PROJECT NUMBER: WZM-1M11

A MAJOR QUALIFYING PROJECT REPORT

Home Energy Automation Technology

Final Report

WORCESTER POLYTECHNIC INSTITUTE

Electrical & Computer Engineering

Submitted by:

Michael Audi
maudi@wpi.edu

Scott Cloutier
sjcloutier@wpi.edu

John Manero
jmanero@wpi.edu

Approved by:

Professor William R. Michalson
wrm@wpi.edu

Professor Stephen J. Bitar
sjbitar@wpi.edu

Date Submitted:
January 2nd, 2011

Acknowledgements

As contributors to this Home Energy Automation Technology (H.E.A.T.) Major Qualifying Project, we would like to recognize the following individuals and organizations listed below for their generous support to this project.

Marc Pepin, Product Marketing Engineer, Intel Corporation

Fred J. Looft, Professor and ECE Department Head, Worcester Polytechnic Institute

Dr. William R. Michalson, Professor and Project Advisor, Worcester Polytechnic Institute

Stephen J. Bitar, Professor and Project Co-Advisor, Worcester Polytechnic Institute

Alexander E. Emanuel, Professor, Worcester Polytechnic Institute

Abstract

The purpose of this project is to explore residential household climate control systems and develop a viable product concept that integrates any and all heating, ventilation, and air conditioning (HVAC) sources into an automated electronic control system. This project will incorporate a microcontroller-based modular system that provides multiple communication mediums to adapt to most household configurations. This system will utilize a web-based control server that implements efficient climate control algorithms, resulting in improved heating and cooling efficiency for residential and small-business consumers.

Executive Summary

This project focuses on the development of a product concept that provides a solution to an existing problem found in households with multi-sourced climate control systems. The procedure taken for this project includes identifying the problem, developing the project concept, constructing and demonstrating the prototype, and analyzing the results rendered from this project.

The Problem

Today's technology has created the possibility for consumers to control the climate inside their homes with the use of various Heating, Ventilation, and/or Air Conditioning (HVAC) systems. This essential comfort however, can prove costly if controlled inefficiently. The scenario below illustrates a common household heating system and how it is typically controlled.

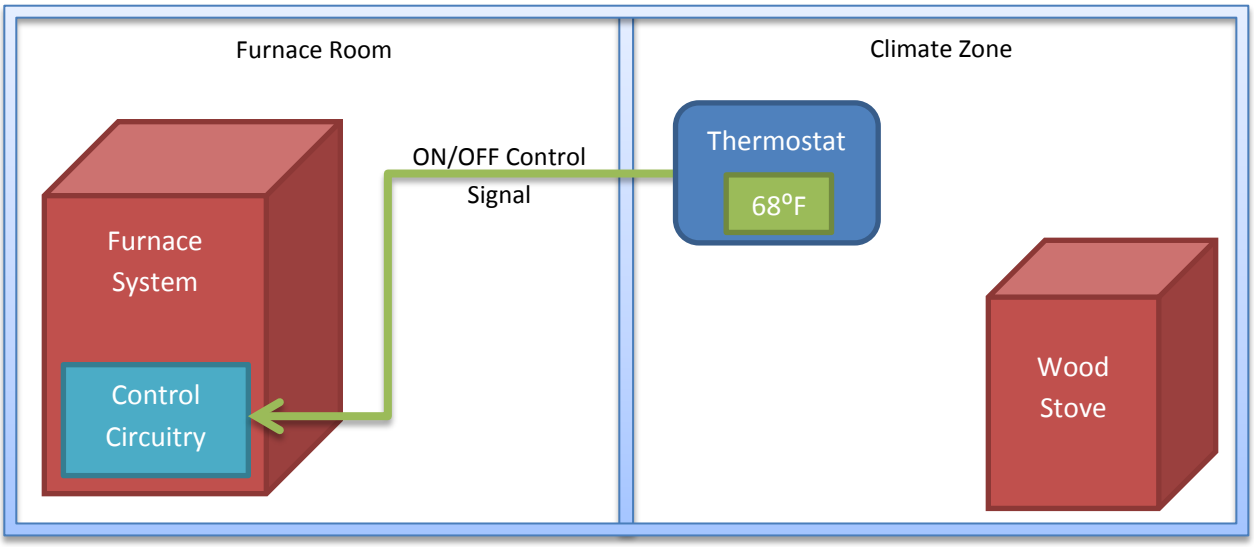

Figure 1 – Typical Household Heating System Control

This illustration above depicts a common scenario of climate control which can prove to be highly inefficient. A wood stove is used to provide heat to a particular portion of a house, but in the case where the wood stove runs out of fuel, the temperature must decrease a substantial amount in the general area in order for the thermostat to signal for the backup heat-source to turn on.

The problem here exists with the use of multiple heating sources to regulate the temperature of the same zone(s) there is no direct control system to maintain the efficiency of all heat sources in this house. Even with just the typical household furnace, there is a lack of communication between the heating system and user interface when the thermostat can only control the ON/OFF state of the heating system.

The Product Concept

To describe the principle functionality behind this concept, this project will be divided into sub-sections to illustrate the different reference points of a household heating system configuration.

The "Back" of the House

The back of the house is the area in which all of the mechanical heating and cooling sources are maintained. This location would commonly be referred to as a basement or storage area where the furnace is kept. The concept of this project utilizes microcontroller-based technology to provide a solution to this HVAC control integration problem. This system is developed to adapt multiple types of heating and cooling sources simultaneously into one central controller. An example configuration for this scenario is illustrated below in Figure 2.

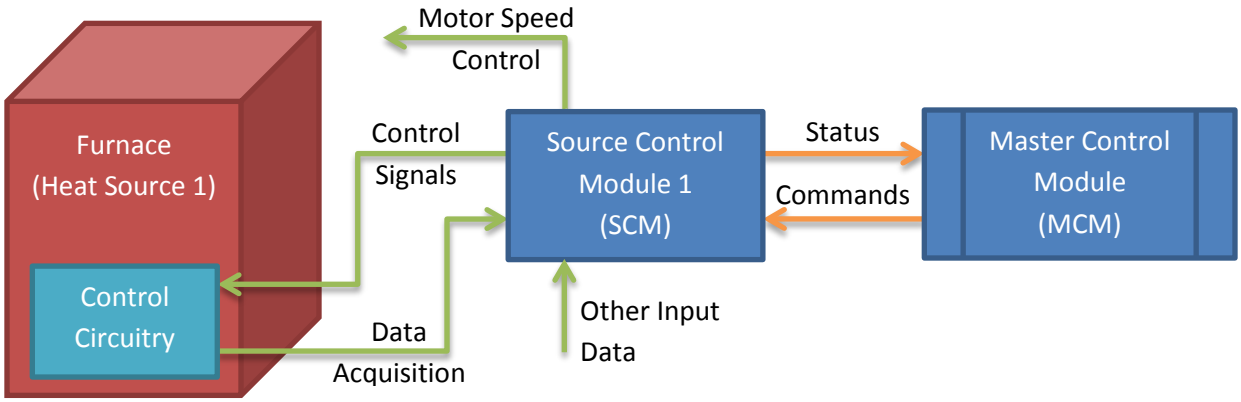

Figure 2 – "Back" of House Proof of Concept

This scenario shown above is meant to illustrate the project concept's capability and advantage over the problem identified earlier. With the use of microprocessor embedded technology, a modular based system can be created to serve as a "medium" for communicating with various types of climate sources. The example above shows how a Source Control Module (SCM) can provide more control over a typical heating furnace. In the case of multi-sourced heating systems, multiple SCMs can be used to provide a multitude of control and data acquisition over each source.

A Master Control Module (MCM) will provide the necessary high-level computational control over the entire system. One master controller will be used to communicate with and control multiple embedded modules in the entire house. This hierarchy of electronic control systems can drastically improve efficiency of household climate control with the write software control methods.

The "Front" of the House

The front of the house is considered the user-side of the entire climate control system. This includes the various climate zones for which the temperature will be controlled. An illustration of possible scenario for the front of the house is shown below in Figure 3.

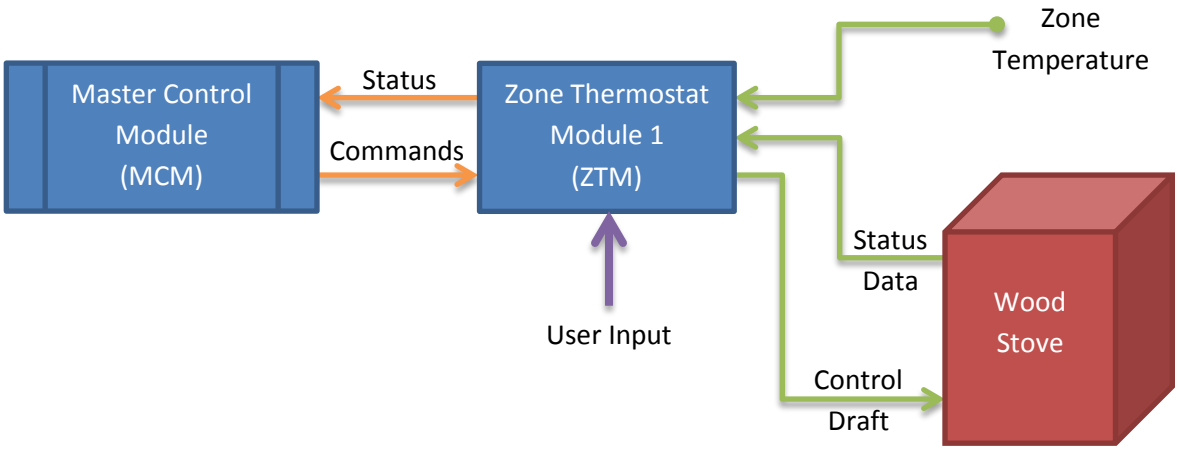

Figure 3 – "Front" of House Proof of Concept

The configuration shown in the figure above is similar to that of the back of house setup. Each climate zone in the house would be adapted with a Zone Thermostat Module (ZTM), which is the same modular-based physical construction of the microprocessor embedded system used for the SCMs. The ZTM is used to provide a communication "medium" to the front of the house for the climate control system. This purpose will allow the MCM to obtain data in each climate zone (i.e. room temperature) and provide the ability to control supplemental heating systems, such as a modern wood stove as shown in the example. The ZTM will also provide a small user-interface thus which the desired room temperature can be set manually in a particular climate zone.

The key component to this system concept is automation. By providing the necessary components to collect data from multiple inputs and be able to control multiple outputs, this climate control system will be able to function autonomously with minimal user interaction. The primary benefactor of this project concept comes from creating a universal modular-based system that can incorporate various and multiple household climate sources. With this level of capability, the system will be able to know what resources are available, which are more efficient ("greener"), and be able to control these sources simultaneously as needed. This form of unified communications will enable the ability to increase overall efficiency over a traditional climate control configuration.

Implementation & Results

The following sections highlight the results of implementing key parts of this project.

The Master Control Server

The figures below illustrate the results of the progress made in implementing the Control Server:

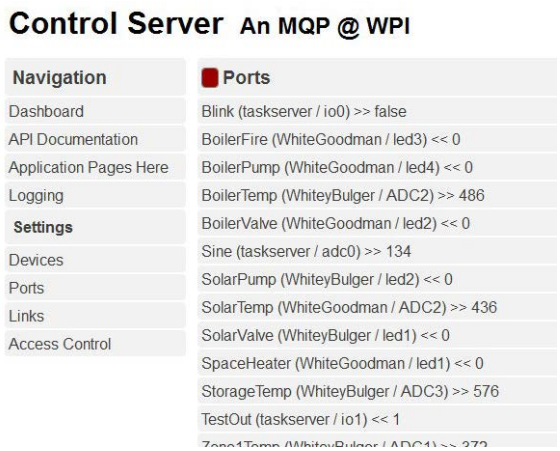

Figure 4 – A List Interface, Displaying Ports on a System

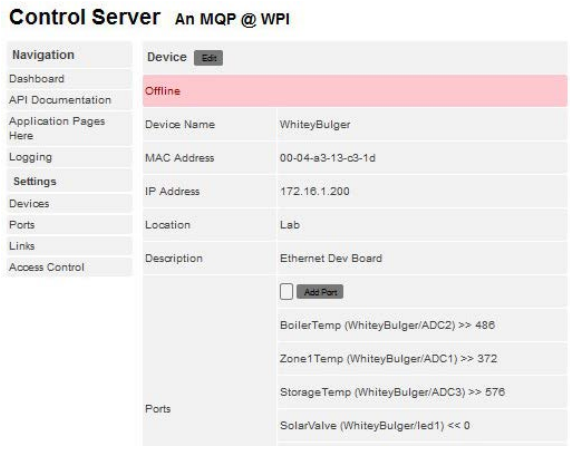

Figure 5 – A View Interface, Displaying a Device

The view above in Figure 4 lists the available ports for a device. These ports are sorted by their system identifier, but also display their parent device and intra-device identifier. Figure 5 is a view interface, which displays a particular device. Devices provide the server with a real-world anchor for ports, defining where to send and receive data to and from, and how to format the data.

The Embedded Modules

The figures below depict the results made from implementing the Embedded Modules:

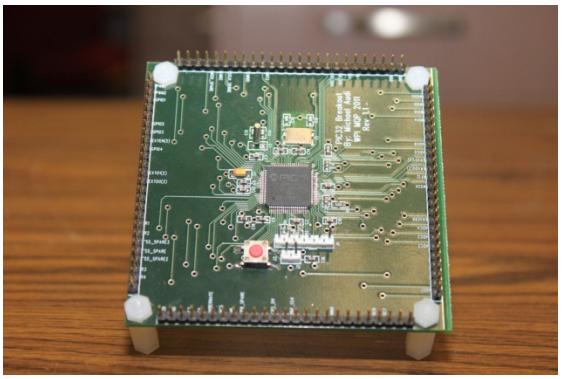

Figure 6 – PIC32 Breakout Board

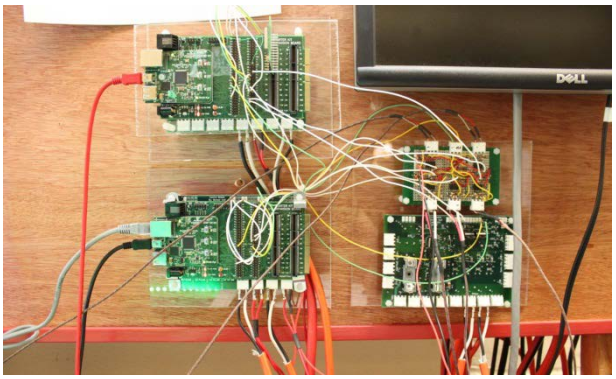

Figure 7 – Demo Board Implementation

The breakout board in Figure 6 above includes the necessary components to stabilize the microcontroller. This board provides the signal pins needed to control devices and peripherals. This

board concept is the basis for the central controller in the next generation system. Figure 7 above shows the three relay boards which include AC motor control, GPIO, and SPI peripheral interfaces. The thermocouple amplifier circuits are also seen in the top right of the picture. These boards are connected to the PIC32 demonstration boards provided by Microchip which handle the communication to the server. Together these boards handle the control of the demonstration platform.

The Demonstration

In order to test the functionality of the project prototype, a demonstration platform was built to provide a simulation of a typical household multi-sourced heating system as shown below in Figure 8.

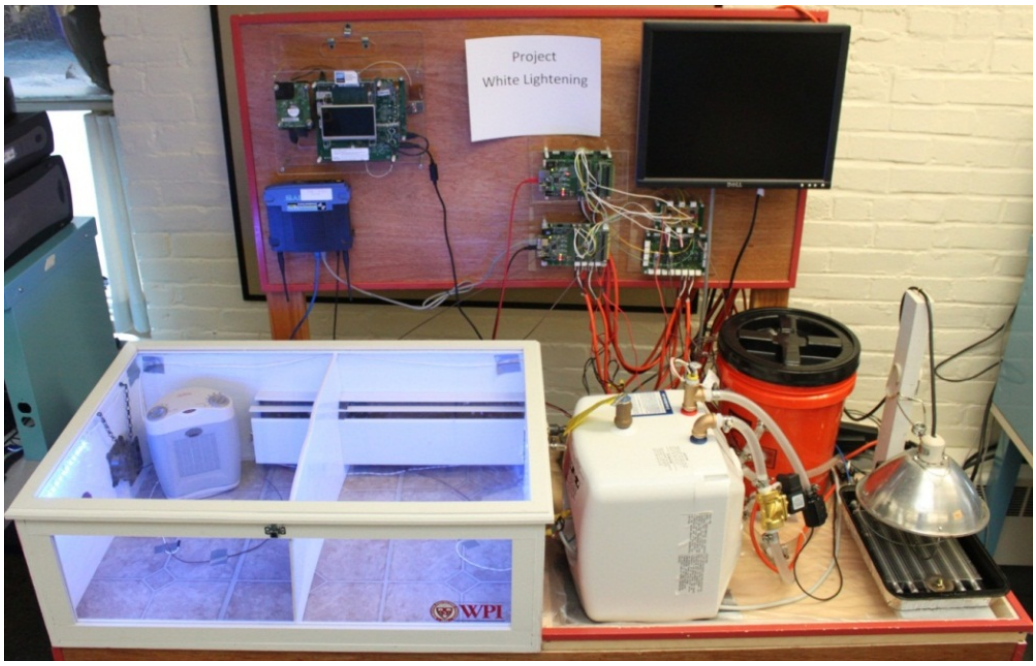

Figure 8 – The Demonstration Platform

This demonstration platform shown in the figure above includes two heat sources: Heat Source 1 – Boiler Simulation and Heat Source 2 – Solar Energy Collector. These heat sources supply a heat storage tank to store heat energy when it is not needed. When the heat energy (hot water) is needed the heat storage tank feeds into a hydronic radiator that is installed inside the climate zone box as shown above. The climate zone box simulates a "house" by incorporating two climate zones on the inside. A supplemental heat source is included in one zone to demonstrate efficient heating when a draft is induced within that zone. The Home Energy Automation Technology (H.E.A.T.) system includes the Master Control Module (MCM), the Source Control Modules (SCMs) and the Zone-Thermostat Module (ZTM). This system controls the necessary components of this heating system simultaneously using efficient and effective climate control methods.

Table of Contents

Table of Figures

Table of Tables

Glossary of Terms

- **API** Application Programming Interface
- **CPU** Central Processing Unit
- **HVAC** Heating, Ventilation, and Air Conditioning systems
- **MCI** Model-Control-Interface
- **MCM** Master Control Module; Sometimes referred to as “Master Control Server”
- **MQP** Major Qualifying Project
- **NTC** Negative-Temperature Coefficient
- **PC** Personal Computer
- **PCB** Printed Circuit Board
- **SCM** Source Control Module
- **WPI** Worcester Polytechnic Institute
- **ZTM** Zone Thermostat Module

Chapter 1: Introduction

As an essential part of human technology and daily life, today's climate control industry has evolved to enable consumers the ability to control the temperature inside their homes and businesses at a comfortable level. This crucial feature of HVAC systems carries an expensive cost to the consumer and the environment. While many home owners have taken the initiative to incorporate "green energy" climate control systems in their homes, there exists no such system that integrates multiple climate sources efficiently and effectively. This project will focus on providing a viable solution to this problem.

The purpose of this project will be to develop a microprocessor-based system that has the ability to incorporate and control various types of heating, cooling, and ventilation systems most commonly found in households. The system will be designed to integrate all climate sources and climate zones in any household and shall provide efficient and effective control methods. Upon completion of this project, the system developed will improve the overall efficiency of climate control in residential and small-business environments.

This report will present the process and procedures taken to design, develop, and test a project prototype that can demonstrate this solution as a product concept. Research is required to identify the various types of heating and cooling systems that are most commonly found in consumer households and to evaluate current climate control methods. A set of customer requirements must be determined in order to identify the product requirements and specifications to begin designing the system. Creating a layout of the system architecture and the design, the process of implementation can begin to develop a working prototype that serves as a foundation for this concept. Given the size and environment that climate sources currently operate in, a scaled-down demonstration platform is required to test and demonstrate the prototype. During this entire process, any and all engineering characteristics and tribulations will be documented in this report.

Given the nature of this project and absence of a current solution, a business proposal will be incorporated into this report that will outline the forecast models and future progress that this project can take to become a marketable product.

Chapter 2: Background Research

This chapter covers the research done on the background of climate control methods and systems. This section will also include similar systems that are defined as prior art to this project.

2.1. Climate Systems

This section will discuss various types of common household climate systems. This includes household heating, cooling, and ventilation systems.

There are many different types of climate sources found in typical households and businesses. Given the time constraints and the nature of this project, it would be unfeasible to cover all possible climate systems. For the scope of this project, only the most common household systems will be researched and defined in this section. Shown below is a list of the general types of climate systems used in households.

- Traditional Hydronic Heating System – Water, Steam
- Air Conditioning (AC) System – Refrigerant
- Solar Heating System – Water
- Geo-Thermal System – Water
- Heat Stove System – Wood, Pellet

One of the most commonly used heating systems consists of a Traditional Hydronic boiler setup, normally fueled by petroleum or natural gas. This type of system is used for heating households and small business buildings during the winter season. For the summer season where temperatures become uncomfortably warm, an Air Conditioning (AC) system is used to cool the air. This system usually consumes electrical energy to operate.

Some households have incorporated “green” energy heating systems to help reduce the consumption of non-renewable fuels and save money overall. Amongst these heating technologies, the Heat Stove System is the most common and has been used the longest in human history. The heat stove traditionally used split wood as a fuel source before it became replaced by more efficient pellet-based fuel. Other newer technologies involving “green” energy heating systems include Solar-Heating and Geo-Thermal systems. These heat systems use the environment as their heat source with water as the heat-transfer agent.

2.1.1. Hydronic Heating Systems

The Hydronic Heating System is one of the most common types of household heating setups used in cold-weather climates. A hydronic heating system is a form of water-based heating, in which water is used as the medium to transport the heat energy to its destination. This typical heating system varies in type and sizes, but often consists of the basic essential equipment listed below:

- **A Boiler** – This is the primary component of hydronic heating systems. A boiler, often referred to as a "furnace" is a containment device that is used to generate heat energy to heat the water.
- **Fuel Source** – There are various types of home heating fuels used in hydronic systems, but is essential for generating the heat energy required to heat the water. Most common fuels used are #2 Heating Oil and Natural Gas.
- **Liquid Medium** – This is the liquid used to transport the heat energy to its destination. In most cases, common tap water is used in this type of heating system and is easily replenished through the main water line in the house.
- **Pipe/Tubing** – Pipe or tubing is typically used to transport the heated water to its destination throughout the house.
- **Heat Exchanger** – This is typically referred to as a "radiator" in which is used to transfer the heat energy from the water to the air surrounding the exchanger.
- **A Pump** – A liquid pump is required to circulate water throughout the entire heating system.

With the basic components listed above, the hydronic heating system can be used to provide heat to an entire house effectively.

Hydronic heating systems come in various types and configurations. A basic boiler system typically found in most households contain only one climate zone with a heat-exchanging element in each room. The boiler itself can be either a hot-water or a steam based system in the hot water or steam is used as the medium to transfer the heat energy from the source to the destination.

How Boilers Work?

In hydronic heating systems, a boiler unit is one of the most common devices used as a heating source. There are several types of boilers available based on the type of fuel source available in the area. An example of a typical hot-water boiler system is shown below in Figure 9.

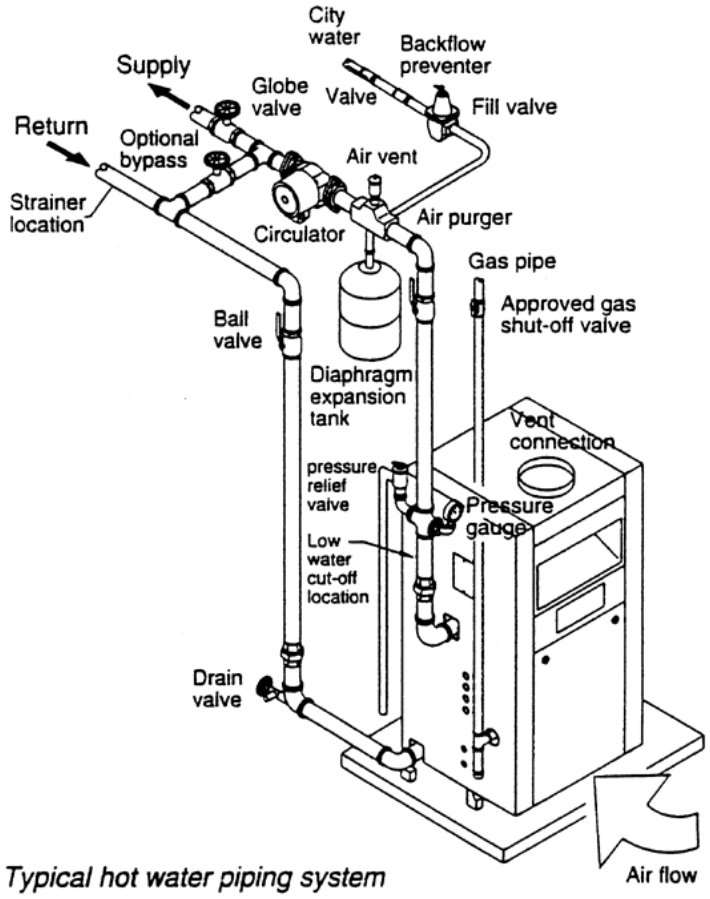

Figure 9 – Gas-Powered Hot Water Boiler

The key part in a boiler's operation is its fuel source. The fuel type can vary from natural gas (as shown in the figure above) to #2 heating oil. These non-renewable heating fuels are ignited inside a chamber in which the fuel burns to produce heat. This heat rises through a heat exchanger pipe inside the boiler in which cold water is pumped through. The water is heated rapidly to a certain temperature and then it is pumped out to radiators inside the house. When the hot water fills the radiators, the heat is then exchanged into the air through the radiator. This cooler water is then returned back to the boiler for re-heating. This process is typically controlled through a thermostat in a central location of a house. When the temperature falls below a certain set threshold, the boiler system is activated to generate heat and increase the temperature of the house back to the desired temperature.

