Milestone 2: Closed Loop Temperature System

Tanner Smith, Russell Binaco
Rowan University

Dr. Schmalzel
Russell Trafford
Section 4
December 1, 2017

1 Abstract

The task presented in milestone 2 is to control the temperature of a voltage regulator by controlling the voltage input of a fan via a microprocessor. This combines concepts presented in previous labs such as PWM and UART, and includes the most recent concepts from labs 5 and 6 that revolve around sensors, high power control and open loop control systems. For this milestone, a high power circuit was designed and a closed loop control system was created to control the temperature of the voltage regulator as a function of its current temperature (with feedback). The MSP430G2553 microprocessor was used to control the power sent to a 24V fan. The design process included decisions about circuit design and the implementation of code including a closed loop control algorithm. The closed loop system successfully controlled the temperature of the voltage regulator with very slight oscillation around a steady-state value. This application note includes an introduction, background, and detailed discussion of the implementation and results of the milestone.

2 Introduction

Closed loop control systems are control systems that account for the current state of the system when determining how to function. This allows the functionality of a system to be accurate, precise, and typically faster than open-loop control systems. Closed loop systems can also react to external changes that are not initially present in the system. Generally, they consider the error, or the difference, between the current state of the system and its goal state, to determine what steps to take in order to achieve the goal state. For this milestone, the system consists of a voltage regulator whose temperature needs to be controlled by a fan. This means the microprocessor needs to assess the current temperature of the voltage regulator and determine how to power the fan in order to achieve a goal temperature (in degrees Celsius). Figure 1 below shows a simple block diagram of this system.

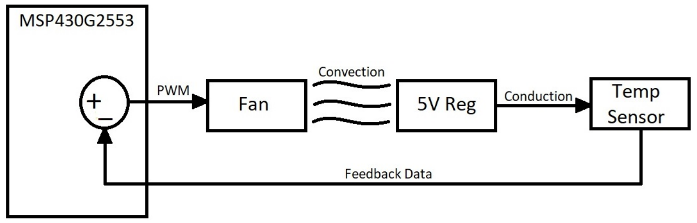


Figure 1: Block Diagram of Closed Loop Control System

In this figure, the microprocessor's inputs are the goal temperature (over UART) and the current temperature (through an analog-to-digital converter), and its output is a pulse-width modulation (PWM) signal that controls the fan. By assessing the difference between the goal temperature and the current temperature, the microprocessor can change the PWM output and, consequently, the temperature of the voltage regulator. If this system were an open loop control system rather than a closed one, the most accurate functionality of the microprocessor would be to use experimental data at different operation levels of the fan to predict what PWM setting would result in the goal temperature. This characterization is part of the closed-loop system, but produces a slower, less accurate change in temperature. This is the top-level description of the system in this milestone; the details of each component of this system will be described in greater detail in the background section of this report.

3 Background

3.1 Universal Asynchronous Receiver Transmitter (UART) Communication Protocol

UART is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. The universal asynchronous receiver-transmitter (UART) takes bytes of data and transmits the individual bits sequentially. At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Serial transmission of digital information (bits) through a single wire or other medium is less costly than parallel transmission through multiple wires.

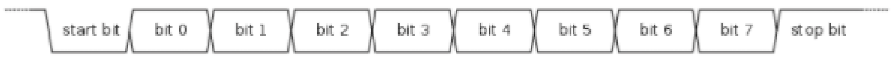


Figure 2: UART Communication Protocol Timing Diagram

The start bit signals the receiver that a new character is coming. The next five to nine bits, depending on the code set employed, represent the character. If a parity bit is used, it would be placed after all of the data bits. The next one or two bits are always in the mark (logic high, i.e., '1') condition and called the stop bit(s). They signal the receiver that the character is completed. Since the start bit is logic low (0) and the stop bit is logic high (1) there are always at least two guaranteed signal changes between characters. If the line is held in the logic low condition for longer than a character time, this is a break condition that can be detected by the UART.

