Application Note

1 December 2017

Closed-loop temperature control using MSP430FR6989

Damon Boorstein, Brendan Nugent & Stephen Glass
Electrical & Computer Engineering, Rowan University

1. Abstract

Microcontrollers are well-suited for closed-loop system control. The MSP430FR6989 is used to interface with an automatic temperature control system with temperature display and potentiometer tuning.

2. Introduction

What makes embedded systems useful is that they can operate independently within an overarching system. In other words, the end user does not have to constantly reprogram the microcontroller if they want it to perform a certain action. In temperature control, microcontrollers can close the loop in what would normally be an open-loop system. Instead of having to switch from a cooling mode to heating mode (or vise versa) to manage the room temperature, the user can simply input a desired temperature. Then, the microcontroller will try to maintain this temperature by constantly sensing and making the necessary adjustments.

This report summarizes the results of the authors' second milestone project in which a closed-loop temperature control system is built with the MSP430FR6989 at its core. Details about the software and hardware components of this project are given, along with an elaboration on how the system works as a whole. System performance, including a measure of steady state and stability, is also discussed.

3. System Design

3.1 MSP430FR6989 Processor

The MSP430FR6989 processor will be used for this application note. The FR6989 has enough hardware addressable pins from the timer modules for hardware PWM applications. Additionally, the FR6989 has a LCD display which we can display the temperature and temperature control setting.

The MSP430 family of ultra-low power microcontrollers consists of several devices featuring peripheral sets targeted for a variety of applications. The architecture, combined with extensive low-power modes, is optimized to achieve extended battery life in portable measurement applications. The microcontroller features a powerful 16-bit RISC CPU, 16-bit registers, and constant generators that contribute to maximum code efficiency.

3.1.1 TimerA Module

The Timer A is a 16-bit timer/counter module included with the MSP430FR6989. The timer module supports multiple capture/compare registers (CCR) and four modes of operation. The value of the CCR and the mode of operation, in conjunction, are used to generate the output signal. With pulse-width modulation support, the timer module can be used to generate a PWM signal directly on at least one of several GPIO pins.

3.1.2 ADC12 Module

The MSP430FR6989 includes an analog-to-digital converter (ADC) peripheral. This module is used in many applications which require an analog voltage output by a sensor to be converted into a readable value by the digital interface of the microcontroller. The module is a 12-bit ADC with a configurable voltage reference, V_{ref}, multiple input channels, and interrupt capabilities.

3.2 Hardware Configuration

The breadboard circuit consists of three sections: the voltage regulator and temperature sensor assembly, the MOSFET switch, and the potentiometer control. The potentiometer is connected to an additional ADC channel of the processor to control the desired temperature. A mechanical tuner was preferred over serial communication to give the operator better control over the system. To set the desired temperature during operation by turning a knob is intuitive, compared to communicating over UART with a serial communication terminal. Additionally, this design uses less microcontroller pins than if UART were to be implemented; two more pins for UART receive and transmit would be required for UART. A picture of the breadboard circuit can be seen in the Figure 1 below. Each of the sections are labelled. Some connections into the microcontroller are not shown for clarity. A 1 Watt power resistor, connected to the output of the voltage regulator, is also not shown.

As can be seen in the figure, the voltage regulator and temperature sensor were secured together using a bare cable, not connected to anything. Touching the sensor directly to the heating element was done to ensure the most accurate possible temperature reading obtainable by the sensor.

Not shown in the image is the fan used for testing. A fan rated at 12 volts was fed with +14VDC on one end and connected to the collector pin of the NMOS on the other. Initially, this fan was directly pointed at the voltage regulator approximately three inches away. The position of the fan was changed throughout the procedure to test system performance. This is explained in the **System Performance** section later on.