2.1.2. Air Conditioning (AC) Systems

Air Conditioning systems are primarily used to cool the temperature inside a room or entire house during the warm climate season. These AC systems come in various forms and sizes, ranging from standalone portable window units to more permanent household solutions typically referred to as Central Air Systems. An example of a typical Central Air system is shown below in Figure 10.

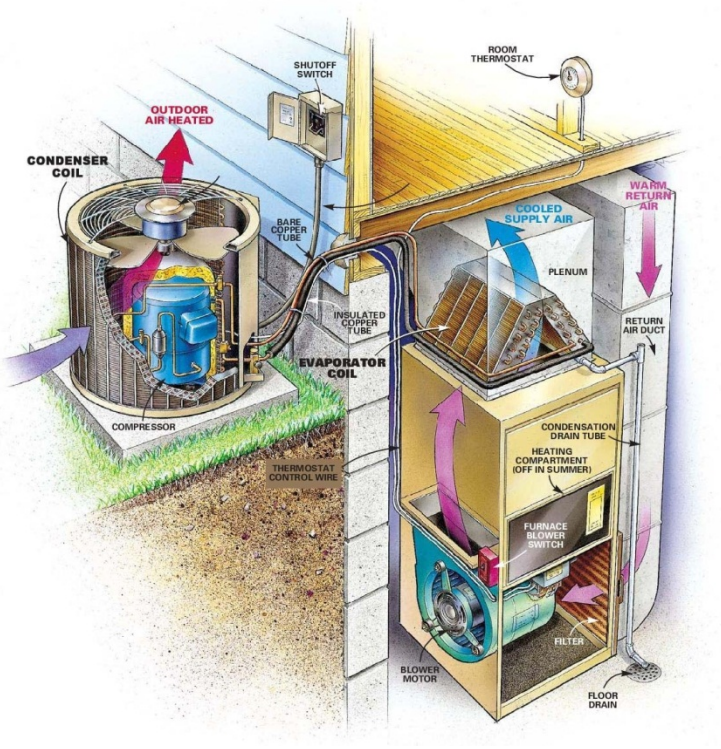

Figure 10 – A Typical Central Air System

The system illustrated above operates through a compressor-condenser-evaporator type configuration to condition the air to a cooler temperature inside the house. The system is comprised of two remote units: An outdoor condenser system and an indoor evaporator system. The two are connected through a cycled gas line to transfer heat to the outside while making the inside of the house cooler. The substance used is typically Freon gas. This gas is compressed in the compressor which increases the temperature of the gas. This gas is then directed through a condenser to dissipate the heat and turn the gas into a liquid. This liquid is then forced through an expansion valve to turn the liquid back into a low-temperature gas. This cooler gas is then transferred into the indoor unit where it runs through an evaporator coil. A blower located on one side of the evaporator coil blows air through the coil thus conditioning the air to a cooler temperature. This gas is then returned to the outdoor compressor unit where it undergoes the same process while the system is in operation.

2.1.3. Solar Heating & Geo-Thermal Systems

Solar Heating and Geo-Thermal Heating Systems are becoming increasingly more popular as the price for home heating fuels rise. These heating systems are referred to as "Green Energy" systems because they use renewable energy as their primary source of heat. An example of a typical solar heating system setup is shown below in Figure 11.

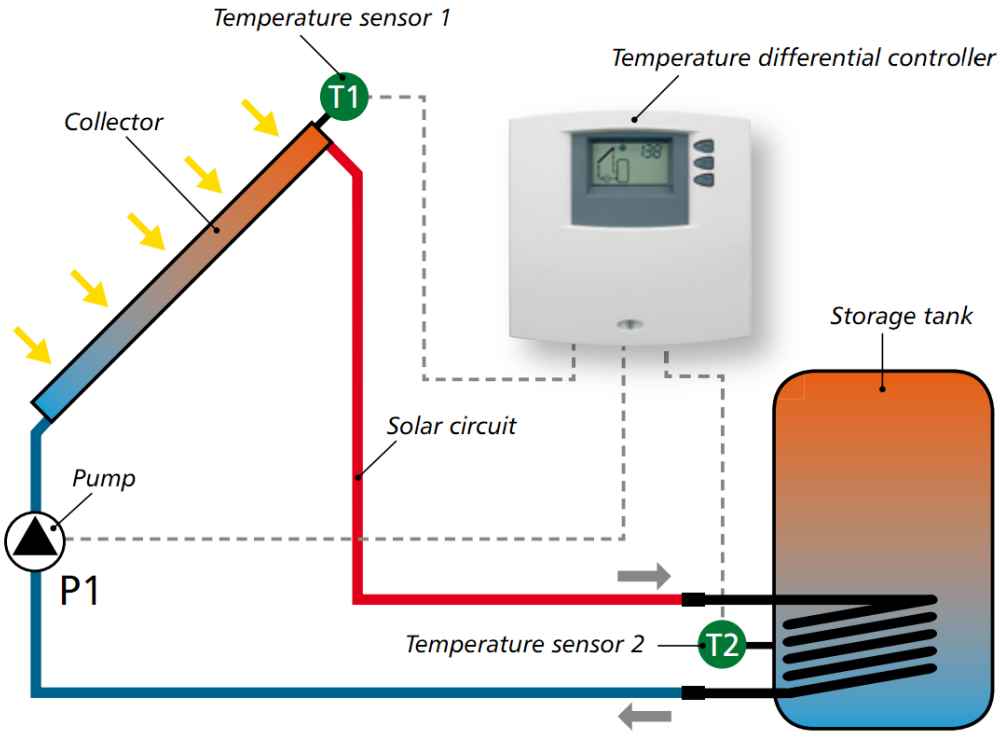

Figure 11 – Solar Energy Collector Heating System

The solar heating system shown in the figure above is a minimal solar collection system. This green energy system functions by using the Sun's heat energy to heat water and store this water in a heat storage tank for use. This heating system uses only the energy provided by the sun, therefore requiring no non-renewable heating fuel or other heat source.

Unfortunately, there are limitations to such a heating system configuration. This system can only collect heat energy on a sunny day during daytime hours. While the amount of heat energy provided by the sun may seem adequate, it is not enough to support the heating requirements of a typical household. Therefore, this type of heating system is typically used as a supplemental system alongside a traditional hydronic or other non-renewable heat source system.

Geo-Thermal Heating Systems operate similarly to that of the Solar Heating System. Rather than using the Sun's energy to heat water like a Solar heating system, the geo-thermal system uses the earth as a means of absorbing or dispersing energy. A typical example of a geo-thermal heating system is shown below in Figure 12.

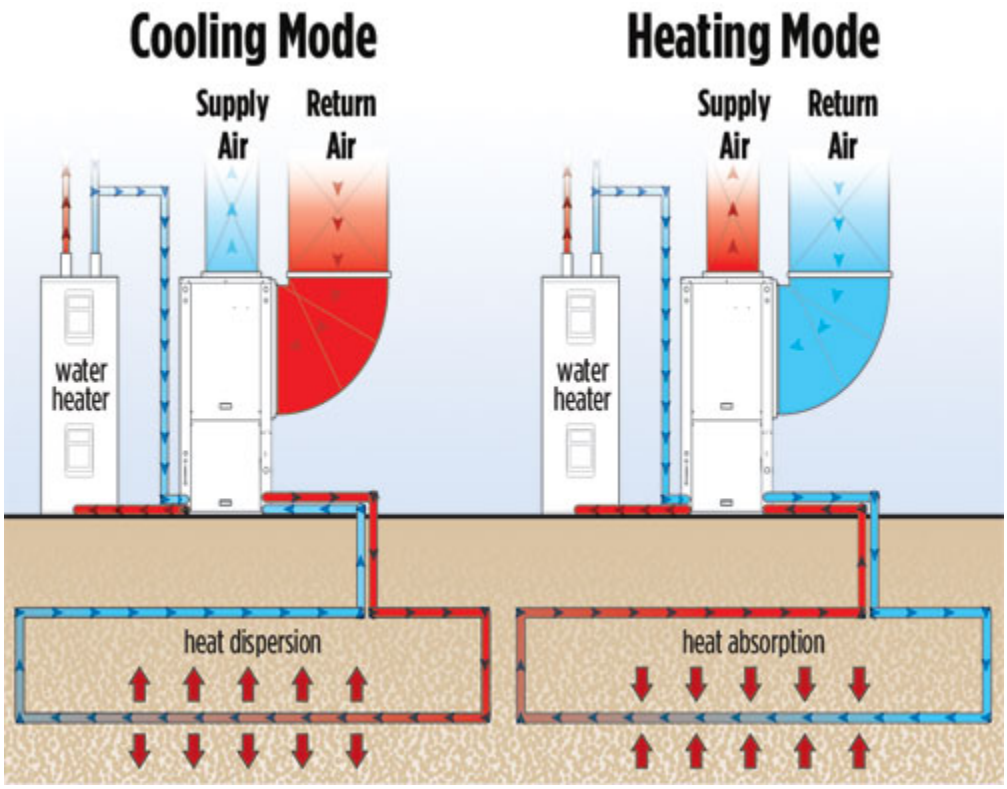

Figure 12 – Geo-Thermal Heating/Cooling System

Unlike solar heat collection systems, geo-thermal systems can be used to absorb heat energy or to dissipate heat energy depending upon the climate outside. While solar heat collector systems are limited by the time of day and appearance of the sun, geo-thermal systems are able to operate throughout the day and night.

However, compared to solar collection systems, geo-thermal transfer systems are less efficient. This is primary due to the limited range of temperature variance underground. This system cannot be used as a standalone heating system, but can help improve the efficiency of an overall household heating and cooling system.

2.1.4. Heat Stove Systems

Heat Stove systems are one of the simplest and oldest types of traditional forms of heating systems. A heat stove which can often be referred to as a wood stove or pellet stove provides heat by burning its fuel source in a containment stove as well as radiates the heat that it generates. An example of a typical wood stove is shown below in Figure 13.

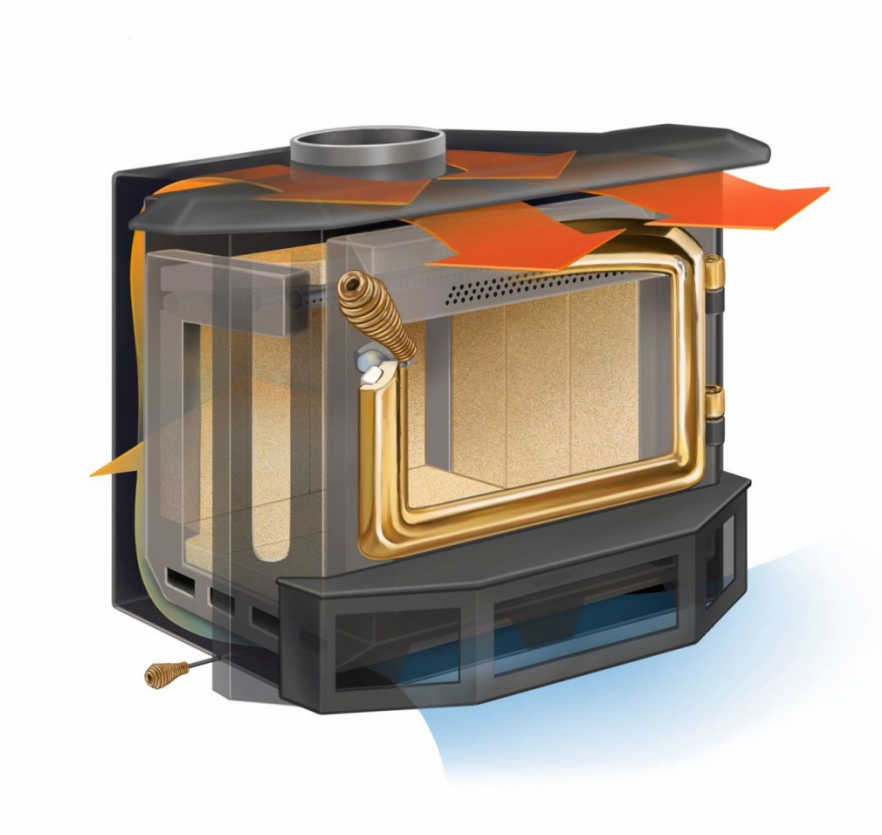

Figure 13 - Typical Stand-Alone Wood Stove

The traditional heat stove shown above in amongst the many variations of heat stoves currently available on the market. This type of heat stove can provide a tremendous amount of heat directly to the surrounding area of the stove. This feature also serves as a limitation as well, given that the heat given off by the stove does not have the ability to distribute the heat evenly throughout the entire house. To compensate for this limitation somewhat, wood stoves have been typically located in the lowest room in the house. This allows the natural convention of the heat generated to spread up through the floors of the house into the rooms in which the heat is desired.

Modern wood stoves have incorporated blower systems to maximize the heat energy dissipated from the stove and spread it throughout the house. Other modern systems have adapted to use a more efficient fuel-type called pellets. A diagram representing a pellet stove is shown below in Figure 14.

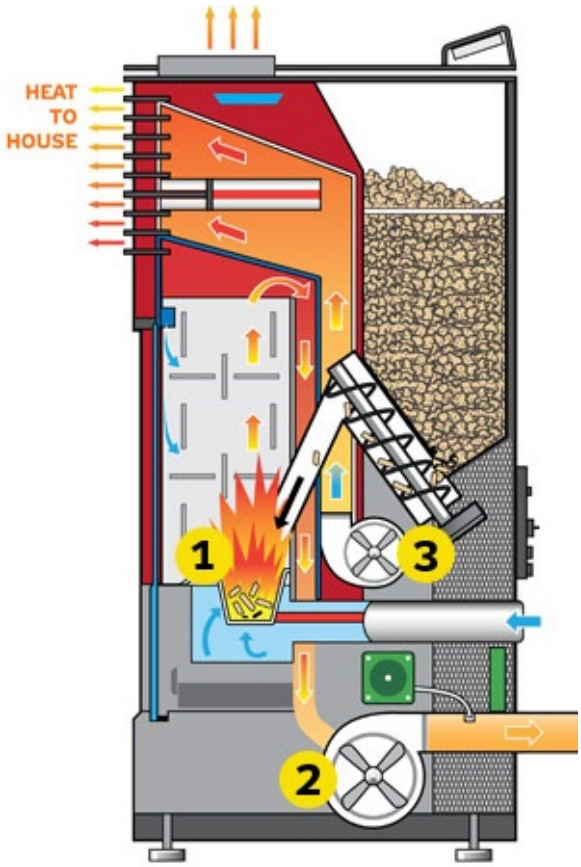

Figure 14 – A Typical Pellet Stove System

This type of heat stove increases the efficiency over traditional wood stoves by using a pellet-type fuel that burns longer when regulated and leaves less waste over regular wood. A pellet stove system can incorporate a blower-vent system to distribute heat throughout the house. The effectiveness of a pellet stove has proven to be a viable heating system during the coldest climate seasons.

2.2. Climate Control

This section will cover methods of control for climate systems. This includes controlling heating, cooling, and ventilation systems.

2.2.1. The Thermostat

The thermostat has become the standard in home heating and cooling control. A simple thermostat provides the automated functionality of controlling any home heating/cooling system by switching the system either ON or OFF dependent upon the current temperature setting with respect to the ambient temperature of the room that the thermostat is located in. An example of a basic analog thermostat is shown below in Figure 15.

Figure 15 – Basic Analog Thermostat

While the basic thermostat provides a simple solution to automated functionality of home climate systems, they have very limited efficiency. The development of technology over the past decade has resulted in digital thermostat systems that provide more control home heating and cooling systems, thus increasing efficiency and comfort.

Increasing Efficiency

With the onset of programmable thermostats, methods have been introduced in attempt to increase home heating efficiency. Such methods include adjustments of the desired temperature setting based on the time of day – i.e. setting the thermostat back 10 degrees for an average of 8 hours per night can save about 5-15% a year in energy expenses. Programmable thermostats provide the feature of autonomy in which these settings can be programmed based upon the time of day, day of week, etc.

Some Science on Heating

Myth – more energy is wasted when the furnace has to work harder to bring the temperature back up from a lower setting. It's actually less efficient to keep the temperature setting at a higher temperature throughout the day; more energy is wasted by maintaining a set temperature that is much greater than that of the outside temperature.

Another fact – at lower temperature settings, a house loses energy much slower. The greater the difference between interior vs. exterior temperature, the faster energy is lost when not maintained.

Programmable vs. Traditional

Most thermostats today come in two different types: digital and electromechanical. Some may combine the two technologies. Digital provides more possibilities and greater control over home heating systems, thus increasing efficiency. An example of a digital programmable thermostat is shown below in Figure 16.

Figure 16 – Digital Programmable Thermostat

The downside to such a device is that it may be complicated to program, in which some might chose an electromechanical thermostat for the "ease of use" factor, disregarding the decreased efficiency.

A major disadvantage of thermostats is that the location of the thermostat can greatly affect the performance and efficiency of any home-heating system. There exist some systems that provide zone control over various heating systems, but at a much larger cost and increased installation difficulty.

Limitations of "Advanced" Thermostats

While these advanced electronic thermostats provide an enhanced experience in controlling climate sources for the user, there are several limitations with these typical household programmable thermostats based upon the desired heating system applications. Some of these limitations are highlighted below:

Heat Pumps – Programmable thermostats are not recommended for heat pumps, because of their inefficient operation when trying to regulate the temperature in heating mode.

Electric Resistance Systems – Requirement for thermostats capable of controlling 120V or 240V circuits, and is a limited manufactured product.

Steam and Radiant Floor Heating – Slow response time; can take up to several hours to take effect. There exist some programmable thermostats that track this behavior to optimize performance.

While advanced technology progresses in digital thermostat control, the thermostat system still contains limitations that prevent this control device from being the most efficient option available.

2.3. Prior Art

This section will discuss research results pertaining to prior products and project concepts that have been developed to address the defined problem in some form. The purpose of this section is to identify similar products and systems that are comparable to the solution that is being developed by this project. This will help construct this project's concept to provide a solution that does not yet exist by utilizing areas in which these prior-art systems did not address.

2.3.1. Control4 Energy Control System

This electronic-based home energy control system is one example product that has been developed to bring household climate control to a "smarter" level. This system replaces your typical thermostat and uses computing technology to enhance your climate control capabilities. An illustration of this product is shown below in Figure 17.

Figure 17 – Control4 Energy Control System[1]

While this climate control product provides an advanced level of control for your typical household heating system, it is limited in the following criteria:

- **Software Capabilities** - Focused on delivering information about energy consumption to user
- **Functionality** - Only provides turning on/off function similar to that of a thermostat
- **Proprietary System** - Does not have the ability to customize the software; i.e. open-source

Given the limitations described above, this product would not provide a viable solution to the initial problem defined in this project. However, this product does include some features like an interactive touch interface and wireless communications that may be incorporated into this project's concept to provide a viable solution.

[1] http://control4.com/energy/products

2.3.2. HAI Energy Management & Lighting Controller

This energy control product provides another more in-direct approach to our initial problem area with home automation technology. This electronically-based energy management system serves the primary purpose of reducing overall energy costs in homes and small businesses. A picture of the hardware components for this product is shown below in Figure 18.

Figure 18 – HAI Energy Management & Lighting Controller[2]

This product shown above developed by Home Automation, Inc. (HAI) is a complete solution to providing efficient control over energy consumption. This product provides the ability to control household appliances such as lighting and climate systems to help improve efficiency and limit energy consumption. While this product provides another "smart" computer-controlled home automated system, there are several limitations to addressing our initial problem:

- **Primary Focus** – Developed primarily to control lighting to conserve energy
- **Software** – Limited control through timing schedule or use of occupancy sensors
- **Functionality** – Only provides on/off function for control; only relay based
- **Proprietary System** – No ability to customize control software; i.e. open-source

This home automation system controller provides an interesting approach to improving efficiency in household energy consumption – a requirement that coexists with the initial problem. This system however isn't directed towards climate control, and therefore cannot be considered as a complete solution to the problem.

[2]http://www.homeauto.com/Products/HAISystems/Luminapro/Luminapro.asp

Chapter 3: Project Overview

This chapter will focus on the development of the project concept. This includes identifying the problem statement and choosing a primary goal to solve this problem. With the primary goal chosen for this project, a list of customer requirements will be created to help determine the necessary product requirements.

3.1. Problem Statement

With the onset of rising home heating costs, the demand for more efficient green energy home heating systems has increased. There are several "green energy" systems that exist in the market today and have been adapted by many home owners at the consumer level and businesses and corporations at the industrial level. While the addition of these various types of green energy systems have improved the overall efficiency of home energy consumption, the methods used by which to control the functionality between multiple heating systems have been deemed inefficient. An example of such a situation would be comprised of no central control system and simply relying on individual thermostats as a means of utilizing any backup heat systems.

3.2. Primary Goal

To solve the problem identified above, a microcontroller based system capable of controlling most traditional household heating systems in addition to green energy systems. The sole purpose of this controller device will be to drastically improve the efficiency over current climate control methods used today.

The completion of this project should render a fully-functional product prototype that will serve as a solution to the problem. This prototype shall fulfill the absence of complete climate control communication in consumer households.

3.3. Target Market

This concept for this project has been derived from a problem identified on the residential level. A similar problem has been addressed on the commercial business level where industrial climate control products have been developed to meet the requirements for large-scale buildings.

This concept has been developed primarily for household and small business users. The ideal market has been identified at the consumer level, consisting of home owners and developers, and energy enthusiasts.

3.4. Customer Requirements

The first step in developing this project concept into a marketable product, a list of customer requirements must be determined in order to identify the requirements and features for the product itself. These customer requirements are typically divided into two categories: Explicit Requirements – where the customer identifies a particular feature request for the product, and the Implicit Requirements – the generalized requirements that must be accommodated to make this product functional and safe.

Explicit Requirements

- **Universal Control** – Product must have the ability to control *most* heating, cooling, and ventilation systems commonly used in households and small businesses.
- **Multi-Source Control** – Product must be able to accommodate as many climate sources and systems that the customer would like to control at the same time.
- **Easy Installation** – This system must be easy to install and require minimal alterations to existing building structure.
- **Low Cost** – The product's overall cost should not exceed an amount the consumer would not be willing to pay.
- **Efficient Functionality** – This product must improve efficiency of climate control over previously used systems within its market.
- **Expandability** – In order to maintain value in its market, this product must be expandable to adapt future climate sources and systems.
- **Customizable Feature Set** – This product should allow the user to configure and adjust settings as desired with a feature set equivalent to or greater than that of a digital programmable thermostat.

Implicit Requirements

- **Safety** – The product must incorporate all necessary procedures to maintain a safe installation and functional environment.
- **Low Maintenance** – Aside from the maintenance already performed on existing systems, this product shall not require additional extenuating maintenance and cost with its operation over time.

Given the proprietary list of features and requirements determined above from the customer standpoint, the product concept can be taken to the next procedure in developing a functional prototype for this project.

3.5. Product Requirements

This section will highlight the derived requirements for the product concept. These specifications will then be used to design the system architecture for this concept and implement a functional prototype to demonstrate the feature of this product.

Universal Control Ability

In order to accommodate the entire target market, this product must be able to have a *universal control* factor such that it will have the ability to control the functionality of most household heating, cooling, and ventilation systems. Some climate sources may have different control requirements based on their size, type, function, etc. This product should incorporate all the necessary hardware and software needed to control any of these systems.

Multi-Source Design

This particular requirement will set this product apart from the traditional thermostat control systems used today. Most household thermostats can control one or two climate sources at most. This product will need to possess the ability to control multiple climate sources at the same time. This particular feature requirement will serve as the primary solution to inter-system communications between all the household climate sources and systems.

Easy Installation

One particular problem that sometimes accompanies installing a new product in an existing environment is an in-depth installation that requires modifications to the current environment. This usually includes drilling holes, running wires, etc. To prevent this type of problem from occurring, the product design will need to incorporate an alternative medium for communications. The product should also include all the necessary hardware to adapt to its installation environment the easiest.

Efficient Control Methods

Primarily the most important key feature for this product is the ability for efficient controlling of climate sources. The main problem with existing climate systems being used in households today is the absence of efficient control of these systems. In order to provide an efficient system and save the consumer money, the product must incorporate the necessary software algorithms to control the various climate sources effectively.

Low Cost & Expandability

In order to provide a low cost product that is an expandable system, the product will be based on a modular system design. Such a system design approach will simplify the system architecture and create a versatile product.

Customizable Feature Set

While this product may handle most of the controlling functionality automatically, it is important to provide a set of customizable features to give the customer more control over the system if they so desire. In comparison, high-end programmable digital thermostats in the market today provide a user interface allowing the customer to set particular temperature, timing, and control settings for their climate system. This product design should incorporate at least this level of functionality for the customer to customize their system.

Safe System Design

Safety is one of the most important aspects that must be addressed in designing any consumer product. A control system for household climate systems plays a crucial role in maintaining a safe environment for the inhabitants of the household. Some climate systems may utilize hazardous materials that can pose a serious risk if not controlled properly. This requires the product to take the necessary procedures to ensure the safe functionality and usage of this product.

Low Maintenance

This requirement can be achieved by utilizing an efficient and reliable system design. A system that can be easily configured and updated automatically through internet access can help keep maintenance costs minimal after installation. Adapting such features will help keep this product an attractive solution on the market for consumers.

3.6. Project Management

This section will cover the plan for managing the project development and synthesis of the product prototype. This includes defining the structure in which the project will be conducted over the course of the given time frame.

3.6.1. Project Milestones

This section will highlight the milestones set for this project in order to organize the method in which this project will progress. These milestones are designated by "phases" in which describes the developmental process of the project. Each phase is contingent upon the completion of its predecessor, which makes this organization crucial to the success of this project's completion.

