

ME-330

ME 330: Mechatronics – Laboratory 4

Temperature Controller

Nicholas Agtual, Dalaney Conte, Evaline Cuatok, Josh Gipson

Department of Mechanical Engineering

Spring 2021

**Abstract**

The Temperature Controller Lab expanded on the introductory Arduino programming utilized in the previous lab, but now focused on a closed-loop operation. A main circuit was constructed and used for all three sections with the code being modified to change the performance of the components. The behavior of the temperature was recorded for each instance. The first section used a bang-bang controller to reach a setpoint temperature of 85 F. The second examined a step input response controller by setting the PWM to a constant value of 100. The third section recorded the results of a proportional controller where the PWM value was set to a value proportional to the error measured with a setpoint temperature of 95 F. Each controller in the lab showcased behavior close to the expected values, with variation due to error.

Table of Contents

[1 Introduction and Objectives 5](#_Toc100777611)

[1.1 Objectives 5](#_Toc100777612)

[1.2 Required Components 5](#_Toc100777613)

[2 Bang – Bang Temperature Controller 7](#_Toc100777614)

[2.1 Physical Circuit 8](#_Toc100777615)

[2.2 Arduino Code 11](#_Toc100777616)

[2.3 Lab Questions 13](#_Toc100777617)

[3 Step Input Response 16](#_Toc100777618)

[3.1 Physical Circuit 16](#_Toc100777619)

[3.2 Arduino Code 16](#_Toc100777620)

[3.3 Lab Questions 18](#_Toc100777621)

[4 Proportional Controller 22](#_Toc100777622)

[4.1 Physical Circuit 23](#_Toc100777623)

[4.2 Arduino Code 23](#_Toc100777624)

[4.3 Lab Questions 25](#_Toc100777625)

[5 Conclusion 27](#_Toc100777626)

[6 Appendices 28](#_Toc100777627)

[Appendix A: MATLAB Code 28](#_Toc100777628)

[7 References 30](#_Toc100777629)

List of Figures

[Figure 1: Arduino Uno Microcontroller 5](#_Toc100777651)

[Figure 2: Heater Assembly 6](#_Toc100777652)

[Figure 3: MOSFET Transistor 6](#_Toc100777653)

[Figure 4: Momentary Pushbutton Switch 7](#_Toc100777654)

[Figure 5: Bang-Bang Temperature Control Response 8](#_Toc100777655)

[Figure 6: Schematic Representation of Heating Circuit 9](#_Toc100777656)

[Figure 7: Schematic Representation of Fan Circuit 9](#_Toc100777657)

[Figure 8: Schematic Representation of Temperature Reading Circuit 10](#_Toc100777658)

[Figure 9: Schematic Representation of Pushbutton Setpoint Circuit 10](#_Toc100777659)

[Figure 10: Electronic Circuit Diagram 11](#_Toc100777660)

[Figure 11: Step Input Output 16](#_Toc100777661)

[Figure 12: Transfer Function with Data Output Overlay 20](#_Toc100777662)

[Figure 13: Transfer Function Block Diagram 21](#_Toc100777663)

[Figure 14: Proportional Controller Output 22](#_Toc100777664)

[Figure 15: Feedback System Diagram 25](#_Toc100777665)

List of Tables

[Table 1: Ts & Tr 20](#_Toc100777713)

# 1 Introduction and Objectives

## Objectives

The goal of the temperature controller lab is to provide an overview of Arduino programming, specifically for closed loop systems. The system of interest is a closed loop temperature controller with PWM. A heater assembly is used in conjunction with an Arduino microcontroller. Three different controllers will be used: Bang-Bang, Step input response, and a proportional controller. They all use the same physical circuit, but the Arduino code is slightly modified for each of them.

