**ELEN2006 Project: Microcontroller Water Temperature Monitor**

**Jared Ping – 704447**

School of Electrical & Information Engineering, University of the Witwatersrand, Private Bag 3, 2050, Johannesburg, South Africa

**Abstract**: A water temperature monitor, which is used to control and indicate the temperature of any water source, was designed, implemented, and designed to solve the well-known problem in which a person gets into a shower with a most unpleasant error in their desired temperature. Major focuses in making this design possible were the implementation of features such as analogue-to-digital conversion and pulse width modulation. The temperature was measured with the use of a thermistor and then displayed on two seven segment displays and illustrated with the aid of an RGB led. The final product was then critically analyzed as to evaluate any tradeoffs, correction of errors or improved functionality.

**Key Words**: Analogue-to-Digital, Pulse Width Modulation, Thermistor, Seven Segment Display, RGB, Lookup Table, Multiplex.

**1. INTRODUCTION**

Attempting to acquire an optimal temperature in the shower is a tedious task which requires time, patience, money, and possibly a high pain threshold. The conclusion is thus obvious, it's inefficient. This entire process could be avoided by having a geyser set to the correct temperature at all times. This would reduce electricity used on raising its temperature much higher than required, thus the design and implementation of a water temperature monitor for a geyser is a necessity. The temperature range of the monitor would scale between 0 and 99 degrees Celsius, with higher values possible but limited by the use of only two seven segment displays. The optimal desired temperature could then be defined and a green light would be displayed when it's reached. Should it fall below or above the optimal range, it would display blue or red respectively.

**2. BACKGROUND INFORMATION**

The water temperature monitor had to be implemented by programming a PIC16F690 microcontroller with the use of MPLABX IDE. The chip was programmed in assembly language, which is typically used due to its low level language capabilities which thus allows it to be run much faster than when compared to a higher level language such as C which could also be used to program microcontrollers. This chip was then placed a circuit with other elements in order to obtain full functionality.

*2.1 Conversion of an Analogue signal to a Digital signal*

The ADC function is implemented in a way that it will classify an upper and lower bound, in this case 5V and 0V respectively. The chip then reads some voltage at the ADC pin and from there, it transforms the resulting input into a 10 bit digital binary result. This means that there is a resolution of 1024 bits in a range of defined values, where in the specified range of the project, this would equate to 4.9mV per a bit. The ADC input is sourced from a thermistor, which is used to measure temperature. The microcontroller is then calibrated to return a set temperature when it reads in a specific voltage at the ADC pin.

*2.2 Pulse Width Modulation*

The PWM function is one which is used convert digital signals into analogue signals by manipulating the width, or duty cycle, of a pulse. This cycle can be defined by a ratio of the time that a digital signal is measured to be at a high voltage signal compared to that of a low voltage signal. The obtained ratio is then obtained and outputted as an average cycle which allows for different fractional values of the high voltage signal to be outputted. This technique was used to implement the smoothed fading and intensities of the RGB led to firstly transition smoothly through different colour states as well as to help identify the temperature based of the intensity of the colour , i.e. brightness, as a brighter red light would indicate a much higher temperature than that of a light red colour.

**3. SOLUTION DESIGN**

*3.1 Assumptions and Constraints*

Certain limitations and assumptions were required to be considered when designing the final product. A constraint was the limitation of a direct current supply voltage of 5V would be used to power the entire system. It was also assumed that there was no limit with regard to component quantities aside from the specified two seven segment displays as well as the single RGB led

*3.2 ADC Calibration*

A thermistor was utilized to measure the temperature. This was then read by the microcontroller as either a rise or drop in voltage input, which corresponded to a specific temperature. Two issues arose with this implementation, the initial being that the microcontroller is only an 8 bit chip, which meant that the input ADC value now had a reduced resolution of 256 bits, thus corresponding to 19.5mV per a bit, which raised some concern over accuracy of temperature readings. This could however be safely ignored as the seven segments were displaying values to the nearest whole value, which allowed for elimination of reading error.

The second issue was the non-linear relationship of the thermistor. An indirectly proportional relationship exists between temperature and its exponential decay in resistance as temperature increases. Thus to form a more proportional relationship[Fig.1], an easy trick is to simply place it in parallel to a large enough resistor (10k in this case)[1]. Another issue with the thermistor was that it’s extremely sensitive to temperature change, thus a running average was required to achieve the most refined calibration.

The equation by which the ADC value can be calculated to be is defined by:

(1)

These values could then be used when programming the microcontroller to tell the controller what temperature to display when a certain ADC value is recognized. These values were called through the use of a lookup table. Which uses program counter low (PCL) to index the ADC value to one of its 256 possible input values, and return a specific value which went through a binary coded decimal (BCD) conversion to display decimal numbers. Due to the lookup table being more than 256 words, the program would go out of bounds when checking high temperatures, thus PCLATH was utilized to move the lookup table to the second page of memory to ensure a central location for the data[2].

