in ASM and tuning the output response of the ASM system was more work. Also ASM is less detailed than C so error detection was a bit more difficult for the ASM. The debugger came in very handy at times to see if values were being properly updated.

### Appendix 1. Error Analysis

This section is intended to analyze the measurement error of the Tea Kettle temperature regulation system. The parameters to be varied to minimize measurement error are $T_a$ and $V_{0°C}$. The table above shows data obtained from observing the PIC micro controller and two thermometers in the first three columns. Columns to the right of this show the calculated temperature and percent error in pairs. The temperature is calculated by the A2D by using Equation 2 with the parameters specified at the top of the column.

When the system was demo'd the parameters $T_a = 9.93$ and $V_{0°C} = 529$ were used with success. The system was able to regulate the temperatures to within a few degrees accurately; however this set of data tells a much different story. The average percent error for temperatures above $70°C$ was 7.1%, which would imply that the micro controller could be as much as $6°C$ off at $90°C$. Although it may seem like it this does not indicate a major fault in the programming, but rather that the values must be modified to reflect the system better.

With this particular set of data it was found that by modifying the parameters to $T_a = 10.2$ and $V_{0°C} = 570$ the average percent error for temperatures above $70°C$ could be reduced to 1.26%. The system would likely have increased accuracy if this change was implement; however there are a few things to consider about this experiment that decrease the value of these parameters.

The largest flaw in the way the experiment was performed was that data was collected as the tea kettle was warming up. In order to obtain proper measurements the water should have time to steady out at a temperature before a reading is taken. Another flaw is that the thermometers sensors are not perfectly accurate; two of the thermometers were different by $2°C$.

*Table 3 PIC Temps with error percents for select parameters listed against Average Temp from Thermometer*

| **Thermometer** | **Data Sheet** | | **Demo** | | **Optimum** | |
|---|---|---|---|---|---|---|
| **Voc=>** | 500 | | 529 | | 570 | |
| **Ta=>** | 10 | | 9.93 | | 10.2 | |
| **Avg Temp** | Temps | % Error | Temps | % Error | Temps | %Error |
| **25.75** | 33.7117647 | 30.91947459 | 31.02896748 | 20.5008446 | 26.18800461 | 1.700988791 |
| **28.45** | 34.9705882 | 22.91946656 | 32.29666489 | 13.5207905 | 27.42214533 | 3.612845945 |
| **31.4** | 38.1176471 | 21.39378044 | 35.46590842 | 12.9487529 | 30.50749712 | 2.842365871 |
| **35.45** | 41.2647059 | 16.40255538 | 38.63515195 | 8.98491381 | 33.5928489 | 5.238790115 |