- **Phase 1:** Background Research & Problem Identification
- **Phase 2:** Project Concept & Requirement Development
- **Phase 3:** Prototype Design & Implementation
- **Phase 4:** Prototype Testing & Evaluation
- **Phase 5:** Revisions & Final Design Completion

Phase 1 is deemed the first phase of this project. This phase includes conducting research on the given area and identifying the problem statement that will become the initial foundation for this project. Once this phase is complete, then the project concept can be shaped with the necessary requirements for development – Phase 2. The completion and thoroughness of this phase is crucial for continuing the project in the next phase. Phase 3, includes creating a design that will defy the system architecture and implementing that design to create a working prototype. This prototype shall demonstrate the features defined by the project concept. Upon completion of Phase 3, testing and evaluation of the working prototype will begin in phase 4. Any and all problems encountered and inefficiencies will be used in the proceeding part of this project – Phase 5. This is the final phase of this project, which includes revising the project concept based on the results and analysis from testing and completing the final design for this project.

Given the initial time constraints for the duration of this project, these milestones will provide an adequate structure for organizing this project. Ideally, if this project is taken into further development, the final phase would be extended by repeating Phase 3 and Phase 4 with the necessary revisions determined from the previous revision.

3.6.2. Assigned Roles & Responsibilities

This section will discuss how this project will be conducted with distributing the work amongst the project team. This includes defining the roles and responsibilities for each project member to ensure the integrity and success of this project.

The assigned roles and responsibilities for this project have been broken into three sections, one for each member of the team. While each member has been assigned to a particular section, this only designates who is responsible for each part of this project. All members will collaborate together as a team throughout the project's development.

Hardware Development

The hardware development section of this project consists of the design and implementation of the physical project hardware. This includes, but is not limited to: schematic design, printed-circuit board (PCB) layout, hardware assembly, etc.

- **Assigned Member:** Michael Audi

Software Development

The software development section for this project consists of the design and implementation of the software for the control and functionality of the project hardware. This includes, but is not limited to: embedded & server software design, writing device drivers, application programming interface (API) configuration, etc.

- **Assigned Member**: John Manero

Demonstration Development & Report Management

The demonstration platform development plays a crucial role in this project because it will provide the means to test and demonstrate the functionality of the project's prototype. This includes, but is not limited to: designing a platform model, constructing a climate source & zone simulation, ensure transparent functionality to full-scale system, etc.

Along with the development of the demonstration platform, the management of the report documentation for this entire project is assigned to the same member. This includes, but is not limited to: report outlining and development, formatting, etc.

- **Assigned Member**: Scott Cloutier

Chapter 4: System Architecture

This chapter will define the scope of this project by deriving the specifications and design for the product concept. The system architecture will be composed of hardware and software design components, as well as a test platform to evaluate the results for the completed prototype.

4.1. Climate Control Methods

This section will highlight and discuss the process taken to identify a method for controlling multiple sources and multiple climate zones efficiently and effectively. This method will be used in the design of the hardware and software architecture for the system.

In order to develop a control method for this system architecture, a baseline system must be established. This baseline will be used as an example household system for which this project can be used to control. Given that a typical system can be referenced by multiple points, the structure will be broken up into two parts: Front of House and Back of House. A set list of requirements for this baseline is detailed below:

- **FRONT of House:**
    - AT LEAST: 2 Climate Zones
- **BACK of House:**
    - AT LEAST: 2 Climate Sources
        - 1 Climate Source – "FREE" Energy Source (i.e. Solar-Collector Source)
        - 1 Climate Source – "COSTLY" Energy Source (i.e. Typical Oil Furnace)
    - AT LEAST: 1 Storage Unit

The baseline setup illustrated by the requirements listed above will provide the necessary components to be incorporated by the control method. These requirements have been divided amongst two sections, both illustrating a point of reference in the climate control system. The FRONT of House takes the position of the consumer. This reference will include the entire set of climate zones – in which are the delivery points of the climate control system. The BACK of House will incorporate all the climate sources needed for the system. The requirement for two climate sources will enable the ability to prioritize between "FREE" sources and "COSTLY" energy sources. To improve efficiency of a climate system, the "FREE" energy source will always be of most priority. The "COSTLY" energy source that is served as a typical household boiler will be used as a back-up system with a last priority. A storage unit is included in these requirements for improved efficiency.

4.1.1. FRONT of House

With the defined list of requirements for developing the method of climate control, the first control loop can be created for the system architecture. Taking the first reference point of FRONT of House, the control loop is illustrated below in Figure 19.

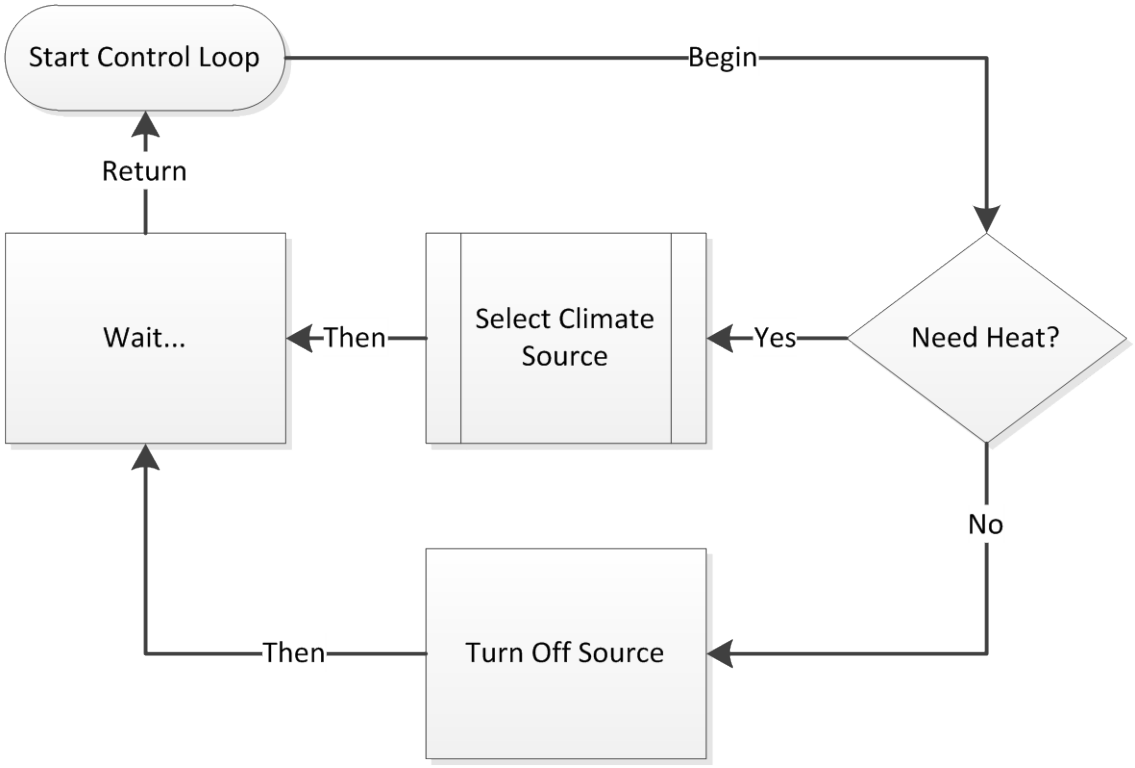

Figure 19 – Main Control Loop

The diagram above illustrates a method of control for this project. Taking the perspective of FRONT of House, the control loop can begin by making the decision for climate change; in this case, the control loop poses the question if heat is needed? This can be taken from measuring the temperature in the climate zone – if heat is needed, then the control system must proceed to select a climate source. If heat is not needed, this means the temperature in the climate zone is at the desired temperature and any climate sources (if initially turned ON) can be turned OFF. Upon completing this process, the system will proceed to wait a small finite amount of time before returning to continue another iteration of the loop.

For the process of selecting a climate source, this control loop will reference another control loop that is located within a different reference point: BACK of House.

4.1.2. BACK of House

Transitioning to the BACK of House reference point, this will be where the climate sources will be controlled. Given the previously defined main control loop in Figure 19, the process of "Select Climate Source" is detailed by a separate process that is illustrated below in Figure 20.

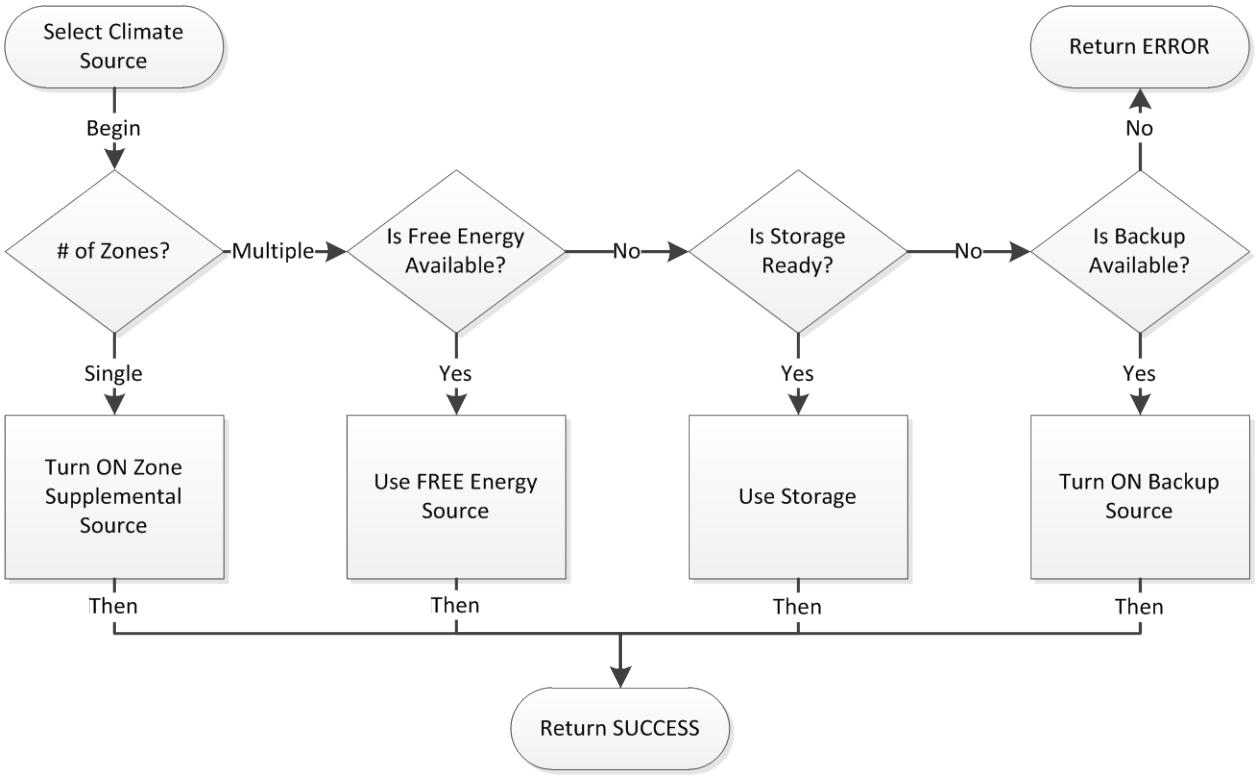

Figure 20 – Select Climate Source Process

The process of selecting a climate source is illustrated above. This process is composed of two phases with respect to the number of climate zones needing heat. If only a single zone requires heat, then the process is called to turn ON that climate zone's supplemental heating system. (i.e. electric heater, wood stove, etc.) If multiple climate zones are calling for heat, then the process continues down a priority list to determine the best available heat source to use. At this point, there are three generalized heat sources to check: Free Energy Source, Storage Unit, and the Backup Source. If the particular source is unavailable or not ready, then process continues to the next-best source that is available. In the case where no climate sources are available, the process shall end with returning some sort of error message. If any source is ready and available, the process will go about initiating that source and then ends with returning a success message to the main control loop.

4.2. Hardware Design Specifications

This section will cover the design specifications derived for the hardware component of this project. This includes defining the input and output requirements, as well as describing the purpose and overall functionality for the hardware component. The hardware design consists of several component modules, as illustrated below in Figure 21.

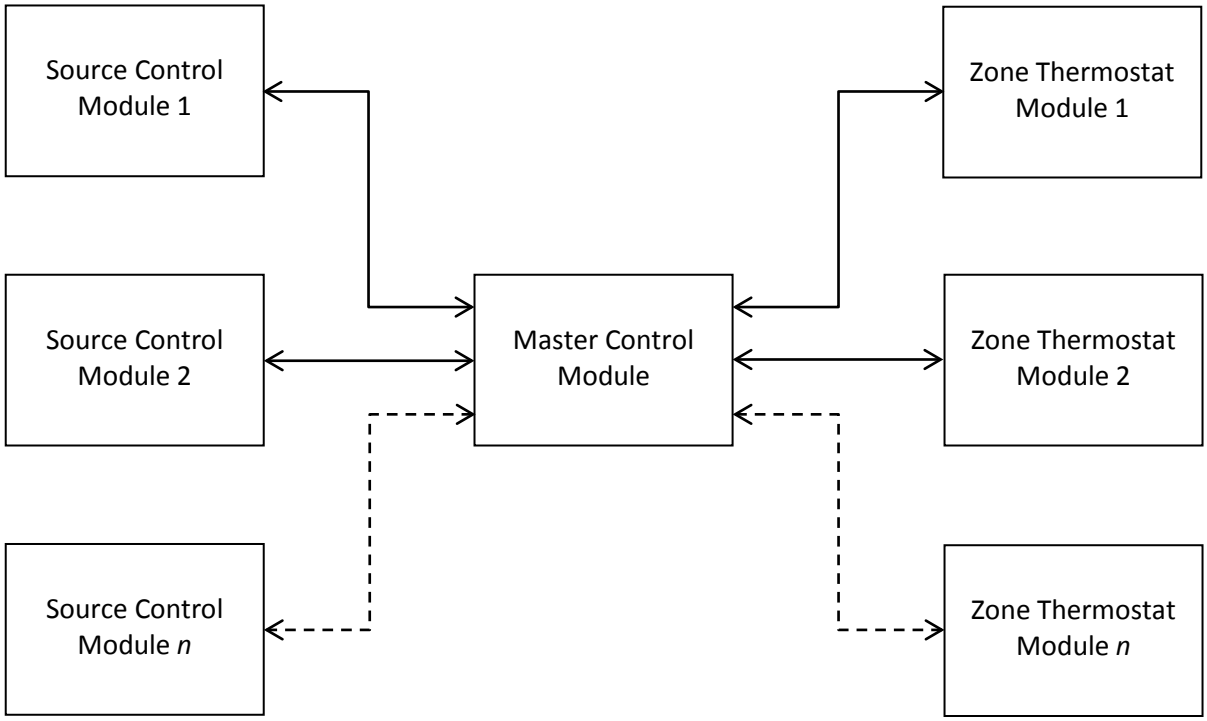

Figure 21 – System Block Diagram

The system-level block diagram above is composed of several modules that are required in order to meet the hardware design specifications. This system includes a Source Control Module, which can be configured as an array with a module for each source. The same goes for the Zone Thermostat Module, in which this module can be replicated for each zone. Both module types will communicate with a Master Control Module that will function as a server to control the entire system.

4.2.1. Source Control Module (SCM)

The Source Control Module or (SCM) will be designed to be adapted to most climate source systems. The primary purpose of this module is to serve as a medium in which the Master Control Server can retrieve input data relayed by the climate source as well as allow the server module the ability to control the functionality of the climate source and its peripherals. A diagram depicting this configuration is shown below in Figure 22.

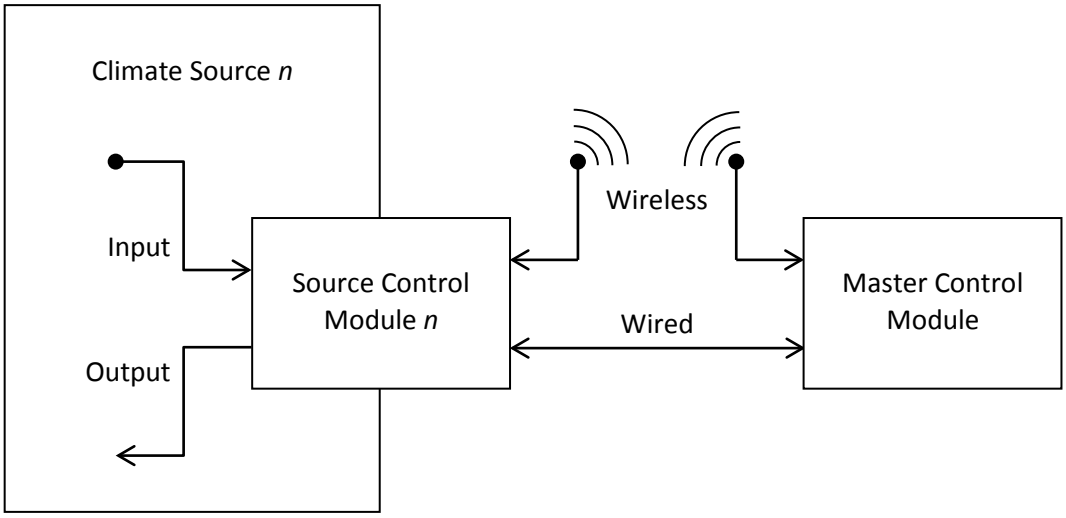

Figure 22 – Source Control Module Diagram

As illustrated above in the figure, the Source Control Module will enable the Master Control Module the ability to communicate with any climate source. Listed below are the input and output specifications for the Source Control Module:

- **INPUT(S)**
  - Analog Signals – to measure temperature, pressure, etc.
  - Digital State Logic – to identify states of devices and peripherals.
- **OUTPUT(S)**
  - Isolated Switching – to control ON/OFF power to devices and peripherals.
  - AC Motor Control – to provide power and speed control to pumps, fans, etc.

With these input and output capabilities, the source control module should be able adapt to any climate source and be able to take measurements of various peripherals and have the ability to identify logic states of devices. With the output specifications, this module will be able to control any switching ON/OFF signal for devices as well as provide control over the speed of AC motors (i.e. fans, pumps, etc.).

4.2.2. Zone Thermostat Module (ZTM)

The Zone Thermostat Module or (ZTM) will serve as a medium for which the Master Control Module can retrieve information and control peripherals pertaining to each climate zone. This module will be similar in hardware design to that of the Source Control Module given similar specification requirements. A diagram illustrating the Zone Thermostat Module configuration is shown below in Figure 23.

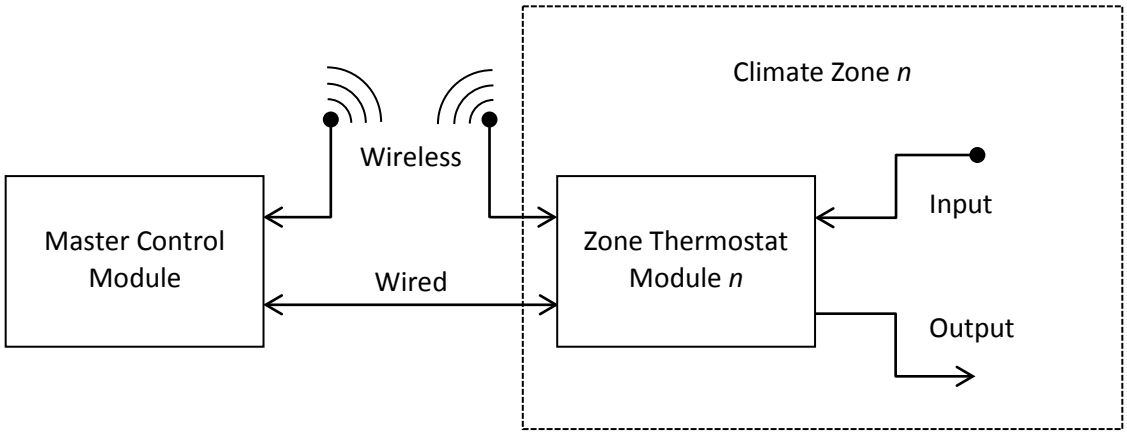

Figure 23 – Zone Thermostat Module Diagram

The configuration shown in the figure above will enable the control server to be able to retain characteristic data for each climate zone destination in order to provide maximum efficiency. The input and output specification requirements are similar to the SCM described in the previous section. A list of specified I/O is shown below:

- **INPUT(S)**
  - Analog Signals – to measure temperature, pressure, etc.
  - Digital State Logic – to identify states of devices and peripherals.
- **OUTPUT(S)**
  - Isolated Switching – to control ON/OFF power to devices and peripherals.

The above I/O specifications should provide the ability to accurately measure data from the climate zone and be able to control the switching functionality of peripheral devices. This will be exclusively advantageous for adapting supplemental heating systems separate to each climate zone to increase efficiency.

4.2.3. Master Control Module (MCM)

The Master Control Module (MCM) or also known as the Master Control Server will function as the primary control device for the entire system. A particular configuration for this module is shown below in Figure 24.

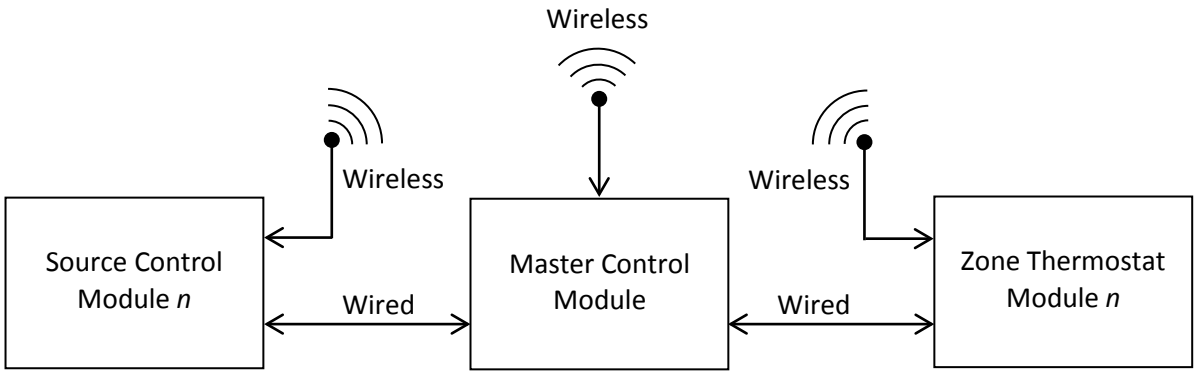

Figure 24 – Master Control Module Diagram

The Master Control Module is the only component in the system that will be singular, whereas both the Source Control Modules and Zone Thermostat Modules are expandable from 1 to *n.* The control server will serve as the primary computing device for running the control software, communicating with the Source Control and Zone Thermostat devices in the system.

To provide the ability for the master control server to communicate with the other modules, a wired medium will be used to connect the devices. To provide flexibility and meet the easy installation requirement for this product, wireless communication protocols will be used as an alternative means for the modules to communicate with the server.

4.3. Software Design Specifications

This section will cover the software design specifications for the project. The software component is crucial to the functionality and efficiency of the entire system. The software component will consist of two parts: low-level and high-level software which refer to the embedded devices and master control server respectively.

4.3.1. Embedded Software Specifications

For the Source Control Modules (SCMs) and Zone-Thermostat Modules (ZTMs), a low-level embedded environment will be used to meet the requirements for these devices. Given the communication requirements for these embedded systems, it has been determined that our system should use TCP/IP, Ethernet, and Wi-Fi for inter-device communication. This allowed the scope of the project to focus upon two specific facets:

1. Producing a high quality hardware module for performing digital and analog I/O at some remote location, and
2. Producing a central module on an off-of-the-shelf server or desktop for reading and processing inputs at all remote modules, and synchronizing the outputs of the remote modules.

Given that constraint, we first focused upon selecting an embedded device which supported or allowed expansion for Ethernet and Wi-Fi, and which included an accompanying open-source TCP/IP API. We found that Microchip Technology produced several products which met these criteria exactly.

Figure 25 – PIC32 Ethernet Starter Kit

Figure 26 – WiFi PICtail Plus Daughter Board

The PIC32 device we selected shown above in Figure 25 is supported by Microchip's TCPIP Stack v5, which includes a large amount of demonstration code. This proved to be functional 'out-of-the-box' with

the PICStart development devices purchased early in the term. We were able to run an HTTP server on the PIC32 which could fetch and set register values based upon URIs and POST request bodies.

4.3.2. Driver Programming

In order to begin the higher-level programming for the control server, the received development platforms had to be configured correctly in order to communicate and function properly.

The Ethernet TCP/IP stack was the first of the hardware devices on the platform to be configured. This was easily accomplished by utilizing Microchip's provided library software drivers for the development platform.

The next step was to get the WIFI PICtail Plus daughter board running properly with the development platform. This proved to be a bit tricky given the lack of documentation and support for the MRF24WB0MA WIFI module. After conducting some research into the hardware combination, it turned out that the particular PIC32 Ethernet Starter Kit was not compatible with the WIFI PICtail Plus daughter board through the I/O Expansion Board. The reason for this was because the required hardware interrupts and SPI channels were being used with the onboard Ethernet components. With some slight modification of the development board's hardware, we were able to get the WIFI module functioning correctly.

The final step in the required embedded configuration was to enable the capability of Program Flash Memory Write-Back. This is necessary in order to be able to store the configuration information needed post-compile for the microprocessor platform such that it will remain available after a hard reset. By enabling this feature, the need for an external EEPROM would be eliminated. The procedure to enable this feature also proved troublesome since it has turned out that this particular development board did not support this feature. Further research produced a special library of software from Microchip that "emulates" an EEPROM on the microprocessor, allowing the storage of data in program memory.

Now as a part of the TCP/IP stack setup on the starter kit, an HTTP configuration page is used to modify such data and store it into the program memory to be retained after reset. This feature is functional across both the WIFI and Ethernet stack setup, so that the HTTP page can be accessed and used to store the configuration files.

With these tools configured, the embedded development and implementation can be completed for this project – which will be discussed in detail in a later section.

4.3.3. Master Control Server Specifications

For the Master Control Module (MCM), a high-level x86 processor-based computer will be used to function as a control server for this system. This control server will provide the necessary computational support and power to provide various levels of control and organization of this multi-source climate control system. A suitable x86 platform being used for development is shown below in Figure 27.