3.2 Pulse Width Modulation (PWM)

Pulse Width Modulation is a technique used to control the power supplied to a device. The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast rate. The longer the switch is on compared to off, the higher the total power supplied to the load. A quantitative term to describe PWM is duty cycle, which refers to the portion of on time compared to the regular interval period.

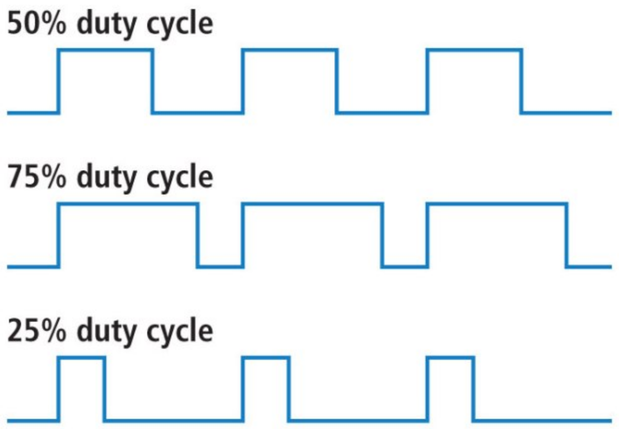


Figure 3: Duty Cycle Example using PWM

In order to power the fan with different voltage strengths, the duty cycle of a PWM can be modified. The ratio of duty cycle to 100% (simply, the decimal form of the duty cycle) is the fraction of Vcc that the fan will see as a DC value. This is a digital-to-analog converter (DAC) since the digital 0 or Vcc is now being output as an average value per PWM period, which provides a resolution equal to how precise the duty cycle percentage can be set.

3.3 Analog to Digital Converter (ADC)

In addition to using a DAC for controlling the fan, the microcontroller needs to be able to read in the current temperature of the voltage regulator via a temperature sensor. This is done through an analog-to-digital converter (ADC). Analog-to-digital converters turn analog signals into a finite, discrete approximation of the analog values. Since digital systems can only store finite-sized numbers, this causes a loss in precision of the values. For example, an 8-bit number has 256 different possible values. Consequently, the resolution of each bit is the quotient of the reference voltage (Vcc) and 2 to the power of the number of bits of the ADC. For the MSP430G2553, the ADC is 10 bits, which means the bit resolution is 3.3V/1024, or 3.2 mV/division. The temperature sensor used in the milestone is 10mV/degree Celsius, so the ADC can store temperature values to the nearest 0.32 degrees. The objective of the milestone is to control the temperature within 3 degrees, so this level of precision is more than enough to precisely determine the current state of the system.

3.4 Open Loop Systems

Open loop systems are control systems which attempt to set their state without knowing whether that goal state and the current state are the same. This means that if the system does not work as expected, the control construct has no way of validating its success, and will not react to inaccuracies or external changes. Given this flaw, the control construct needs some way to know how to try to set the state of the system. This is normally done through the use of experimental data to characterize the expected output of the system. By iterating through all possible control signals and measuring the output, the system can be characterized to associate each possible control setting with a given goal state of the system.

With respect to the milestone, the control signal is a PWM signal. This means that possible control settings range from a 0% duty cycle to a 100% duty cycle. Ideally, this translates to testing these settings and acquiring at least 101 data points to fully characterize the system. Practically, since temperature systems are very slow to reach steady-state, collecting this many points of data would take a very long time. For the milestone, the duty cycle was iterated over in increments of 20%. This results in a plot of duty cycle versus temperature. By fitting a line to this data, any given input temperature can now map to a corresponding duty cycle. Since the experimental data is captured in a particular external setting, this characterization will be inaccurate if factors such as the current room temperature change when testing. However, this predicted duty cycle can be used as a baseline when the system transitions from open loop to closed loop.

4 Results and Discussion

In order to create a closed loop system, first an open loop system was designed to model the speed of the fan against temperature to help find the correct temperature for the closed loop system.