## 1.2 Required Components

The components that are necessary to build the main circuit are listed within the section:

**Arduino UNO Microcontroller**

**A close-up of a circuit board

Description automatically generated with medium confidence**

Figure 1: Arduino Uno Microcontroller from ref. (9)

The Arduino UNO microcontroller is an electronics platform that reads inputs and transforms it into an output. This can be altered by editing and uploading instructions utilizing the accompanying Arduino program to control isolated portions of the circuit. The Arduino is used in this lab to modify the function of the circuit.

**Heater Assembly**

**Diagram

Description automatically generated with medium confidence**

Figure 2: Heater Assembly from ref. (9)

The heater assembly used in this lab is composed of a 50 x 15 mm blower fan, a nichrome wire heating element, and a 100K thermistor. The thermistor works by measuring the increased resistance that occurs when the temperature increases.

**P30N06LE MOSFET**

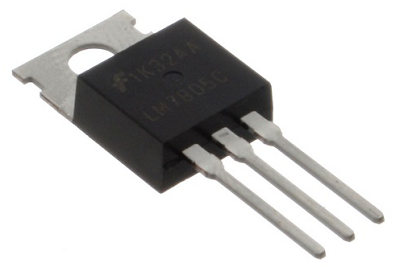


Figure 3: MOSFET Transistor from ref. (4)

The MOSFET is a voltage driven device, meaning the voltage provided at the gate terminal will determine how much current flows through the drain of the transistor. MOSFETs are used in larger applications and since the alternative BJT is current driven it is often used in more simplified circuits [2].

**Pushbutton**



Figure 4: Momentary Pushbutton Switch from ref. (9)

The pushbutton is used in the circuit to toggle between activating and deactivating the power within the circuit. This component adds a factor of convenience due to not having to turn the main power supply on and off.

# Bang – Bang Temperature Controller

A bang-bang temperature controller operates by turning the system on or off depending on when a targeted temperature is recorded. If the desired temperature is not achieved, then the control system either switches heater on or off. The Bang-Bang controller causes a range of values for the temperature to fluctuate which is determined by the setpoint temperature in the inputted Arduino code. This type of control system is also referred to as an on-off control as it is either fully on or off depending on the if the temperature reading is at the setpoint temperature.

Chart

Description automatically generated

Figure 5: Bang-Bang Temperature Control Response

## 2.1 Physical Circuit

The physical circuit built for this lab was comprised of multiple subcircuits. A combination of a heating circuit, fan circuit, temperature reading circuit, and a pushbutton setpoint circuit were used to build the final result. The main power source used for the circuit to power the heater and fan came from the ELVIS rather than the Arduino power supply, which was only utilized in the thermistor circuit. The circuit was built in the same sequential order as suggested in the lab instructions, and then the final schematic as seen in Figure 11 was referenced in order to wire all of the components together. Constructing the subcircuits was straightforward and troubleshooting only occurred once everything was connected.

**Heating Circuit**

**Diagram, schematic

Description automatically generated**

Figure 6: Schematic Representation of Heating Circuit

The heating circuit connects the nichrome heating element to the supply voltage on the ELVIS board and a MOSFET transistor. The heating element is driven by pin 3 on the Arduino which is controlled by the MOSFET through the voltage it supplies from the external power supply. All components that were connected to ground, in addition to the ground of the Arduino, were done through the (-) power rails on the breadboard.

**Fan Circuit**

**Diagram, schematic

Description automatically generated**

Figure 7: Schematic Representation of Fan Circuit

The fan circuit connected the two fan terminals to the ELVIS 5V power source as well as to another MOSFET, similar to the configuration of the heater circuit. The main difference between the two being that the fan circuit is wired to pin 6 on the Arduino.

**Temperature Reading Circuit**

**Diagram

Description automatically generated**

Figure 8: Schematic Representation of Temperature Reading Circuit

For the thermistor circuit, the power supply on the Arduino board is utilized to minimize noise from the circuit. The thermistor was wired using the terminals that were connected to the heater assembly and used in conjunction with a resistor. The parts in this subcircuit were controlled by pin of the Arduino.

