OPTIMAL CONTROL FOR EFFICIENT ELECTRIC HEATING

A thesis presented to the faculty of the Graduate School of

Western Carolina University in partial fulfillment of the

requirements for the degree of Master of Science in Technology

By

Jairo Nevarez

Director: Dr. Bora Karayaka

Associate Professor

School of Engineering and Technology

Committee Members:

Dr. Martin L. Tanaka, School of Engineering and Technology

Dr. Peter Tay, School of Engineering and Technology

April 2018

This thesis is dedicated to my family.

Gracias Familia.

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor Dr. Bora Karayaka for the continuous support during graduate studies, for his patience, motivation, and enthusiasm. I would like to thank my thesis committee members Dr. Martin L. Tanaka and Dr. Peter Tay for all the help, time, and immense knowledge throughout my graduate studies and classes. My sincere thanks to the College of Engineering +Technology, Rapid Center, cohort and anyone who made my stay at Western Carolina University not only educational but also enjoyable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

OPTIMAL CONTROL FOR EFFICIENT ELECTRIC HEATING

Jairo Nevarez, M.S.T.

Western Carolina University (April 2018)

Director: Dr. Bora Karayaka

Abstract—The purpose of this study is to investigate methods of reducing the cost of electricity consumption. Utility companies must forecast and adjust for power demand. Utilities desire a 1:1 load factor ratio between peak energy usage and average usage. During peak hours, electricity production is most expensive. There are two major methods for reducing the peak power for Thermostatically Controlled Loads (TCL), such as electric water heaters, air conditioners, or heat pumps: a) Classic Demand Side Management (DSM) methods such as demand shifting and electricity pricing tariffs, and b) Advanced DSM load control methods. This thesis will focus on analyzing the advanced control methods to reduce peak power and to save energy. The use of space heating and TCL loads for reducing electricity consumption and peak demand production is an important research area, considering that the energy consumption of most of US single-family residential homes is from controllable appliances. An experimental thermal identification system utilizing first and second order mathematical models has been developed at WCU.

Using these models, a new proportional (P-Only) and proportional integral (PI) controller are investigated and assessed for improvements of reduction of peak power and energy savings for a TCL compared to the traditional Bang-Bang Controller in a resistive space heating prototype. Comparative results between simulation and experimental work validated the linearity of power electronics controller. Linearization was achieved by identifying a mathematical relationship that eliminates quadratic power function as well as Buck converter’s nonlinearity. Temperature disparity and input power characteristics were improved using this new converter for controlling the space heater. The system developed is an important step toward energy savings, temperature improvements and demand side management for reducing peak demand.

CHAPTER 1: INTRODUCTION

1.1 Key Terms

m	Slope of linear proportion	e	Error signal
D	Duty Cycle	$k(s)$	Transfer function of linearized actuator
L	Inductor Value	$T_{TH}(s)$	Transfer function of the thermal system
R	Resistor Value	$T_C(s)$	Transfer function of the controller
f_s	Switching frequency	T_{out}	Temperature output of the system
P	Power to the thermal system	T_{ref}	Reference temperature
V	Voltage across the thermal load	$T_{measured}$	Measured temperature of the system
V_{in}	Input Voltage to Buck Converter	K_p	Gain of Proportional Controller
	Critical current of Inductor		
d_R	Output of the controller	K_i	Gain of Integral Controller
$k_e()$	Linearization function comprising PWM Output of *ke*	E_{bb}	Bang-Bang controller energy
d_A	Output of *ke*	E_{bc}	Buck converter controller energy
BC	Buck Converter	T_{Fdiff}	Bang-Bang controller temperature difference
c	Proportional Constant	T_{F1diff}	BC controller temperature difference

1.2 Problem Statement

Ideally, the load demand for a power system should be constant, which can be met by constant generation as well. Since consistent generation at a fixed level is not possible, utility companies must forecast and schedule generation for power demand. This causes power production to become expensive during peak hours of power consumption. Peak power times are times in which electricity is in highest demand and as a result the most expensive. In order to keep up with peak load demands, utility companies must adjust by producing more energy utilizing peak load power plants, such as hydro and gas plants, which generally contributes to the cost to produce power. Utility companies may also opt to purchase power from other utility companies within the same continental interconnection. Purchasing power from other sources also adds to the cost of power production. Most energy consumption of US single-family residential homes is from controllable appliances [1]. It would be cost beneficial if peak power demand was decreased in single-family residential homes.

This thesis focuses on reducing the peak power for Thermostatically Controlled Loads (TCL). There are two major methods for reducing the peak power for TCL, such as electric water heaters, air conditioners, refrigerators, heat pumps, etc. The classical Demand Side Management (DSM) method is demand (load) shifting or by utilizing energy efficient appliances. The demand shifting method requires households to utilize appliance during non-peak hours or by imposing pricing tariffs during peak hours. The main goal of this study is to minimize the cost of energy consumption for existing space heating appliances by improving energy efficiency and reducing peak power demand.

A typical controllable space heating appliance utilizes a simple two-state Bang-Bang feedback controller. This thesis describes a thermostatically controlled space heater using a Buck

Converter (BC) as a linear power actuator. A simulation based on the mathematical model of the proposed system is used to determine the performance of the proposed linear thermal power controller. The simulation of the space heater’s thermal circuit is based on a transfer function that is a second order mathematical model, which had been experimentally identified in [2]. The proposed linear control method is compared against the traditional Bang-Bang controller for saving energy and improving temperature disparity.

CHAPTER 2: LITERATURE REVIEW

This section focuses on the demand side management methods for TCL. The first work concentrates on a direct-load algorithm of electric water heaters. The second review item investigates temperature control through K-factor control approach. The work in this thesis is a continuation of the future work proposed in [2]. Although only the work in [2], [3], and [4] are present in this section, other related studies can be found in references [5] and [6].

2.1 Water Heater Load Potential

In [3] the direct-load control algorithm is presented to control a two-element electric water heater (EWH) for the purpose of load regulation. Major operational problems in power systems have been identified as increased ramp rates and capacity requirements. Some options that could provide fast-response ancillary services include pumped-hydro plants, flywheels, batteries, DSM, and distributed generation resources. DSM is the option that has been the minimally explored and utilized. Due to strict telemetry requirements, most of the participants in studies concerning DSM have been industrial consumers. Even though a majority of the studies focus on industrial consumers, a smart grid can provide more flexible tools to residential and commercial customers as highlighted in [3].

There are two control methods for DSM, direct and indirect load control. Direct load control is done through the utility company and gives the consumer little control. This is very efficient however, does not consider personal preferences. In contrast, indirect load control is set by the consumer or by appliances where peak hour usage is not allowed. For a Thermostatically Controlled Appliance (TCA) to be suitable for regulation, it must always remain in operation in order to be continuously monitored. That is the reason [3] concentrates on the direct control of EWH.

A conventional EWH in the United States has two heating elements and only one element can be turned on at a time. The water in the tank is divided into hot water on the top, cold water on the bottom, and a mixing layer in the middle. The layering in the tank occurs since water density varies with temperature. A thermostat is located in the hot water level and the cold-water level of the tank.

A modified circuit was used to control only the bottom heating element. The circuit checked the temperature of the water every one minute. Changes to the element were limited to five minutes so that there was not a regular off-on action. The average load of the EWH is a continuously changing curve, though the power load is an irregular pulse train. The modified circuit is shown in Fig. 2.1.

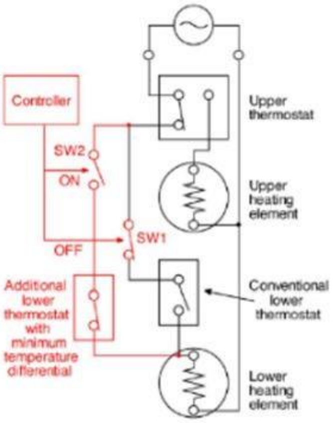

Controller
SW2
ON
SW1
OFF
Additional lower thermostat with minimum temperature differential
Upper thermostat
Upper heating element
Conventional lower thermostat
Lower heating element

Fig. 2.1: EWH modified circuit

The results of this study indicate that DSM with a smart grid is possible and could be monetarily rewarding for consumers. The simulations displayed that multiple EWHs using a direct-load control could provide a 2-MW regulation service as well as account for a customer with regular water consumption. The modeling results concluded that continuous 2-MW regulation service for 24-hour would take approximately 33,000 EWHs, however, the load regulation service between 6:00 a.m. to midnight, only requires 20,000 EWHs.

2.2 Improving the Centralized Control of Thermostatically Controlled Appliances by Obtaining the Right Information

This paper [4] focused on TCA control through two-way communication to provide load balancing services. With two way communications, it would not only provide the traditional on-off control, but also the control of rates of temperature increase and temperature decrease. By including the rates of temperatures, the central control could predict the temperature performance, therefore, reducing communication between the TCAs and central control. Linear model will be used to predict temperatures of TCAs.

Forecasting with a simplified linear graph as opposed to a more sophisticated exponential model could hurt the accuracy of the temperature especially considering that not all TCAs behave the same way. These results take into consideration that with larger numbers of TCAs, reducing the communication between the central control and the TCAs would be essential for performance improvement and data flow.

2.3 Thermal Load Characterization and Regulation

As noted in [2], there are two main types of power plants, base load plants and peak load power plants. Base load plants are manufactured to generate continuous reliable power at low cost. Examples of base load plants include coal, solar, and wind power plants. Peak load power plants have a quick ramp rate and are utilized during peak load hours.

Examples of peak load power plants include hydro and gas fired power plants. Utility companies typically use two primary methods to reduce peak power times. As mentioned previously, peak power times are times in which electricity is the most expensive due to high demand. Increasing production during peak power times normally increases the cost to produce electricity. Utility companies typically attempt to reduce peak power demand by pricing tariffs and/or encouraging customers to shift when they use electricity during peak hours. The second method that could be used is direct load control. An example of using direct load control would be changing the temperature set-point of consumer appliances directly by the power company. As stated earlier, for single family, residential homes, much of the energy consumption comes from controllable appliances. Some examples of controlled appliances include electric water heaters, ovens, air conditioning, refrigerators, and dishwashers. One significant method to promote a more leveled power demand is to draw power continuously, as opposed to drawing power in pulses. The proposed strategy in [2] replaced traditional Bang-Bang controller with the PI controller designed through K-factor approach on a thermal system model that was identified as shown in Fig. 2.2.