| **38.15** | 44.4117647 | 16.41353789 | 41.80439547 | 9.57901828 | 36.67820069 | 3.857927413 |
| **41.7** | 47.5588235 | 14.04993652 | 44.973639 | 7.85045324 | 39.76355248 | 4.643759041 |
| **47.45** | 50.7058824 | 6.861712019 | 48.14288253 | 1.46023716 | 42.84890427 | 9.69672441 |
| **50.05** | 53.8529412 | 7.598284069 | 51.31212606 | 2.52173039 | 45.93425606 | 8.223264625 |
| **52.15** | 57 | 9.300095877 | 54.48136959 | 4.47050736 | 49.01960784 | 6.002669524 |
| **56.75** | 60.1470588 | 5.986006737 | 57.65061312 | 1.58698346 | 52.10495963 | 8.185093161 |
| **59.1** | 63.2941176 | 7.096645765 | 60.81985664 | 2.91007892 | 55.19031142 | 6.61537831 |
| **62.65** | 66.4411765 | 6.051359091 | 63.98910017 | 2.13743044 | 58.27566321 | 6.982181634 |
| **66.4** | 69.5882353 | 4.801559178 | 67.1583437 | 1.14208389 | 61.36101499 | 7.58883284 |
| **60.4** | 72.7352941 | 20.42267238 | 70.32758723 | 16.4364027 | 64.44636678 | 6.699282752 |
| **63.7** | 72.7352941 | 14.18413519 | 70.32758723 | 10.4043756 | 64.44636678 | 1.171690396 |
| **65.9** | 75.8823529 | 15.14772829 | 73.49683076 | 11.527816 | 67.53171857 | 2.476052458 |
| **66.9** | 75.8823529 | 13.42653653 | 73.49683076 | 9.86073357 | 67.53171857 | 0.9442729 |
| **68.7** | 79.0294118 | 15.03553386 | 76.66607428 | 11.5954502 | 70.61707036 | 2.790495426 |
| **73.1** | 82.1764706 | 12.41651243 | 79.83531781 | 9.21384106 | 73.70242215 | 0.824106902 |
| **77.65** | 85.3235294 | 9.882201432 | 83.00456134 | 6.89576477 | 76.78777393 | 1.110400601 |
| **78.65** | 87.2117647 | 10.88590554 | 84.90610746 | 7.95436422 | 78.63898501 | 0.014005078 |
| **80.6** | 88.4705882 | 9.764997811 | 86.17380487 | 6.91539066 | 79.87312572 | 0.90182913 |
| **77.65** | 88.4705882 | 13.93507822 | 86.17380487 | 10.9772117 | 79.87312572 | 2.863008011 |
| **80.35** | 89.7294118 | 11.67319448 | 87.44150228 | 8.82576513 | 81.10726644 | 0.942459783 |
| **84.05** | 92.8764706 | 10.50145222 | 90.61074581 | 7.80576539 | 84.19261822 | 0.169682598 |
| **85.8** | 92.8764706 | 8.247634718 | 90.61074581 | 5.60692985 | 84.19261822 | 1.873405334 |
| **88.05** | 94.7647059 | 7.626014631 | 92.51229193 | 5.06790679 | 86.0438293 | 2.278444865 |
| **86.05** | 94.7647059 | 10.12749086 | 92.51229193 | 7.5099267 | 86.0438293 | 0.007171067 |
| **90.25** | 96.6529412 | 7.094671664 | 94.41383804 | 4.61367096 | 87.89504037 | 2.609373552 |
| **92.05** | 98.5411765 | 7.051794102 | 96.31538416 | 4.63376878 | 89.74625144 | 2.502714349 |
| **91.45** | 99.8 | 9.130672499 | 97.58308157 | 6.70648614 | 90.98039216 | 0.513513224 |
| **93.65** | 101.058824 | 7.911183694 | 98.85077898 | 5.55342123 | 92.21453287 | 1.532799923 |
| **95.05** | 102.947059 | 8.308320698 | 100.7523251 | 5.99928995 | 94.06574394 | 1.035513998 |
| **94.3** | 103.576471 | 9.837190444 | 101.3861738 | 7.51450032 | 94.6828143 | 0.405953661 |
| **96** | 106.723529 | 11.17034314 | 104.5554173 | 8.91189306 | 97.76816609 | 1.841839677 |