**Pushbutton Setpoint Circuit**

**Diagram, schematic

Description automatically generated**

Figure 9: Schematic Representation of Pushbutton Setpoint Circuit

The last subcircuit built was made to operate the switch button required to control the step input. The orientation of the switch button was considered for the element to be functional. The button was connected to the power supply on the breadboard in addition to pin 12 on the Arduino and a second resistor.

Diagram, schematic

Description automatically generated

Figure 10: Electronic Circuit Diagram

The final result of the physical circuit can be seen in the representation above. The only element not included was the optional potentiometer used to define a setpoint temperature. Troubleshooting of the physical circuit occurred for most of the duration of the lab as the control system was not responding to the temperature values. After checking and securing the individual wires of every subcircuit, the components became operational.

## 2.2 Arduino Code

// Defining integer variables

int thermistor = 0;

int fan = 6;

int coil = 3;

int button = 12;

// Defining constants for the thermistor as floats

float RThermistor = 100000;

float Resistor = 100000;

float Bvalue = 3950;

float RTempRated = 25;

// Defining temperature set-point (deg C)

float tempset = 85;

// A flag is defined to record when the pushbutton has been pressed:

bool flag=0;

// The default PWMs for the fan and coil are defined so that the fan starts on at 40% PWM and the coil is

// off:

int fanPWM = 100;

int coilPWM = 0;

// The constants for recording the time between iterations are defined:

int dt;

int tic;

int toc;

void setup() {

 // Defining inputs and outputs

 pinMode(thermistor,INPUT);

 pinMode(button,INPUT);

 pinMode(fan, OUTPUT);

 pinMode(coil, OUTPUT);

**Serial**.begin(9600);

}

void loop() {

analogWrite(fan,fanPWM); //Turn on fan

int buttonreading = digitalRead(button); //Read from button

float Farenheit = readTemperature(thermistor); //Store temperature value

delay(10); // 10ms delay

if (buttonreading == 1){ //Check if button is pressed

     flag = abs(flag-1); //If pressed change the flag to the opposite value

     delay(500); //Delay to negate contact bounce

}

if (flag==1){ //If flag is 1 turn on circuit (the lines below this will

       //be modified in subsequent circuits

// If setpoint is less than measured set coil to full power

if (Farenheit<tempset)

     coilPWM=255;

else

     coilPWM=0; //If setpoint is greater than measured set coil to 0

 }

 else

{

       coilPWM=0;      //If flag is 0, set coil to zero

}

analogWrite(coil,coilPWM); //Write PWM value to coil

tic = millis();    //Record current time

dt  = tic - toc;   //Compute time between iterations

toc = tic;         //Store current time to previous time

//Print out all variables

**Serial**.print("FanPWM ");

**Serial**.print(fanPWM);

**Serial**.print(" CoilPWM ");

**Serial**.print(coilPWM);

**Serial**.print(" dt ");

**Serial**.print(dt);

**Serial**.print(" Temp ");

**Serial**.println(Farenheit);

}

//Finally, we will define our temperature measuring function

double readTemperature(int thermistor) //Function for reading from thermistor

{

float reading = analogRead(thermistor); //Read analog input from

                                       //thermistor

float frac = reading/(1023 - reading); //Compute fractional voltage

float Rval = Resistor \* frac;   //Compute resistance from

//thermistor

float celsius = Rval/RThermistor;  //Remaining lines compute

// temperature in Celsius

celsius = log(celsius);

celsius /= Bvalue;

celsius += 1.0/(RTempRated+273.15);

celsius = 1.0/celsius;

celsius -= 273.15;

return celsius\*9.0/5.0 + 32.0;   //Return Fahrenheit

## 2.3 Lab Questions

1. **Based on the results, describe the performance of the bang-bang controller.**

As seen in Figure 6, the behavior of the temperature readings behaved as expected. With a setpoint of 85 F, the bang-bang controller operated as the temperature was below the specified value by turning the heater on. The performance of the control system was not recorded as it began to level out at the setpoint temperature, which can be attributed to human error for not allowing an appropriate duration to show the results. In addition, the recorded temperature began a downtrend after the reaching the setpoint, which is expected of a bang-bang controller due to the delayed response of the system and the restriction of either being fully operational or completely off. If the trendline continued, a set fluctuation of values that converged to 85 would occur as the bang-bang controller turned the heater on and off periodically.