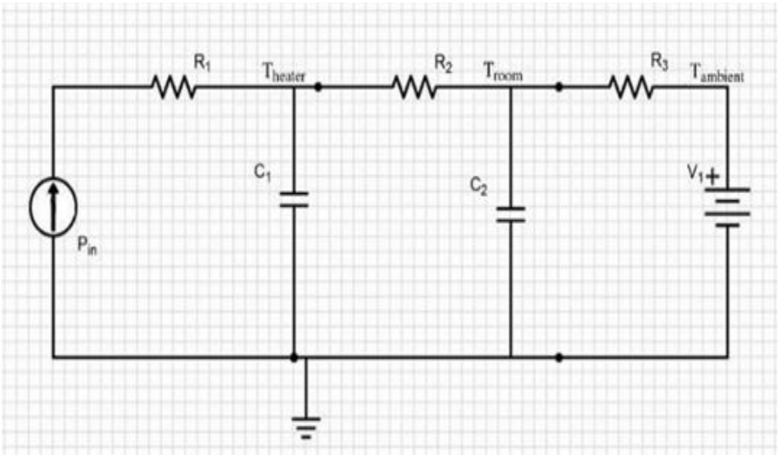


Fig. 2.2: Thermal system model

The actuator in this system was also a buck converter that regulated power to the load continuously, though in a non-linear fashion.

The conclusion reached in [2] claims that the methodology had five benefits compared to Bang-Bang controllers. The benefits were decreased peak power, evened ramp rates, eliminated in rush currents, continuous power, and improved temperature stability. The conclusion in [2] suggested future research should focus on improving the energy efficiency of TCLs.

CHAPTER 3: METHODOLOGY

Fig. 3.1 is a block diagram of the proposed control system in *s* domain. The feedback system consists of a proportional controller, signal conditioning system to linearize the BC, BC, the thermal system, and a temperature sensor. The input to the proposed feedback system is a setpoint temperature and the actual ambient temperature.

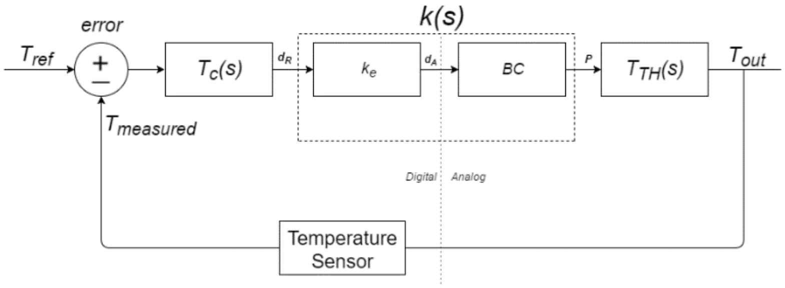


Fig. 3.1: Control system block diagram

3.1 Proportional Controller

A proportional controller is defined as a linear controller, in which the output is calculated by multiplying the error with a constant. The constant is known as the proportional gain K_p and is varied by how the system needs to react. Due to the fact that the controller system is a proportion of the error, this often causes steady-state error.

$$d_R(t) = K_p \times e(t) \tag{3.1}$$

The error of the feedback system $e(t)$ is calculated simply by subtracting the reference, or set point, temperature and the measured temperature of the system.

$$e(t) = T_{ref} - T_{measured} \tag{3.2}$$

3.2 Linearizing the BC Power Actuator

The BC takes a pulse width modulated (PWM) input signal d_A and produces an output voltage or power P. The BC used in the experimentation can function at a wide range of switching frequencies. However, higher frequencies were found to be prone to higher distortion. The BC output voltage is determined by the duty cycle of the PWM and the relationship is linear [2]. However, the BC input duty cycle vs. output power relationship is not linear. This is due to the fact that the power produced for a resistive thermal load is proportional to the squared value of voltage applied to the heater. This study aimed for a linear relationship between the input d_R (duty cycle), which determined the BC PWM duty cycle, and output heater power.

That is

$$P = c \times d_R \tag{3.3}$$

where c is a constant and P is output power of the BC.

Another important concern that impacts the linearity for a BC is discontinuous conduction mode (DCM). If the BC enters DCM, the voltage to PWM duty cycle ratio would no longer be linear.

The critical inductor current for the system can be found:

$$I_{Lcrit} = \frac{V_{in}}{2Lfs} D(1 - D) \tag{3.4}$$

where V_{in} is supply voltage of the BC, L is the BC’s inductor, f_s is BC switching frequency, D is

BC PWM duty cycle, and I_{Lcrit} is the BC critical inductor current. When the current through the BC inductor falls below I_{Lcrit} then the input and output of the BC is no longer linear.

The power used by the thermal system can be found using Ohm's law for power:

$$P = \frac{{V_{out}}^2}{R} \qquad (3.5)$$

where V_{out} output voltage of BC, R heater load resistance, and P is power.

A problem with this power formula of equation (3.5) as mentioned earlier, is that the equation does not represent a linear relationship between BC voltage and power. Therefore, a linear relationship between the power and duty cycle relationship of the system needs to be established by introducing a $k(s)$ transfer function.

$$k(s) = c \qquad (3.6)$$

Empirical evidence has shown that the BC voltage and d_A relation can be approximated by (3.7).

Fig. 3.2 shows the linear relationship between V_{out} and d_A determined through linear regression analysis.

$$V_{out} = 79.7d_A + 8.34 \qquad (3.7)$$

In order to obtain (3.6) or (3.3), d_A can be solved from (3.5) and (3.7) as follows

$$d_A = \frac{\sqrt{d_R \times c \times R} - 8.34}{79.7} \qquad (3.8)$$

Nonlinear relationship given by (3.8) practically defines k_e block shown in Fig. 3.1.

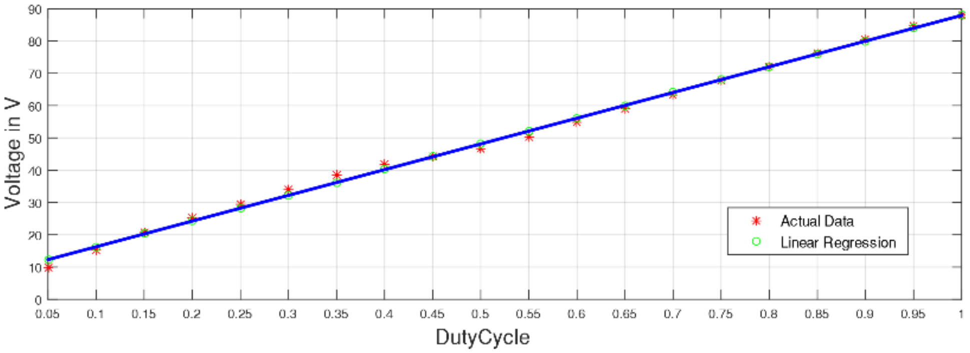


Fig.3.2: BC voltage V_{out} and d_A comparison of the proposed control system

The k_e function can therefore be represented as:

$$d_A = k_e(d_R) \tag{3.9}$$

Although k_e is a nonlinear function, it would result in a linear transfer function $k(s) = c$ when connected with the BC. Experimental analysis showed that the BC goes into DCM as dictated by (3.1) below 5 % duty cycle. As a result, the voltage to duty cycle ratio is no longer linear, thus, k_e block output was limited as such from 0.05 to 1.0.

Since d_R is a product of the error and the proportional controller gain, the error needs to be limited to only positive values. Otherwise, negative error would result in a complex d_R as can be seen in (3.8).

The constant c is selected as 88 to compensate for the full range of 0.05 to 1.0 for k_e. If c was inappropriately set, the BC would not be able to utilize the full range and the system's performance would suffer. The load resistance R is empirically determined as 86Ω. This value reflects the multimeter value of the resistive heater load of the system.

3.3 Temperature Averaging

Due to the noise in the real time data acquisition medium caused by various sources, such as electromagnetic interference and analog digital conversion, it is necessary to compensate for outliers in temperature sensing. Temperature sampling in this study occurs every 500 ms. For every sampling, an array is built by multiple back to back temperature readings. The elements in these array are later averaged using (3.10). These readings are compared against one another since temperature cannot change quickly between these readings. If the absolute error was greater than the allotted tolerance, the array element was set to 0, therefore, not included in averaging. If all elements of the array are set to zero, the average temperature is set to the old temperature sample in memory.

$$\bar{x} = \frac{1}{n}\sum_{i=0}^{n-1} x_i = \frac{1}{n}(x_0 + \ldots + x_{n-1}) \qquad (3.10)$$

3.4 Infinite Impulse Response (IIR) Filter

An IIR filter is also utilized to condition and smooth the data. The filter is applied to the system twice, for the acquisition of temperature and the output of *ke* to succor the performance of the BC. The transfer function of the first order IIR filter is constructed as follows

$$H(z) = \frac{Y(z)}{X(z)} = \frac{m}{1-(1-m)\,z^{-1}} \qquad (3.11)$$

where $X(z)$ and $Y\,(z)$ are the z-transforms of the input signal and output signal respectively.

In addition

$$m = \frac{T_s}{\tau + T_s} \qquad (3.12)$$

where $T\ s$ (sampling time) is 0.5 s, and τ (filter time constant) is 1.5 s. Since $\tau \geq 2T_{s,}$ Nyquist/Shannon sampling theorem guarantees no aliasing occurs.

Based on the values of τ and T_s , m can be calculated as follows:

$$m = \frac{0.5}{1.5+0.5} = 0.25 \tag{3.13}$$

Resulting discrete time filter equation can be written accordingly as,

$$y(n) = 0.25x(n) + 0.75y(n - 1) \tag{3.14}$$

The filter is designed to mostly consider the previous value of the data iteration, this could be adjusted accordingly by adjusting the time constant of the filter.

3.5 A Proportional Integral Controller

A proportional integral (PI) controller is a linear controller in which the output is calculated by adding the error multiplied by the proportional gain constant to the integral error multiplied by the integral gain constant (3.15). The cumulative integral error can be calculated as in (3.16).

$$d_R(t) = K_p \times e(t) + K_i * integral\_error \tag{3.15}$$

$$integral\_error = integral\_error + sampling\ time \times e(t) \tag{3.16}$$

When implementing the integral parameter of the PI controller the sampling time has to be multiplied by the error to discretize the term. By adding an integral part of the controller, the system’s steady-state error is eliminated, meaning the setpoint and measurement difference will be removed during the steady-state of the system.