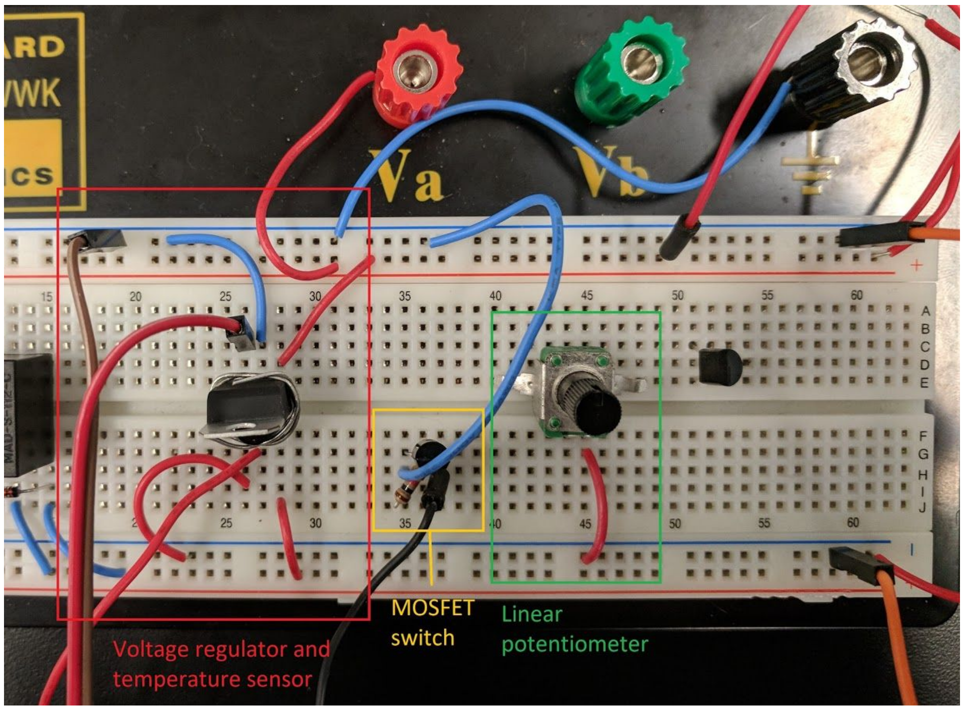

Figure 1: Full breadboard picture. Each section is labeled.

The schematic for the circuit is shown in Figure 2 below.

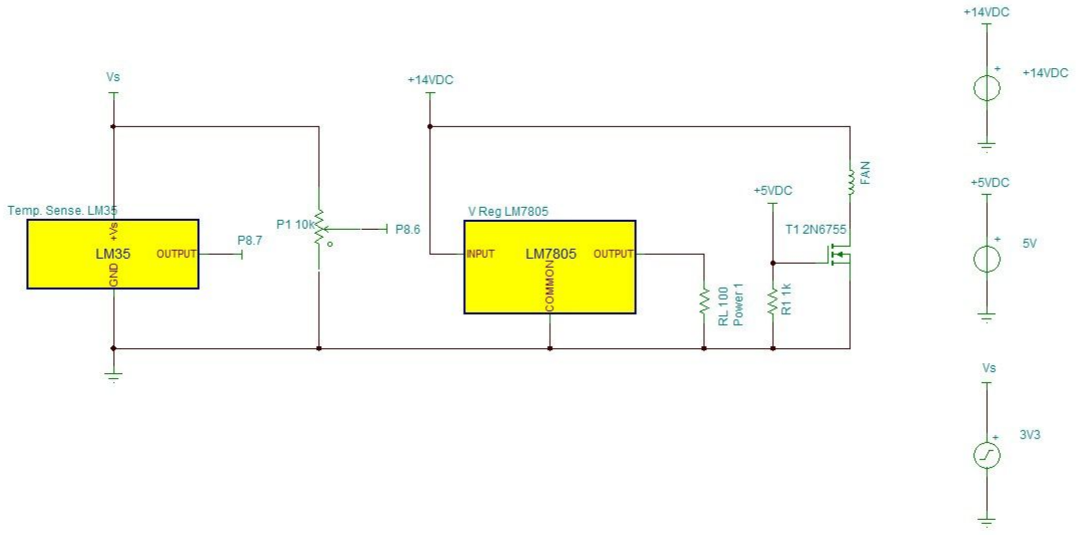

Figure 2: Full schematic. Each section is labelled the same as in the figure above

.