1. **How would you expect the bang-bang controller to perform if the fan were off? Why?**

If the fan were off and not constantly set at a controlled speed due to a constant PWM, it would take a longer time for the bang-bang controller to turn the heater back on as it would take a longer duration for the temperature to cool back to a value of 85. The controller would still operate as intended by turning the heater on and off depending on when the setpoint temperature was reached, but the interval it would take for the heater to be required to turn back on would increase. Additionally, there would be a decrease in the range recorded below 85 as the system takes a longer cooling time, allowing for the bang-bang controller to turn the heater on without the fan increasing the declining rate.

1. **How would you expect the bang-bang controller to perform if the fan were at full speed? Why?**

If the fan were at full speed, the rate at which the system would be cooled would shorten. However, the range between the values fluctuating to converge at 85 would increase due to this rate. In addition to the cooling rate increasing, the heating rate would also increase due to the heat generated by the fan motor being fully operational. The bang-bang controller would continue to operate as expected, but there would be a noticeable oscillation in the graphed results due to the increased rate in both heating and cooling.

1. **Notice there is a delay when the button is pressed. Look up switch bounce and explain why it is there. Is there a better solution?**

Switch bounce is a physical phenomenon that occurs when two mechanical components come into contact. It is expected that when a switch is closed, that the two parts connect and maintain contact after the initial press. However, there is a constant separation and connection that occurs when the switch is first closed. The circuit registers this as the button being pressed multiple times in a short time interval. The contact between the two parts increases after each ‘bounce’ until the switch is completely closed with a stable output. The switch bounce effect can be detected within this lab in Figure 6, as the graph repeatedly dips in the beginning due to the software recognizes the switch as repeatedly opening. There are a few solutions to combat switch bounce. One method is to add a setback to the software for the program to account for the delay in the hardware. This would be performed by adding a delay in the recording of data in the Arduino code. Another option is to add a capacitor in parallel with the switch to produce a result known as RC debouncing. This minimizes switch bounce by filtering the change in input signal by stopping the input voltage across the targeted resistor until it reaches the source voltage [7].

# Step Input Response

Chart, line chart

Description automatically generated

Figure 11: Step Input Output

## 3.1 Physical Circuit

Refer to Section 2.1 for details about the physical circuit used in this section.

## 3.2 Arduino Code

int thermistor = 0;

int fan = 6;

int coil = 3;

int button = 12;

// Next the constants for the thermistor are defined

float RThermistor = 100000;

float Resistor = 100000;

float Bvalue = 3950;

float RTempRated = 25;

// The temperature setpoint (in Fahrenheit) is defined below:

float tempset = 85;

// A flag is defined to record when the pushbutton has been pressed:

bool flag=0;

// The default PWMs for the fan and coil are defined so that the fan starts on at 40% PWM and the coil is

// off:

int fanPWM = 100;

int coilPWM = 0;

// The constants for recording the time between iterations are defined:

int dt;

int tic;

int toc;

void setup() {

 pinMode(thermistor,INPUT);

 pinMode(button,INPUT);

 pinMode(fan, OUTPUT);

 pinMode(coil, OUTPUT);

**Serial**.begin(9600);

}

void loop() {

 // put your main code here, to run repeatedly:

analogWrite(fan,fanPWM);       //Turn on fan

int buttonreading = digitalRead(button);   //Read from button

float Farenheit = readTemperature(thermistor);  //Store temperature value

delay(10);

if (buttonreading == 1){   //Check if button is pressed

     flag = abs(flag-1);  //If pressed change the flag to the opposite value

            delay(500);