3.6 Materials and Methods

An mbed LPC1768 microcontroller board based on Advanced RISC Machines (ARM) core was used to control the duty cycle of an amplified pulse-width modulated signal. This signal switches a high speed metaloxide semiconductor field-effect transistor (MOSFET). Switching of the MOSFET controls the DC output of a buck-boost converter voltage. The BC schematic is shown in Fig. 3.3. The voltage of the resistive heating element is ultimately regulated by the manipulation of the duty cycle in PWM signal generated by the ARM microcontroller.

An open-loop control system is a control system in which the control (regulating) action is independent of the output [7]. In addition, open-loop systems have no automatic correction to the output of the system [8]. Therefore, measuring the open loop system output voltage with no outside influence will give a base output voltage to duty cycle relation. Thus, voltage measurements to the resistive heating load by modifying the duty cycle were documented first. The empirical tested values of single-input, single output system (SISO) system shows the input to output ratio. In other words, the voltage to duty cycle relation could be represented by a line as shown in Fig. 3.2.

The next part of the research consisted of a few closed loop system techniques. A feedback-loop system is a process where the output of the system is continuously monitored and compared with the reference or set point [9]. The set point represents the desired value and, in this case, references a constant value representing a desired temperature of a heated space or enclosure (Fig. 3.4).

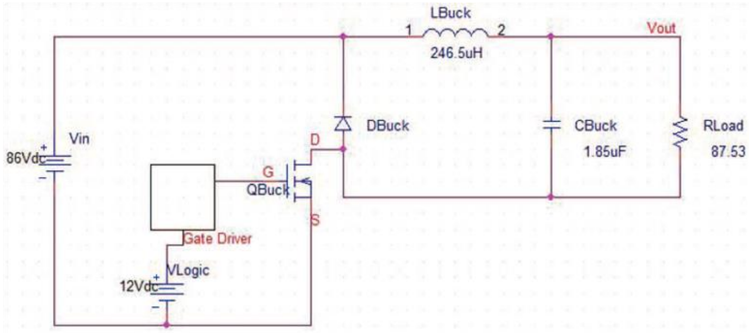


Fig. 3.3: Buck converter schematic [2]

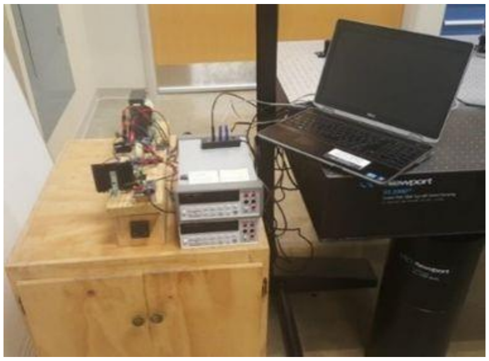


Fig. 3.4: Temperature controlled enclosure and experimental setup

The heated enclosure temperature is monitored by an MCP9808 temperature sensor. The heating element is kept in this enclosure to maintain the temperature isolated from room temperature. The temperature sensor has Inter-Integrated Circuit (I2C) communications capabilities and communicates with the mbed ARM microcontroller through the protocol. The microcontroller compares the set point to the temperature of the heated enclosure. The result comparison of the reference point to the current temperature of the system is called the error signal. Afterwards, the error signal can be used as control point in a close-loop system.

The system increases voltage if the current temperature is too low, likewise, if the existing temperature is too high for the system, the system reduces the voltage output. How the system increases and decreases the system voltage is exclusively dependent on the feedback loop strategy being implemented. A simplified close-loop system diagram for this project is illustrated in Fig. 3.5.

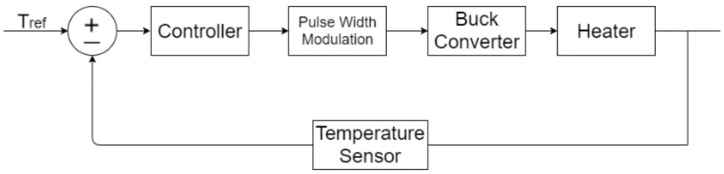


Fig. 3.5: Basic diagram of a closed-loop system

The mbed LPC1768 board provides a digital pulse-width modulation that could be set to significantly high frequencies that can efficiently drive the BC. The mbed board also provides a 32-bit ARM 96MHz Cortex that could be utilized for all the calculations required by the controller, filtering and linearization function k_e. In addition, an external circuit was built to drive the MOSFET at both the desired voltage and frequency. An example of how the duty cycle of the pulse-width modulation occurs can be seen in Fig. 3.6.

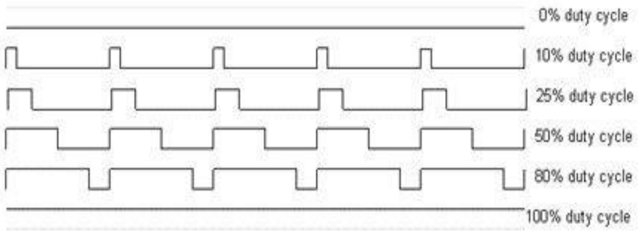


Fig. 3.6: Duty cycle percentages [9]

The MOSFET driver circuit has an operating voltage of 12V and requires a 5V PWM signal. The mbed microcontroller operates at 3.3V. However, the driver circuitry requires a PWM input of 5V. This was solved by including a Low-Power Dual-Channel Digital Isolator. The isolator takes the 3.3V PWM input and outputs a 5V PWM with the same frequency and duty cycle. The isolator also divides the microcontroller from the analog side of the circuit with a semiconductor isolation barrier. A 12V voltage regulator was added to regulate the voltage and output of the MOSFET driver. A 5V voltage regulator was also used to regulate the output of the Dual-Channel Digital Isolator.

Other miscellaneous items used in the circuit design include: terminal connectors for I2C connection and the 12V PWM driver IC, barrel connector for the power supply and a right-angle switch to turn on/off the power applied to the circuit. The schematic and the board layout can be seen in Fig. 3.7 and Fig. 3.8.

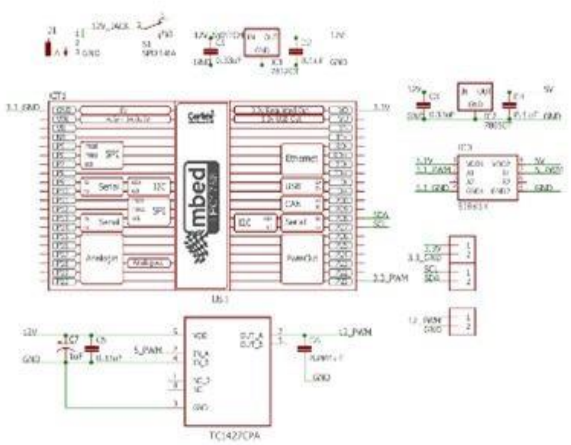


Fig. 3.7: Schematic layout of PWM MOSFET driver

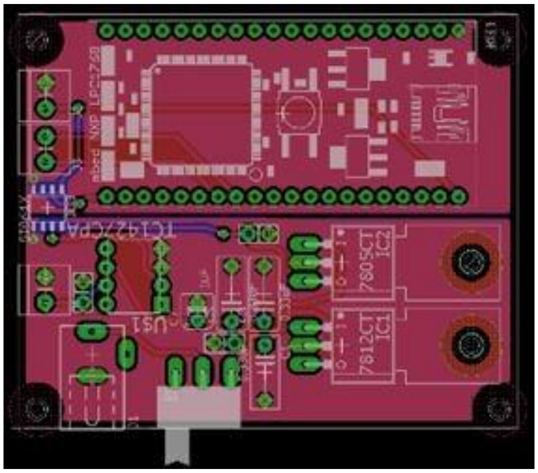


Fig. 3.8: Board layout of PWM MOSFET driver

Digital multimeters with data logging capability were chosen to measure the power inputs to the both Bang-Bang and BC systems. Agilent 34410A digital multimeters were utilized for this purpose to measure both DC/AC voltages and currents. The voltage and current values were logged to a local laptop PC through a RS232 serial connection. The procedure to measure both Bang-Bang controller and Buck converter voltages/currents is shown in Fig. 3.9.

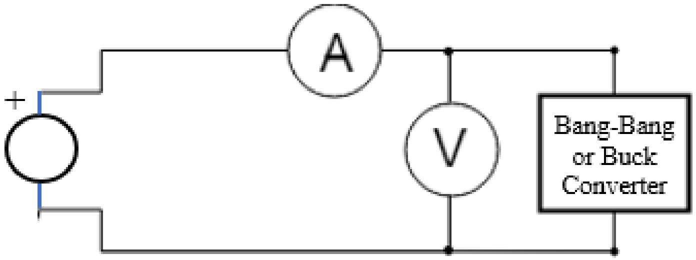
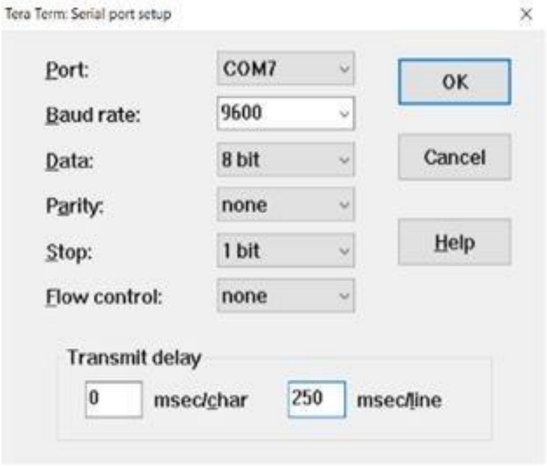


Fig. 3.9: Power measurement scheme for bang-bang or buck converter

To utilize the serial communication feature from the digital multimeters, a few accommodations were made, considering that most modern computers do not have built in serial interface. A 9-pin null male to male connector is attached to the multimeter, followed by a Universal Serial Bus (USB) Serial connector to the computer. The physical setup can be seen in Fig. 3.4. To enable serial communication on an Agilent multimeter, 'Talk Only' mode must be enabled and set. This is achieved by setting GPIB address to '31' in the front-panel I/O Menu of the multimeter. Subsequently, the RS232 communication protocol must be configured and set. The I/O menu can set the baud rate, data bits, parity and stop bits.