3.2.1 Voltage Regulator and Temperature Sensor

The TI LM35 temperature sensor, LM7805 voltage regulator, and 1W power resistor were used in conjunction. The goal with the LM7805 was to measure and control its temperature when heating up while supplying voltage to the power resistor. Temperature measurement was done by the LM35, whose voltage output is calibrated to the temperature in Celsius on a 10-mV/℃ scale. The stimulus received from the temperature sensor is the input to the ADC module in the FR6989.

3.2.2 MOSFET Switch

A MOSFET switch is a device which can actively switch a higher-power device using a lower-power input signal. For instance, if the user wants to control a fan rated at 12V using a 3.3V signal input, the user could use a MOSFET switch where the signal input was pulse-width modulated. This is what was done here, with the input signal originating from the FR6989 and the output of a high-side NMOS switch driving the fan.

3.3 System Operation

In normal operation, this closed system is able to adjust its output based on stimuli from its environment. The microcontroller accesses its ADC12 and Timer A peripherals to perform these calculations, and then outputs to a select set of pins. The signals generated from these pins are then used to drive the hardware, which in turn, affects the environment of the system itself. An algorithm was developed to model this closed-loop control system.

3.3.1 Temperature Sensing and Analog-to-Digital Conversion

The 12-bit ADC is used to read the current temperature through the LM35 and the desired temperature through a potentiometer. By reading the voltage output from the LM35, the programmed microcontroller converts, by the IC's transfer function, to a temperature in degrees celsius. The conversion is shown in Equation 1:

$$T = \frac{Vref \times ADC12MEM0}{10mV \times 2^{12}} = \frac{1.2V \times ADC12MEM0}{0.010V \times 4096} = ADC12MEM0 \times 0.029$$

Equation 1: Current temperature conversion.

The reference voltage is set to 1.2V for this application. `ADC12MEM0` is the value held in the ADC memory buffer, related to the voltage output of the LM35 temperature sensor. This voltage is obtained through ADC input channel four, located on P8.7 (A4). The circuit is discussed in-depth in the **Hardware** section of this note.

Moreover, the voltage output by the potentiometer, used as an adjustment knob, is taken through ADC input channel five, located on P8.6 (A5). This ADC reading is converted to a desired temperature through Equation 2.

$$T_{desired} = ADC12MEM1 \times 0.0244$$

Equation 2: Desired temperature conversion.

This equation is derived from the fact that the maximum ADC reading is 4096, 2^{12}, or `0x0FFF`, so the ADC contents must be multiplied by $\frac{100}{4096}$, or approximately 0.024414. According to this desired temperature and the current temperature, the PWM duty cycle is set and the fan speed is altered. The algorithm is also discussed in depth in the **Algorithm** section.

3.3.2 Pulse-Width Modulation

For this application, the timer is set to operate in up mode, in which the timer counts to the value held in `CCR0` before immediately resetting to `0x00`. The output mode of the timer is set to reset/set, or `OUTMOD7`. With `CCR1` less than the value of `CCR0`, the timer's output is reset when the timer equals the value held in `CCR1` and set high when the timer equals the value held in `CCR0`.

With this the timer in up mode and its output set to reset/set, the duty cycle of the PWM output can be controlled by changing the value of `CCR1`. The duty cycle can then be calculated from the ratio of `CCR1` to `CCR0`. For example, with `CCR0` set permanently to 100, the duty cycle is 50% when the value held in `CCR1` is 50. The timer module is set to put its output on P1.6. The signal seen by P1.6 is shown in Figure 3.