4.1 Open Loop Setup

4.1.1 Hardware Design

The hardware design for this system was thought out and planned before any physical circuit was made. A MSP430G2553 was used because of its low cost and replaceability in case it was accidentally damaged. The only required hardware device given to complete the system was a fan so the rest of the circuit had to be designed around that component. The fan was marked with its average voltage and current ratings which were 24V and 0.255A respectively. Since the MSP430G2553 can only output a voltage of 3.3V and current of 15mA a high power control system had to be implemented. Out of the two high power control systems used in previous labs, a NMOS low side switch was used due to its extremely low current draw and high max switching speed. The fan was connected to 24V and the drain of the NMOS. The source was attached to ground and the gate was connected to the P2.1 GPIO pin.

The next design decision was how to heat up a 5V regulator. It was suggested that 15v to 20V was input into 5V regulator and a power resistor was attached to the 5V output. A decision was made to attach 20V to the 5V regulator to heat it up faster. Also, various sized power resistors were tested to determine the best fit for the desired heat up speed. A 27Ω 5W power resistor was used in the final design.

To measure the temperature of the 5V regulator, a temperature sensor had to be chosen. The LM35 temperature sensor was used because of its familiarity and availability due to use in previous labs. The LM35 required a minimum supply voltage of 5V to output a value of 1° C per 10mV. That means the range of 0° C to 100° C corresponds to 0V to 1V which can be easily read through an ADC pin on the MSP430G2553. P1.7 was used as the ADC pin.

After all circuit components were decided on, a problem appeared where each component needed a different voltage. A total of 4 voltages were needed: 3.3V for the processor, 5V for the temperature sensor, 20V for the 5V regulator and 24V for the fan. However, the bench power supplies used for the milestone could only output 3 different voltages. So in the deepest and darkest hours of the wee night, many pencils were broken and brains fried, three dusty old OPA548 $15 power op amps started to shine brighter and brighter as a whisper sang through the air, ”...use the op amps...”. The whisper was from none other than the man, the mease, the legend himself: Phil Mease. And so the power op amps were used in a non-inverting amplifier fashion to boost a 3.3V signal to 5V, 20V and 24V using 0V and 24V as the rails. The power issue was solved and Phil Mease rested peacefully that night.

The final circuit used for the milestone after all design considerations were finalized can be seen in Figures 4 and 5.