On the computer side, serial communication is usually configured and connected through software. When the RS232 USB interface connector is used, it is important to confirm the correct communication (COM) port is utilized by the computer to communicate to the multimeters. It is also essential that the software being utilized supports logging of data. Most modern serial communication tools support this feature. An example of a computer software serial configuration in software is shown in Fig. 3.10.



Port: COM7
Baud rate: 9600
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none

Transmit delay
0 msec/char 250 msec/line

Fig. 3.10: Software serial COM configuration

CHAPTER 4: RESULTS

The results investigate the different solutions that were collected through the implementation of proposed methods. The first section analyzes the linearization of the system. The second section compares the performance of the controller with BC and the traditional Bang-Bang controller. Two main factors that are examined consist of temperature disparity and more importantly energy consumption. The third section focuses on modeling the temperature system utilizing second order modeling functions. And finally, the results explore the validation of the thermal models developed via simulations in MATLAB/Simulink environment.

4.1 Linearization

A focus of this research is the linearization of the thermal power generation for the purpose of energy saving. The input of the buck converter d_R must be linearly related to the output power P of the buck converter. The theoretical BC power curve based on controller's output d_R can be seen in Fig. 4.1.

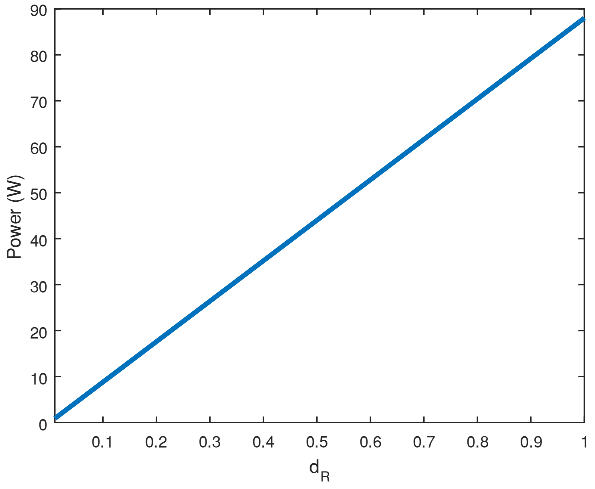


Fig. 4.1: Theoretical BC power curve based on controller's output d_R

Fig. 4.2 displays how the voltage (in blue) reacts to d_R input of the system. It is clear that the voltage to d_R ratio is not consistent and linearity is questionable. However, the power response to d_R ratio is approximately constant as expected or linear after 900 s. Due to the limitations in BC's output voltage (or 100% limit in duty cycle), early part of Fig. 4.2 before 900 s is variable for both cases and linearity cannot be justified. This is primarily due to the fact that the voltage output of the BC cannot keep up with the proportional controller's output and hits the upper saturation limit.

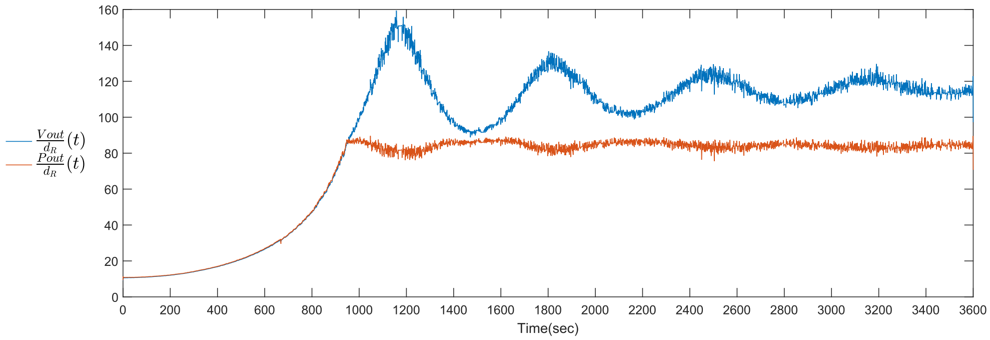


Fig. 4.2: BC voltage V_{out} to d_R ratio (blue) and power to d_R ratio (red)

4.2 Comparison to traditional heating method

Bang-Bang controller is the traditional heating method application. A Bang-Bang controller processes the error when the current temperature is compared to the set point temperature, which is selected by the user. A Bang-Bang controller has two states, ON or OFF. It is a basic feedback system, if the current temperature is less than the set point plus or minus the tolerance the system will be ON. In comparison, if the system meets or exceeds the desired temperature the system will be in the OFF state. Because of the natural temperature dissipation, the temperature will fall and cool the system, eventually the temperature would continue to decrease until the Bang-Bang controller will need to change to the ON state. The time and temperature that the system utilizes to warm up and cool down is called the system cycle. The Bang-Bang controller is often used due to the simplicity of the control mechanism.

To compare the Bang-Bang controller and the P-Only controller utilizing the buck converter fairly, the Bang-Bang controller was set to an initial temperature. Unfortunately, because of the nature of the Bang-Bang controller and the built-in tolerance of the controller, the temperature achieved is not really equal to the set temperature when this controller was utilized. To deal with this inconsistency, an average temperature of the heated space was calculated through the whole heating and cooling cycle.

After the averaging procedure, the temperature set point for the buck converter heating application was decided. It is important to note that proportional controller cannot also achieve perfect tracking for reference temperature. Therefore, a trial and error process for proper reference temperature selection was applied. This process directs the equality of temperature integrals for the duration of the test.

The comparison of the Bang-Bang controller and the BC temperature profiles for 1 hour duration can be seen in Fig. 4.3. The results demonstrated that the temperate profile is approximately constant for the BC at steady state with less temperature disparity.

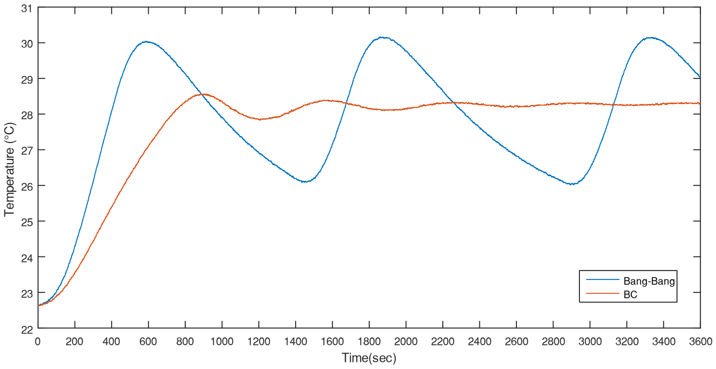


Fig. 4.3: Temperature profiles with bang-bang and buck converter

The power drawn for both systems was measured employing the Agilent 34410A, as discussed previously. The placement of measurement devices for both controllers is critically important for a fair judgement. The digital multimeters for the Bang-Bang controller (Love Series 16A) is set directly before the semiconductor triac switch and measured rms current and voltage values. Meanwhile, the digital multimeters for the buck converter's input power measurement were placed right after 86V DC source in reference to Fig. 3.3 and Fig. 3.9. In this case, the meters were configured for DC current and voltage measurement. Logged voltage and current values were eventually multiplied to calculate active/average power drawn. A sample power input profiles for both cases can be seen in Fig. 4.4.

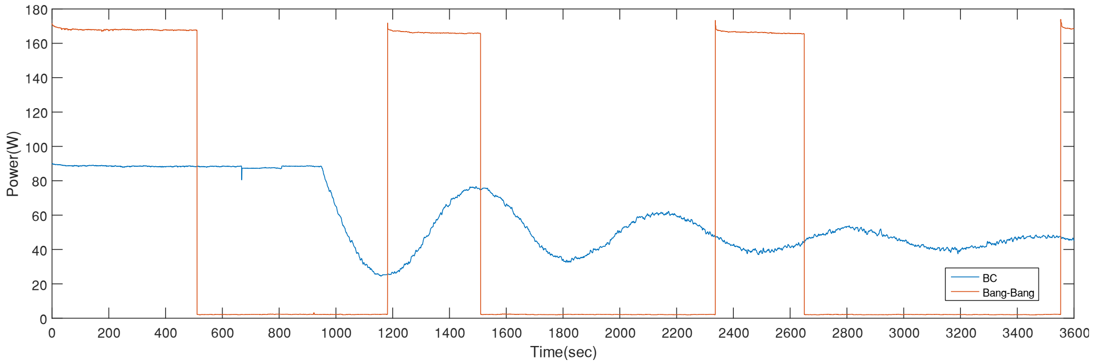


Fig. 4.4: Power profiles with bang-bang and buck converter

4.3 Thermal Modeling

The thermal system was identified through the second order mathematical model used in the prior research [2]. Identified thermal circuit equivalent for this model can be seen in Fig. 4.5. R_1 cannot be identified due to the construction of the model. The model input/output variables include electrical power input P_{in}, heated space temperature output T_{room}, and ambient temperature input $T_{ambient}$. The identified system parameters are used to calculate temperature output T_{room} based on given temperature input $T_{ambient}$ and electrical power input P_{in}.

Fig. 4.6 displays how the physical system response and the model compare to each other. Quality of fit between these two waveforms is 93.2%, as shown in Fig. 4.6. According to the figure, the simulated and the actual system transients closely match while the responses towards steady state also compare reasonably well.