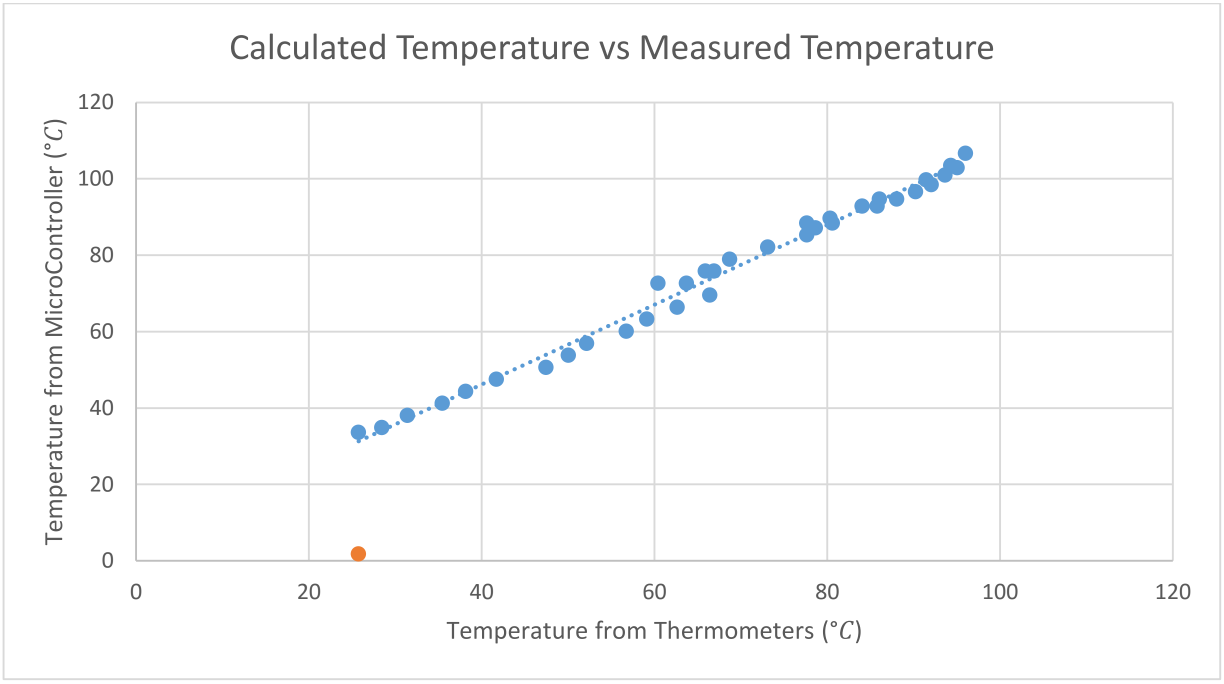

*Figure 4 PIC Temps using Parameters from Datasheet vs Average Temps*

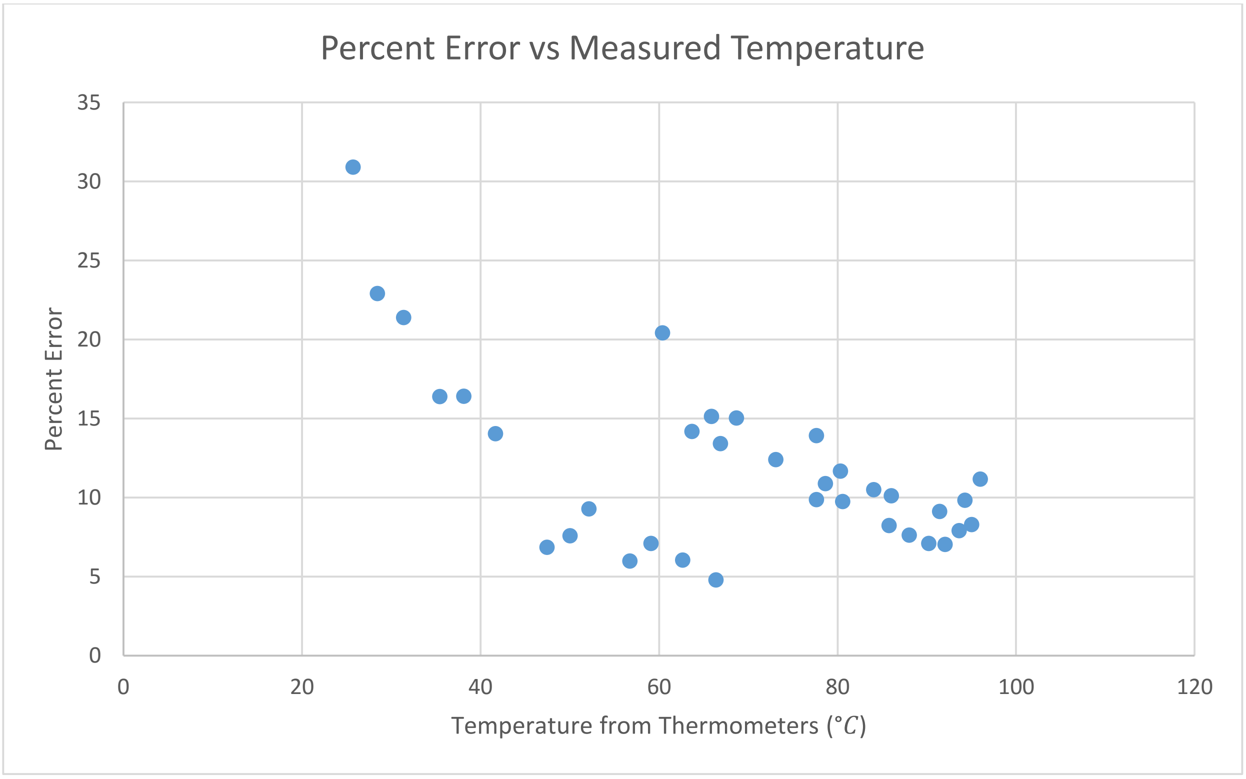