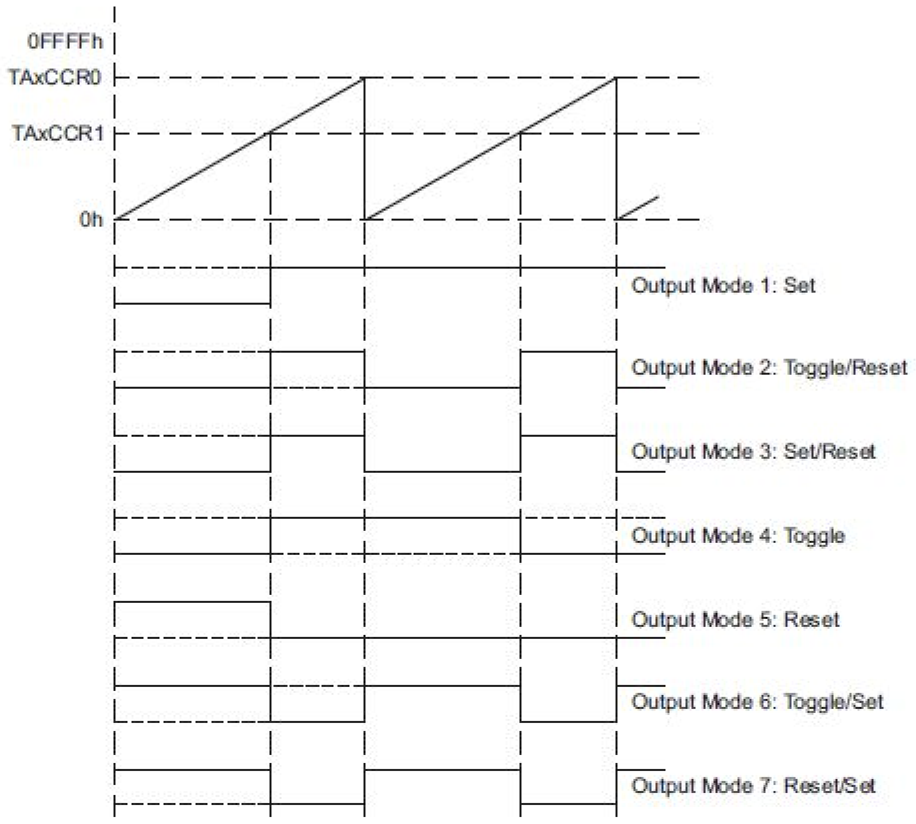

Figure 3: Configurable output modes.

3.3.3 Relating Duty Cycle and Desired Temperature

Before the system could be implemented, it was necessary to draw a relationship between the current temperature and the fan's PWM duty cycle. By plotting the steady-state temperature against the duty cycle supplied to the fan, a function was created to relate the parameters. The plot is shown in Figure 4**.** This function is shown in Appendix A.1, within the C function used to calculate the necessary duty cycle.