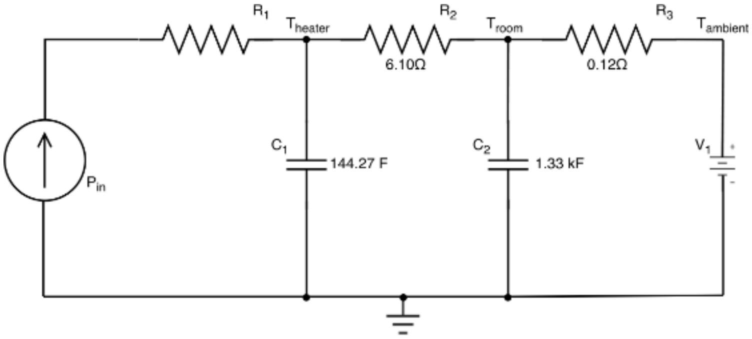
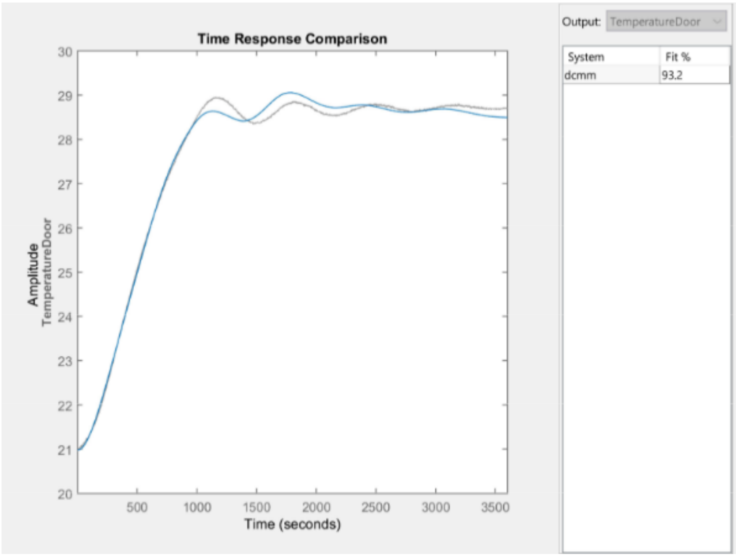


Fig. 4.5: Identified thermal circuit model



Output: TemperatureDoor

System	Fit %
dcmm	93.2

Fig. 4.6: Modelled (blue) vs actual heated space temperature (gray)

4.4 Model Validation

Utilizing the identified thermal circuit model, a feedback loop controller model was designed utilizing Simulink, and can be seen in Fig. 4.7. The model duplicates all parts of the experimental controller system. The first block from the left is the controller, which accounts for the linearization function of the system. A breakdown of the controller subsystem titled "P-Controller" can be seen in Fig. 4.8. Since the proportional controller is only multiplying by one, the proportion was left out of the current subsystem.

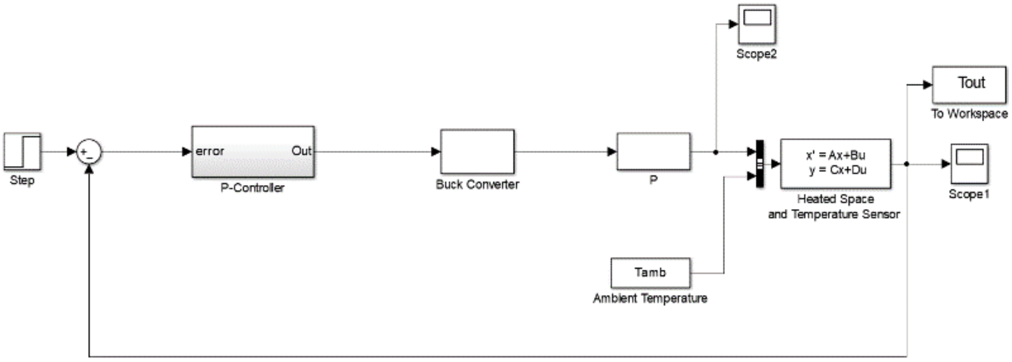
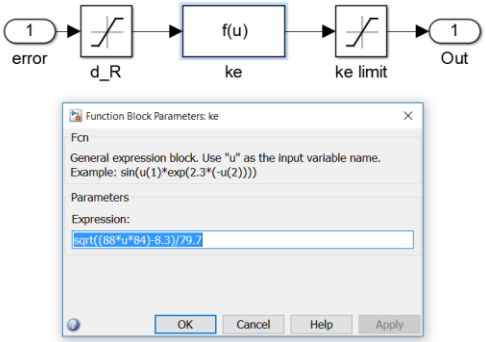


Fig. 4.7: Simulink block diagram



Function Block Parameters: ke

Fcn

General expression block. Use "u" as the input variable name.
Example: sin(u(1)*exp(2.3*(-u(2))))

Parameters

Expression:

sqrt((88*u*84)-8.3)/79.7

Fig. 4.8: P-controller with linearization function

The first saturation block (d_R) in the subsystem limits the input to positive values for the linearization block k_e to account for the square root function. After the linearization block, the k_e saturation block is used. This block models the real constraints of the BC. The output is limited from 5 to 100 percent of the duty cycle as explained in Section 3.2.

The next blocks in the model simulate the linear gain of the BC, as well as the final output power conversion of the system. The inputs of the thermal model identified earlier includes the electrical power by the BC as well as the ambient temperature. A final simulated output of the system is the temperature for the heated space and is shown in Fig. 4.9. Along with the simulated temperature Fig. 4.9 also plots the real temperature response of the system for the same inputs, displaying the similarities between real and simulated temperature. The earlier part of the transient state of the simulated system fitted the real data almost perfectly. The later part of transient state response had some minor variation. Steady state response also matched each other very closely. This simulation will later be used to determine how the system will react to different controller algorithms and techniques. A real test takes about an hour to complete, with the model developed it will be known how the system approximately reacts almost instantaneously. The model can run a simulated test multiple times to ensure that the results are evaluated before a real test is attempted.

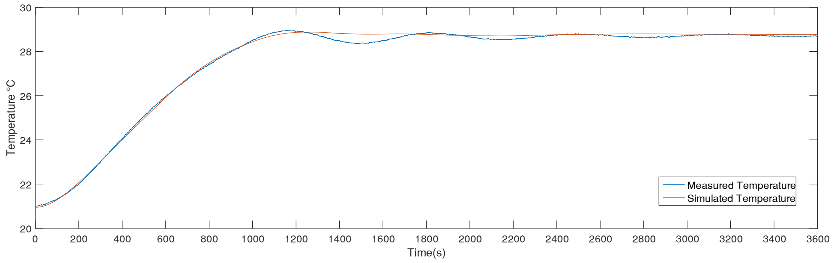


Fig. 4.9: Simulated vs measured temperatures

4.5 Simulink PI Modeling

Utilizing the Simulink model of the proportional controller and the thermal system, different parameters for the system can be tested and simulated. A Proportional Integral (PI) controller is simulated by replacing the P-Only controller in the model and replacing it with the PI controller block.

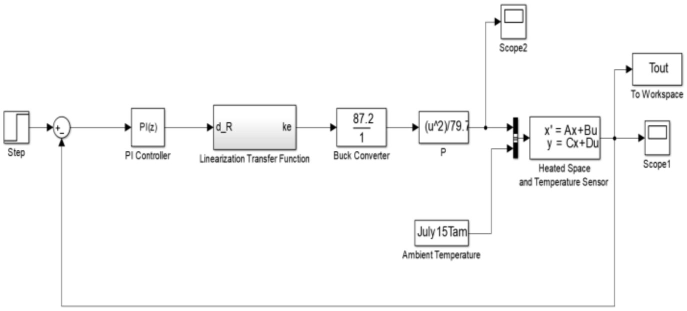


Fig. 4.10: PI Controller with linearization function

4.6 Manual Tuning of PI Controller

Manually tuning is used to improve the performance of the PI controller, the PI controller improvements to the system are reducing the overshoot and steady-state error while reducing the settling time of the controller. The two parameters being manually tuned of the PI controller are mutually influenced by the other, therefore, proper adjustments are important. Table 4.1 shows the effects of increasing the parameters individually of the controller.

Table 4.1: PI individual parameter effects [15]

Parameter	**Rise Time**	**Overshoot**	**Settling Time**	**Steady State Error**
$\boldsymbol{K_p}$	Decrease	Increase	Minor Change	Decrease
$\boldsymbol{K_i}$	Decrease	Increase	Increase	Eliminate

When correcting the overshoot of the system both proportional and integral gains influence the response, the proportional gain has a more direct effect on the overshoot. The integral gain is important in the elimination of steady-state error, while negatively affecting the setting time of the system.

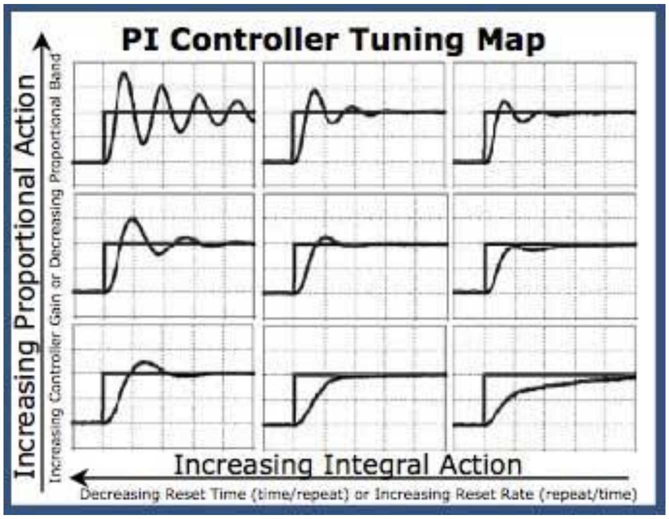


Fig. 4.11: PI tuning map [16]

When considering the effects of the gains, the parameters must be reduced accordingly. The effects of gains of the controller on the output temperature of the system can be summarize in Table 4.1. The result of manually tuning the controller can be summarized in Fig. 4.11. The final parameters utilized by the PI are K_p is equal to 0.25 and K_i is equal to 0.009.

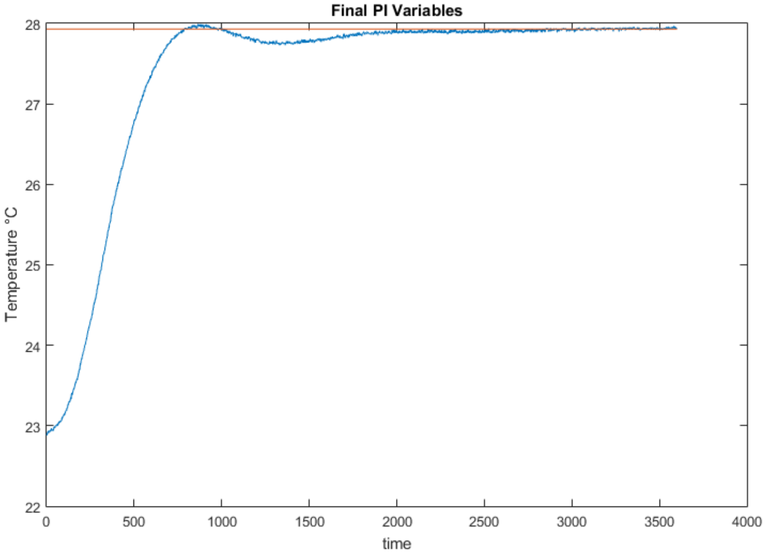


Fig. 4.12: PI results utilizing Manual Tuning of Controller Variables

To validate the veracity of the Simulink model, the final parameters of the PI controller are substituted into the model. The resulting simulated and measured temperatures comparison can be seen in Fig. 4.13. The transient state response and steady state response of the simulated and actual systems matched each other comparably well.