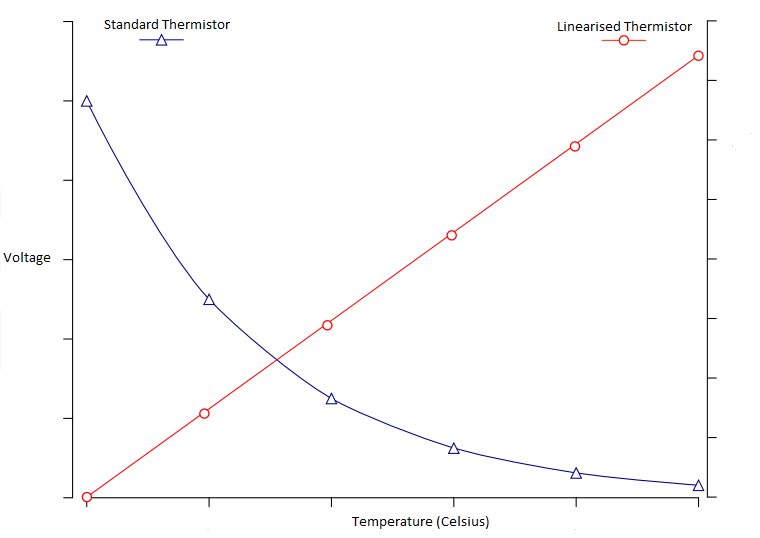


Fig. 1. Relationships of thermistor in different configurations

*3.3 Seven Segment Displays*

A seven segment display is a combination of leds in an orientation that can allows for numbers and letters to be visually displayed. This is done by lighting several of the seven pins available (decimal point pin not required). An issue that arose with using two segment displays was that the microcontroller simply did not feature enough output pins to control both displays independently, as this would require 14 pins and the microcontroller itself only features 17 usable pins, leaving no space for any other functionality.

This issue was easily solved with the implementation of multiplexing. This is a technique where the displays are connected to the same pins and output is sent to both, however PNP transistors which are connected between the microcontroller and the displays’ power pins, which allows for each display to be turned on and off sequentially at a fast enough rate that both displays appear to be powered and showing different values[Fig.2]. These pins were connected to both PortB and PortC so that the PWM pins on PortC could remain open for future use.

Since the displays run off the same pins, one would show some concern for the amount of current being drawn from each pin as all three ports can only source an absolute maximum of 200mA. Thanks to the use of multiplexing, half the current of powering both displays is used. The total current draw from the displays is thus:

This current draw falls well within the limitations of the microcontroller whilst still allowing some headway for PWM implementation.

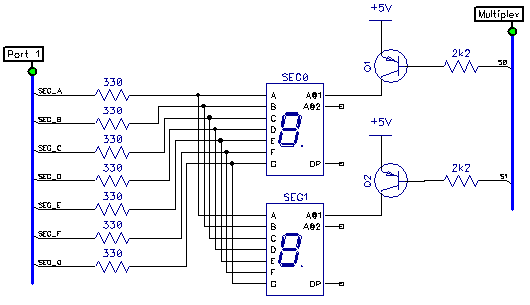


Fig. 2. Circuit diagram of two seven segment displays being multiplexed adapted from [3].

*3.4 RGB led*

The final element in the implementation of the project is the RGB led using PWM. An RGB led is simply a single led which has three leds within it, being red, green, and blue, which thus allows for a full colour spectrum to be created. This four pin led had its common anode connected to the power rail and the other three pins going to the PWM pins on PORTC.

Due to the RGB led being common anode, to show a specific colour, all other pins have to be given a high voltage signal whilst the required colour will be given a low voltage signal. When multiple colours are required, pins will start sending different voltage levels to the pins in order to generate different colour changes.

The optimal temperature was set to be between 30 and 40 degrees Celsius, with anything lower showing blue and above showing red. The use of PWM allowed for a smooth transition of brightness and colour intensity for each colour range showing for example when the temperature was warmer than desired or much hotter than desired.

*3.5 Flow Chart of Circuit Functionality*

Figure 3 shows the logic flow of the assembly program implemented in creating the water temperature monitor.

Set pins to be input/output/ADC/PWM

Get ADC value

No value

Get temp value from lookup table

Pass value through BCD conversion

Switch on first display – display tens unit

Switch off first display, switch on second display – display ones unit

Display RGB colours using PWM

Figure 3: Flow diagram showing logic flow of program implemented on microcontroller.