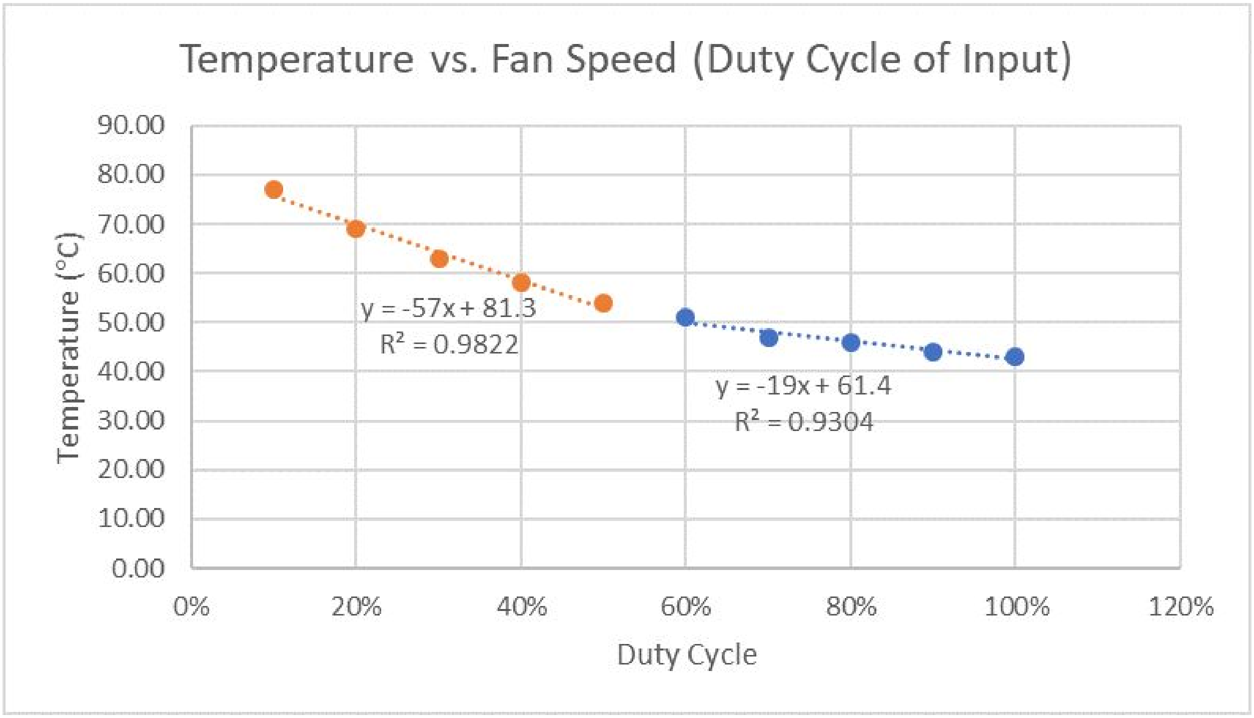

Figure 4: PWM and temperature relationship.

This duty cycle is then used in an equation to set the value held in `CCR1`. However, it is important to note that this specific duty cycle calculation is not the only factor in determining the fan's final duty cycle. In addition, the difference between the current temperature and the desired temperature is taken into account. This difference is amplified by a gain of 2.5 when it is greater than 1℃ and only by 0.5 when it is not. In other words, proportion control is used, correcting heavily when the error is large and correcting less when it is not.

3.4 System Performance

Tantamount to ensuring a system functions properly is maximizing its performance. Additional criteria for this system included steady-state operation and system stability. A steady state is a condition in which the system's output falls within a relatively unchanging region. Stability is a measure of how infrequently a system oscillates. It also refers to its resilience against disturbances introduced into the environment. This system is able to accurately and precisely maintain a stable, steady state temperature within ±3°C. It is also resistant to improper air flow. For example, the system will adjust accordingly if airflow is restricted to half of its full capability. This was tested by turning the fan

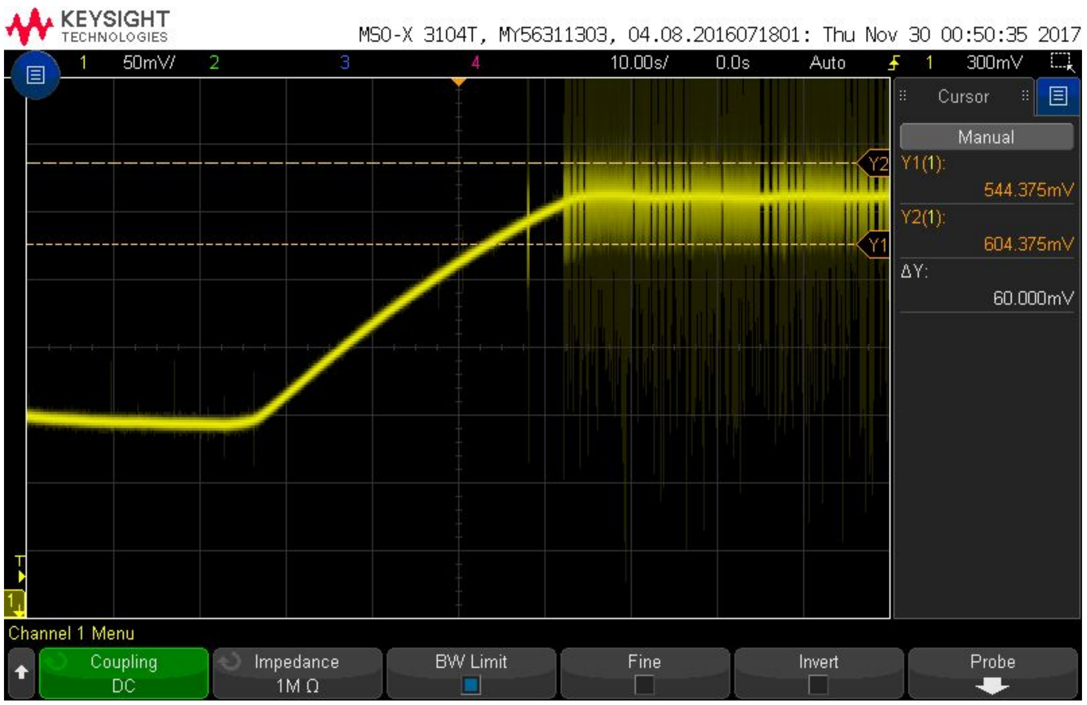

Figure 5: Steady-state temperature read through LM35 after heating.