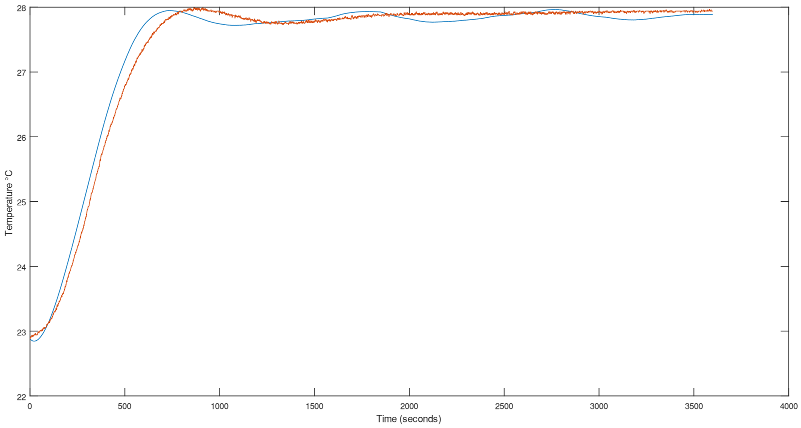


Fig. 4.13: PI Simulated (blue) vs measured temperatures (red)

4.7 Pulse-width modulation switching frequencies

Originally, the testing frequency of pulse-width modulation was much higher than final frequency used. But operating the mbed microcontroller at higher frequencies prove too much of an endeavor for the ARM processor. Multitasking proved difficult amidst the dual I2C temperature sensors, the feedback controller, the linearization function, RS232 communication, the pulse-width modulation and the real-time clock (RTC) for the timing. The accumulation of all these processes and especially the problem of switching noise caused by the BC, were potential reasons why the microcontroller hangs up at high switching frequencies over 30-40kHz.

Complications constitute RS232 communication improper intermittent transmission, timing error, unacceptable temperature readings, which eventually cause heavily jittery power output. Lowering the pulse-width modulation frequency was ultimate solution to the problem. The initial proportional controller test was successfully using the default period of 0.020 second pulse-width modulation of the microcontroller. The remaining proportional and PI controller tests, which were implemented later are successfully tested using the pulse-width modulation period of 0.00005 second or 20 kHz frequency.

4.8 Energy Savings

4.8.1 Energy Savings Calculation

A MATLAB script is used to calculate and compare the temperature and energy consumption of both Bang-Bang and BC controllers. The script utilizes imported column vectors of time, temperatures, voltage and electric current of both tests. The measured voltage and current value vectors were multiplied to calculate the power output to the resistive thermal load. To compensate for vector size disparities, a data interpolation in combination with a linearly spaced time vector is implemented. A trapezoidal numerical integration of the power with respect to the time vector is applied to calculate the total usage of the system. A similar, additional trapezoidal numerical integration is used for both the internal and ambient temperature of the tests. The ambient is subtracted from internal temperature to calculate a quantity proportional to the total heat escaping the system. The relative power is used as multiplier to normalize the BC to Bang-Bang temperature disparities.

The energy savings formula used in this study can be given by:

$$Energy\ Savings \ = \ 100 \times \frac{E_{bb} - (E_{bc} \times \frac{T_{Fdiff}}{T_{F1diff}})}{E_{bb}} \qquad (4.1)$$

where E_{bb} is Bang-Bang controller's energy consumption, E_{bc} is the BC's energy consumption, T_{Fdiff} is Bang-Bang controller's heat escape factor, and T_{F1diff} is the BC's heat escape factor.

4.8.2 P-Only Energy Savings

Energy savings are found in Test 1 through Test 3 utilizing the P-Only controller. All P-Only test were ran with a K_p gain of 1. Test 1 through Test 3 were done during similar overnight times which equates to similar ambient temperatures. Test 1 and Test 2 were ran using the default mbed frequency of 50Hz. Test 3 and Test 4 were ran using 20 kHz frequency.

Table 4.2: P-Only Energy Savings

Test	***T_{fdiff} (°C)***	***T_{f1diff} (°C)***	***E_{bb} (W)***	***E_{bc} (W)***	**Savings (%)**	**Duration (Hrs.)**
1	4.8714	5.8955	30.9028	30.6415	8.6144	17
2	4.8714	5.2848	30.8554	30.6663	8.3870	17
3	5.5526	5.6589	33.2108	32.9844	2.5465	15
4	5.9000	6.3124	31.9060	35.0746	-2.7529	15

Test 4 Bang-Bang controller and BC (P-controller) test was conducted on a different time of day from Test 1 through Test 3, resulting in an overall increase of temperature differences compared to the rest of the P-Only tests. Recalculating all energy savings by removing the temperature differences multiplier from the Bang-Bang controller and BC controller yields a positive energy savings for all tests; E_{bc} average power is more efficient than E_{bb} average power in all P-Only tests, however, the temperature differences in Test 4 results in a negative percent savings. The average energy savings calculating all four test seen in Table 4.2 equate to 4.1986 percent savings.

4.6.2 PI Energy Savings

Test 5 through Test 8 were implemented utilizing PI controller. All PI controller tests compared to equivalent Bang-Bang controller tests yield a positive energy savings. All PI test results can be seen in Table 4.3.

Table 4.3: PI Energy Savings

Test	T_{fdiff} (°C)	T_{f1diff} (°C)	E_{bb} (W)	E_{bc} (W)	**Savings (%)**	**Duration (Hrs.)**
5	5.4308	4.8914	45.67	40.63	1.22	1
6	5.1555	4.8033	28.86	25.94	4.44	48
7	4.9905	5.9936	27.10	31.24	4.02	48
8	5.5526	5.9824	33.21	31.83	11.05	15

Test 5 is a one hour test and it represents the savings resulting from the perspective of rise time transients of the thermal system. Test 6 and Test 7 are ran for 48 hours to illustrate the savings over multiple days. Test 8 operating time is similar to the running duration of the P-Only tests, which is useful in the comparison of controller methodologies.

CHAPTER 5: DISCUSSION

5.1 Ambient Temperature

Overall, the tests led to an inconclusive result, even though there is a positive trend toward energy savings. One of eight tests resulted in energy loss, further test and investigation must be conducted associated with this negative energy savings. One factor that contributed to the uncertainty of the results was the ambient temperature that the enclosure exposed to. The user has very little control of the ambient temperature of the room, the ambient temperature has many components that could lead to variations. Including the amount of people in the ambient room, the current yearly season, and facilities changes to the overall temperature patterns cause by school vacations and weekends, to name a few. The factor of ambient temperature could be resolved by utilizing an environmental chamber or a room that the ambient temperature could be controlled at all times; Unfortunately, Western Carolina University does not have access to such facilities. If an environmental facility was to be utilized, another consideration must be realized by moving the enclosure; a new thermal system model must be calculated and taken into account with new calculations. A simplified solution could be to raise the internal setpoint of the system, by using a higher internal setpoint the ambient temperature factor would decrease in the calculations, perhaps, using this method the energy savings calculations consistency would increase clarifying the energy savings fluctuation.

5.2 Proportional, Integral and Derivative controller

Though, the P-Only controller results were adequate for temperature control because of its simplicity and robustness, P-Only controller has notable weakness that is succored by deploying an integral component. When P-Only controller is implemented, the result will always have a steady-state error, this is because a P-Only controller is not equipped to handle and account for temperate loss in the system. The implementation of a P-Only controller will create a temperature setpoint offset from actual temperature output of the system. To compensate for the temperature offset of the controller the temperature setpoint must be set higher than the desired temperature, which is an undesired process for the controller.

Consequently, for the controller temperature output to match the temperature setpoint, an integral component must be added to the controller. The integral component of the controller sums the historic error of the controller and remedies the shortcomings of the P-Only controller by utilizing the integral gain to eliminate the steady-state error. The PI controller reduces the overshoot of the system without sacrificing the rise time controlled by the proportional gain of the controller, although it can increase the settling time of the system. However, if the proportional gain is set too high, the system will oscillate around the setpoint. The two gains interact and even oppose mutually amid adjustments when searching for the desired response of the thermal system.

Additionally, a derivative component can be added to controller. The derivative term of the PID controller calculates the slope or rate of change of the error over time and utilized as a term to predict the response of the system, making the system more flexible to sudden changes in the temperature. Also, the inclusion of the derivative term allows for a more aggressive tuning of the proportional and integrative gains without the inclusion of an overshoot [16]. Unfortunately, the derivative component reacts poorly to the noise of the system, amplifying the noise and creating an overall unstable system. An adaption of the PID controller was tested but considered unfeasible to the control of the thermal system when both manual tuning and the Ziegler–Nichols tuning method yielded unstable results. Considering that most systems are inherently noisy the derivative component is often not used, making the PI controller the most common configuration of the controller in industry [16].

5.3 Energy Calculations

To better understand the energy calculation formula (4.1) can be rewritten as follows:

$$Energy\ Savings\ =\ 100 \times (1 - \frac{E_{bc} \times T_{fdiff}}{E_{bb} \times T_{f1diff}}) \tag{5.1}$$

The equation 5.1 is used to calculate and compare the BC power and the Bang-Bang controller power. T_{fdiff} and T_{f1diff} is accounted into the formula to account for ambient temperature differences among the tests. When all components of the formula are equivalent, the energy savings calculates to a savings of zero. In the same fashion, a positive energy savings translates to the BC being more efficient, therefore using less energy than the Bang-Bang controller. Similarly, a negative energy savings renders the Bang-Bang converter more efficient, the energy savings is represented negatively because the energy comparison is in perspective of the BC vs the Bang-Bang controller. Additionally, an undefined solution is possible by the formula when formula variables are equal to zero. This can be achieved by not activating either or both controllers, this in turn makes both numerator and denominator of the fraction zero causing an undefined solution. Therefore, a constraint of utilizing the energy savings formula is that both controllers should be running for heating of the enclosure.