}     //Delay to negate contact bounce

if (flag==1){      //If flag is 1 turn on circuit (the lines below this will

       //be modified in subsequent circuits

coilPWM =100;

 }

 else

{

       coilPWM=0;      //If flag is 0, set coil to zero

}

analogWrite(coil,coilPWM);   //Write PWM value to coil

tic = millis();   //Record current time

dt  = tic - toc;   //Compute time between iterations

toc = tic;    //Store current time to previous time

     //Print out all variables

**Serial**.print("FanPWM ");

**Serial**.print(fanPWM);

**Serial**.print(" CoilPWM ");

**Serial**.print(coilPWM);

**Serial**.print(" dt ");

**Serial**.print(dt);

**Serial**.print(" Temp ");

**Serial**.println(Farenheit);

}

double readTemperature(int thermistor) //Function for reading from thermistor

{

float reading = analogRead(thermistor); //Read analog input from

                                       //thermistor

float frac = reading/(1023 - reading); //Compute fractional voltage

float Rval = Resistor \* frac;   //Compute resistance from

//thermistor

float celsius = Rval/RThermistor;  //Remaining lines compute

// temperature in Celsius

celsius = log(celsius);

celsius /= Bvalue;

celsius += 1.0/(RTempRated+273.15);

celsius = 1.0/celsius;

celsius -= 273.15;

return celsius\*9.0/5.0 + 32.0;   //Return Fahrenheit

## 3.3 Lab Questions

1. **Based on the response and the input find the transfer function for the system (See Appendix B). Notice that the temperature does not start at zero. Subtract the initial temperature so the response starts at zero. You can then add the initial temperature back on at the end. Follow the procedure in the lecture. There is a slight modification because our step input was not 1:**

**K=**

Accounting for the fact that the step input was not 1.

Finding the time constant.

Solving for the transfer function gain.

where

Constructing the transfer function.

1. **Graph the output of the transfer function from (1) and overlay it on your data output.**

Chart

Description automatically generated

Figure 12: Transfer Function with Data Output Overlay

1. **Compare Tr and Ts of the transfer function and the actual data output.**

is the time for the waveform to go from .1 to .9 of its final value and can be represented by for a first order system. is the time for the response to reach and stay within 90% of its final value and can be approximated by .

Table 1: Ts & Tr

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Transfer Function |  |  |
| Data Output |  |  |

1. **Compare the graphs between at least two team members. Explain what might account for the differences.**

There are slight variations in the graphs that different team members plotted. This can be due to error in interpreting the final temperature (set point or actual average value) and interpreting the true staring time. These numbers affect the values of the transfer function thereby affecting the plots.

1. **Draw out the block diagram for the transfer function. Label the signals. Can this be easily used for temperature feedback? (Hint: what would the signal flow look like for temperature feedback)**

P(s)

Q(s)

H(s)

Controller

System

Feedback Gain

+

-

Error

Input

Temperature Output

R(s)

E(s)

M(s)

C(s)

Temperature Setpoint

Figure 13: Transfer Function Block Diagram

This can be easily used for temperature feedback because we have temperature setpoint and an output by our system that involves a feedback loop back to the controller.

1. **Notice that the noise bounces between discrete values, why does this occur?**

Within all our experiments there is always a degree of noise but within out experiment of temperature readings the noise is clearly seen in our graphs. The noise in this experiment could have been caused by the quality of the thermistor or the quality of the heater. The thermistor was also measuring the temperature of the air as it is mixing with the air from the heater and the outside air. This could cause the noise in our data because as the air is a heterogeneous mixture and the temperature varies with time.

1. **If we used some capacitors, how would we filter this noise? (Recall Lab 1)**

To filter out the noise we must provide a clean curve. Capacitors work with AC current because it brings impedance to the system. With the impedance the noise could be filtered out and provide that clean curve we are looking for.