In the Figure 5 above, the temperature of the voltage regulator is at a steady-state temperature. Using the potentiometer, we then increased the desired temperature. The voltage regulator successfully heats up the system maintains the regulator at that desired temperature.

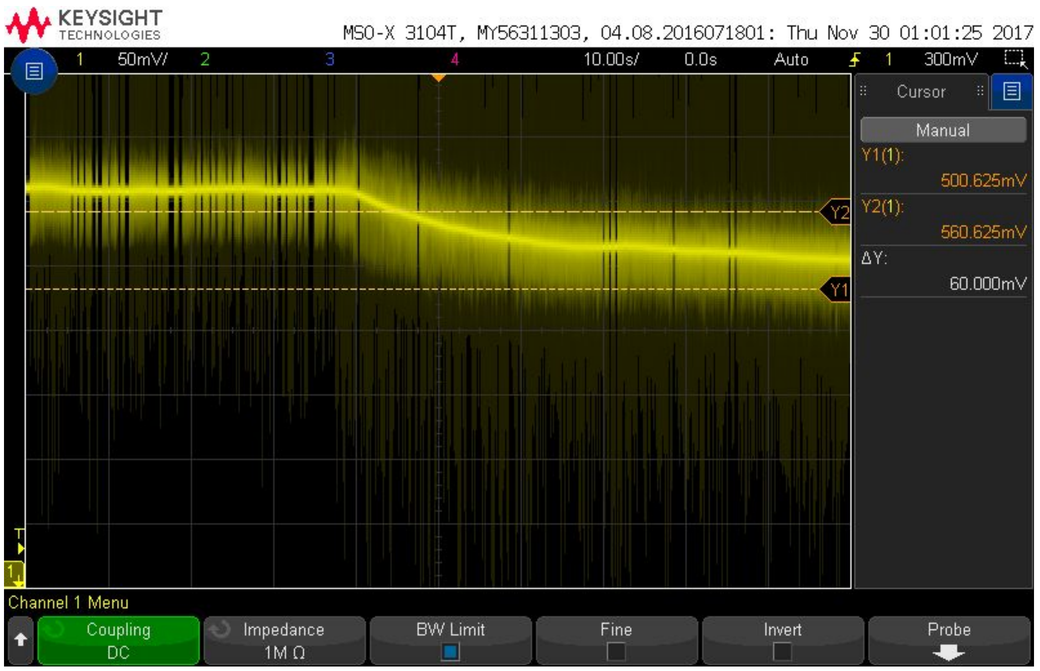

Figure 6: Steady state after cooling.

Inversely, in Figure 6, the temperature of the voltage regulator decreases from steady-state. The fan increases the PWM to ensure the voltage regulator can be maintained at a cooler temperature. The system successfully maintains the regulator at the decreased temperature.

4. System Summary

Overall, the system functions as shown in Figure 7. The flowchart may be used to simplify the understanding of the functionality of the software-hardware interaction.