Ultimately, if further tests are conducted the linearized BC controller could decrease the cost of energy consumption for existing space heating appliances by improving energy efficiency and reducing peak power demand that controllable appliances such as heaters cause on the grid. The BC feedback controller would also maintain the temperature level of enclosure rooms more comfortable by decreasing the temperature displacement and fluctuation that is caused by the traditional Bang-Bang controller. Also, by utilizing other controller methods in conjunction with Buck Converter could further increase the performance of the system. Other systems could be modeled in Simulink before real life implementation to both confirm that the new control method behaves adequately and most importantly simulate almost instantaneously a multi hour test that could save the researcher valuable research time.

CHAPTER 6: CONCLUSION AND FUTURE WORK

A linear thermal power controller was designed and implemented by involving various technical aspects such as power electronics, digital signal processing, and linear control theory. The buck converter with a linearization mechanism was successfully tested and validated for expected behavior through experiments. In addition, the new heater control framework reduced temperature disparity significantly. A second order thermal model utilizing an equivalent circuit was also identified and validated using Simulink simulations and physical experiments. Future work will include a developed data collection method for energy consumption and comparison between two controllers. The ambient temperature differences led to inconclusive data, an environmental chamber would alleviate this variable. Due to the linearity of control shown in this work, other feedback control methods such an optimal control method can be investigated for efficiency improvement in the future. Furthermore, higher order linear filtering methods such as Butterworth or Chebyshev filters will be tested for performance improvement in comparison to the IIR filter. Ultimately, the most efficient and reliable controller can be achieved by using a more robust microcontroller and properly designed temperature control framework.

REFERENCES

[1] "Energy Use in Homes - Energy Explained, Your Guide To Understanding Energy - Energy Information Administration", eia.gov, 2017. [Online]. Available: https://www.eia.gov/energyexplained/index.cfm/data/index.cfm?page=us_energy_homes. [Accessed: 10- Apr- 2017].

[2] H. Karayaka, L. Holland, M. Tanaka and A. Ball, "Power Systems: Thermal Load Characterization and Regulation", Encyclopedia of Energy Engineering and Technology, Second Edition, pp. 1-23, 2014.

[3] D. Hammerstrom, N. Lu and J. Kondoh, "An evaluation of the water heater load potential for providing regulation service", IEEE Transactions on Power Systems, vol. 26, no. 3, pp. 1309-1316, 2011.

[4] M. Vanouni and N. Lu. "Improving the centralized control of thermostatically controlled appliances by obtaining the right information", IEEE Transactions on Smart Grid, vol. 6, no. 3, pp. 946 – 948, 2015.

[5] L. Holland, H. Karayaka, M. Tanaka and A. Ball, "An Investigation of Parametric Load Leveling Control Methodologies for Resistive Heaters in Smart Grids", 2014 Sixth Annual IEEE Green Technologies Conference, 2014.

[6] L. Holland, H. Bora Karayaka, M. Tanaka and A. Ball, "An empirical method for estimating thermal system parameters based on operating data in smart grids", ISGT 2014, 2014.

[7] S. Gupta, Elements of control systems, 1st ed. Upper Saddle River, N.J.: Prentice Hall, 2002.

[8] N. Macia and G. Thaler, Modeling and control of dynamic systems, 1st ed. Clifton Park: Thomson Learning, 2005.

[9] H. Bischoff and D. Hofmann, Process Control System, 1st ed. Dresden: Festo Didactic GmbH & Co., 1997.

[10] "Arduino - Secrets Of Arduino PWM", Arduino.cc, 2017. [Online]. Available: https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM. [Accessed: 24- Apr- 2017].

[11] "Arduino - Compare”, Arduino.cc, 2017. [Online]. Available: https://www.arduino.cc/en/Products/Compare. [Accessed: 21- Apr- 2017].

[12] M. Vanouni and N. Lu, "Improving the Centralized Control of Thermostatically Controlled Appliances by Obtaining the Right Information", IEEE Transactions on Smart Grid, vol. 6, no. 2, pp. 946-948, 2015.

[13] Control Systems Theory With Engineering Applications, 1st ed. Birkhauser, 2013.

[14] O'Dwyer, Handbook of PI and PID controller tuning rules, 3rd ed. London: Imperial College Press, 2009.

[15] Zhong, “PID Controller Tuning: A Short Tutorial”, 2006. [PowerPoint]. Available http://saba.kntu.ac.ir/eecd/pcl/download/PIDtutorial.pdf [Accessed: 3-Jan-2018].

[16] “PID Turning in Distributed Control Systems”, Yokogawa, 2010. [Online]. Available https://www.yokogawa.com/library/resources/white-papers/pid-tuning-in-distributed-control-syst ems/ [Accessed: 15-Jan-2018].

APPENDIX A: mbed P-Only Code

```
#include "mbed.h"
#include "FastPWM.h"
#include "MCP9808.h"

FastPWM fastpwm(p21);
Serial pc(USBTX, USBRX);
Ticker timer;

MCP9808::MCP9808(PinName sda, PinName scl) : i2c(sda,
scl)
{
}

MCP9808 sensor(p28,p27);

// read temperature from MCP9808
float MCP9808::internal_readTemp()
{
    data_write[0] = MCP9808_REG_TEMP;
    i2c.write(0x30, data_write, 1, 1); // no stop
    i2c.read(0x30, data_read, 2, 0);

    if(data_read[0] & 0xE0) {
        data_read[0] = data_read[0] & 0x1F;  // clear
flag bits
    }
    if((data_read[0] & 0x10) == 0x10) { // < 0 C
        data_read[0] = data_read[0] & 0x0F;
        tempval = 256 - (data_read[0] * 16) +
(data_read[1] / 16.0);
        tempval = tempval * -1;
    } else { // > 0 C
        tempval = (data_read[0] * 16) + (data_read[1] /
16.0);
    }
    return tempval;
}

float MCP9808::ambient_readTemp()
```

```
{
    //i2c.frequency(400000);
    data_write[0] = MCP9808_REG_TEMP;
    i2c.write(0x32, data_write, 1, 1); // no stop
    i2c.read(0x32, data_read, 2, 0);

    if(data_read[0] & 0xE0) {
        data_read[0] = data_read[0] & 0x1F;  // clear
flag bits
    }
    if((data_read[0] & 0x10) == 0x10) { // < 0 C
        data_read[0] = data_read[0] & 0x0F;
        tempval = 256 - (data_read[0] * 16) +
(data_read[1] / 16.0);
        tempval = tempval * -1;
    } else { // > 0 C
        tempval = (data_read[0] * 16) + (data_read[1] /
16.0);
    }
    return tempval;
}

float R = 84;
float k = 88;

float kp = 0.25;
float ki = 0.009;
float old_error = 0;
float integral = 0;

float setpoint = 27.83;
float ke;
float temperature_internal;
float temperature_ambient;
float old_ke = 1.00;

float count=0;
float sampling_period=0.5;

void attime()
{
    count=count+sampling_period;
}

int main()
{
    float old_count=0.1;
    float setFrequency=96000/20; //set to 50kHz,
clock/frequecy
    fastpwm.period_ticks (setFrequency); //setup the
period for 150Khz

    /*********** Initial Temperature Internal
********************/
    int p_inter=0;
    float initial_temp_inter [4]= {};
    float temp_m_inter[4]= {};

    while(p_inter<=3) {
        int m_inter=0;
        while(m_inter<=3) {

temp_m_inter[m_inter]=sensor.internal_readTemp();
            m_inter++;
        }

        float
sum_array_m_inter=temp_m_inter[0]+temp_m_inter[1]+temp_
m_inter[2]+temp_m_inter[3];
        initial_temp_inter[p_inter]=
(sum_array_m_inter)/4;
        p_inter++;
    }
    /*********** Initial Temperature External
********************/
    int p_exter=0;
    float initial_temp_exter [4]= {};
    float temp_m_exter[4]= {};

    while(p_exter<=3) {
        int m_exter=0;
        while(m_exter<=3) {
```

```
temp_m_exter[m_exter]=sensor.ambient_readTemp();
            m_exter++;
        }

        float
sum_array_m_exter=temp_m_exter[0]+temp_m_exter[1]+temp_
m_exter[2]+temp_m_exter[3];
        initial_temp_exter[p_exter]=
(sum_array_m_exter)/4;
        p_exter++;
    }
    /*********** Initial Temperatures Print
********************/
    float old_temp_inter= (initial_temp_inter
[0]+initial_temp_inter [1]+initial_temp_inter
[2]+initial_temp_inter [3])/4;
    float old_temp_exter= (initial_temp_exter
[0]+initial_temp_exter [1]+initial_temp_exter
[2]+initial_temp_exter [3])/4;
    pc.printf("Internal: %.2f\r\n",old_temp_inter);
    pc.printf("Ambient: %.2f\r\n",old_temp_exter);
    wait(1.0);

    timer.attach(&attime, sampling_period);
    while (count <=172800) {
        if(count != old_count) {
            /*********** Temperature Internal
********************/
            float temp_inter [8]= {};
            int i_inter=0;
            int zeros_inter=0;

            while (i_inter<=7) {

temp_inter[i_inter]=sensor.internal_readTemp();
                if(abs(temp_inter[i_inter]-
old_temp_inter)>.25) {
                    temp_inter[i_inter]=0;
                    zeros_inter++;
                }
```

```
                    i_inter++;
                }
                float
sum_array_inter=temp_inter[0]+temp_inter[1]+temp_inter[
2]+temp_inter[3]

+temp_inter[4]+temp_inter[5]+temp_inter[6]+temp_inter[7
];
                float
temp_average_inter=sum_array_inter/(8-zeros_inter);
                if (zeros_inter==8) {
                    temp_average_inter=old_temp_inter;
                }
                /*********** Temperature External
********************/
                float temp_exter [8]= {};
                int i_exter=0;
                int zeros_exter=0;

                while (i_exter<=7) {

temp_exter[i_exter]=sensor.ambient_readTemp();
                    if(abs(temp_exter[i_exter]-
old_temp_exter)>.25) {
                        temp_exter[i_exter]=0;
                        zeros_exter++;
                    }
                    i_exter++;
                }
                float
sum_array_exter=temp_exter[0]+temp_exter[1]+temp_exter[
2]+temp_exter[3]

+temp_exter[4]+temp_exter[5]+temp_exter[6]+temp_exter[7
];
                float
temp_average_exter=sum_array_exter/(8-zeros_exter);
                if (zeros_exter==8) {
                    temp_average_exter=old_temp_exter;
                }
```

```
temperature_internal=0.25*temp_average_inter+0.75*old_t
emp_inter;

temperature_ambient=0.25*temp_average_exter+0.75*old_te
mp_exter;
            /*********** PID Controller
********************/
            float error = setpoint-
temperature_internal;
            float integral = integral + error *
sampling_period;
            float d_R = kp*error + ki*integral;

            if (d_R < 0.01) {
                d_R = 0.01;
            }

            ke = (sqrt(k*d_R*R)-8.3)/79.7;

            if (ke < 0.05) {
                ke = 0.05;
            } else if (ke > 1.0) {
                ke = 1.0;
            } else {}

            float newke=0.25*ke+0.75*old_ke;
            fastpwm.write(newke);//setup duty cycle
            pc.printf("%.2f\t %.2f\t %.2f\t %.2f\t
%.2f\r\n",count,newke,d_R,temperature_internal,temperat
ure_ambient);
            old_count=count;
            old_temp_inter=temperature_internal;
            old_temp_exter=temperature_ambient;
            old_ke=newke;
            old_error=error;
        }
    }
    fastpwm.write(0.00); // Test Concluded
}
```