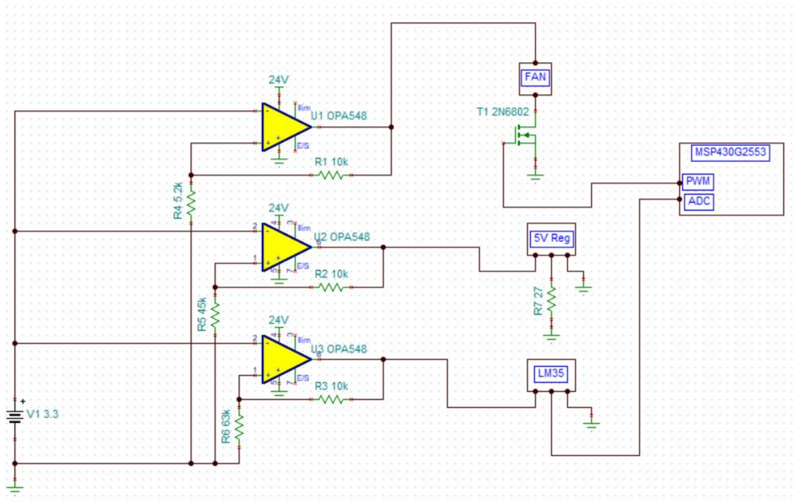


Figure 4: Circuit Schematic

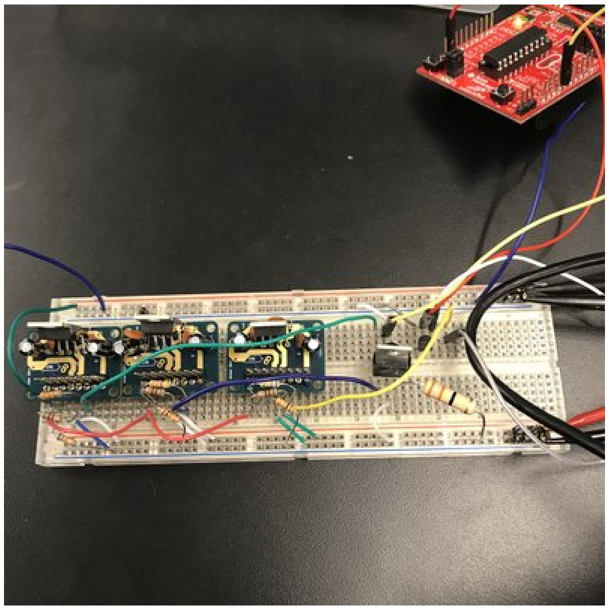


Figure 5: Final Circuit

4.1.2 Software Design

After finishing the hardware, it was time to integrate everything together with software. The software consisted of using four basic peripherals: UART, ADC and two timers. UART was used to send and receive temperature data from the user. The ADC was used to read temperature data from the LM35. One timer peripheral was used to set the sample rate of the ADC and the other timer was used to output a hardware PWM signal to control the fan speed. Each peripheral had their own set up function to configure any necessary parameters to make the system run. The system was then run based on interrupts.

When a user sent a desired temperature to the system, the system would read that data and act appropriately. Whenever a new temperature was received the code would set the correct duty cycle based on a hard coded duty cycle vs temperature curve. The curve was decided based on various testing points taken between 0% duty cycle and 100% duty cycle in increments of 20%. The data can be seen below.

Fan Duty Cycle	**Temperature**
0%	76.7°C
20%	45.4°C
40%	31.0°C
60%	23.5°C
80%	21.6°C
100%	19.7°C

Table 1: Duty Cycle vs Temperature

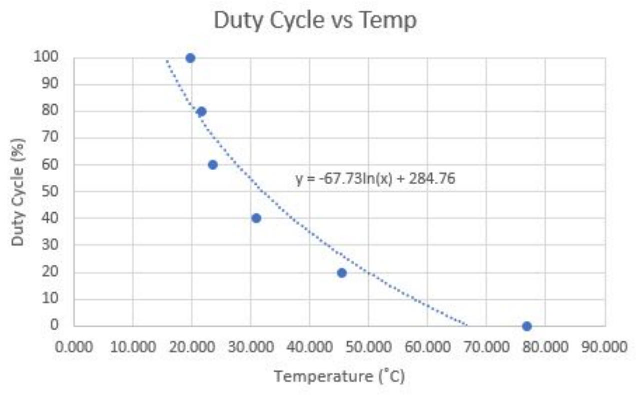


Figure 6: Duty Cycle vs Temperature

The graph shows the line of best fit as a logarithmic function however, this line does not fit the points as well as desired. So the data was split up into two linear halves: duty cycles from 0% to 40% and 40% to 100%.