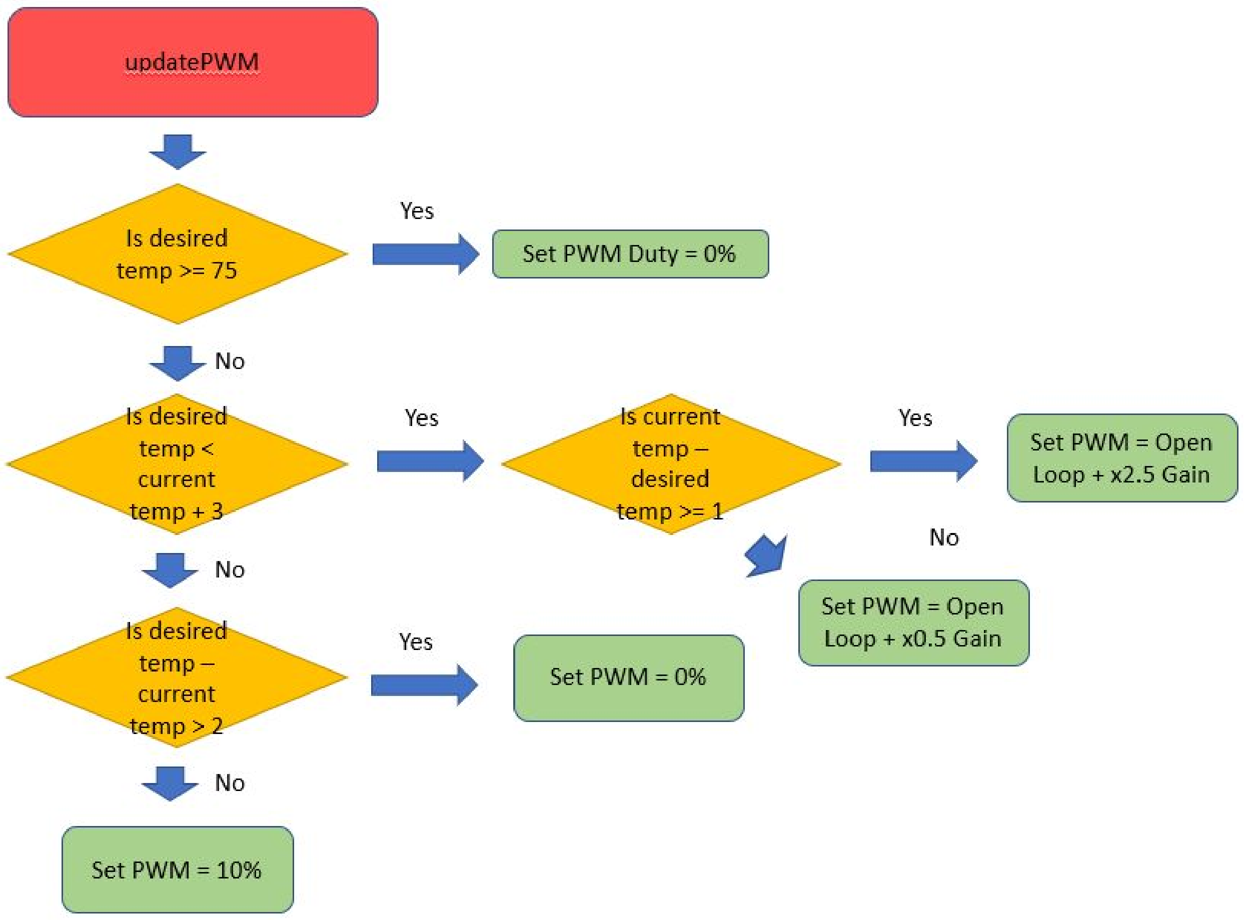
updatePWM
Is desired temp >= 75
Yes
Set PWM Duty = 0%
No
Is desired temp < current temp + 3
Yes
Is current temp – desired temp >= 1
Yes
Set PWM = Open Loop + x2.5 Gain
No
Set PWM = Open Loop + x0.5 Gain
No
Is desired temp – current temp > 2
Yes
Set PWM = 0%
No
Set PWM = 10%

Figure 7: System Flowchart

The system is fully-operational in the range of 43℃ to 77℃. Using the LCD-displayed current temperature and desired temperature, as well as the oscilloscope reading, the system's operation is verified. Additionally, the oscilloscope readings show that the voltage oscillates about the system's input temperature, as desired.

A. Appendix

A.1 Duty Cycle Calculation

```
void setTemperature(int temp)
{
    int duty = 0;
    //y = -36.485x + 75.267
    if(temp >= 54) // first piecewise function
    {
        duty = (int)(-1.754386*(temp-81.3));
    }
    else
    {
        duty = (int)(-5.263158*(temp-61.4));
    }
    TA0CCR1 = duty;
}
```

A.2 System Operation

A video of the system running can be viewed on YouTube:
https://www.youtube.com/watch?v=xSZJFwcNqmo