APPENDIX B: mbed PI Code

```
#include "mbed.h"
#include "FastPWM.h"
#include "MCP9808.h"

FastPWM fastpwm(p21);
Serial pc(USBTX, USBRX);
Ticker timer;

MCP9808::MCP9808(PinName sda, PinName scl) : i2c(sda,
scl)
{
}

MCP9808 sensor(p28,p27);

// read temperature from MCP9808
float MCP9808::internal_readTemp()
{
    data_write[0] = MCP9808_REG_TEMP;
    i2c.write(0x30, data_write, 1, 1); // no stop
    i2c.read(0x30, data_read, 2, 0);

    if(data_read[0] & 0xE0) {
        data_read[0] = data_read[0] & 0x1F;  // clear
flag bits
    }
    if((data_read[0] & 0x10) == 0x10) { // < 0 C
        data_read[0] = data_read[0] & 0x0F;
        tempval = 256 - (data_read[0] * 16) +
(data_read[1] / 16.0);
        tempval = tempval * -1;
    } else { // > 0 C
        tempval = (data_read[0] * 16) + (data_read[1] /
16.0);
    }
    return tempval;
}

float MCP9808::ambient_readTemp()
```

```
{
    //i2c.frequency(400000);
    data_write[0] = MCP9808_REG_TEMP;
    i2c.write(0x32, data_write, 1, 1); // no stop
    i2c.read(0x32, data_read, 2, 0);

    if(data_read[0] & 0xE0) {
        data_read[0] = data_read[0] & 0x1F;  // clear
flag bits
    }
    if((data_read[0] & 0x10) == 0x10) { // < 0 C
        data_read[0] = data_read[0] & 0x0F;
        tempval = 256 - (data_read[0] * 16) +
(data_read[1] / 16.0);
        tempval = tempval * -1;
    } else { // > 0 C
        tempval = (data_read[0] * 16) + (data_read[1] /
16.0);
    }
    return tempval;
}

float R = 84;
float k = 88;

float kp = 0.25;
float ki = 0.009;
float old_error = 0;
float integral = 0;

float setpoint = 27.83;
float ke;
float temperature_internal;
float temperature_ambient;
float old_ke = 1.00;

float count=0;
float sampling_period=0.5;

void attime()
{
    count=count+sampling_period;
}

int main()
{
    float old_count=0.1;
    float setFrequency=96000/20; //set to 50kHz,
clock/frequecy
    fastpwm.period_ticks (setFrequency); //setup the
period for 150Khz

    /*********** Initial Temperature Internal
********************/
    int p_inter=0;
    float initial_temp_inter [4]= {};
    float temp_m_inter[4]= {};

    while(p_inter<=3) {
        int m_inter=0;
        while(m_inter<=3) {

temp_m_inter[m_inter]=sensor.internal_readTemp();
            m_inter++;
        }

        float
sum_array_m_inter=temp_m_inter[0]+temp_m_inter[1]+temp_
m_inter[2]+temp_m_inter[3];
        initial_temp_inter[p_inter]=
(sum_array_m_inter)/4;
        p_inter++;
    }
    /*********** Initial Temperature External
********************/
    int p_exter=0;
    float initial_temp_exter [4]= {};
    float temp_m_exter[4]= {};

    while(p_exter<=3) {
        int m_exter=0;
        while(m_exter<=3) {
```

```
temp_m_exter[m_exter]=sensor.ambient_readTemp();
            m_exter++;
        }

        float
sum_array_m_exter=temp_m_exter[0]+temp_m_exter[1]+temp_
m_exter[2]+temp_m_exter[3];
        initial_temp_exter[p_exter]=
(sum_array_m_exter)/4;
        p_exter++;
    }
    /*********** Initial Temperatures Print
********************/
    float old_temp_inter= (initial_temp_inter
[0]+initial_temp_inter [1]+initial_temp_inter
[2]+initial_temp_inter [3])/4;
    float old_temp_exter= (initial_temp_exter
[0]+initial_temp_exter [1]+initial_temp_exter
[2]+initial_temp_exter [3])/4;
    pc.printf("Internal: %.2f\r\n",old_temp_inter);
    pc.printf("Ambient: %.2f\r\n",old_temp_exter);
    wait(1.0);

    timer.attach(&attime, sampling_period);
    while (count <=172800) {
        if(count != old_count) {
            /*********** Temperature Internal
********************/
            float temp_inter [8]= {};
            int i_inter=0;
            int zeros_inter=0;

            while (i_inter<=7) {

temp_inter[i_inter]=sensor.internal_readTemp();
                if(abs(temp_inter[i_inter]-
old_temp_inter)>.25) {
                    temp_inter[i_inter]=0;
                    zeros_inter++;
                }
                i_inter++;
            }
            float
sum_array_inter=temp_inter[0]+temp_inter[1]+temp_inter[
2]+temp_inter[3]

+temp_inter[4]+temp_inter[5]+temp_inter[6]+temp_inter[7
];
            float
temp_average_inter=sum_array_inter/(8-zeros_inter);
            if (zeros_inter==8) {
                temp_average_inter=old_temp_inter;
            }
            /*********** Temperature External
********************/
            float temp_exter [8]= {};
            int i_exter=0;
            int zeros_exter=0;

            while (i_exter<=7) {

temp_exter[i_exter]=sensor.ambient_readTemp();
                if(abs(temp_exter[i_exter]-
old_temp_exter)>.25) {
                    temp_exter[i_exter]=0;
                    zeros_exter++;
                }
                i_exter++;
            }
            float
sum_array_exter=temp_exter[0]+temp_exter[1]+temp_exter[
2]+temp_exter[3]

+temp_exter[4]+temp_exter[5]+temp_exter[6]+temp_exter[7
];
            float
temp_average_exter=sum_array_exter/(8-zeros_exter);
            if (zeros_exter==8) {
                temp_average_exter=old_temp_exter;
            }
```

```
temperature_internal=0.25*temp_average_inter+0.75*old_t
emp_inter;

temperature_ambient=0.25*temp_average_exter+0.75*old_te
mp_exter;
            /*********** PID Controller
********************/
            float error = setpoint-
temperature_internal;
            float integral = integral + error *
sampling_period;
            float d_R = kp*error + ki*integral;

            if (d_R < 0.01) {
                d_R = 0.01;
            }

            ke = (sqrt(k*d_R*R)-8.3)/79.7;

            if (ke < 0.05) {
                ke = 0.05;
            } else if (ke > 1.0) {
                ke = 1.0;
            } else {}

            float newke=0.25*ke+0.75*old_ke;
            fastpwm.write(newke);//setup duty cycle
            pc.printf("%.2f\t %.2f\t %.2f\t %.2f\t
%.2f\r\n",count,newke,d_R,temperature_internal,temperat
ure_ambient);
            old_count=count;
            old_temp_inter=temperature_internal;
            old_temp_exter=temperature_ambient;
            old_ke=newke;
            old_error=error;
        }
    }
    fastpwm.write(0.00); // Test Concluded
}
```

APPENDIX C: MATLAB Percent Savings

```
%Import bang bang data
% plot(time,Ti,time,Ta)
plot(time,Ti)
grid
hold
TFi=trapz(tAime,Ti);
TFa=trapz(time,Ta);
TFdiff=TFi-TFa

%import buck conv data
% plot(time1,Ti1,time1,Ta1)
plot(time1,Ti1)
TFa1=trapz(time1,Ta1);
TFi1=trapz(time1,Ti1);
TF1diff=TFi1-TFa1

nn=time(end);%How long is the test
%import Vrms and Irms for BB
time_Irms=linspace(0,nn,length(Irms))';
time_Vrms=linspace(0,nn,length(Vrms))';
time_Idc=linspace(0,nn,length(Idc))';
time_Vdc=linspace(0,nn,length(Vdc))';

if length(Vrms)<length(Irms)
    Vrms = interp1(time_Vrms,Vrms,time_Irms);
    time2=time_Irms;
else
    Irms = interp1(time_Irms,Irms,time_Vrms);
    time2=time_Vrms;
end
Ebb=trapz(time2,Irms.*Vrms)/nn

%import Vdc and Idc for BC
if length(Vdc)<length(Idc)
    Vdc = interp1(time_Vdc,Vdc,time_Idc);
    time3=time_Idc;
else
    Idc = interp1(time_Idc,Idc,time_Vdc);
    time3=time_Vdc;
end
Ebc=trapz(time3,Idc.*Vdc)/nn

figure
time_BB=linspace(0,3600,length(Irms))';
time_BC=linspace(0,3600,length(Idc))';
plot(time2,Irms.*Vrms)
hold on
plot(time3,Idc.*Vdc)

Per_Saving=100*(Ebb-(Ebc*TFdiff/TF1diff))/Ebb
```