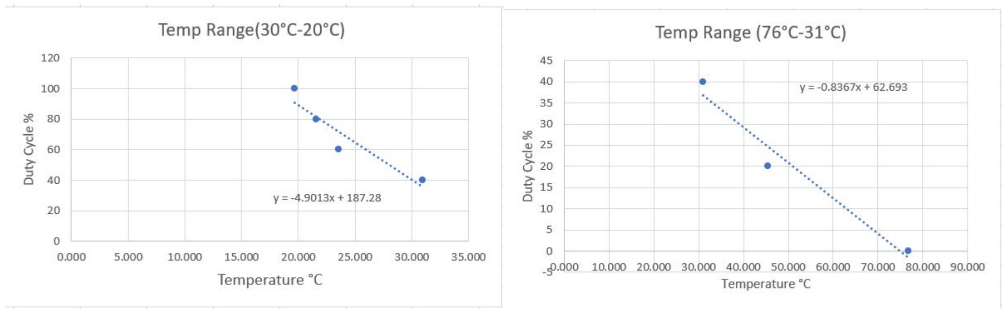
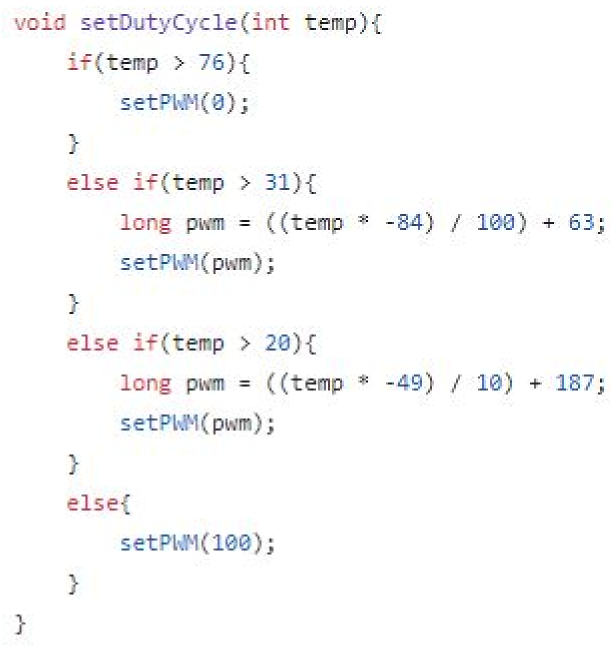


Figure 7: Duty Cycle vs Temperature Linearization

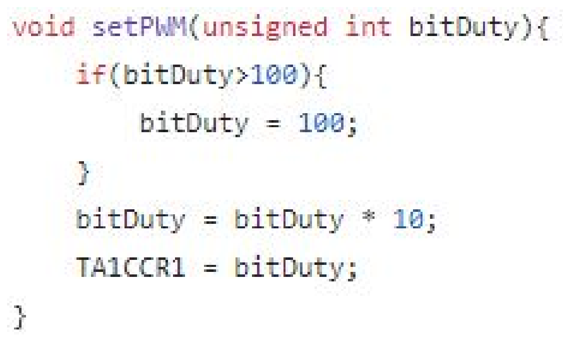
These two linear functions were implemented into code using an if statement to determine which temperature range the input temperature fell within.



```
void setDutyCycle(int temp){
    if(temp > 76){
        setPWM(0);
    }
    else if(temp > 31){
        long pwm = ((temp * -84) / 100) + 63;
        setPWM(pwm);
    }
    else if(temp > 20){
        long pwm = ((temp * -49) / 10) + 187;
        setPWM(pwm);
    }
    else{
        setPWM(100);
    }
}
```