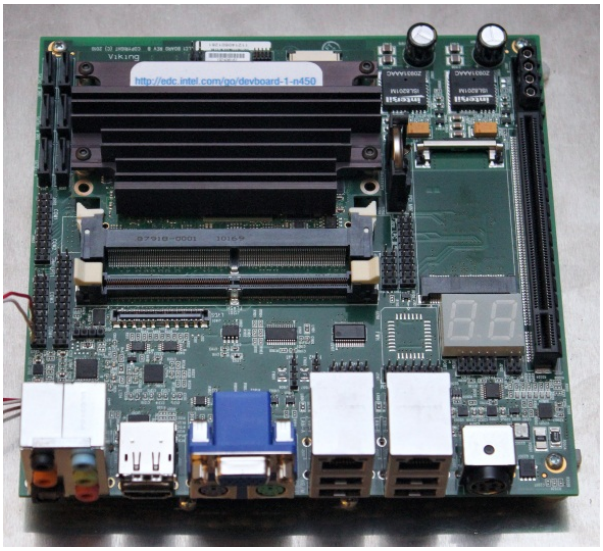

Figure 27 – Intel ATOM 1-N450 x86 Development Platform

The development board shown in the figure above will provide the necessary processing power and peripheral I/O for the MCM. For this project, this particular development board will be used to develop the software and will function as the control server in the project's prototype system.

The control server has been designed to meet several requirements:

1. Implement with modular, extensible multi-platform code
2. Perform calculations using data from multiple network devices
3. Provide a simple web-based user interface

The following sections will describe each requirement in greater detail.

Modular, Extensible Multi-Platform Code

The finished server must be a highly flexible framework for future applications to use the hardware devices discussed earlier. For this reason, the server must be written in a common language with a native packaging mechanism and multi-operating system support.

Perform Calculations

The underlying purpose of the control server is to perform calculations on large, complex datasets compiled from multiple network sources. This functionality turns the hardware devices, whose design represents a large part of this project, into a distributed control system.

To accomplish this end:

1. The server must be able to store a hierarchical description of its constituent network devices and their ports. A design similar to the figure below will allow the server to periodically send and receive updates to and from devices while simultaneously performing internal calculations using ports without regard for their parent devices.

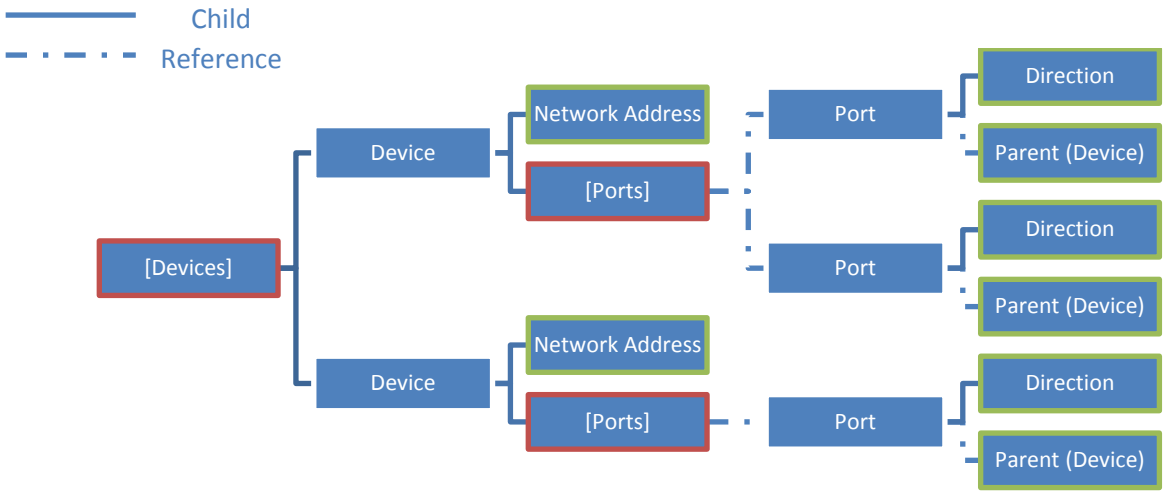

Figure 28 – Device/Port Hierarchy

2. The server must be able to communicate with network devices. It should be able to use 'plug-able' connector modules to ensure that future revisions of device firmware with different APIs can be easily supported without major software upgrades.

Simple Web-Based User Interface

The server will provide a central user interface for the distributed system. This interface must be easily modified to suit future applications. For this reason, the server must also provide a simple internal API for accessing and modifying objects in its data structure described above. The implementation of the user interface, data structure, and device interface will be described in detail in section 6.2.

4.4. Test Platform Specifications

This section will set forth the requirements for the demonstration platform. These requirements will be used to determine the criteria for which the model will be created from. To develop a project model, whether it is for a product or a demonstration platform, a list of requirements must be determined in order to identify the constraints of the project when developing a model. Shown below is a list of explicit and implicit requirements determined for the demonstration platform.

Explicit Requirements

- To have at least: 1 build revision of the platform
- To include at least: 2 primary heat sources, 1 backup heat source, 1 climate zone
- To use less-than: $500 per build

Implicit Requirements

- Safety
- Low Maintenance
- Aesthetics

To develop a solid demonstration platform that highlights all key features of a product, it has been determined that multiple revisions of the platform. This will ensure that any minor issues or inefficiencies in the first build can be refined in the next build. Shown below is a set of criteria for constructing the demonstration platform, broken up based on the build version.

- **First Platform Build – Rev. 1**
    - *Functionality* – This build will be primarily focused on creating a functional system.
- **Second Platform Build – Rev. 2**
    - *Efficiency* – The focus will be on improving the efficiency of the demonstration platform.
    - *Expandability* – Improvements by expanding the system will be incorporated.
    - *Aesthetics* – This build will take into account visual refinements as necessary.
- **Third Platform Build – Rev. 3 – *(Optional)***

A third build has been included as a possibility based on the results derived from the first two builds. If any additional refinements are needed, this build will be used to polish the demonstration platform. The requirements and criteria described in this section are subject to change during the course of the semester.

4.4.1. Developing a Model

This section introduces a possible working model for a demonstration platform based on the requirements and criteria determined above. For the initial build, the figure below is a model of one possible idea for a demonstration unit using two unique heat sources, a heat storage tank, and a radiator inside a concealed climate zone.

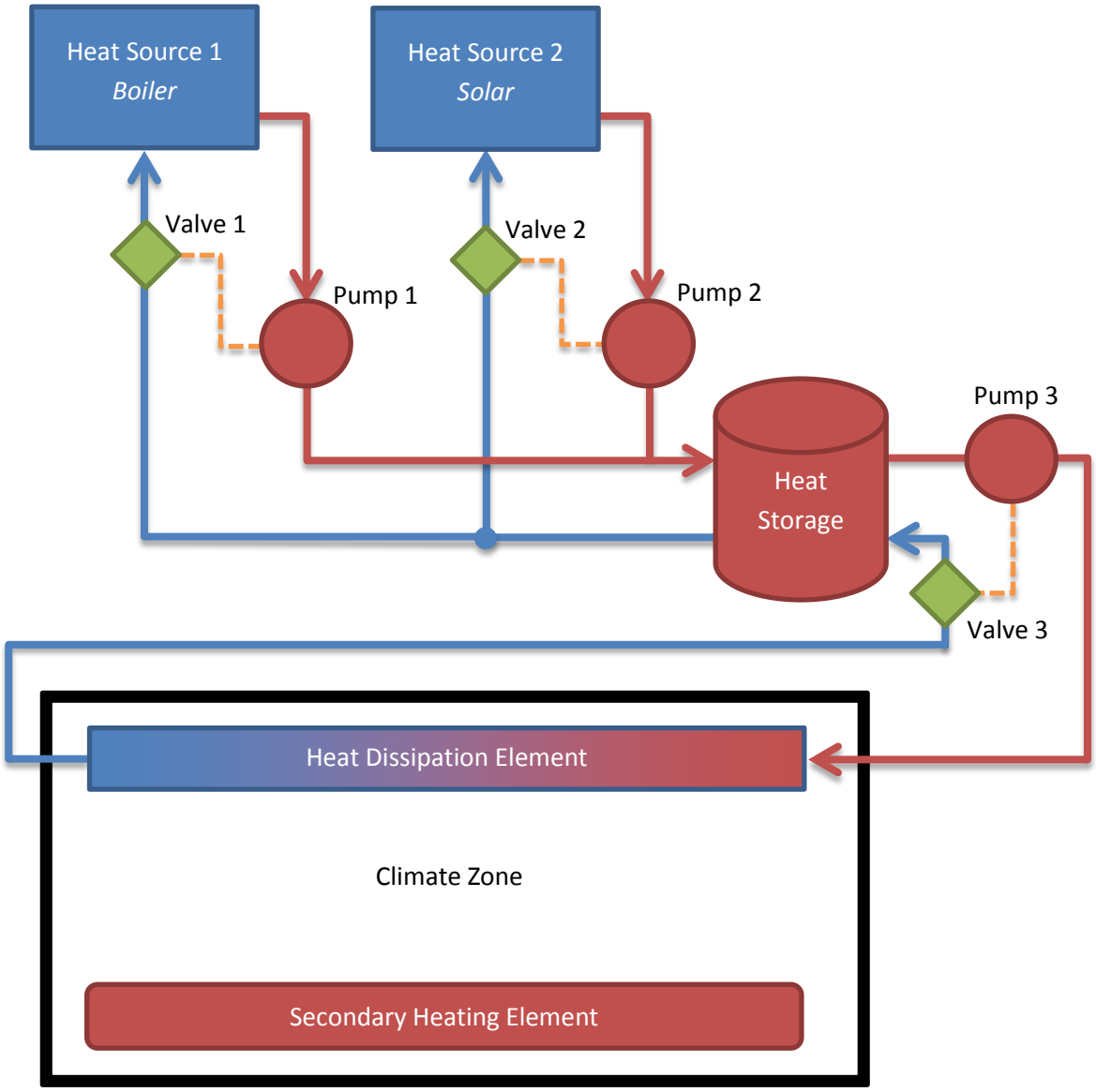

Figure 29 – Demonstration Platform

The general idea of the diagram shown above is to demonstrate the efficient heating of a climate zone using our Universal Climate Control System to control and maintain the operation of two primary heating sources and a backup electric heat source.

4.4.2. Primary Devices

This section will dive into the model described above by highlighting in detail the devices and configurations of each device. These devices are the primary heat sources, heat storage tank, and climate zone.

Heat Source 1 – Boiler Simulation

The first heat source will represent a typical boiler setup, through the use of a heating element as the source and water as a carrier. The figure below is a diagram that depicts a possible way of setting up Heat Source 1.

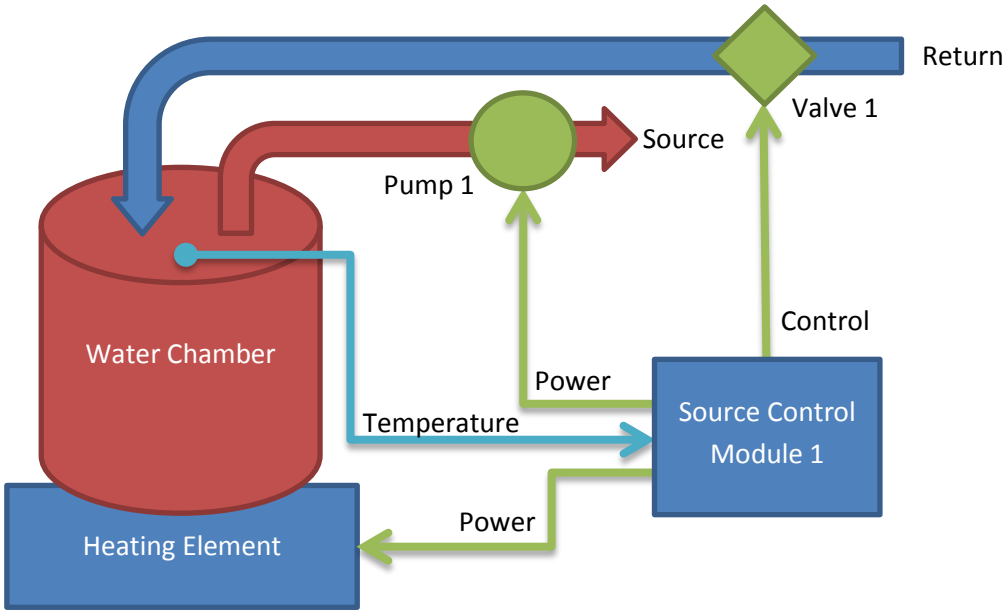

Figure 30 – Heat Source 1 Model

The illustration shown above in Figure 30 is one particular possibility for configuring Heat Source 1. To simulate a typical boiler-type heating system, a hot plate is being used as a heating element to heat a chamber filled with water. This substitution is primarily for safety concerns with utilizing a fuel-burning (gas or oil) boiler system in the demonstration platform.

With the configuration above, the Source Control Module (SCM) will be used to control the power to the heating element as needed. Two additional control lines have been added to control the function of the source pump and return valve. The SCM will open the valve and supply power to the pump to transfer the heated water to the system and allow colder water to return to the water chamber for re-heating when needed.

Heat Source 2 – Solar Simulation

For systems that may utilize a source of green energy heating, solar heating has been chosen to represent Heat Source 2. The figure below illustrates a particular configuration in which this heat source can be set up.

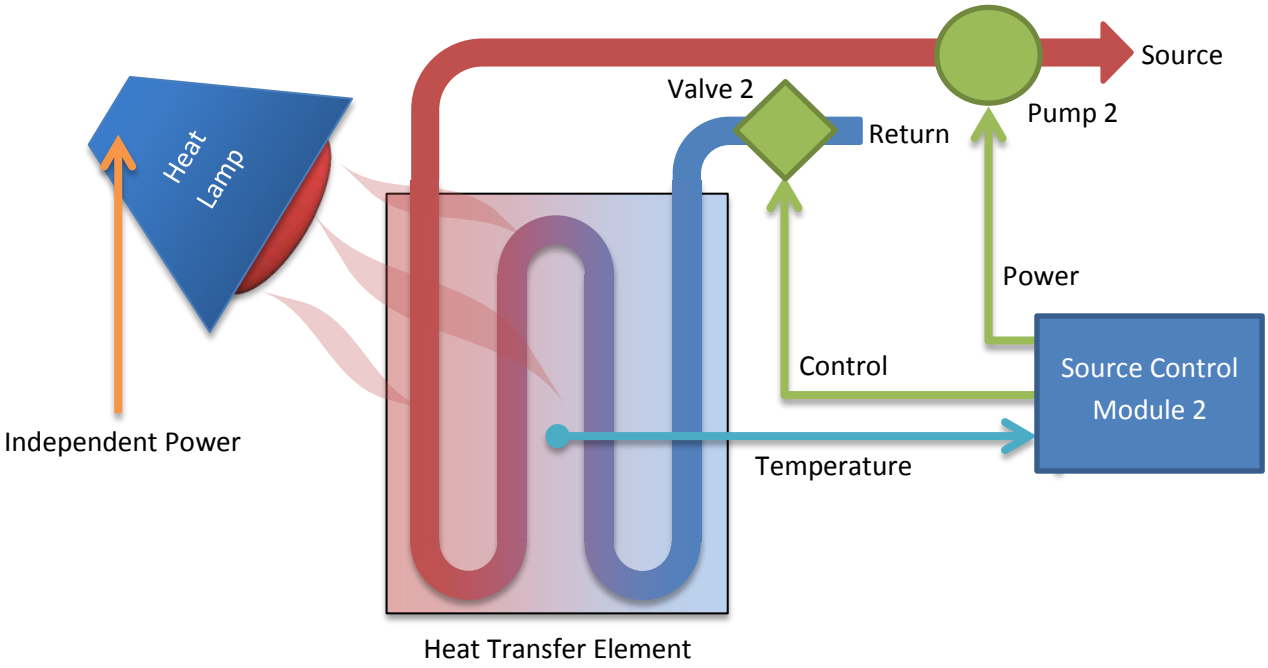

Figure 31 – Heat Source 2 Model

In Figure 31 above, a solar-heat simulation is being constructed as a model for Heat Source 2. The primary source of heat will be produced from a heat lamp that is controlled independently from the system. This lamp will be used to simulate solar-heat energy. The heat energy produced by the heat lamp will be directed on a heat-transfer element in which the heat will be absorbed by cold water passing through the element. The Source Control Module (SCM) will be used to monitor the temperature of the heat transfer element. If the temperature reaches a certain threshold, this will provide an indication that "solar" heat energy is present allowing the system to utilize this renewable energy. The SCM will control the power to Pump 2 which will direct water through the heat-transfer element when needed.

Heat Storage Unit

In order to store heated water when it is not needed, a storage tank can be used. This storage tank is shown below in Figure 32.

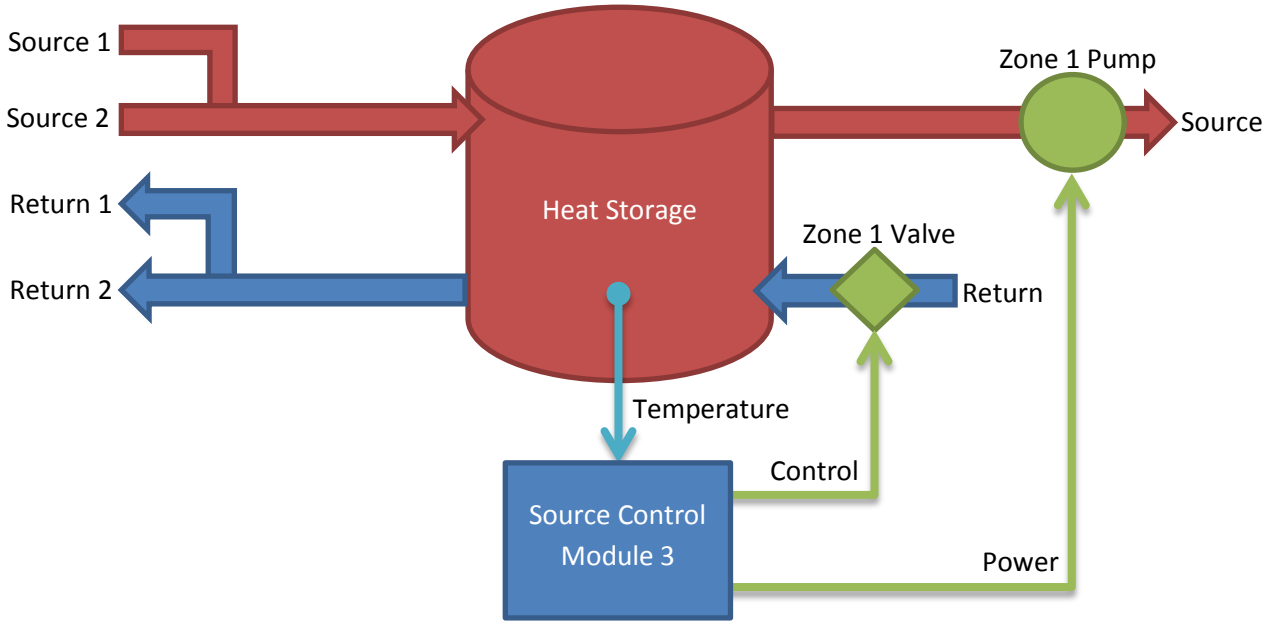

Figure 32 – Heat Storage Model

The configuration shown above in Figure 32 is one possible model for setting up a heat storage tank for the system. This tank will serve as a multi-sourced heat storage to increase the efficiency of the system.

There are several variations and configurations that can be made to the model setup shown above. This one in particular uses a Source Control Module (SCM) as a controller for the Heat Storage tank system. This module will be used to monitor the internal temperature of the heat storage tank, as well as control the source pump and return valve for the system. This setup has been configured for only one zone in particular, but can be easily configured for multiple zones by adding additional sources and returns to this storage tank.

Climate Zone 1

The climate zone is the area in which the temperature will be regulated by this Universal Climate Control system. The climate zone can be any room or a number of rooms adjacent to each other located in a building or house. This system can support multiple zone configurations, but for the first build of this demonstration unit, only one zone will be constructed. The figure below is a model of how the climate zone could be configured.

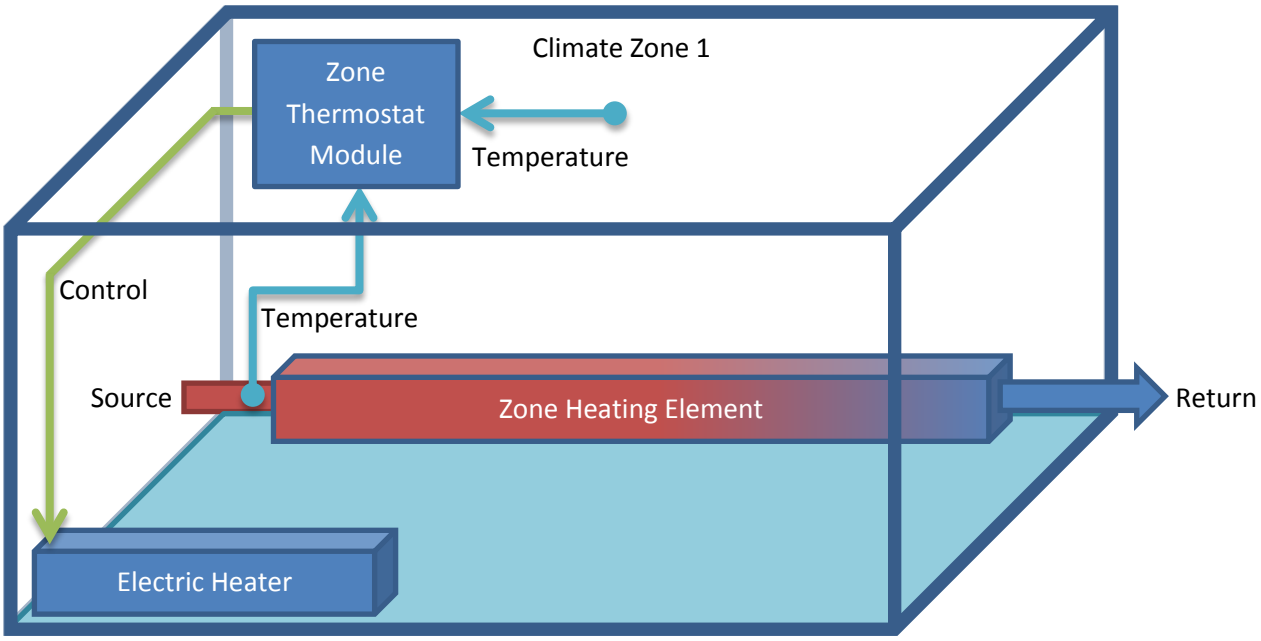

Figure 33 – Climate Zone Model

The figure above is a model of how Climate Zone 1 can be configured. This model will incorporate a Zone Thermostat Module (ZTM), which serves the primary purpose to measure the temperature of the zone. This Module is setup to measure one point of temperature, but can be configured to measure the temperature of multiple points placed throughout the zone.

In this particular configuration, an electric heater is set to be controlled through the Zone Thermostat Module. This heater will serve as a supplemental heating source to improve the efficiency of the entire control system. This particular heating source can be any secondary system, such as: an electric heater, pellet stove, wood stove, etc.

Chapter 5: Hardware Implementation

This section will highlight the progress made in terms of hardware development for the project. This section details the work done from the start of the project up to the current state for the proprietary hardware. The sub-sections represent the hardware design process and flow sequentially as described below.

A method used for conducting the implementation of the hardware is broken down into three phases: Design, Implementation, and Results. This process is illustrated in the diagram shown below in Figure 34.

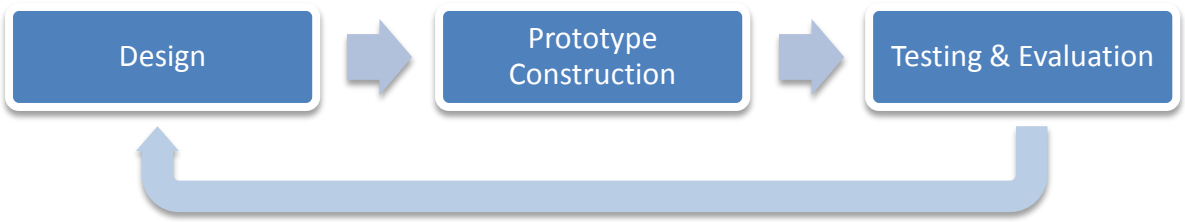

Figure 34 – Hardware Design Process

The process shown above depicts a particular method used for implementing the hardware for this project. This implementation process is described in detail below:

- **Design Phase:** The actual hardware implementation is contingent upon the completion of the design phase. This phase includes taking the initial product requirements derived in a previous section and finding the necessary components and circuits that will meet these requirements.
- **Construction Phase:** Prototype construction begins when the design phase has been completed. This includes the assembly of the physical circuits on printed circuit boards (PCBs) based on the schematic plans that have been created in the design phase.
- **Evaluation Phase:** There are multiple tasks that occur in the evaluation phase. Once the prototype hardware has been constructed in the previous phase, then testing and evaluation is done to identify problems, inefficiencies, and any other areas in which the prototype hardware does not meet the product requirements. Completion of this phase will provide the necessary results that are needed to go back and start the Design Phase again with a new revision.

Upon completing this hardware design process multiple times, the hardware implementation can be completed such that the prototype meets all requirements and is ready to be manufactured and brought to a market.

5.1. Design Implementation

The first step in design is to identify a set of requirements. For the scope of this project, only prototype hardware was required to demonstrate functionality. This is largely due to time and resource constraints. Hardware design is an iterative process; where an initial design receives varying levels of modification before finalization. Available resources, mainly time, employees, experience, and money typically define the length of this process. As mentioned before the scope of this project entailed a functional demonstrable prototype, which defined the majority of the requirements. In order to produce as accurate of a prototype as possible the following hardware requirements were selected.