**4. TESTING AND RESULTS**

The testing process involved powering the circuit with a 5V direct current source. Water was then boiled and placed in a container where the thermistor, which had its wires insulated with heat-shrink tubing, was then submerged and both the seven segment displays and RGB led were both observed for expected results.

Upon testing, all expectations were successfully completed. The temperature readings were within close proximity of actual temperature values. Calibrated temperature values from the linearized thermistor compared to actual readings taken can be observed in table 1.

Table 1: Measured temperature values compared to actual temperature values with error margins.

|  |  |  |
| --- | --- | --- |
| Displayed Temp(˚C) | Actual Temp(˚C) | Error Margin(˚C) |
| 0 | 1 | 1 |
| 10 | 10 | 0 |
| 20 | 19 | 1 |
| 30 | 31 | 1 |
| 40 | 39 | 1 |
| 50 | 50 | 0 |
| 60 | 61 | 1 |
| 70 | 72 | 2 |
| 80 | 80 | 0 |
| 90 | 89 | 1 |

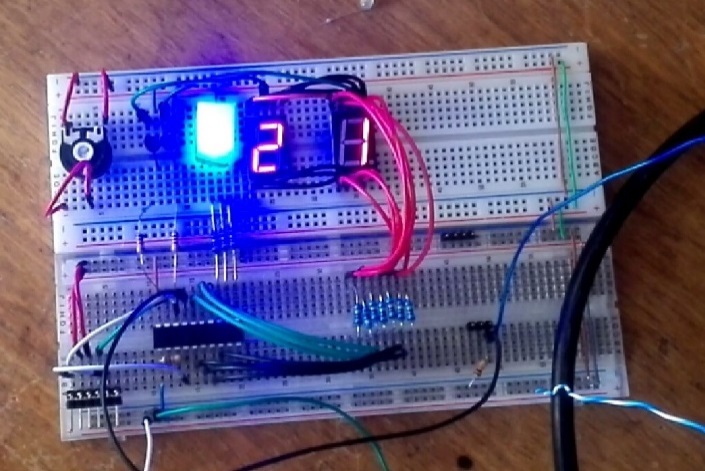


Figure 4: Testing circuit before final testing and circuit simplification.

By referring to figure 4, it’s possible to see what an output would be at room temperature, showing that the temperature is undesirable and thus the geyser would need to be turned on in order to raise the temperature.

**5. DISCUSSION**

*5.1 Improvements*

Due to the sever sensitivity of the thermistor, some calibration values were impossible to accurately calculate as well as having to deal with the non-linear relationship of the device would have led me to have the implementation of a better temperature reading device, such as the lm35 which has a scale of 10mV per a degree Celsius with an error margin or 0.02mV allowing for no calibration being required which would speed up production efficiency if the monitor were to find itself on the consumer market. A second temperature sensor could have also been implemented to account for failure of the primary sensor in the future.

Since the project is ideally modelled to be applied in situations such as monitoring geyser temperatures for optimal efficiency. It would make sense that some functionality could have been implemented that could change the thermostat value of the geyser, or simply control it’s power source to turn off the geyser off when a certain margin above the desired temperature had been reached. The implementation of an LCD screen instead of the seven segments could have allowed for more information to be displayed to the user making it more consumer orientated.

*5.2 Design Tradeoffs*

Were someone wanting to implement any further functionality within this design, they would find a lack of pins to do so. A microcontroller with more pins create many more opportunities with regard to functionality and creativity.

Although assembly is much faster than a high level language such as C, it would serve much easier to a person who might wish to add additional functionality to the design in their home, or even to simply modify their desired temperature without assistance.

**6. CONCLUSION**

A fully functional water temperature monitor was created using a low cost PIC16F690 microcontroller which met all requirements in the project brief. It was realized that the design had much more potential and was found to not be consumer friendly. Some transitions on the RGB led did not seem as smooth as they could have been which could have been dealt with but did not affect the output of the final design. It was also realized that there were features that have been implemented as fail-safes with regard to good engineering practice but the use of those components fell without the constraints with which the project had to be designed.

**RERERENCES**

[1] *Using a Thermistor* [ONLINE] Available at: https://learn.adafruit.com/thermistor/using-a-thermistor. [Accessed 20 September 2015].

[2] Elmer 160 - Lesson 14. 2015. [ONLINE] Available at: http://www.amqrp.org/elmer160/lessons/E160L14.pdf. [Accessed 20 September 2015].

[3] *7 Segment Multiplexing With ULN2003 & PNP Transistors* [ONLINE] Available at: http://codeandlife.com/2012/02/24/7-segment-multiplexing-with-uln2003-pnp-transistors/. [Accessed 20 September 2015].