Figure 8: Set Duty Cycle Function

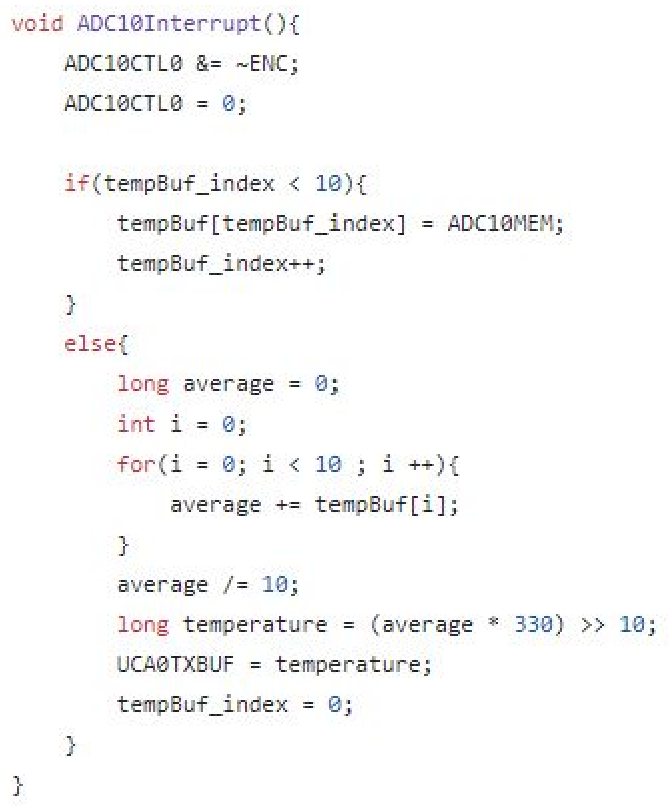
This created a baseline for duty cycle values corresponding to specific temperatures. The setPWM() function updated a capture and compare register for a timer which also changed the fan's duty cycle.



```
void setPWM(unsigned int bitDuty){
    if(bitDuty>100){
        bitDuty = 100;
    }
    bitDuty = bitDuty * 10;
    TA1CCR1 = bitDuty;
}
```

Figure 9: Set PWM Function

The final piece of the open loop control system was sending the current temperature back to the user. This was done by sampling the ADC at a specific rate and sending the current data through UART. The ADC interrupt turned off its own interrupt enable bit and stored the temperatures in a buffer of size 10.



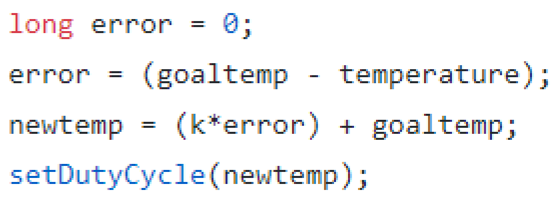
```
void ADC10Interrupt(){
    ADC10CTL0 &= ~ENC;
    ADC10CTL0 = 0;

    if(tempBuf_index < 10){
        tempBuf[tempBuf_index] = ADC10MEM;
        tempBuf_index++;
    }
    else{
        long average = 0;
        int i = 0;
        for(i = 0; i < 10 ; i ++){
            average += tempBuf[i];
        }
        average /= 10;
        long temperature = (average * 330) >> 10;
        UCA0TXBUF = temperature;
        tempBuf_index = 0;
    }
}
```

Figure 10: Storing and Sending Data within the ADC Interrupt

The purpose of turning off the ADC interrupt enable was to set a desired sample rate. A timer interrupt was set to fire at a rate of 2Hz and it consisted of re-enabling the ADC. The purpose of storing each temperature in a buffer was to minimize error. The lower the sample size, the less accurate the data is. So to increase accuracy, a buffer of size 10 was filled every 5 seconds and then the average of that data was sent over UART. If there was a random undesired spike of temperature, the average would minimize its affect on the sent data. This method for sending data provided an accurate reading to the user every 5 seconds.

4.2 Closing the Loop

At this point, the open-loop system received an input temperature and outputted a PWM signal. But since external systems are dynamic, the control construct needed to react to inaccuracies. If the goal temperature and the current temperature were not equal after running the predicted optimal duty cycle, the PWM needed to change. To do this, the PWM value was updated after every temperature reading. In essence, the control algorithm to account for error was simple. The code is shown in Figure 11. The error was the difference between the goal temperature and the current temperature. This error was then multiplied by a constant and added to the current temperature. For example, if the current temperature was too high, the error would be negative, so the new temperature value that would be used to set the duty cycle would be less than the goal temperature. This will raise the duty cycle, decreasing the temperature of the voltage regulator. As this value approached the goal temperature, the duty cycle would slowly decrease until steady state is reached at the goal temperature.



```
long error = 0;
error = (goaltemp - temperature);
newtemp = (k*error) + goaltemp;
setDutyCycle(newtemp);
```

Figure 11: Closed Loop Code

The only parameter in this algorithm was the constant that was multiplied by the error. What this constant was doing (assuming it was greater than 1) is penalizing the error. Since a single degree was only 10 mV, one degree of error would not cause a significant change in duty cycle. For a k value of 3, for example, one degree of error now corresponded to a three degree change in calculating the new duty cycle. This consequentially minimized the steady-state error by reacting faster to a change in temperature.