Explicit Requirements

- Wi-Fi and Ethernet Communication
- USB
- Multiple Power Sources (120VAC, 24VAC, USB, 24VDC)
- 8 12-bit ADC Ports (3.3V & 5V)
- 4 Mechanical Relays
- External SPI Port
- 2 AC Motor Speed Controllers
- 5 General Purpose I/Os (GPIOs)
- User Interface with LCD

Implicit Requirements

- Flexible Design
- Simple Installation
- Reasonable Cost ($$)
- Simple Testing and Debugging
- Attractive Form Factor

The design and implementation of the requirements was focused on available resources (time, experience, and money) and proof of concept rather than reliability and manufacturability. This prototype design concept was often the main factor in the design decision process and will be pointed our repeatedly throughout the following sections. The majority of the design was completed utilizing PADS Logic for schematic creation; however analog simulations were completed within Multisim.

Based on the available resources and to simplify the process a single module was designed to function as both a ZTM and a SCM. The core required components for each module include basic I/O signal processing, power, and communication. The design theory makes use of this commonality with detachable modules to satisfy additional needs. This module design is separated into five sections, processing, communication, peripherals, power, and the user interface, which are detailed below.

5.1.1. Processing

The processing components control the entire module and are major defining components. This means that the selection of the processor will define other components selected, such as power requirements. Processor selection is one of the most critical areas in rapid prototype design. Important factors to consider are software development time, technical support, flexibility, and device stability. For this project a PIC32 was selected.

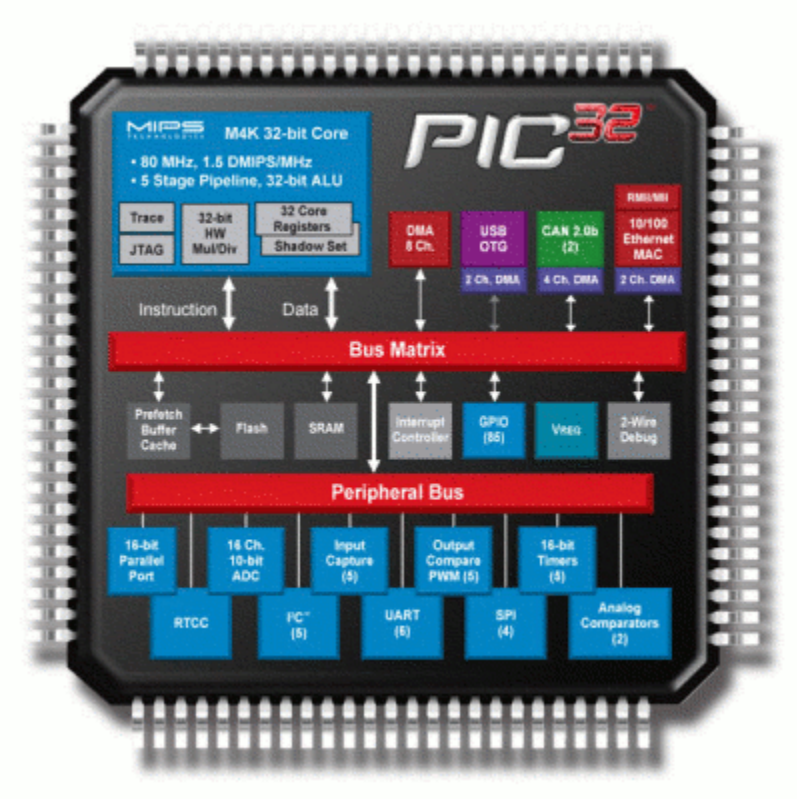

Figure 35 – PIC32 MCU Core Layout

The PIC32 is a relatively inexpensive 32-bit microcontroller with large online support. The manufacturer also provides simple Wi-Fi and Ethernet communication modules specifically for interfacing with their MCU product line. Other microcontroller lines considered were the ARM 7 and Cortex M3. However, the simple Wi-Fi communication was the driving factor in processor selection due to the team's limited experience with embedded Wi-Fi communication solutions.

The Processor

The specific PIC32 selected is the PIC32MX795F512L-80I/PT-ND; it offers 85 general purpose I/O (GPIOs) pins, 512KB of flash memory, and 16 10 bit analog to digital converters. The GPIOs are configurable to support necessary embedded communication protocols, such as SPI, CAN, I2C, UART, USB, and Ethernet. The controller requires several basic components for operation; these are bypass capacitors, external oscillators, a programmer, and a reset circuit.

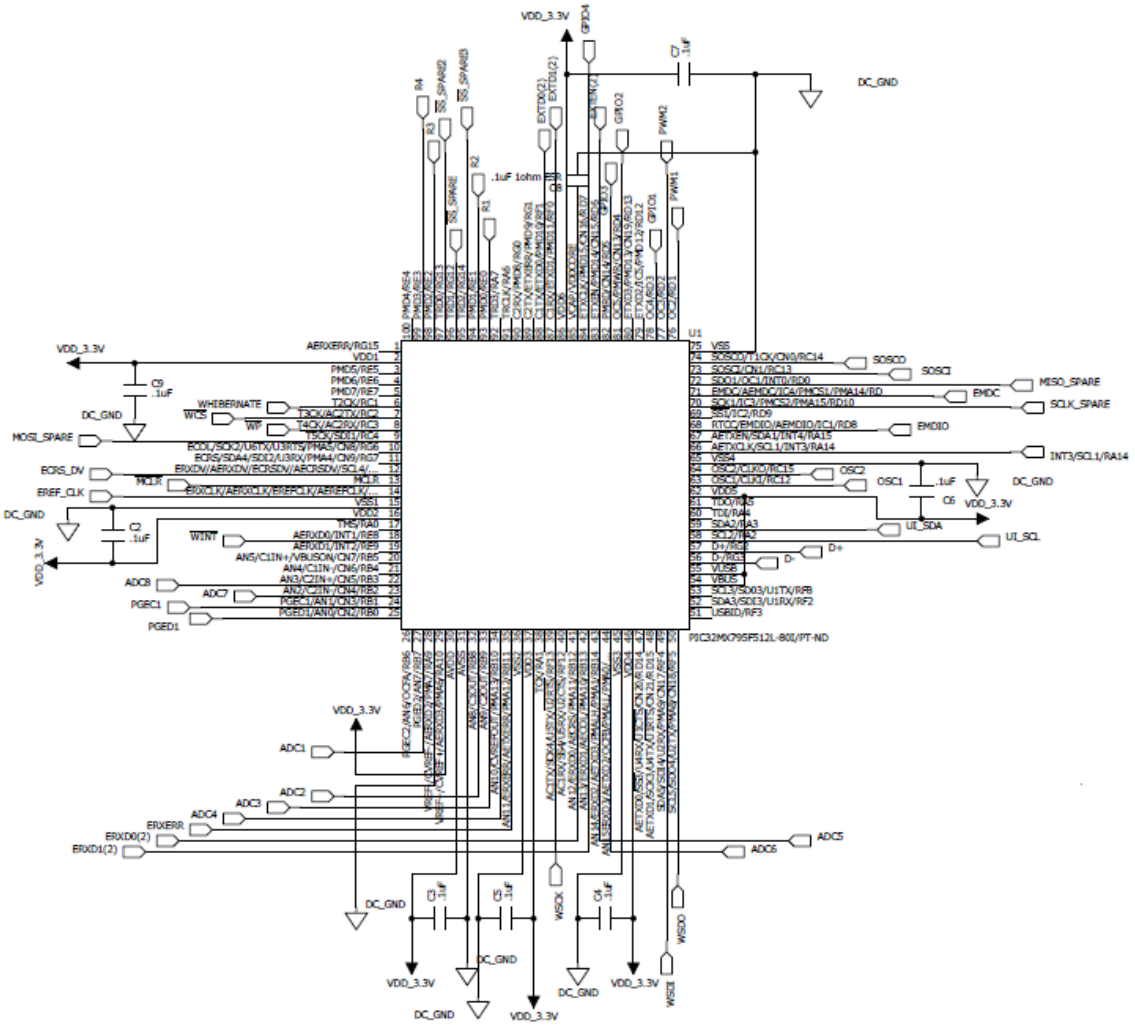

Figure 36 - PIC32MX795F512L Schematic Layout

Bypass capacitors are critical for power supply stability; they need to be placed within 6mm from each VDD and VSS pin on the PCB. These capacitors are .1uF and shown with the PIC32 above in Figure 36.

The PIC32 utilizes an internal oscillator for several applications; however, to increase the number of timers available and for high speed communications, external oscillators are connected. The schematic below illustrates the configuration of these external oscillators.

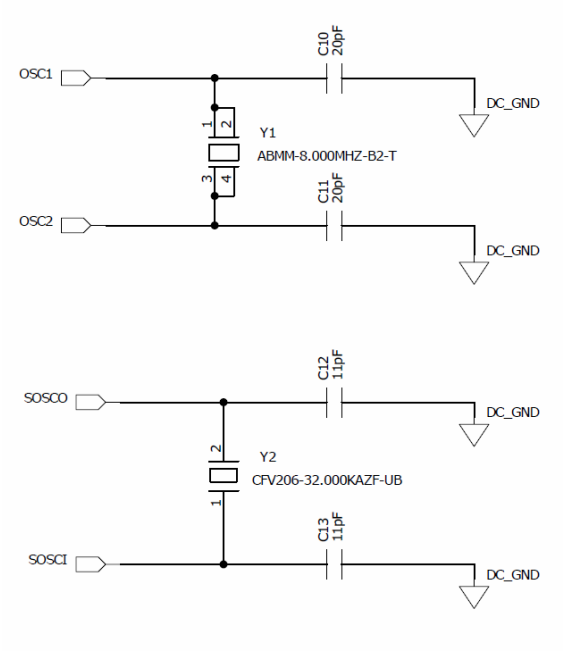

Figure 37 – Oscillator Circuits

An 8 MHz crystal is used along with a secondary low speed 32 KHz crystal as shown in the schematic above in Figure 37. The PIC32 includes internal PLL dividers and multipliers for modification of timer speeds.

Programmer & Reset Circuit

Microchip (PIC32 Manufacturer) utilizes a proprietary serial programming and debugging standard called in circuit serial programming (ICSP). ICSP is a two wire bi-directional standard, it only works with controllers designed specifically for it. Microchip also offers a free integrated development environment (IDE) compatible with ICSP programming and debugging. This method also requires an external programmer and debugger, the current model being the PICkit 3. This device utilizes a six pin header, requiring power, ground, the two bi-directional pins, and the device ~MCLR shown below in Figure 38.

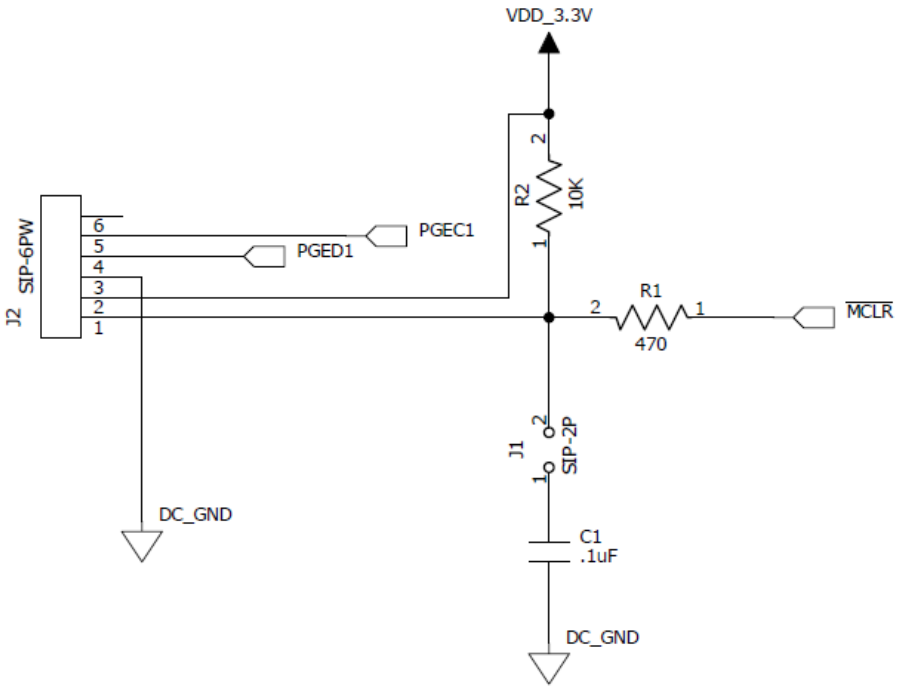

Figure 38 – ICSP Circuit

The ~MCLR pin must be pulled high (to 3.3V) for controller operation. The impedance on the pin needs to be considered because of high sensitivity to cutoff tolerances and timing. The manufacturer recommended configuration of a 470 Ohm resistor in series and a 10K pull up resistor is used (Figure 38). In addition, a .1uF capacitor to ground is recommended to hold pin voltage constant in case of system glitches, during normal operation. It is often convenient during testing to be able to reset the controller.

For the reset switch, the circuit illustrated below in Figure 39 is a common solution with microcontrollers. It includes hardware debouncing, a technique used to minimize switching errors. This is accomplished by producing a delay between logic level changes, making sure the controller isn't reset multiple times in a row potentially damaging the unit. A simple RC circuit coupled with a Schmitt trigger is used.

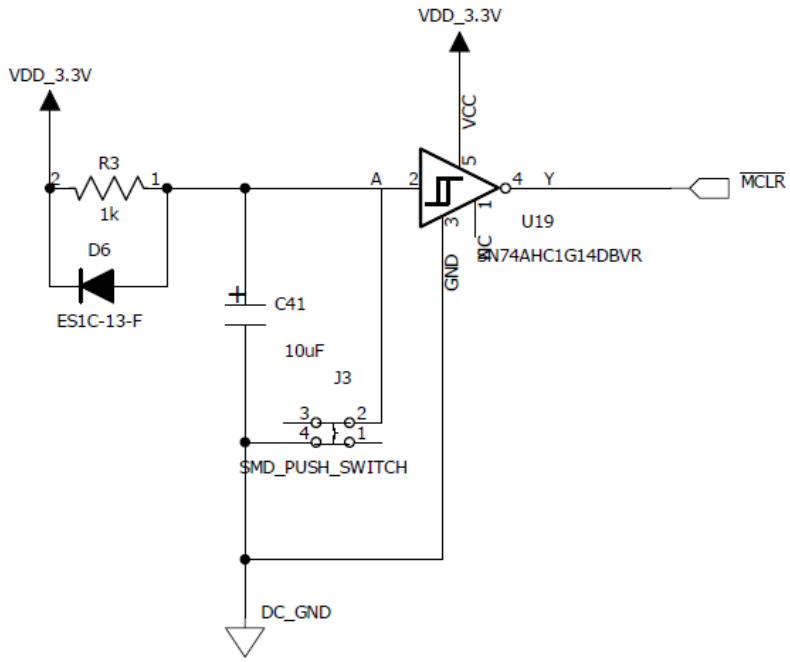

Figure 39 – Reset Switch Circuit

A Schmitt Trigger is a threshold circuit; it utilizes hysteresis to maintain a constant output until a certain cutoff value is met. The RC circuit provides a slow timed charging (10ms) of the capacitor C41 after each reset, defining the voltage input to the Schmitt Trigger. The voltage input with respect to time is modeled by the equation:

$$Vc = V(1 - e^{-t/RC})$$

Diode D6 creates a top voltage input level of 3.3 volts. It also stops the reset from pulling the entire 3.3V DC bus to ground.

5.1.2. Communication

An important requirement for this project is communication. For simple integration and development a standard TCP/IP stack is used. This was selected over the common low power protocol zigbee because it provides a simple common interface to the master controller. Microchip provides a free TCP/IP stack for use with their PIC MCUs. This will be used for initial prototype purposes with the intention to move to a commercial TCP/IP stack during later development.

Ethernet Communication

To accomplish the task of TCP/IP communication Ethernet and Wi-Fi protocols are used. The PIC32 has pre-packaged pins dedicated to communicating with an Ethernet physical layer chip (PHY) shown below in Figure 40.

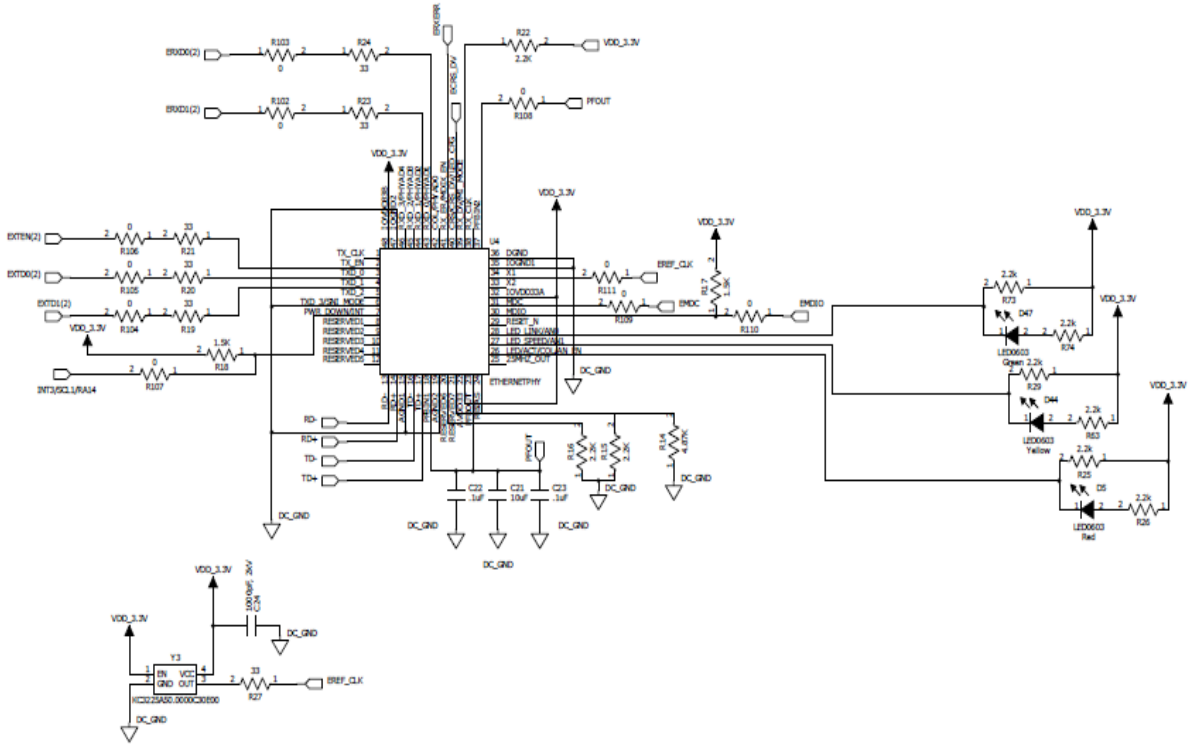

Figure 40 – Ethernet Physical Layer Circuit

The PHY is responsible for implementing the hardware send and receive functions of Ethernet frames. It provides the link between the physical medium to the PIC32. The specific PHY chip (DP83848C) was selected because it is used on the PIC32 Ethernet Starter Kit, a demo board available for purchase from Microchip. Since this specific PHY chip has already been implemented and tested for production, it simplifies the development process. It also requires a high speed 50 MHz crystal.

In addition to the PHY, a component in Ethernet communication commonly referred to as the magnetics is required. Magnetics are required for RX & TX +/- pins in order to isolate the device. The magnetics are simply a 1:1 signal transformer with a few passive components defining voltage levels (Figure 7).

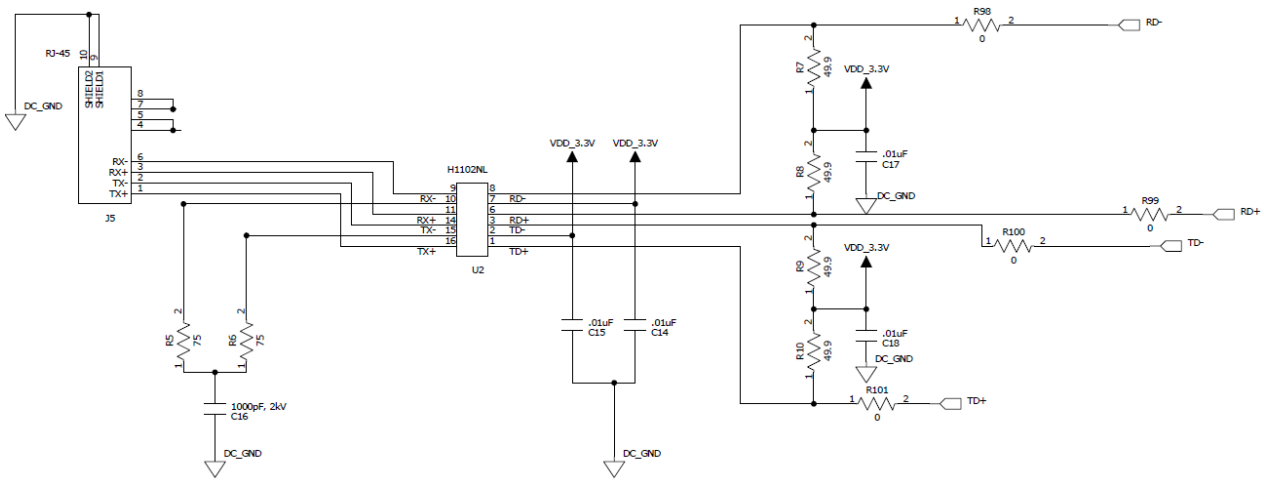

Figure 41 – Ethernet Magnetics Circuit

Often they are embedded directly into the RJ-45 connector. However, due to the desire to have the possibility for Power over Ethernet (PoE), the magnetics are separate. This allows access to the extra unused pin pairs required for PoE.

Wi-Fi Communication

For quick development Microchip's MRF24WB0MB pre-packaged Wi-Fi module was selected which is compatible with IEEE standard 802.11 b/g/n protocols. This module is shown below in Figure 43.

Figure 42 – Microchip's MRF24WB0MB Wi-Fi Module

This module consists of the necessary physical layer device, necessary passive components, an antenna, and a mini co-axial connector antenna extender. The module is on a single surface mount 36 pin PCB and is connected via the circuit shown below in Figure 43.

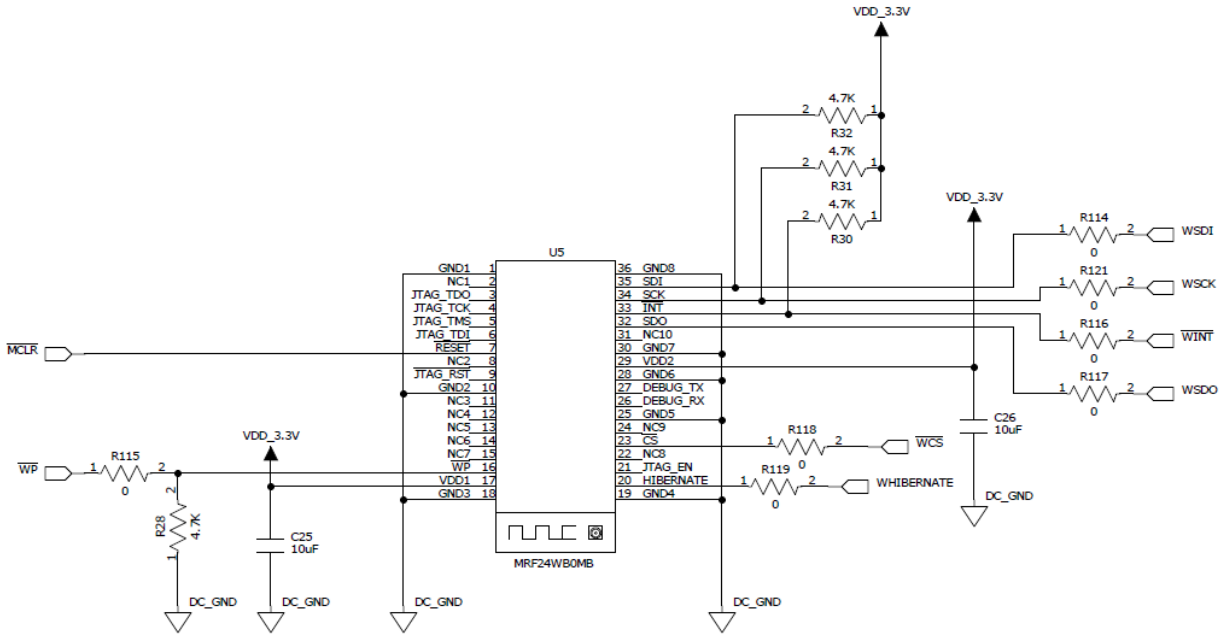

Figure 43 – WIFI Module Circuit

The module communicates to the PIC32 via the communication protocol SPI, an external interrupt ~WINT, and a write protect pin ~WP. In addition to the TCP/IP stack provided by Microchip, source files for operating this module with a PIC32 are supplied APPENDIX X

USB Communication

USB communication is implemented on the device. The device is set up as a USB slave device; this provides the user a method for writing directly to the flash memory. This was added as a method for setting up each device. A simple application can be used to allow the user to write set-up data such as a network security key to the flash memory upon installation. The PIC32 includes the capabilities of a USB host and USB On-The-Go (OTG) however these features are unnecessary and not included. The circuit depicting this configuration is shown below in Figure 44.

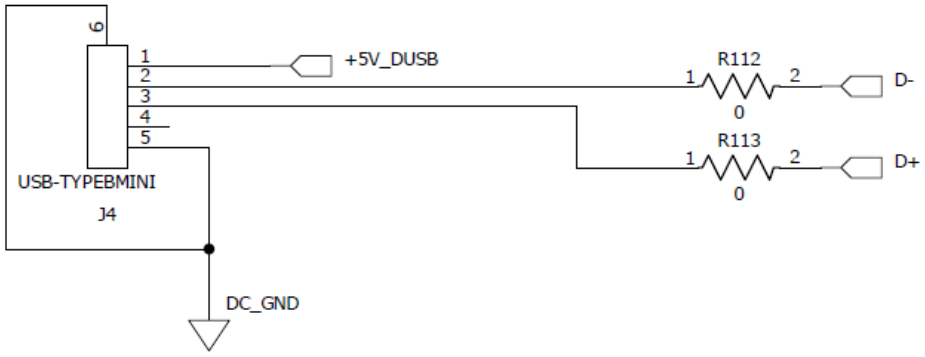

Figure 44 – Universal Serial BUS Circuit

A USB slave connection requires only the D+ and D- bus connections (Figure 44). These lines are a differential pair and PCB traces should keep a constant distance of .007 inches from one another and avoid any unnecessary angles. Connection to the 5VDC power bus is also included to provide limited power during installation.