# 4 Proportional Controller

The proportional controller is a type of feedback system where the output signal is proportional to the error signal. This means that it corrects an applied control variable between a desired value and the actual value. Proportional controllers are useful in cases where correction needs to be applied quickly.

Chart

Description automatically generated

Figure 14: Proportional Controller Output

## 4.1 Physical Circuit

The physical circuit for this portion of the lab is the same circuit set up as the Bang-Bang Temperature Controller and the Step Input Response. The only thing that changes between each section is the Arduino Code which is shown below.

## 4.2 Arduino Code

int thermistor = 0;

int fan = 6;

int coil = 3;

int button = 12;

// Next the constants for the thermistor are defined

float RThermistor = 100000;

float Resistor = 100000;

float Bvalue = 3950;

float RTempRated = 25;

// The temperature setpoint (in Fahrenheit) is defined below:

float tempset = 85;

// A flag is defined to record when the pushbutton has been pressed:

bool flag=0;

// The default PWMs for the fan and coil are defined so that the fan starts on at 40% PWM and the coil is

// off:

int fanPWM = 100;

int coilPWM = 0;

// The constants for recording the time between iterations are defined:

int dt;

int tic;

int toc;

void setup() {

 pinMode(thermistor,INPUT);

 pinMode(button,INPUT);

 pinMode(fan, OUTPUT);

 pinMode(coil, OUTPUT);

**Serial**.begin(9600);

}

void loop() {

 // put your main code here, to run repeatedly:

analogWrite(fan,fanPWM);       //Turn on fan

int buttonreading = digitalRead(button);   //Read from button

float Farenheit = readTemperature(thermistor);  //Store temperature value

delay(10);

if (buttonreading == 1){   //Check if button is pressed

     flag = abs(flag-1);  //If pressed change the flag to the opposite value

            delay(500);

}     //Delay to negate contact bounce

if (flag==1){      //If flag is 1 turn on circuit (the lines below this will

       //be modified in subsequent circuits

coilPWM =100;

float error = tempset-Farenheit; //compute error

int pConstant = 10;     //define proportional constant

coilPWM =error\*pConstant;    //define PWM by multiplying error by constant

 }

 else

{

       coilPWM=0;      //If flag is 0, set coil to zero

}

analogWrite(coil,coilPWM);   //Write PWM value to coil

tic = millis();   //Record current time

dt  = tic - toc;   //Compute time between iterations

toc = tic;    //Store current time to previous time

     //Print out all variables

**Serial**.print("FanPWM ");

**Serial**.print(fanPWM);

**Serial**.print(" CoilPWM ");

**Serial**.print(coilPWM);

**Serial**.print(" dt ");

**Serial**.print(dt);

**Serial**.print(" Temp ");

**Serial**.println(Farenheit);

}

double readTemperature(int thermistor) //Function for reading from thermistor

{

float reading = analogRead(thermistor); //Read analog input from

                                       //thermistor

float frac = reading/(1023 - reading); //Compute fractional voltage

float Rval = Resistor \* frac;   //Compute resistance from

//thermistor

float celsius = Rval/RThermistor;  //Remaining lines compute

// temperature in Celsius

celsius = log(celsius);

celsius /= Bvalue;

celsius += 1.0/(RTempRated+273.15);

celsius = 1.0/celsius;

celsius -= 273.15;

return celsius\*9.0/5.0 + 32.0;   //Return Fahrenheit

}

## 4.3 Lab Questions

1. **Draw the feedback system diagram for the proportional controller and label all signals.**