Two main issues arose from this control scheme: steady-state oscillations and a failure to account for the mechanical limitations of the system. Steady-state oscillations of the system showed a failure to keep the temperature exactly at the goal temperature.

With a very high k value, one degree of error would correspond to a significant change in duty cycle. This correction may overshoot the actual goal temperature, causing the temperature to change by a greater margin in the opposite direction. Next, the duty cycle would change even more to account for that increased error. This results in divergence, where the system will never reach steady-state. Even if k is not so large that it causes divergence, there will be a small oscillation around the goal temperature as the duty cycle cools and heats the system slightly relative to the goal temperature. An average of ten temperature readings taken at a rate of 2 Hz means a new data point was produced every five seconds. This means that the temperature had a possibility of diverging for five seconds before the PWM changes again. So, while the average of 10 temperature values prevented noise, it also introduced a greater oscillation to the system.

Secondly, the mechanical limitations of the system arose at high goal temperatures. If the goal temperature was high enough for the open-loop characterization to expect a very low duty cycle, the average voltage going to the fan would be very low. The expected behavior of the fan would be a slow rate of rotation, but since the system typically starts at room temperature and heats up, the fan would be off while the system accounts for the significant error before the system gets close to the goal temperature. Since mechanical systems are affected by friction, the low average voltage to the fan was not enough to make the fan start rotating. This caused a large initial oscillation as the temperature needed to rise significantly higher than the goal temperature in order to make the PWM signal strong enough to turn on the fan.

4.3 Performance

The performance evaluation demo consisted of starting at room temperature and setting the system to 62 degrees, then 40 degrees, then 58 degrees. At each of these temperatures, steady-state temperature and oscillation were evaluated. Finally, the fan was moved four inches away and held at a 45 degree angle, and the temperature was set to 50 degrees. Figure 12 shows the non-disturbed testing, and Figure 13 shows the testing once the fan was moved.

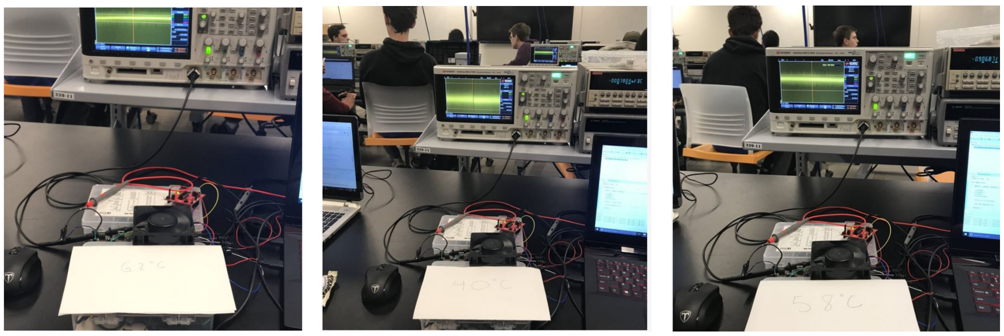


Figure 12: Non-Disturbed Tests

For the first three tests, the system worked as expected. The dotted lines are placed one and a half degrees above and below the goal temperature, and the temperature stayed between these limits successfully. Also, the system reached steady-state within a minute, which is fast for a temperature system. Minor oscillations occurred as a result of the combination of slow PWM updating and error penalization as mentioned in section 4.2 above. The only other noticeable error was due to the frictional forces also mentioned in section 4.2. For the tests that caused the duty cycle to be set to zero while the temperature of the system rose, there was one initial oscillation significantly higher than steady-state since the very low PWM value at that temperature was not enough to make the fan start rotating. Once the fan overcame friction it did not stop moving again since the kinetic coefficient of friction for a material is smaller than its static coefficient of friction, so the lower average voltage was enough to keep the fan moving.