5.1.3. Peripherals

Peripherals form a significant component of the device structure. Using different peripheral controls the device interfaces with external systems. These external systems range from pumps, motors and fans, to temperature, pressure, and level sensors. To accommodate this wide range of devices several interfaces exist in the hardware. These interfaces are defined by the explicit requirements, they are:

- 8 12-bit ADC Ports (3.3V & 5V)
- 2 AC Motor Speed Controllers (0-120VAC)
- 4 120VAC Electromechanical Relays
- 3 External SPI Ports
- 5 General Purpose I/Os (GPIOs)

Analog to Digital Converters

The PIC32 includes 16 channels of 12-Bit ADCs used for reading analog data from sensors. Accurately reading a large set of sensors is integral to the system design. Sensors outputs range in value; two of the most common inputs are 3.3V and 5V sensors. To simplify the design and to accommodate a wide range of sensor inputs only 5V and 3.3V direct ADC inputs are provided in the hardware shown in Figure 45.

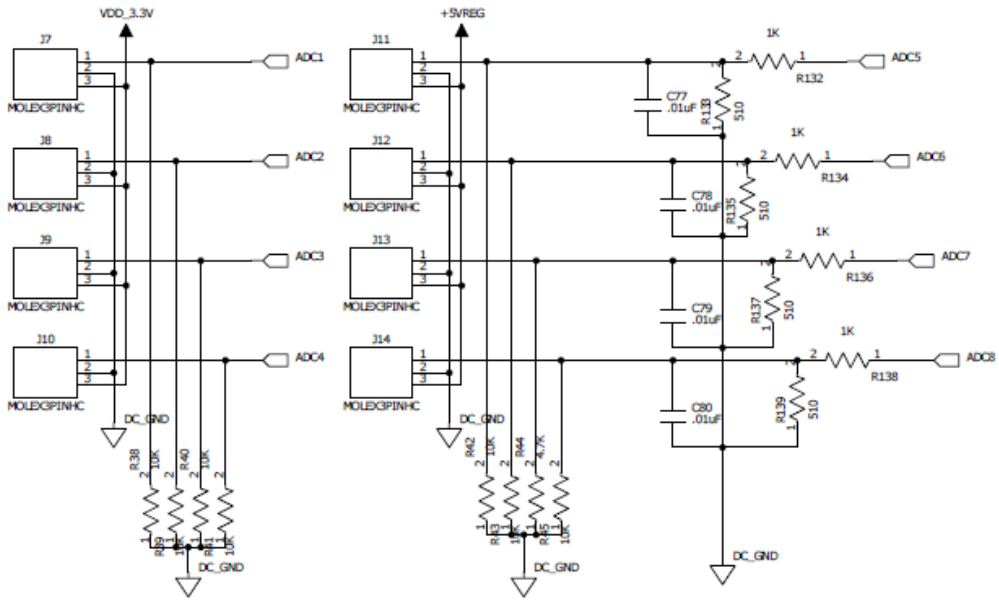

Figure 45 – Analog-to-Digital Input Circuits

The assumption is that necessary external circuitry will be used between the sensor and the interface. Microchip recommends the use of a pull down resistor on potentially unused pins to limit power consumption. These are seen on both the 3.3V and 5V ADC input lines. The ADC pins on the PIC32 are

only 3.3V tolerant. A voltage divider with a smoothing capacitor is used to accommodate 5VDC sensors, while the 3.3VDC sensors are connected directly. The voltage divider steps the input down from 5V to 3.3V using this equation:

$$Vout = Vin\left(\frac{R2}{R1 + R2}\right)$$

When selecting resistor values input impedance must be kept below the tolerance level of 10KΩ.

For testing and demonstration purposes, thermocouples needed to be connected to the PIC32 ADC pins. Thermocouples are a measurement device made from two conductors producing a voltage proportional to the temperature between either ends of the conductors. A typical thermocouple is shown below in Figure 46.

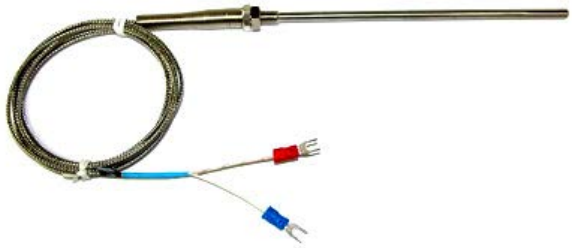

Figure 46 – Typical Immersion Thermocouple Probe

Commonly used for temperature measurement in HVAC applications, due to the high temperature accuracy. In addition, unlike thermistors, thermocouples are not self-warming, simplifying the circuitry and software. The thermocouples selected for demonstration purposes, produce a linear voltage output of -.5mV to 2.7mV from 50°F to 160°F determined empirically. However, the PIC32 ADC resolution is only 3.2mA, calculated by:

$$\frac{Vmax}{2^{ADC\ Resolution}}$$

To use these thermocouples with our device hardware, an amplifier circuit needed to be constructed. The signal -5mV to 2.7mV needs to become a signal from .5V to 2.7V. The amplification technique selected is a differential amplifier, common in thermocouple measurement.

The circuit amplifies the difference between the two input terminals. The amplification uses an operational amplifier (op-amp), a device which using feedback can amplify a signal between its two power rails. Typical op-amp power rails are +/- voltages (i.e. +5v and -5V), putting the ideal operating region at the center (0 volts).

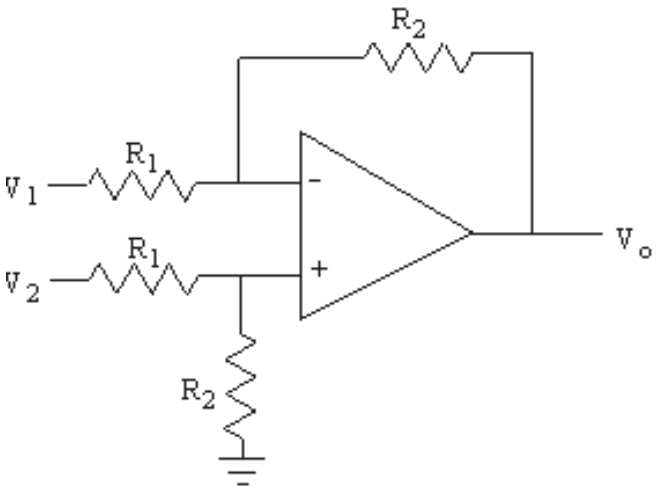

Figure 47 – Differential Amplifier Circuit

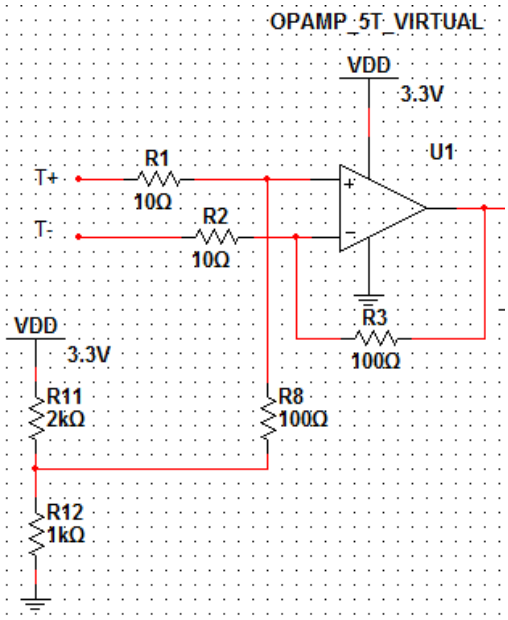

Figure 48 – Differential Amplifier with DC Offset

In our system, the available voltages are +3V and GND with the ideal operating region at 1.6V. This becomes troublesome because op-amps also require a minimum turn on voltage. To meet this requirement, a voltage divider is used to create a DC offset (Figure 48), this shifts the thermocouple input voltage to varying between 1.095 VDC and 1.1027 VDC. This difference between the thermocouple input pins after the offset is amplified 10 times, calculated using the equation:

$$Gain = \frac{R2}{R1}$$

This output then enters another differential amplifier with a gain of 100. To keep the output within the operating region of 0-3.3V, the input is compared to the same fixed DC offset as in the first amplifier (Figure 13). However, it makes use of an independent voltage divider otherwise the circuit experiences gain issues due to the feedback. Two op-amp circuits are required because R8 directly affects the gain within the circuit, to achieve the overall gain of 1000 a 10K Ohm resistor would be required, which would disrupt the DC offset applied. Figure 14 depicts an oscilloscope comparison of the thermocouple

input to the amplifier output. Channel B is the thermocouple input measured on a time scale of 5mV/Div. The input was simulated using a saw wave representing the thermocouple range of operation. Channel A is the amplifier output to the ADC measured on a time scale of 1V/Div.

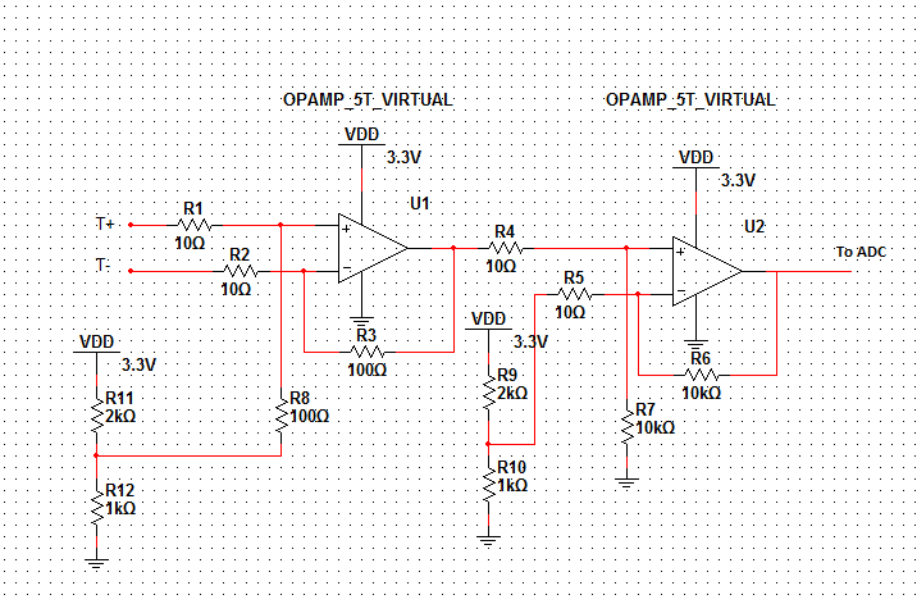

Figure 49 – Thermocouple Signal Amplifier Circuit

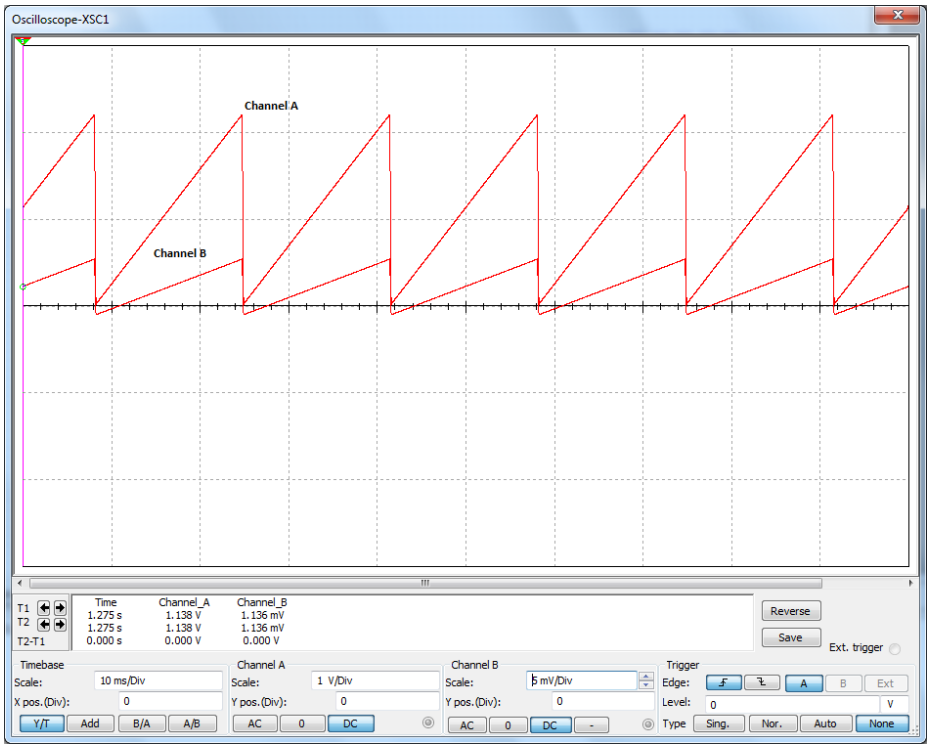

Figure 50 – Amplifier Input/output Comparison

Spare SPI and GPIOs

External motor speed controllers, sensors, and other devices are common practice. The serial peripheral interface (SPI) is one of the most commonly used communication protocols for these applications. It allows multiple SPI devices to be connected to a single bus with each device utilizing a chip select signal. When data is to be written or read from a device the chip select value is set. The hardware provides a single spare SPI bus with three chip select signals as illustrated in the schematic below in Figure 51.

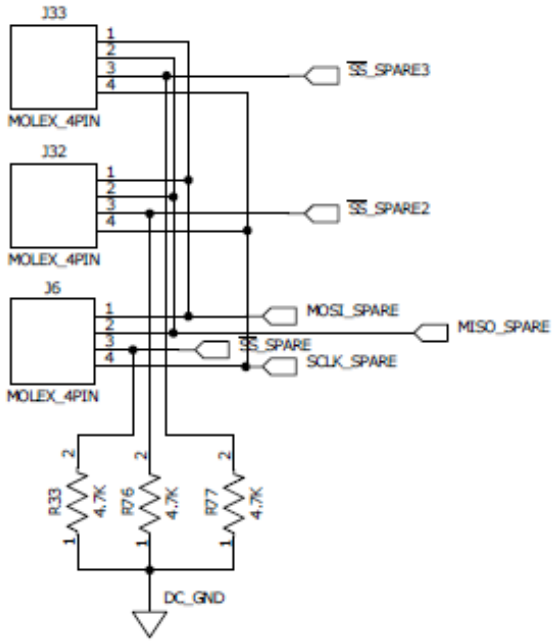

Figure 51 – External SPI Circuit

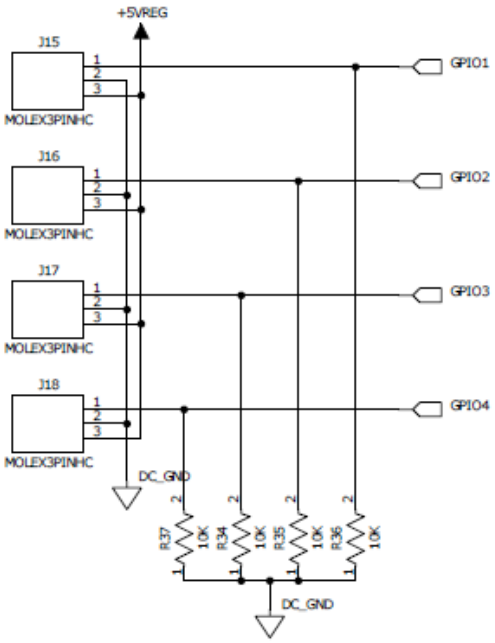

Figure 52 – General Purpose I/O Circuit

An external spare SPI bus increases the systems compatibility for minimal cost and effort. The user is also provided with four GPIOs (Figure 52) similar to the SPI connections, providing increased compatibility and flexibility for the user at minimal cost. The GPIO pins are all 5V tolerant and include pull down resistors for power conservation. These pins can be configured as either an input or an output allowing the user to integrate external switches, relays, lights, and etcetera.

Mechanical Relays

The majority of heating sources and their corresponding hardware (pumps, valves, fans, etc.) are controlled by mechanical relay switches. Figure 53 depicts a single relay switching circuit; however, three other identical circuits exist in the hardware.

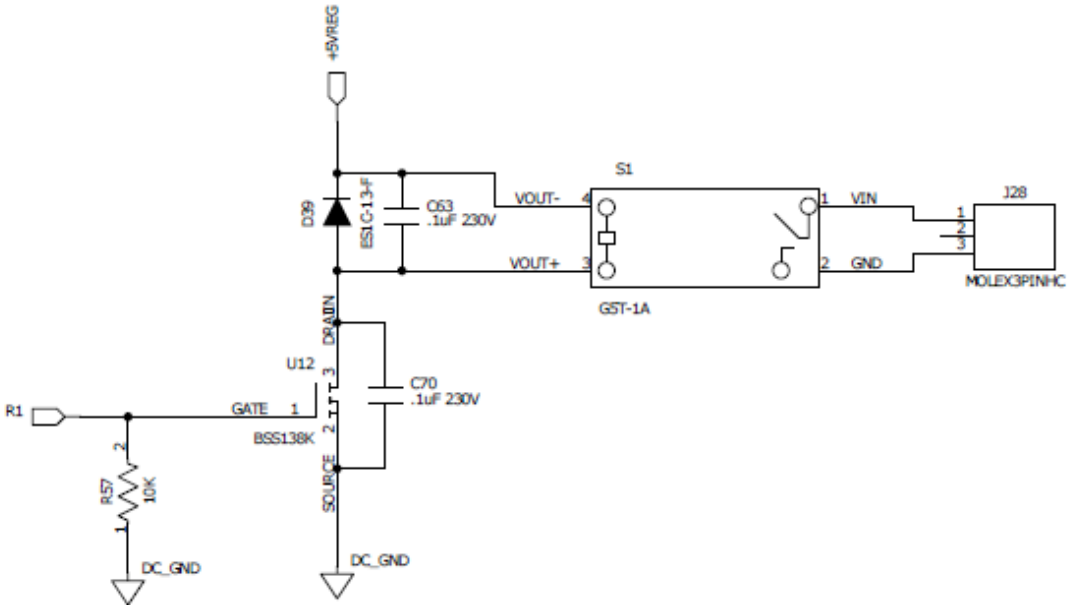

Figure 53 - Relay Switching Circuit

The switching uses an N-Channel MOSFET to drive the mechanical relay with 5V. The circuit includes two capacitors in parallel, one to the relay, and another across the MOSFET. These help eliminate dangerous voltage spikes caused from the initial opening of the relay. A freewheeling diode connected in parallel to the relay also provides back-emf protection to the hardware.

AC Motor Drive

The system required two AC motor speed control circuits, capable of driving up to a .25 HP motor at any voltage from 0-120VAC. A single ac motor speed control circuit is shown below in the Figure 55; however, another identical one is included in the hardware. Traditional motor speed control circuits use a TRIAC; these circuits require a set voltage. To accomplish the goal of a variable voltage speed controller, the team modified a published design by Freescale Semiconductor.

The design illustrated below in Figure 54 works by utilizing a full-wave rectifier as a bi-directional switch. A bi-directional switch allows both positive and negative current to pass (i.e. AC current). This bi-directional switch output is connected across the motor load with the control signals connected to an insulated-gate bipolar transistor (IGBT). These act as a MOSFET at the base and a BJT on for the collector and emitter. When the IGBT is off, the motor load is connected to the AC input and vice versa. By rapidly switching the IGBT, the AC input to the motor is controlled. A PWM signal generated by the PIC 32 controls the rapid switching of the IGBT (16 KHz). Because of the high voltage and current requirements, the IGBT control circuitry is separated from the PIC32 controller. To accomplish this, an opto-isolator is used. An opto-isolator is simply a switch consisting of an LED and a photo sensor; the switch turns on when a voltage is applied across the LED. This creates a physical separation of voltage levels.

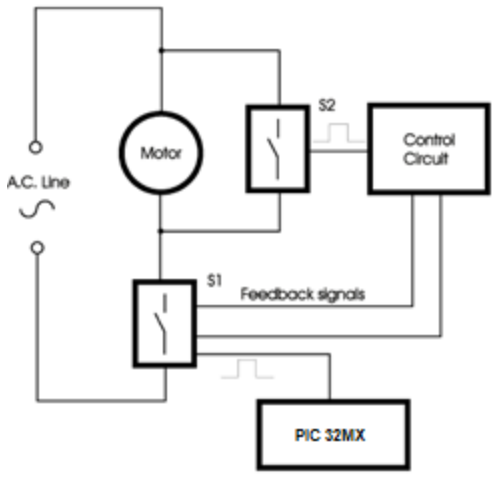

Figure 54 – AC Motor Block Diagram

However, because AC motors are inductive, free-wheeling circuitry is needed to compliment the switching circuitry. This free-wheeling circuitry must operate opposite to the switching circuitry, they can never both be on at the same time. This circuitry requires a second bi-directional switch in series with the motor load. Its control signals connect across another IGBT.

The two input channels of the opto-isolator are connected inversely allowing a single input to control both outputs. A time delay exists on each line to eliminate any switching collisions, created by RC filters from components C36, C37, R49, and R50. For increased circuit protection, three 60Ohm, 2MHz, ferrite beads connected in series dampen any spikes in voltage, along with another high voltage transient suppressor diode and a capacitor.

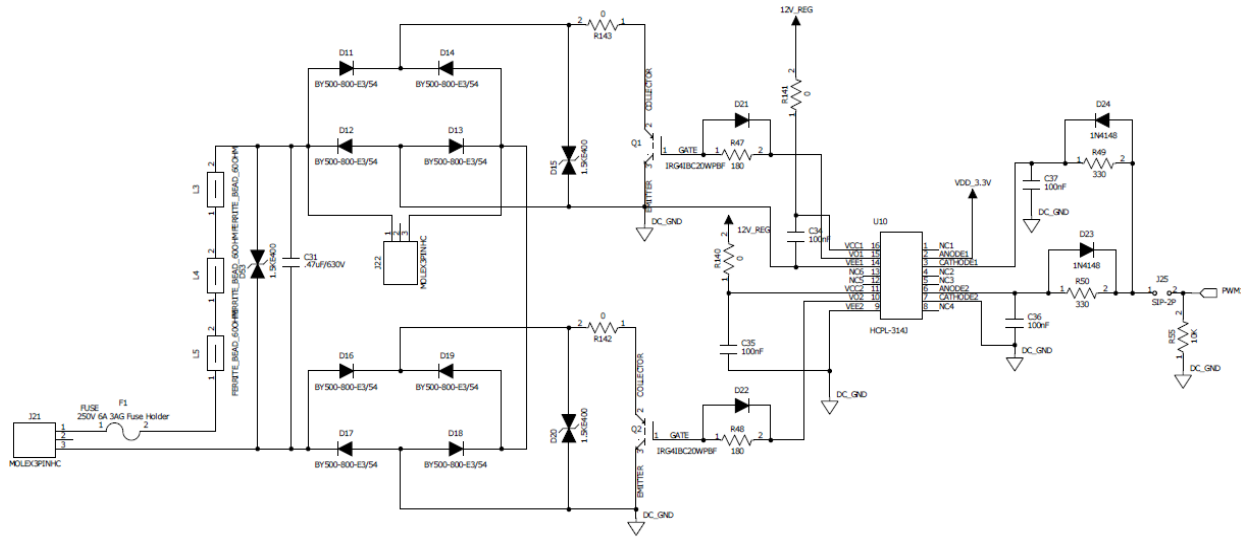

Figure 55 – AC Motor Drive Circuit

A simulation to demonstrate the functionality of the AC motor drive circuit was created using Matlab. The source code is located in Appendix X. The simulation works by creating matrices of the input AC voltage and the ideal target output voltage. The values within these matrices are then compared to determine when to turn on and off the IGBTs connecting the motor to the AC input. The Matlab code can be simply translated into efficient C code for operation on a PIC32 microcontroller.

Simulation outputs for 5 cycles of 70%, 50%, and 30% of a 120VAC input with a resolution of 30 outputs per cycle are shown in the figures below.

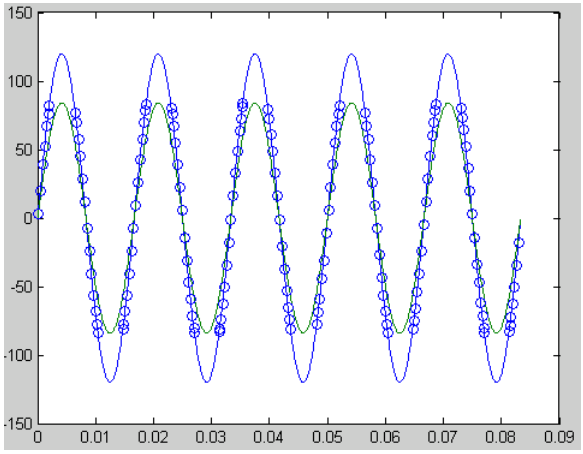

Figure 56 - 70% Output
[RMS_{OUT},AC_{OUT}]=GetMvals(70,30,120,60,16000,5)

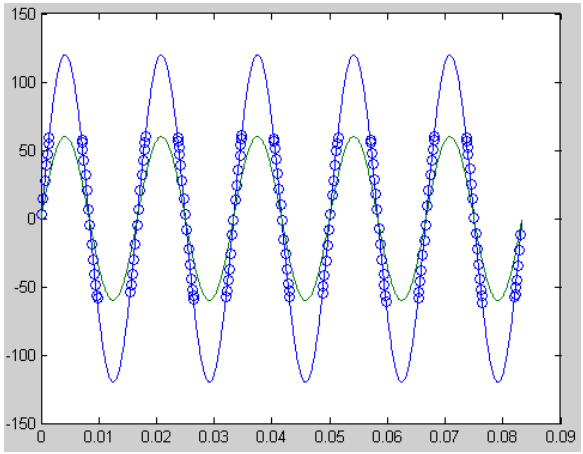

Figure 57 - 50% Output
[RMS_{OUT},AC_{OUT}]=GetMvals(70,30,120,60,16000,5)

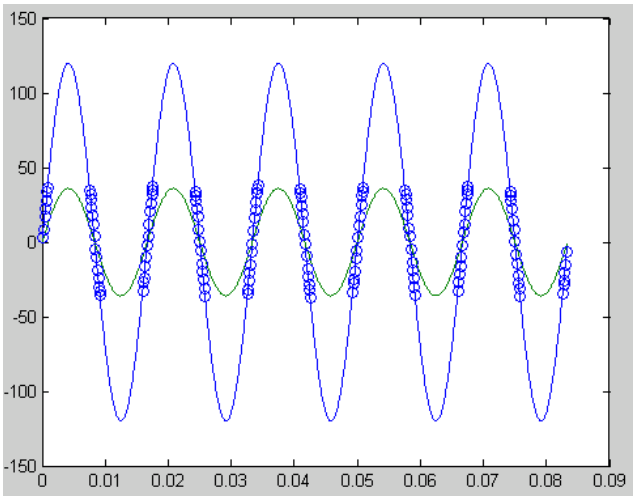

Figure 58 - 30% Output [RMS_{OUT},AC_{OUT}]=GetMvals(70,30,120,60,16000,5)

5.1.4. Power

The hardware design is required to be simple to install and universal. To allow for user flexibility multiple power options are available. These options are 120VAC, 16-24VAC, USB (No 12V line), and PoE (Power over Ethernet). These options cover the majority of situations for home installers. To control all necessary the aspects of the system, multiple regulated voltage lines are available across the board. These are 12VDC, 5VDC, and 3.3VDC. The power distribution across the system is capable of handling over 500mA. The design provides large current capabilities for the initial design, making prototyping and testing simpler. Lower power modules will be implemented during later revisions.