*Figure 5 PIC Temps using Parameters from Datasheet and the Percent Error*

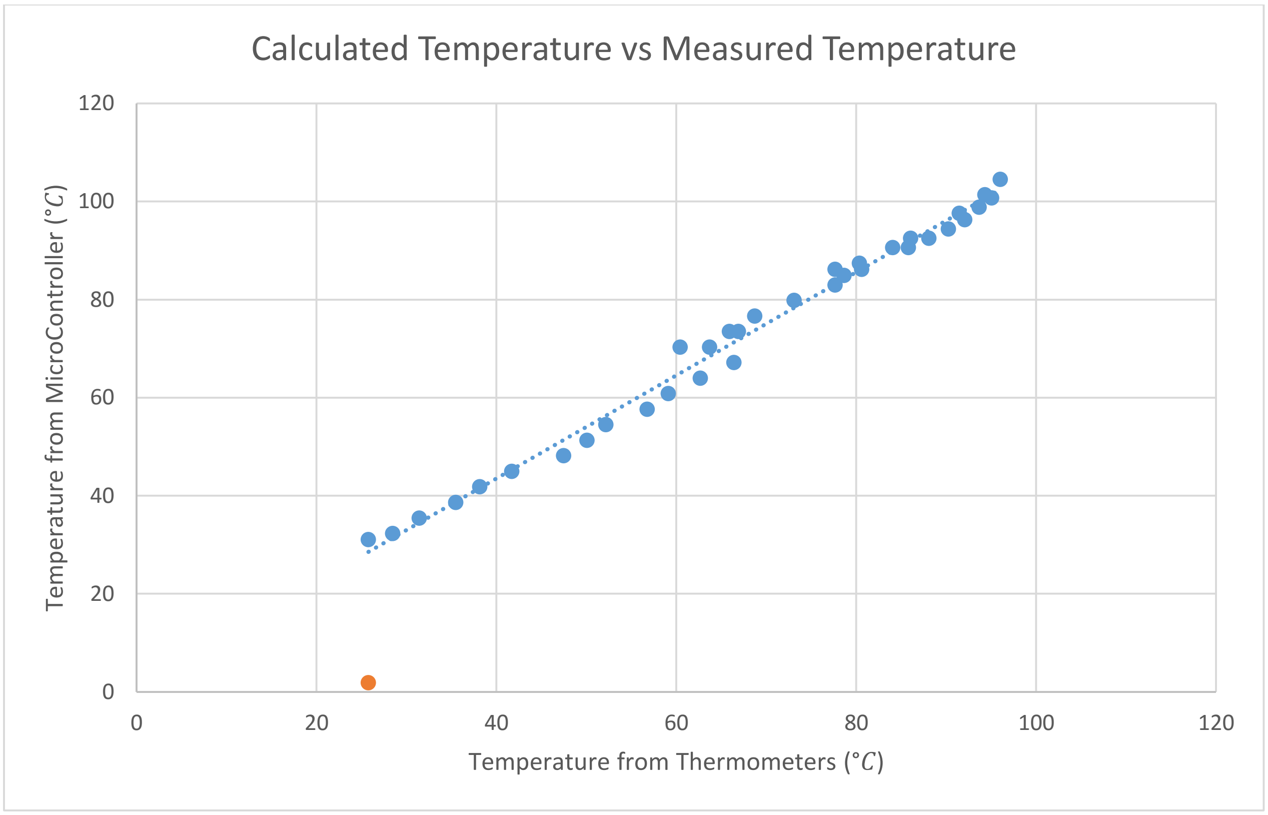

*Figure 6 PIC Temps using Parameters from Demo vs Average Temps*