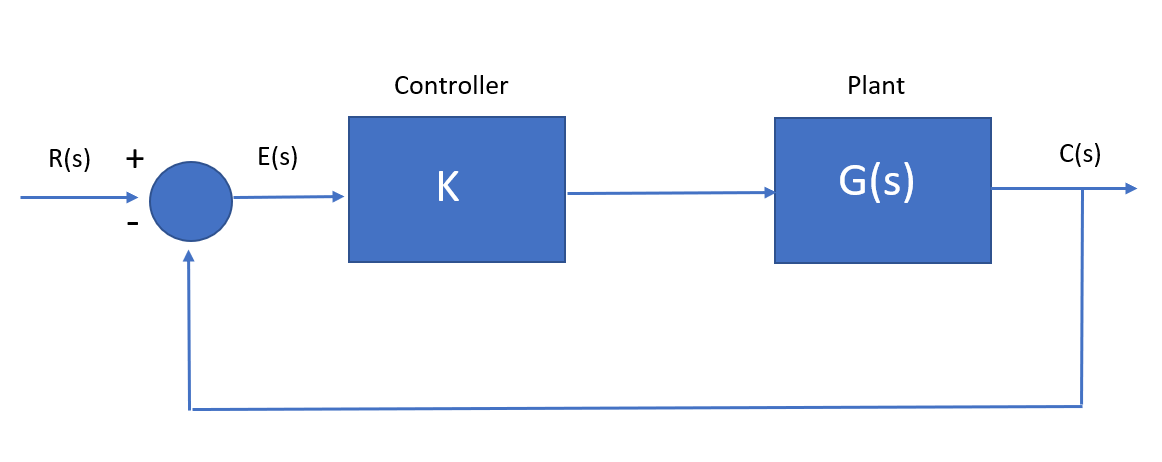


Figure 15: Feedback System Diagram

1. **Did the data converge to the specified value?**

The data did not converge to the specified value of 85. From the graph in figure 15, the data converged to a slightly higher value of approximately 86.

1. **Perform step (8) How did the steady-state error of the model compare to the real system-when the gain was increased? What might account for the differences?**

Equation 1 =-

= 86-78.5=7.5

Equation 2 =-

=85-78.5= 6.5

Equation 3 = (1-)

=0.6327.5=4.74

Adding back on: 78.5+4.74=83.2.

From the graph in figure 15, at =83.2, is approximately equal to 9.75

Equation 4 a=

a==0.103

Equation 5 K=

K==0.118

Equation 6 T(s)= where G(s)=

G(s)=

T(s)==

The steady state error of the model was lower than that of the real system when the gain was increased. To find the steady state error, the equation E(s)=R(s)-H(s)C(s) where E(s) represents the error signal, R(s) represents the setpoint signal (or desired value), H(s) represents the feedback gain signal, and C(s) represents the output signal [8]. As the feedback gain is increased, this gain is fed back into the system in order to get the system closer to the desired setpoint. Thus, an increase in feedback gain means that the error is getting smaller, and the system is closer to its setpoint.

1. **Compare the closed-loop transfer function here to the one derived in Section 3, question 1.**

The closed loop transfer function derived in Section 3 question 1 was found to be while the closed loop transfer function found just above was found to be . The transfer function from Section 3 is significantly smaller than the transfer function from above.

# 5 Conclusion

From this lab, we were able to grasp a better understanding of how to use Arduino in conjunction with a closed loop temperature control with PWM. During the lab, we faced some difficulties when it came to getting our circuits to function properly. With the first section of the lab, we could not get the temperature to reach room temperature. When this was accomplished, we had difficulty getting the temperature to reach our desired setpoint. In order to solve the issue, the circuit had to be rewired and some of the circuit components had to be switched out. Additionally, the Arduino code had to be edited in order for the correct data to be collected. Despite the difficulties, the lab was completed successfully and a deeper understanding of the control system analysis and transfer function computation was achieved.

# 6 Appendices

## Appendix A: MATLAB Code

close all; clear; clc

Importing Excel Workbook

fileName = 'Lab 4 - Data';

Bang-Bang Temperature Controller

% Time and temperature data

tempBangBang = xlsread(fileName, 'Bang Bang Temp Controller', 'H2:H1412');

timeBangBang = xlsread(fileName, 'Bang Bang Temp Controller', ...