DC Power

The DC power distribution provides regulated 12VDC, 5VDC, and 3.3VDC to the system. The power distribution consists of three charge pump buck convertors cascaded. This circuit is shown below in Figure 59.

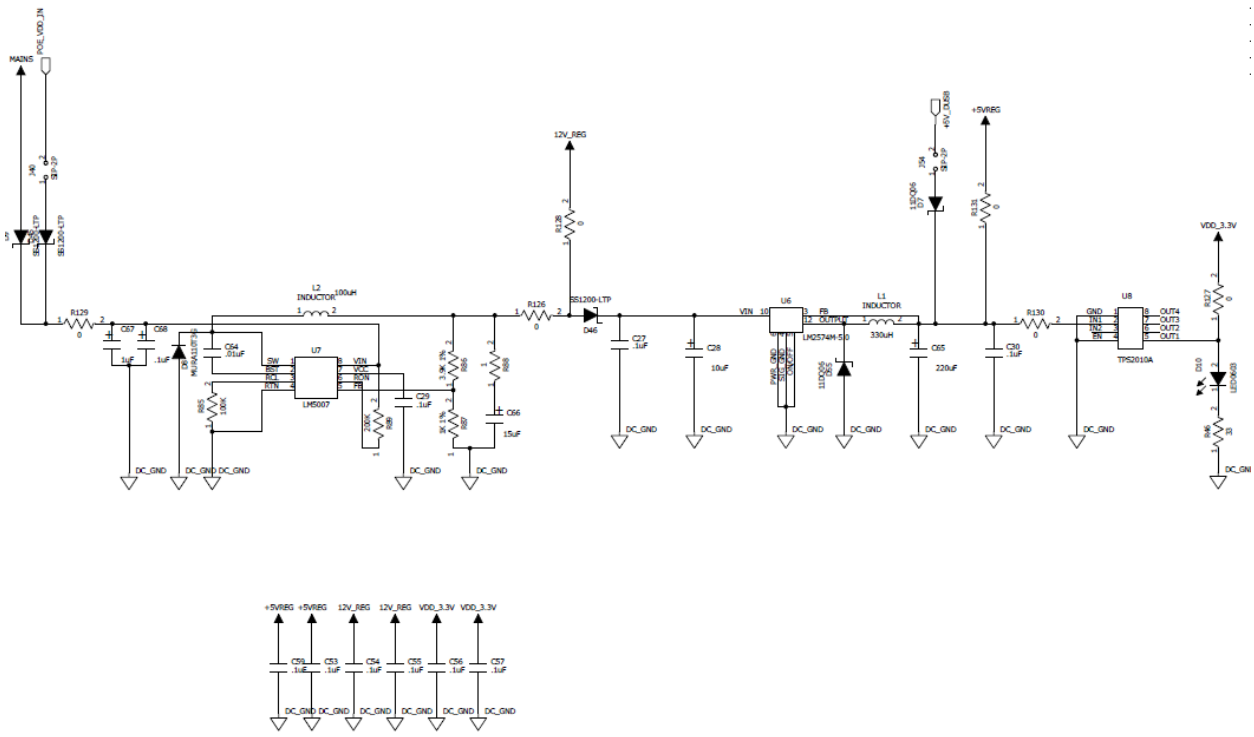

Figure 59 – Power Distribution Circuit

This method is cheapest, and wastes the least power, because the voltage difference at each module is small. Compared to if each line was individually regulated. However, if the 12VDC regulator fails, the rest of the power system fails. The 12VDC regulator has been specified for up to 80VDC

because the PoE circuit provides the highest voltage input to this circuit at 52VDC. Each regulator is coupled with an inductor. The exact value of the inductor selected depends upon the voltage and current expected through it, without the proper inductor the voltage output will not be correct or stable. The datasheets include exact specifications for inductor selection. Each power input and output is complimented with a Zener diode providing isolation between circuits. USB power is included providing a constant 5VDC line at 500mA. It will allow the user to power all lines except for the 12VDC line (used for AC motor speed control).

Power over Ethernet (PoE)

PoE is becoming more and more standard in modern houses. It allows a single port to provide data and power simultaneously. Our design encompasses PoE because it provides a simple, convenient, inexpensive, power option for users. This circuit is shown below in Figure 60.

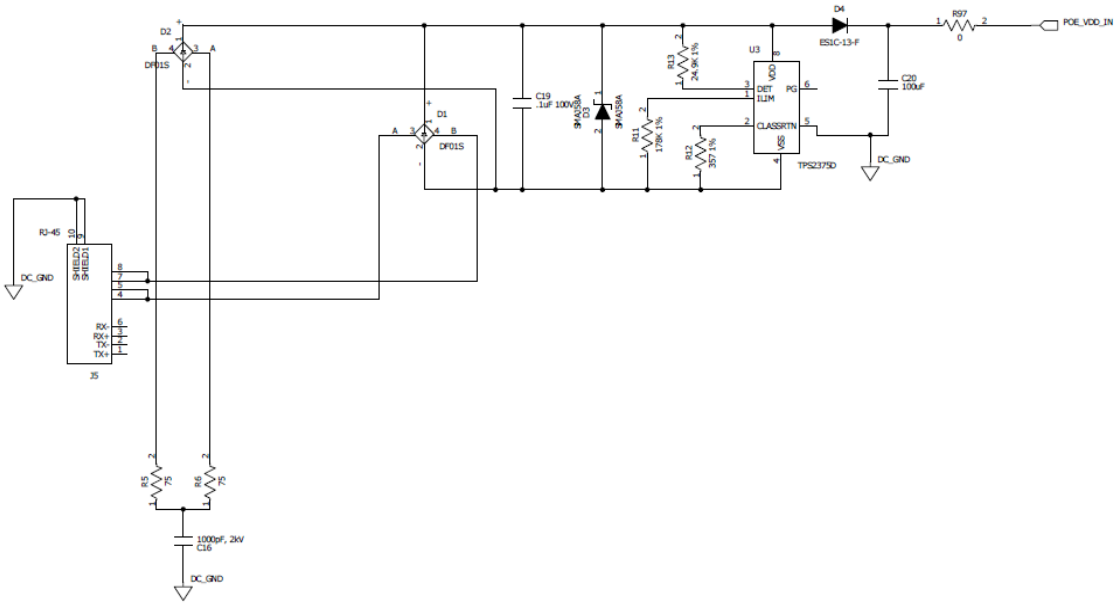

Figure 60 – Power over Ethernet (PoE) Circuit

The PoE protocol makes use of the four unused wire pairs in standard 100Mbps Ethernet. The voltages along the lines passes through full wave rectifiers used to create a common independent ground. PoE protocol requires a system of classes, each with a different resistance value corresponding to a voltage applied. To comply with the IEEE 802.3af standard, a class 3 device controller (TPS2375D) is used. This controller handles the necessary resistance changes providing a stable output. The passive components included with the controller are within the datasheet. The schematic will provide up to 52VDC to the hardware over PoE.

AC Power

The AC power has two distinct parts, 120VAC and 16-24VAC converter. These parts are shown in the circuit below in Figure 61. The two parts work almost identically; bridge rectifiers convert an AC signal into rough DC, followed by a buck boost converter creating the 12VDC line.

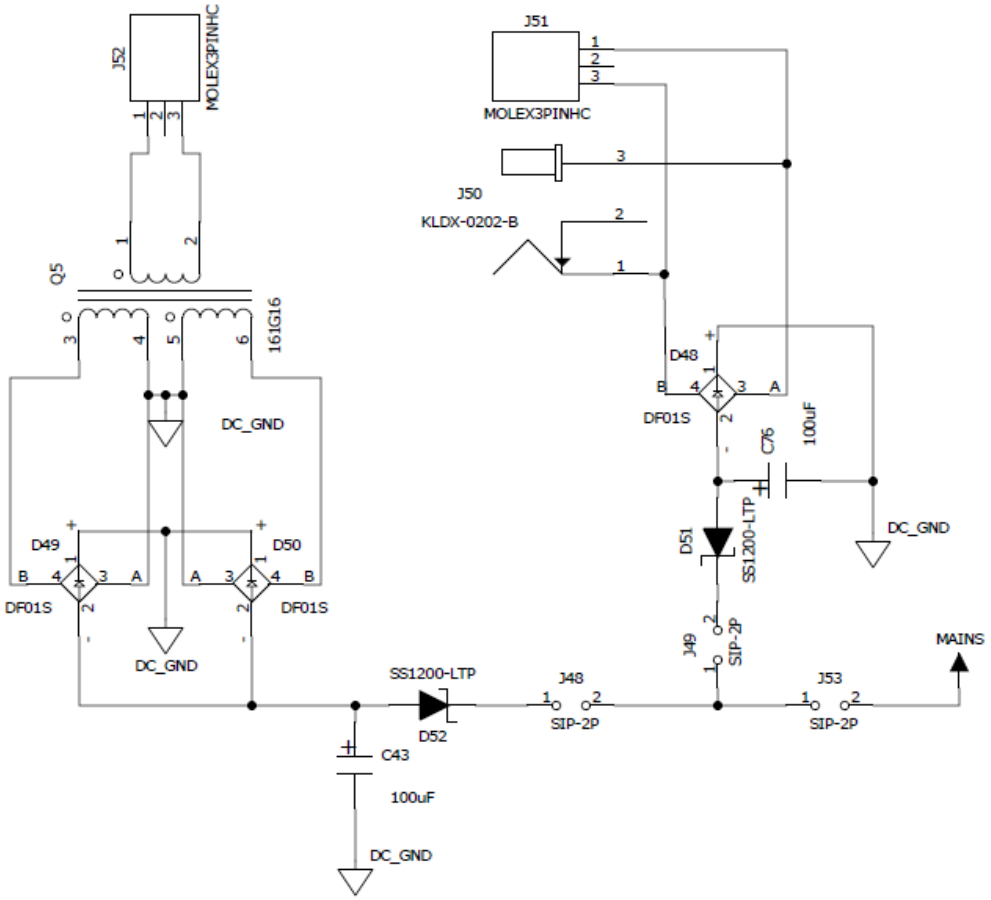

Figure 61 – AC Power Circuit

The difference between the 120VAC and 16-24VAC parts is a center tap transformer. The buck boost converter is only capable of regulating up to an 80V line; therefore, a 16V center tap transformer is used. The resulting outputs are rectified and combined to provide 32V rough DC. The 16-24VAC line has connector capabilities for either a center pin connector from a wall transformer or a direct Molex connector block. Zener Diodes are included on power input lines to keep sources independent without applying a substantial load on the line.

5.1.5. User Interface (Daughter Board)

A user interface is required for setting thermostat temperatures, but it also provides the opportunity to display metrics and for the user to change configurations and settings onsite. The core component of the user interface is the display. This display defines controller and power used; a 1.8" TFT display with an 8-bit controller was selected. The user interface encompasses its own microcontroller which communicates to the controller module via I2C. This method abstracts the dependency of the user interface module from the controller module.

Master Connector

Power and communication to the master controller is transferred via a 25 pin connector. This connector also forms a mount for the module in Figure 62 & Figure 63.

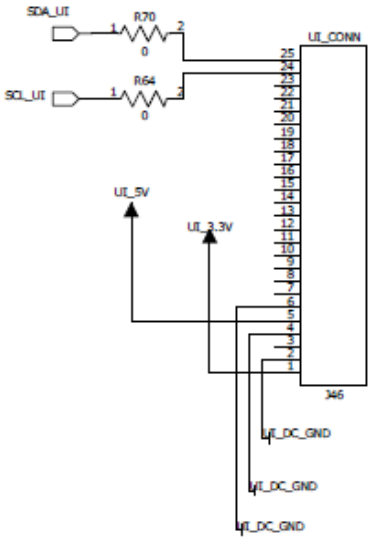

Figure 62 – Master Connector Circuit: UI Side

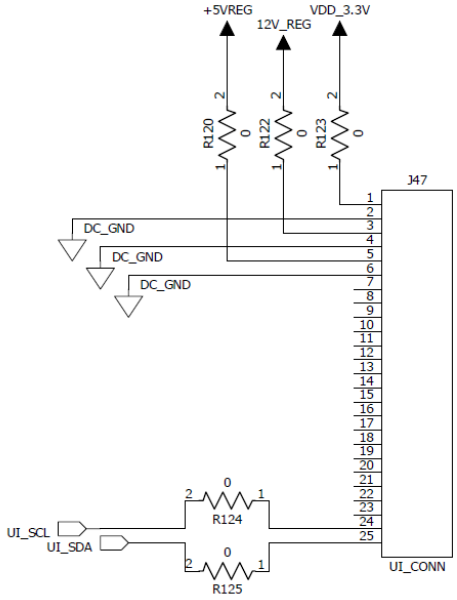

Figure 63 – User Interface Connector

The intent is to provide a connection interface that is useable by almost any system. Therefore if the user interface was re-designed, the team would not need to re-design the controller module also. To accomplish this, the controller uses the I2C communication protocol, which almost every microcontroller uses. I2C is a simple two wire bi directional data bus for communicating between embedded devices. With the I2C, the connector includes signals for 3.3VDC, 5VDC, 12VDC, and ground.

Microcontroller

A PIC24F08KA102 is the microcontroller used to control the user interface module. This circuit is shown below in Figure 64.

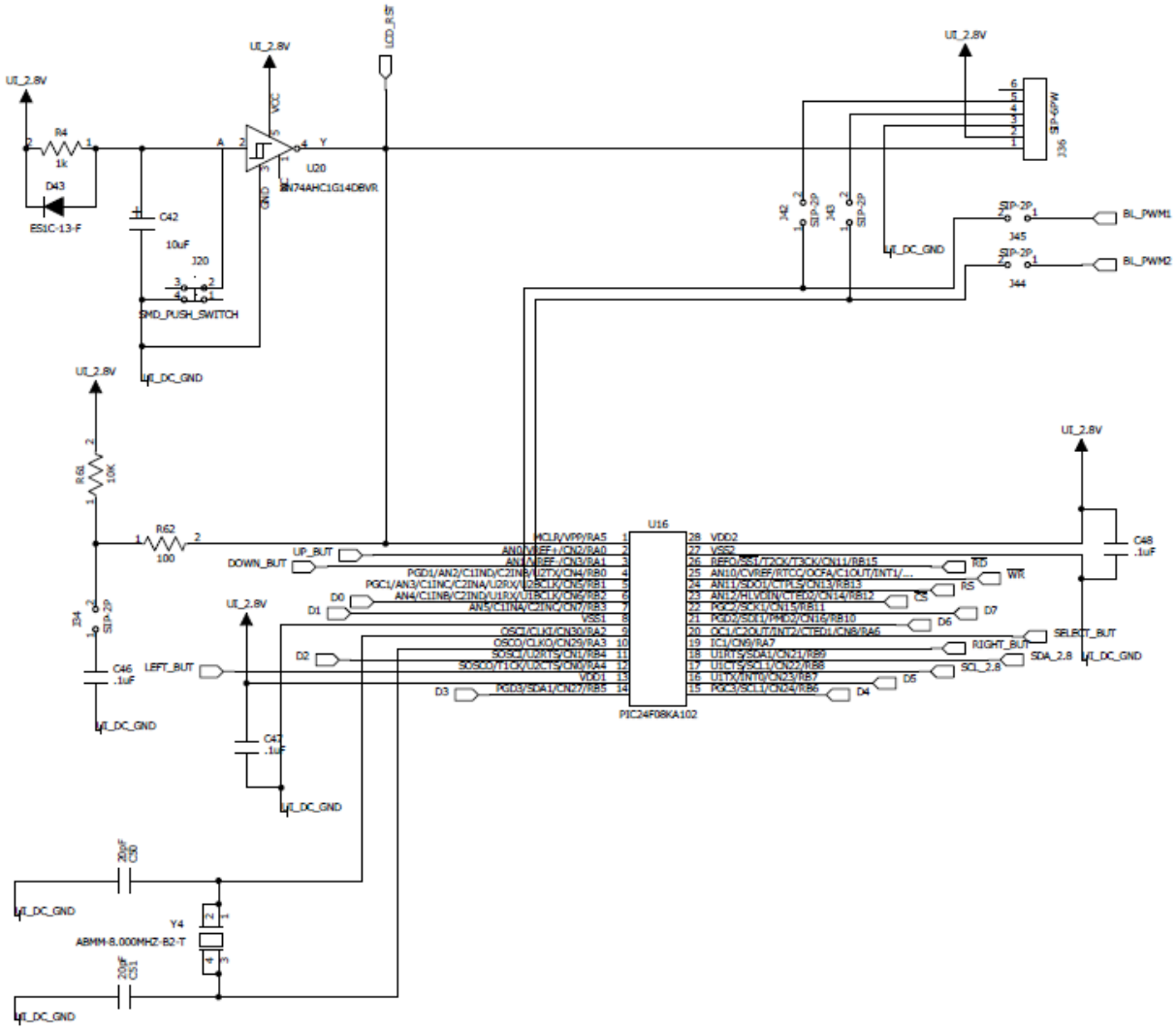

Figure 64 – Interface Controller Circuit

The schematic includes the microcontroller, the ICSD circuit for programming and debugging, the necessary oscillators, and the reset circuit. The design for these circuits is the same as previously discussed.

Liquid Crystal Display (LCD)

A 1.8” TFT LCD display is used in the user interface module. An 8-bit data bus controls the LCD screen. The schematic depicting this LCD circuit is shown below in Figure 65.

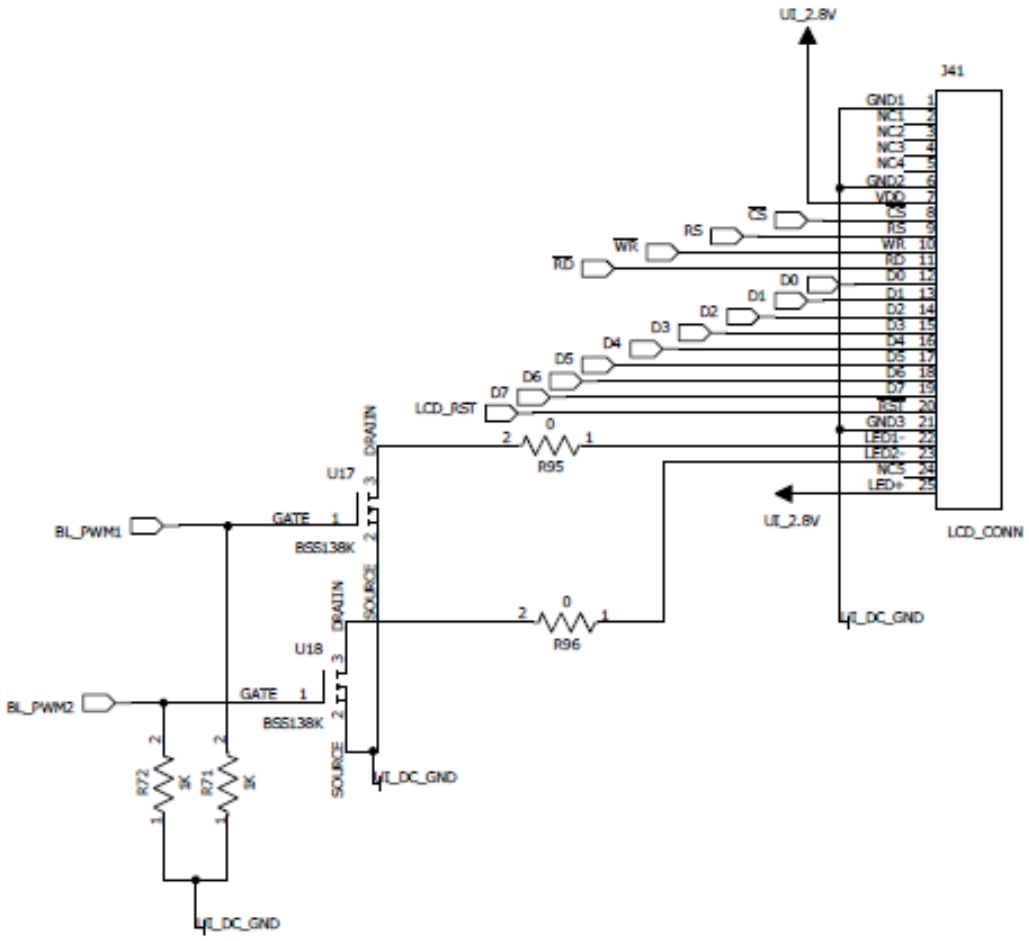

Figure 65 – Liquid Crystal Display (LCD) Circuit

Four control signals determine the direction and type of data. Two PWM signals control the brightness of the display. These PWM signals switch an N-channel MOSFET toggling the backlight LEDs, this method is used because only limited current can be placed on the output of a PIC32 pin.

Buttons

Five buttons are included in the user interface module, these are up, down, left, right, and select. They are simple push-button switch circuits (Figure 66) which when pressed provide a low signal to the PIC32 pin.

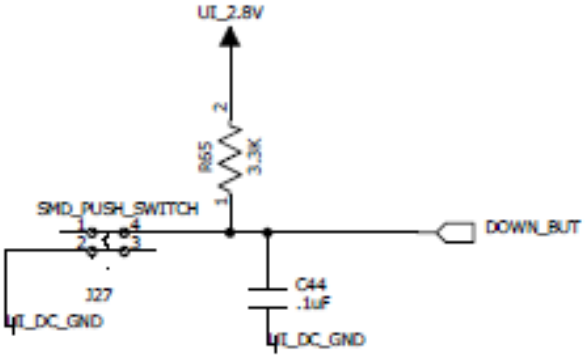

Figure 66 – Button Circuit

Power

The selected LCD screen requires 2.8VDC; this is within the operating voltage of the selected PIC24. Therefore, to drive the selected system a simple linear 2.8VDC regulator (Figure 67) is used.

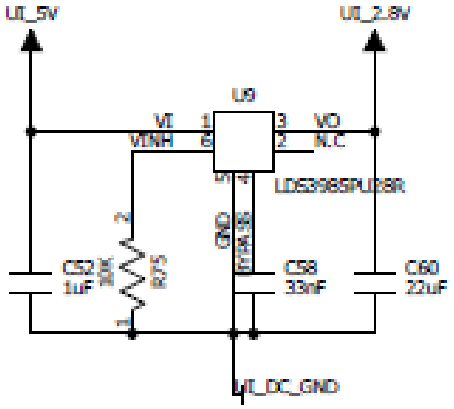

Figure 67 – Power Circuit for LCD

This power circuit was selected opposed to a charge pump similar to the other regulators in the system because of the low current requirements.

Opto-Isolation of I2C

Opto-Isolation of I2C is popular within complex embedded systems. Often, different embedded systems require communication between them over minimal pins. The communication protocols for these communication methods (I2C, SPI, etc.) require a fixed voltage. Yet, the embedded systems may not all run on the save voltage. This is when I2C opto-isolation is used. The concept is the same as opto-isolating any other line, a led turns on/off telling an IR receiver to turn on/off separating the voltage levels.

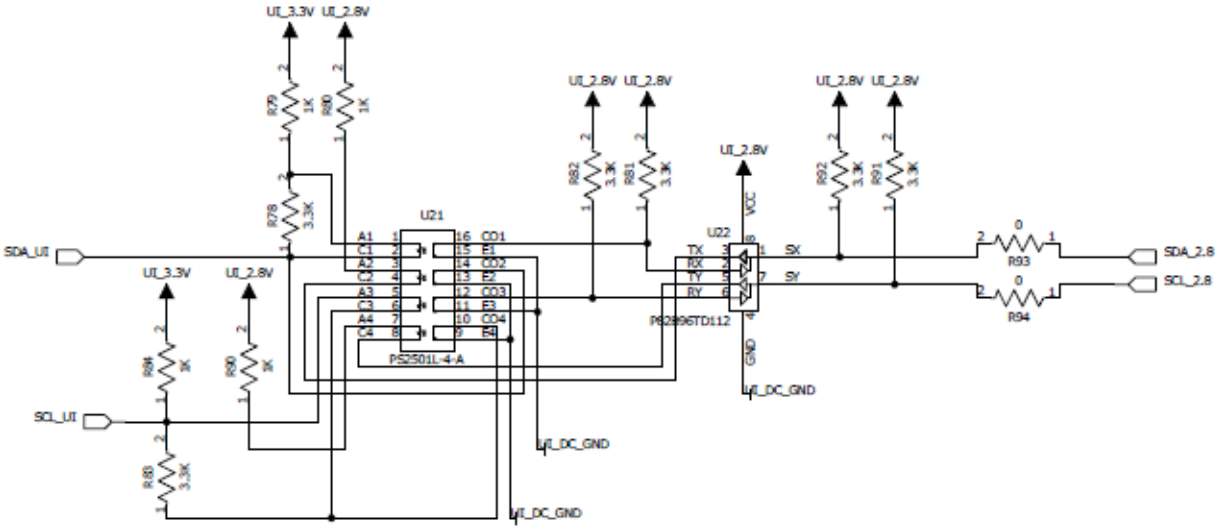

Figure 68 – I2C Circuit

I2C utilizes two bi-directional lines. For opto-isolation, each line must be split in half for each direction. I2C is a relatively slow communication protocol, but compared to typical signal lines, 5 KHz is fast. This causes a problem when re-combining the split signals; data becomes corrupted because not every opto-isolator switches identically. To correct this, special bi-directional buffers exist for use in recombining opto-isolated I2C signals. The schematic employing this technique is shown above in Figure 68.