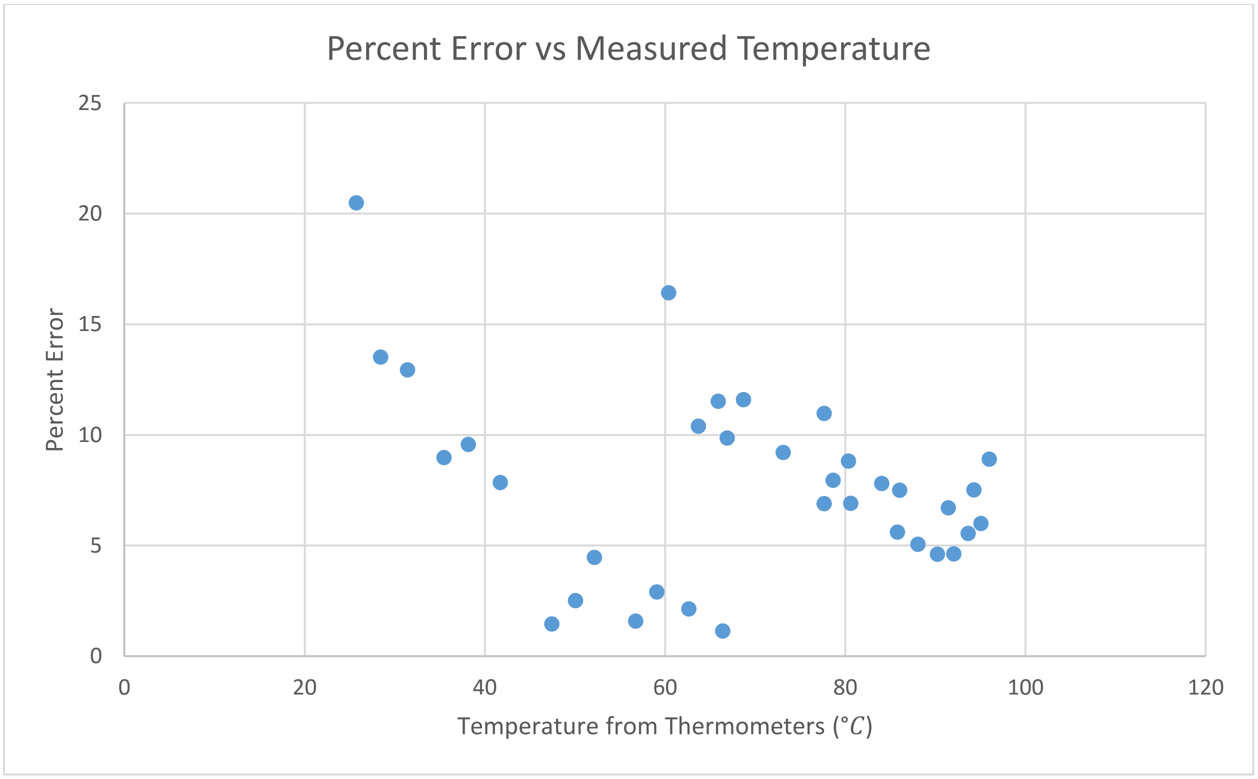

*Figure 7 PIC Temps using Parameters from Demo and the Percent Error*

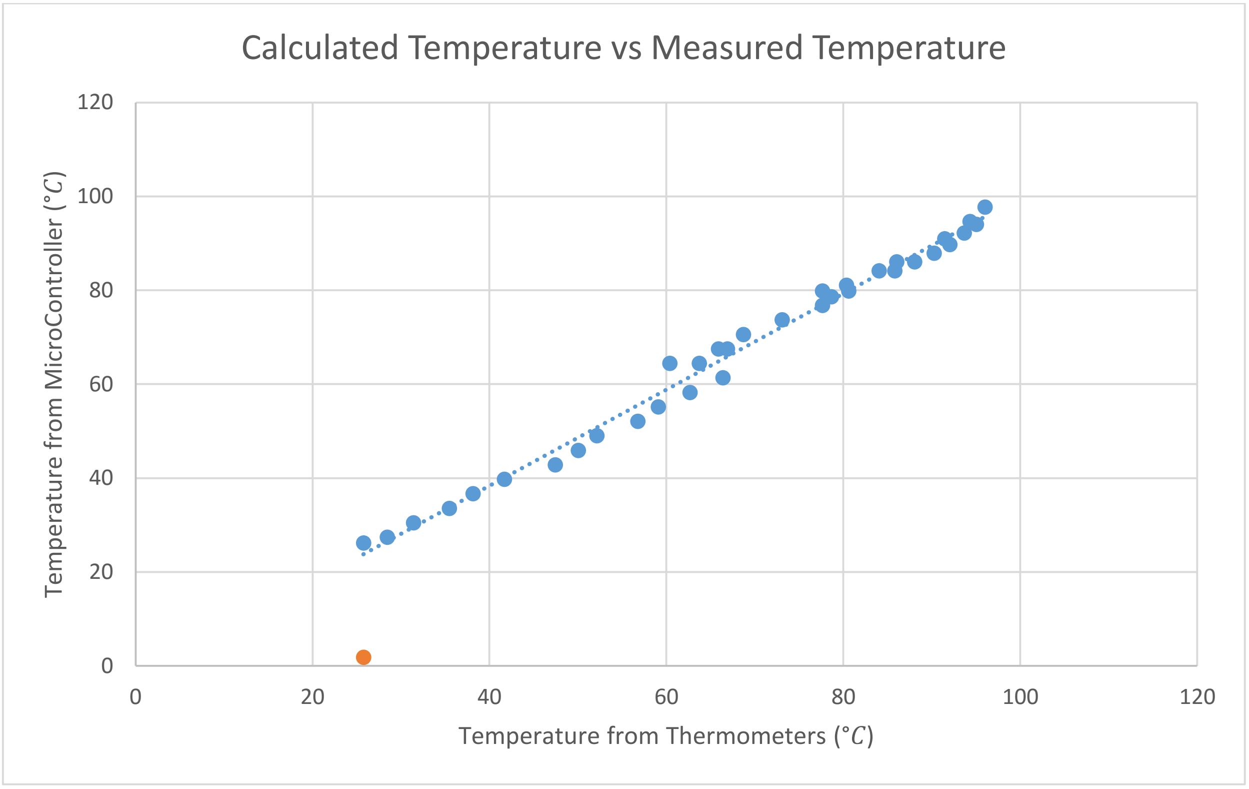