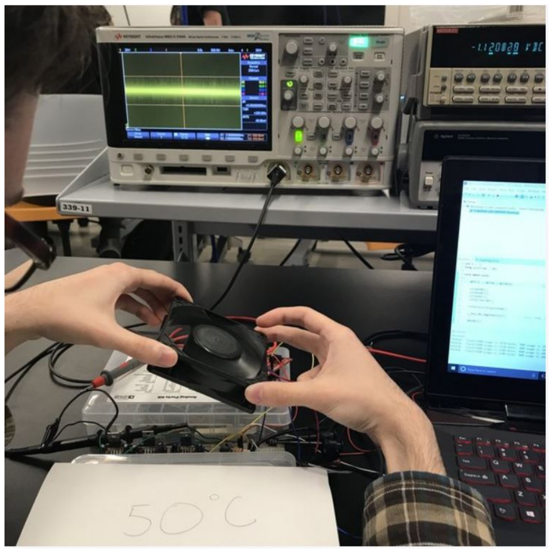


Figure 13: Angled, Distanced Fan Test

For the test that disturbed the system by moving the fan, the system did not quite work as expected: there was a more significant oscillation with a steady-state temperature approximately one degree higher than the goal temperature. If the fan is farther away (and pointed partially away) from the voltage regulator, it will have to work harder to maintain the same temperature. This means that the characterized open-loop temperature will not be as accurate. Initially, the temperature will rise too high and cause the fan to run faster. Then, as the fan starts to cool the voltage regulator, the microprocessor will lower the duty cycle. When the system reaches the goal temperature, the duty cycle will be set as if the fan was running adjacent to the regulator, causing the temperature to increase, increasing the error and PWM signal again. This caused the oscillation as well as the slight increase in steady-state temperature.

5 Conclusion

The overall objective to create a closed-loop control system for this milestone was successful. The constraint of temperature control within three degrees of a given value was satisfied with a small amount of oscillation, even when the fan was moved away from the location at which it was characterized. Ultimately, two significant issues arose from the final design of the system; oscillation was a result of a relatively slow reaction time and unrefined k value, and one large oscillation occurred when increasing in temperature due to the fans inability to turn on.

In order to reduce oscillation, two potential solutions could be attempted moving forward. First, the k value that was chosen was not tested exhaustively. Comparable to hyperparameters of machine learning algorithms, there is no exact mathematical formula to determine the optimal value for k, but the intuition of low k values creating a slow system and high k values causing oscillation could be used to sweep over many options for k and determine a good value. Also, the use of an average of 10 values while sampling only at 2 Hz could be easily remedied. This 5 second reaction time is very slow, even for typically slow temperature systems. By increasing the sample rate, the control signal can react faster to a change in temperature, reducing oscillation by preventing large instantaneous changes in duty cycle. In order to reduce the problem of friction, the algorithm will need a way to detect when the fan is trying to turn on and cannot. One way to do this is to include a flag for decreasing negative error, and temporarily set the duty cycle to some threshold value that will get the fan moving, then immediately reducing the duty cycle back to a low value. Another way would be to have any low nonzero PWM value to result in setting the duty cycle to that same threshold, but this will most likely be less accurate than the former suggestion.

Ultimately, the milestone was successful and regions of improvement have been identified in order to move forward with this closed loop control system had the project continued. Additionally, other algorithms or hardware configurations could be considered in a deeper investigation of systems and control, but this is beyond the scope of the introduction to embedded systems course. The simple closed-loop control construct accomplished the task presented in the milestone while utilizing many of the microprocessors capabilities and other design considerations that have been presented in this course.