'I2:I1412');

% Plot

figure(1)

plot(timeBangBang \* 10 ^ -3, tempBangBang);

% Plot features

hold on

grid on

grid minor

ylim([77 88]);

xlim([0 45]);

% Plot Descriptors

xlabel('\emph {Time (Seconds)}', 'fontsize', 14, 'Interpreter', 'latex');

ylabel('\emph {Temperature ($^{\circ}$C)}', 'fontsize', 14, ...

'Interpreter', 'latex');

title('\emph {Bang-Bang Controller Output (85 $^{\circ}$ Setpoint)}', ...

'fontsize', 16, 'Interpreter', 'latex');

legend('Bang Bang Controller', 'fontsize', 10, 'Interpreter', 'latex');

Step Input Response

% Time and temperature data

tempStepInput = xlsread(fileName, 'Step Response', 'H2:H1772');

timeStepInput = xlsread(fileName, 'Step Response', 'I2:I1772');

% Plot

figure(2)

plot(timeStepInput \* 10 ^ -3, tempStepInput);

% Plot features

hold on

grid on

grid minor

xlim([0 timeStepInput(end) \* 10 ^ -3]);

ylim([76 93]);

% Plot Descriptors

xlabel('\emph {Time (Seconds)}', 'fontsize', 14, 'Interpreter', 'latex');

ylabel('\emph {Temperature ($^{\circ}$C)}', 'fontsize', 14, ...

'Interpreter', 'latex');

title('\emph {Natural response (100 PWM)}', ...

'fontsize', 16, 'Interpreter', 'latex');

legend('Step Input response', 'fontsize', 10, 'Interpreter', 'latex');

Proportional Controller

% Time and temperature data

tempProportionalController = xlsread(fileName, ...

'Proportional Controller', 'H2:H1184');

timeProportionalController = xlsread(fileName, ...

'Proportional Controller', 'I2:I1184');

% Plot

figure(3)

plot(timeProportionalController \* 10 ^ -3, tempProportionalController);

% Plot features

hold on

grid on

grid minor

xlim([0 timeProportionalController(end) \* 10 ^ -3]);

ylim([78 88]);

% Plot Descriptors

xlabel('\emph {Time (Seconds)}', 'fontsize', 14, 'Interpreter', 'latex');

ylabel('\emph {Temperature ($^{\circ}$C)}', 'fontsize', 14, ...

'Interpreter', 'latex');

title('\emph {Proportional Controller Output (10X Gain \& 95 $^{\circ}$ Setpoint)}', ...

'fontsize', 16, 'Interpreter', 'latex');

legend('Proportional Controller response', 'fontsize', 10, ...

'Interpreter', 'latex');

# 7 References

1. Escalona, L. “Electrical Circuits and Components.” ME 330 – Mechatronics. San Diego State University. Lecture
2. Escalona, L. “Analysis of Electrical Networks.” ME 330 – Mechatronics. San Diego State University. Lecture
3. Escalona, L. “Sensors and Actuators Used in Mechatronics Systems.” ME 330 – Mechatronics. San Diego State University. Lecture
4. Escalona, L. “Semiconductor Devices: The Field Effect Transmitter.” Me 330 – Mechatronics. San Diego State University. Lecture
5. Escalona, L. “Operational Amplifier Configurations.” ME 330 – Mechatronics. San Diego State University. Lecture
6. Escalona, L. “Electric Motors.” ME 330 – Mechatronics. San Diego State University. Lecture
7. Christofferson, J. “Switch Bounce and How to Deal With It” All About Circuits. Article
8. Escalona, L. “Lab 5 Background.” ME 330 – Mechatronics. San Diego State University. Lecture.
9. Escalona L. “Lab 4 Background.” ME 330 – Mechatronics. San Diego State University. Lecture