*Figure 8 PIC Temps using Optimal Parameters vs the Average Temperature*

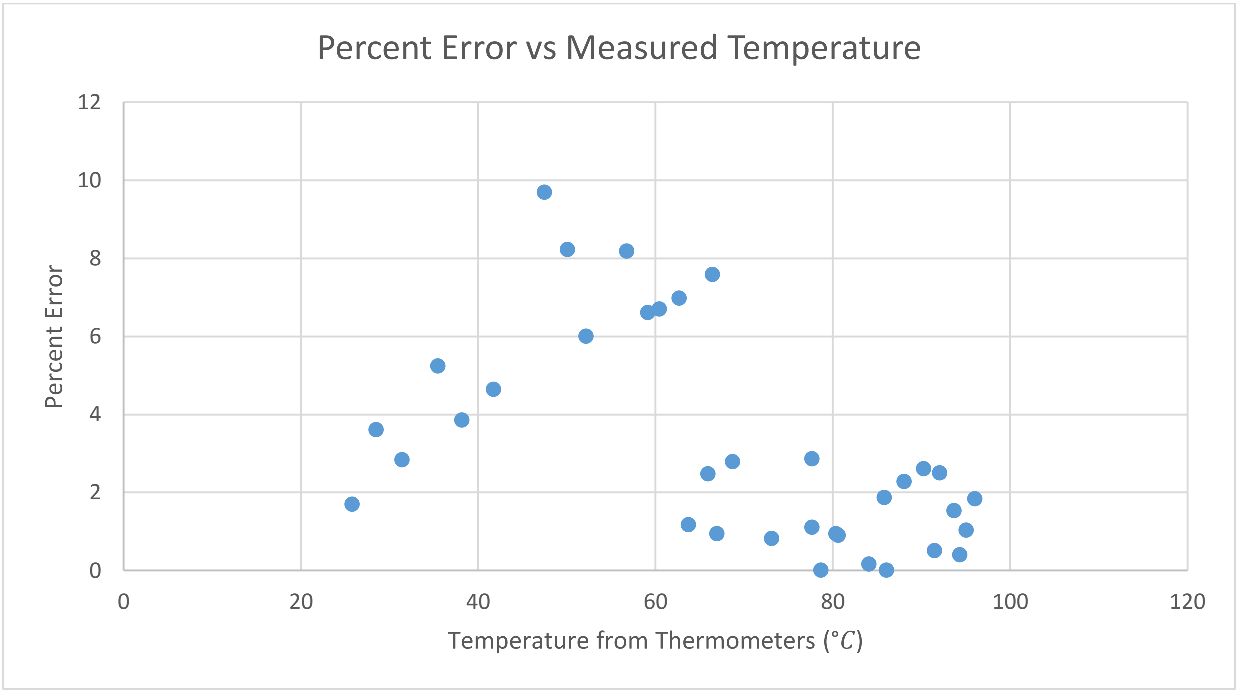

*Figure 9 PIC Temps using Optimal Parameters from Datasheet and the Percent Error*