5.2. Implementation

The implementation section covers the details from taking the initial schematic design and making it a reality. It covers the PCB design, testing, and functionality analysis of the initial hardware design for the project.

5.2.1. Printed Circuit Board (PCB) Design

Upon the completion and approval of the schematic, a team member designed a PCB in PADS Layout. The design is a single board to be cut as necessary to function as a thermostat or a heat source control module. The design consists of the three distinct modules combined on one large board. This was largely due to the cost of printing multiple board batches. Due to the complexity, size, and current requirements of the system the design consists of four main layers. The top and bottom layers include components and routing, the second layer consists of a split/mixed power planes, and the third layer is a ground plane. The other board layers are a top solder-mask, a bottom solder-mask, a top silkscreen, and a bottom silkscreen (appendix X).

The first step to designing a board is to create your component footprints. The component footprint is the physical space, including the pads soldered to on the board. There are many footprint libraries available, however, when a designer is first starting, it is recommended to stick to common footprints. Unique components will typically require the user to define the footprint based off of a drawing provided in the datasheet. If the user does not have access to a large component library, they may be required to define almost all of the footprints for their design, as was the case for the team. When designing a footprint, keep in mind of who is going to assemble the boards, and design to their capabilities, this may mean elongating or widening pads to simplify the soldering process. Following the completion of creating the component footprints, the components are placed in the design.

The component placement and layout were completed using clusters. A user builds clusters by placing all of the components for a specific part of the circuit together. For example, the all of the DC power distribution components are together in one section of the board. This makes routing, building, and testing the board all significantly simpler. A good component layout is integral to a good PCB design. With larger designs, components will be placed on both sides of the board to save space. The designer must keep in my physical space requirements, especially if boards will be interconnecting.

The next step is the routing of traces. Every designer has their own method to doing this; however, there are common trends. For a board with two routing layers, each layer will be given a general

direction, either vertical or horizontal routing. As board space becomes sparse, this will help keep traces from crossing one another. Next internal planes are defined and assigned net values. After this, nets are assigned rules, defining the width, spacing, and via size. Trace width calculation consists of knowing the current, the copper thickness, the available temperature rise, and the ambient temperature. However, typical boards use a copper thickness of 1 oz/ft^2, room temperature is 25 degrees Celsius, and a typical temperature rise is 10 degrees Celsius. Therefore, when designing boards a common standard to use is a 12 mil trace width on external layers and a 30 mil trace width on internal layers per Amp. On the team's board a 35 mil trace is used anywhere that could experience 500-800mA. Once the designer finishes defining the net rules, it is time to start to route the traces. The designer can start anywhere, but it is recommended to start with the "special needs" traces and then move down difficulty as you go. This will help again when board space becomes limited. Special needs traces are sensitive traces with very specific routing rules. This typically includes, bypass capacitors, oscillators, differential signal pairs (D+, D-), and others. These should be routed first, because they become exponentially more difficult as board space is used. Other difficult components are small pitch processors or microcontrollers, typically the designer uses a fan out to route the pins. After every net is connected, it is suggested to run verification tests. These tests catch missed connections, possible acid traps, and more.

When the team's design was completed and approved 15 boards were ordered at $31.48 from advanced circuits. Enough components were ordered to build and test the following number of modules:

- 3 full UI modules
- 3 full Control modules
- 5 processor modules
    - 3 POE circuits
    - 3 AC/DC circuits

These numbers were selected to provide the team with enough modules to test and debug, while still having functional modules for controlling the demonstration platform.

5.2.2. Construction & Testing

Construction and testing of the hardware consists of a couple phases. The hardware was constructed on site in segments, which were tested sequentially. This technique isolates circuits from one another simplifying identification of problems. The hardware was assembled on site using a microscope and a combination of precision solder paste an iron techniques. The sequence for assembling the hardware is below:

- DC Power Regulation (3.3V->5V->12V)
- AC Power Input
- PIC32 Processor
- Reset Circuit
- Programming Circuit
- Relay Circuits
- SPI & GPIOs
- Wi-Fi Communication
- Ethernet Communication
- Power over Ethernet
- AC Motor Control

Each hardware circuit section undergoes testing after construction. The DC power regulation circuit is constructed starting with the 3.3VDC rail and working backwards. Each rail is constructed and the output checked using a digital multi meter, the current draw is also noted, as it should not be suspiciously high (over ~ 50mA). Upon successfully passing this test the 5VDC and then the 12VDC rails are constructed and tested in the same manner. The 3.3VDC output also triggers an LED when on, confirmation of this light being on is necessary for the 5 and 12 VDC lines. The AC power input circuit was constructed next because if there is a fault in the AC power circuit, the only damage would be to the DC regulators. If necessary, the board could be powered with externally regulated 12VDC removing the need for AC power. The AC power circuit is tested by using an oscilloscope to view the transformer output and the DC voltage input line for proper signals. If all checked out, a jumper is placed connecting the AC power input to the DC regulators.

Next step is to mount the PIC32 processor and surrounding bypass capacitors and oscillators. This must be completed carefully under the microscope using a pressurized solder paste dispenser. A

connectivity check is completed to make sure none of the power and ground pins have been shorted together and that the chip is not faulty. Next, the reset circuit is built and an oscilloscope is connected to the output. The button, when pressed should change the output signal from 3.3V-0V consistently. Next the programming circuit needs to be connected and tested. This only includes a 6-pin connector and two resistors. However, once mounted successfully the user can program the PIC32. The testing for this stage requires connecting the PIC32 to power and the PICKit3 programmer. The user then uses MPLab to detect the PIC32 and upload a small program to its memory. The final sections are all tested similarly. They are constructed individually and a test program is run on the PIC32 to verify functionality. For example, the relays are tested with a program that sequentially toggles each one on and then off in order. This technique of building and testing each part of the hardware individually allows for simplistic debugging and isolation of problems.

5.2.3. Functionality

As new hardware is tested, problems inevitably occur. This section details the problems determined with the hardware design and the corresponding solutions.

AC Input Transformer

The transformer originally selected supplied 14 volts to the 12 VDC regulators. However, this was not a high enough input to drive a constant 12VDC output. To fix this problem a transformer with the same footprint and a higher output voltage was purchased. This transformer (161G40) supplies 20VDC at 500mA.

AC Input Rectification

Two full-wave rectifiers are connected to the output from the transformer. These full-wave rectifiers do not work the way they are connected. Instead the outputs cancel one another out. These full-wave rectifiers are replaced with regular diodes, creating a half-wave rectifier. This half-wave is used to power the 12VDC rail.

AVDD & VDD Connected

The AVDD & VDD lines need to be isolated from one another even if both values are the same (i.e. 3.3VDC). Microchip recommends using a simple 10Ohm resistor for isolation. This is missing in the hardware, it can be hacked in, but future designs must incorporate this resistor.

MCLR Programming Pin Has Inline Resistance

The MCLR pin on the PIC32 requires an inline resistor for the reset circuit. However, the PICkit3 programming pin for controlling the MCLR cannot have any inline resistance. This issue can cause propagation delays when resetting the controller during programming. The temporary solution for this is to hack in a parallel trace to the MCLR programming pin. However, once again, this must be incorporated into future designs.

Reset Circuit Impedance Too Large

The MCLR pin also has a tight tolerance on the allowable impedance. The reset circuit utilizing hardware de-bouncing has too high of an impedance for the pin. The suggested design is simply a button directly to ground.

Bypass Capacitors Over 6mm from Pins

Microchip recommends that all bypass capacitors be within 6mm from their respective pins. The current hardware design does not have all bypass capacitors safely within this specification. These can also be hacked into the right range but fixed in future designs.

Reset Resistors Over 6mm from Pins

The resistors in the MCLR reset circuit must also follow the 6mm spacing rule. They are not in the current design and should be moved. It will increase programming and resetting stability.

Incorrect 3.3 VDC Regulator Selected

The 3.3VDC regulator selected initially was a power mux. It was mistakenly selected under the pretense that a liner regulator was also included in the package. To remedy this issue a 3.3VDC linear regulator was selected with the same package dimensions. This package required only two jumper wires to be placed on the board for a constant 3.3VDC output.

Oscillator Pinout Incorrect

The 8 MHz oscillator selected has pins 2 & 4 connected to the outside cover as a ground. In the current hardware design these pads are tied to the signal pins. To fix this the oscillator can be mounted diagonally and pins 2&4 not connected, future designs should take note of the proper pad connections.

LCD Screen Connector Inverted

The LCD screen connector for the user interface was placed in reverse. The screen can be connected and tested with the hardware, however future designs need to reverse the connector for proper mounting.

10uF 1 Ohm ESR Tantalum Capacitor on VCore

For a stable power supply of 1.8VDC the PIC32 controller requires an additional 10 uF 1 Ohm ESR tantalum capacitor on the Vcore pin tied to ground. This capacitor is placed in parallel to the current .1uF bypass capacitor connected to ground.

These are the problems determined through the testing and implementation stage of the hardware design. However, due to the culmination of these problems the PIC32 could not be programmed reliably. Due to this, the functionality of the following components have not been fully tested. Ethernet communication, Wi-Fi communication, PoE, UI, and AC motor speed control. However, Wi-Fi and Ethernet communication has been implemented on Microchip starter kits provided. The details of this are in the embedded software section. For a proof of concept design, the Microchip starter kits have been connected to the relay switching boards designed. These modules have been used to measure temperature in zones and sources and switch relays accordingly. The main functionality of the system has been successfully implemented with future work left to be completed in the additional features of the device (PoE, AC motor speed control, etc.).

Chapter 6: Software Implementation

The implementation of the software design for this project includes two particular levels of development that is divided by the two proprietary components: the Embedded System and the Master Control Server. This chapter will discuss the procedure taken to implement the required software for this project for each processor-based device.

6.1. Embedded System Software

Embedded software is linkage between the hardware design, the physical world, and the master controller (main server). It directly controls the microcontroller (PIC32) and provides the base layer for communication and processing within the system. This project's embedded software is primarily responsible for maintaining network communications and basic peripheral interaction. The majority of the software consists of a TCP/IP stack, which provides the network communication protocol for interaction with the master controller. The TCP/IP stack is free from Microchip, which also includes the necessary software drivers for interfacing with the Wi-Fi chip via SPI and a demo HTTP server. The TCP/IP stack includes the majority of the software needed for a prototype. However, for the long term a real time operating system (RTOS) needs to be implemented for increased peripheral interaction. The current Microchip software will not be suitable because the RTOS will require a specific compatible TCP/IP stack. The next two sections are the peripheral interface and HTTP server software. If the TCP/IP stack is changed, the concepts covered in these sections will remain mostly unaffected.

6.1.1. Peripheral Interface

The peripheral interface covers the software required for interaction with the specified peripherals. These peripherals are the relays, GPIO, Wi-Fi chip, temperature sensors, and a wireless router. Software is changed and added within the provided demo to accommodate interaction with these peripherals.

Configurations

The hardware configurations for the mapping of specific microcontroller ports are found in a single file labeled HWP PIC32…h. This file defines I/O pin mappings which are be used for both relays and GPIO. The Wi-Fi SPI pin configurations are also included in the file. The demo software allocates eight GPIO pins under the name LED0-LED7. For the prototype, more than eight I/O pins were not needed, so to save time the relay and GPIO pins were kept under the name LED. The pin mappings for the hardware are shown below. The #define LED_PUT(a) is the default output for each port when the hardware is

initialized. Currently each port is set to be off upon startup. The #define LED_GET() gets the initial value of the defined ports upon startup.

```
// LEDs
#define LED0_TRIS          (TRISBbits.TRISB6)  // Port B6 Unused
#define LED0_IO            (LATBbits.LATB6)
#define LED1_TRIS          (TRISDbits.TRISD1)  // Port D1 Relay 1
#define LED1_IO            (LATDbits.LATD1)
#define LED2_TRIS          (TRISDbits.TRISD2)  // Port D2 Relay 2
#define LED2_IO            (LATDbits.LATD2)
#define LED3_TRIS          (TRISDbits.TRISD3)  // Port D3 Relay 3
#define LED3_IO            (LATDbits.LATD3)
#define LED4_TRIS          (TRISDbits.TRISD4)  // Port D4 Relay 4
#define LED4_IO            (LATDbits.LATD4)
#define LED5_TRIS          (TRISDbits.TRISD5)  // Port D5 GPIO
#define LED5_IO            (LATDbits.LATD5)
#define LED6_TRIS          (TRISDbits.TRISD12) // Port D12 GPIO
#define LED6_IO            (LATDbits.LATD12)
#define LED7_TRIS          (TRISDbits.TRISD8)  // Port D8 GPIO
#define LED7_IO            (LATDbits.LATD8)
#define LED_GET()          ((unsigned char)LATD & 0x113E)
#define LED_PUT(a)         do{LATD = (LATD & 0xEE00) | ((a)&0x11FF);}while(0)
```

The following code is the hardware definitions provided by Microchip for using the Wi-Fi chip with SPI1. The pin assignments include ~CS, SDI, SDO, Reset, Hibernate, and Interrupt

```
#define MRF24WB0M_IN_SPI1
//#define MRF24WB0M_IN_SPI2
#if defined(MRF24WB0M_IN_SPI1)
    // MRF24WB0M in SPI1 slot
    #define WF_CS_TRIS         (TRISBbits.TRISB2)
    #define WF_CS_IO       (LATBbits.LATB2)
    #define WF_SDI_TRIS        (TRISFbits.TRISF7)
    #define WF_SCK_TRIS        (TRISFbits.TRISF6)
    #define WF_SDO_TRIS        (TRISFbits.TRISF8)
    #define WF_RESET_TRIS      (TRISFbits.TRISF0)
    #define WF_RESET_IO        (LATFbits.LATF0)
    #define WF_INT_TRIS        (TRISEbits.TRISE8)  // INT1
    #define WF_INT_IO          (PORTEbits.RE8)
    #define WF_HIBERNATE_TRIS  (TRISFbits.TRISF1)
    #define WF_HIBERNATE_IO    (PORTFbits.RF1)
    #define WF_INT_EDGE        (INTCONbits.INT1EP)
    #define WF_INT_IE          (IEC0bits.INT1IE)
    #define WF_INT_IF          (IFS0bits.INT1IF)
    #define WF_INT_IE_CLEAR    IEC0CLR
    #define WF_INT_IF_CLEAR    IFS0CLR
    #define WF_INT_IE_SET      IEC0SET
    #define WF_INT_IF_SET      IFS0SET
    #define WF_INT_BIT         0x00000080
    #define WF_INT_IPCSET      IPC1SET
    #define WF_INT_IPCCLR      IPC1CLR
    #define WF_INT_IPC_MASK    0xFF000000
```

```
#define WF_INT_IPC_VALUE   0x0C000000

#define WF_SSPBUF          (SPI1BUF)
#define WF_SPISTAT         (SPI1STAT)
#define WF_SPISTATbits     (SPI1STATbits)
#define WF_SPICON1         (SPI1CON)
#define WF_SPICON1bits     (SPI1CONbits)
#define WF_SPI_IE_CLEAR    IEC0CLR
#define WF_SPI_IF_CLEAR    IFS0CLR
#define WF_SPI_INT_BITS    0x03800000
#define WF_SPI_BRG         (SPI1BRG)
#define WF_MAX_SPI_FREQ    (10000000ul) // Hz
```

Analog-to-Digital Converter (ADC)

The demo application did not include software to read different ADC channels. However, in order to read temperature data from the thermocouple amplifiers, this feature required an additional driver. The ADC driver consists of two small files, CustomADC.c containing the necessary hardware configurations and CUSTOMADC.h, which only includes the function declaration. CustomADC.c contains the function RunADC(int channel), which accepts an ADC channel 1-8 and returns the value on that channel. Large 2 millisecond delays are used in between each sample to make sure the data buffer ADC1BUF0 clears properly since speed is not a major concern in the devices temperature sampling. CustomADC.c only selects the appropriate port and read the data, since the initialization happens during the startup initialization. The registers set for the initialization of the ADC are below followed by the function RunADC(int channel).

```
// ADC
    AD1CON1 = 0x84E4;       // Turn on, auto sample start, auto-convert, 12 bit mode
    AD1CON2 = 0x0404;       // AVdd, AVss, int every 2 conversions, MUXA only, scan
    AD1CON3 = 0x1003;       // 16 Tad auto-sample, Tad = 3*Tcy
    AD1CSSL = 0x0000;

//custom ADC app to run for a single channel AN1-8
//ADC initialized prior
#include "TCPIP Stack/TCPIP.h"
int RunADC(int channel)
{
    int val = 0;
    AD1PCFG = 0;
    switch(channel)
    {
    case 0:
    break;
    case 1:
        AD1PCFG = 1<<1; //set to analog port
        AD1CSSL = 1<<1; //channel scan
        DelayMs(2);
```

```
        return ADC1BUF0;
    case 2:
        AD1PCFG = 1<<2; //set to analog port
        AD1CSSL = 1<<2; //channel scan
        DelayMs(2);
        return ADC1BUF0;
    case 3:
        AD1PCFG = 1<<3; //set to analog port
        AD1CSSL = 1<<3; //channel scan
        DelayMs(2);
        return ADC1BUF0;
    case 4:
        AD1PCFG = 1<<4; //set to analog port
        AD1CSSL = 1<<4; //channel scan
        DelayMs(2);
        return ADC1BUF0;
    case 5:
        AD1PCFG = 1<<5; //set to analog port
        AD1CSSL = 1<<5; //channel scan
        DelayMs(2);
        return ADC1BUF0;
    case 6:
        AD1PCFG = 1<<6; //set to analog port
        AD1CSSL = 1<<6; //channel scan
        DelayMs(2);
        return ADC1BUF0;
    case 7:
        AD1PCFG = 1<<7; //set to analog port
        AD1CSSL = 1<<7; //channel scan
        DelayMs(2);
        return ADC1BUF0;
    case 8:
        AD1PCFG = 1<<8; //set to analog port
        AD1CSSL = 1<<8; //channel scan
        DelayMs(2);
        return ADC1BUF0;
    default:
        return 0;
    }
}
```

6.1.2. HTTP Server

The HTTP server is the back end for which the master controller (server) communicates with each device. The server includes its own webpage, which is convenient for debugging. The webpage is stored in the flash memory of the PIC32 at compile time. The files are typically written in JavaScript or XML and a program by Microchip (MPFS2) converts them into C files. The HTTP server also includes a few functions for communication of data to and from the microcontroller. These functions are covered in detail in the next few sections.

HTTP Get

HTTP Get is a bi-directional variable update function. When a HTTP Get is called on a specific variable, the value of the variable within the server is updated to match the microcontroller. For example if a button is pressed changing an I/O pin from a one to a zero, the corresponding variable for communicating this data via the HTTP server will not update or change unless a HTTP Get is called. HTTP Get functions are a key method for getting and setting data in the system. Because they are bi-directional, the master controller can change a variable within the devices http server and then when a Get is called, the microcontroller updates its value to match accordingly. The function in the Microchip TCP/IP stack is called HTTPExecuteGet(void) is shown below which updates all of the LED ports. Additional I/O ports can be added following the same structure.

```
Function:
    HTTP_IO_RESULT HTTPExecuteGet(void)
  Internal:
    See documentation in the TCP/IP Stack API or HTTP2.h for details.
  ********************************************************************/
HTTP_IO_RESULT HTTPExecuteGet(void)
{
       else if(!memcmppgm2ram(filename, "leds.cgi", 8))
    {
       // Determine which LED to toggle
       ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"led");

       // Toggle the specified LED
       switch(*ptr) {
          case '1':
             LED1_IO ^= 1;
             break;
          case '2':
             LED2_IO ^= 1;
             break;
          case '3':
             LED3_IO ^= 1;
             break;
          case '4':
             LED4_IO ^= 1;
             break;
          case '5':
             LED5_IO ^= 1;
```

```
            break;
         case '6':
            LED6_IO ^= 1;
            break;
         case '7':
            LED7_IO ^= 1;
            break;
      }
   }

   return HTTP_IO_DONE;
}
```

HTTP Print

HTTP Print is simply a variable status print function. The function prints the value of the variable to the HTTP server. This is useful because once the value has been printed to the server, the master controller or other devices can access the data. Any data that needs to be published to the HTTP server needs an associated Print function. The three functions for printing the ADC values to the HTTP server are shown below. The function utilizes a sub function called TCPPutString which arguments are the type of server to print to and a string to print.

```
Function:
   void HTTPPrint_varname(void)

  Internal:
   See documentation in the TCP/IP Stack API or HTTP2.h for details.
  ********************************************************************/

void HTTPPrint_ADC1(void)
{
   TCPPutString(sktHTTP, AN0String);
}

void HTTPPrint_ADC2(void)
{
   TCPPutString(sktHTTP, AN1String);
}

void HTTPPrint_ADC3(void)
{
   TCPPutString(sktHTTP, AN2String);
}
```

HTTP Form Processing

HTTP Forms are useful because they allow the simultaneous update of multiple variables. They are similar to an HTTP Get except they are unidirectional and set only by an external user (master controller). In this TCP/IP stack, forms are included in the HTTPExecuteGet(void) function. Below is the other half of the function that was left out in HTTP Get. The function below will allow simultaneous submission of up to six LED port values. However, this can also be increased to any number or port name by following the syntax.

```
Function:
    HTTP_IO_RESULT HTTPExecuteGet(void)
  Internal:
    See documentation in the TCP/IP Stack API or HTTP2.h for details.
  ********************************************************************/
HTTP_IO_RESULT HTTPExecuteGet(void)
{
    BYTE *ptr;
    BYTE filename[20];
    // Load the file name
    // Make sure BYTE filename[] above is large enough for your longest name
    MPFSGetFilename(curHTTP.file, filename, 20);

    // If its the forms.htm page
    if(!memcmppgm2ram(filename, "forms.htm", 9))
    {
        // Seek out each of the four LED strings, and if it exists set the LED states
        ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"led6");
        if(ptr)
            LED6_IO = (*ptr == '1');
        ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"led5");
        if(ptr)
            LED5_IO = (*ptr == '1');
        ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"led4");
        if(ptr)
            LED4_IO = (*ptr == '1');
        ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"led3");
        if(ptr)
            LED3_IO = (*ptr == '1');
        ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"led2");
        if(ptr)
            LED2_IO = (*ptr == '1');
        ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"led1");
        if(ptr)
            LED1_IO = (*ptr == '1');
    }
    // If it's the LED updater file
    else if(!memcmppgm2ram(filename, "cookies.htm", 11))
    {
        // This is very simple.  The names and values we want are already in
        // the data array.  We just set the hasArgs value to indicate how many
        // name/value pairs we want stored as cookies.
        // To add the second cookie, just increment this value.
        // remember to also add a dynamic variable callback to control the printout.
        curHTTP.hasArgs = 0x01;
    }
    return HTTP_IO_DONE;
}
```

6.2. Master Control Server

The Master Control Module (MCM) as described previously serves as the high-level processor-based computer for the climate control system. To implement the MCM, section 4.3.3 describes the necessity for modular software which is accomplished with a three part hierarchy called a model control interface (MCI). The functions of each layer of the MCI will be described in detail in subsequent sections. To implement this software architecture, a physical hardware platform is required – which is described in the next section.

6.2.1. The Platform

Continuing on utilizing the Intel ATOM 1-N450 Development Board mentioned back in section 4.3.3, this x86 processor-based micro-ATX computer platform has been interfaced with the necessary peripherals for MCM functionality. This platform setup is shown below in Figure 69.

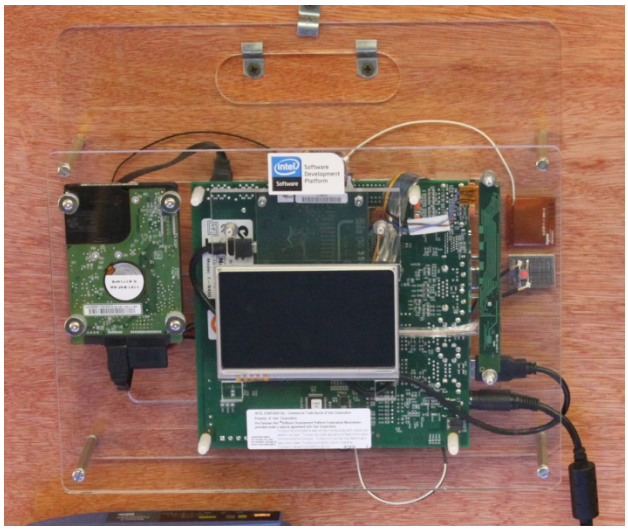

Figure 69 – Intel x 86 Platforms for Master Control Server

The platform setup for the MCM shown in the figure above utilizes multiple I/O interfaces suitable to the control server's functionality. The following peripherals have been integrated into this platform:

- Built-in 802.11b/g/n Wi-Fi mini PCI-e Adapter
- 5-Inch diagonal LCD-TFT Touch Screen

This platform is developmental setup and only represents the functionality of the control server. An ideal revision of this setup would include a complete enclosure, improved interfacing, and smaller form factor.

6.2.2. MCI Layers

The model control interface is a collection of three layers which perform distinct tasks in the operation of a program. A diagram depicting these layers is illustrated below in Figure 70.

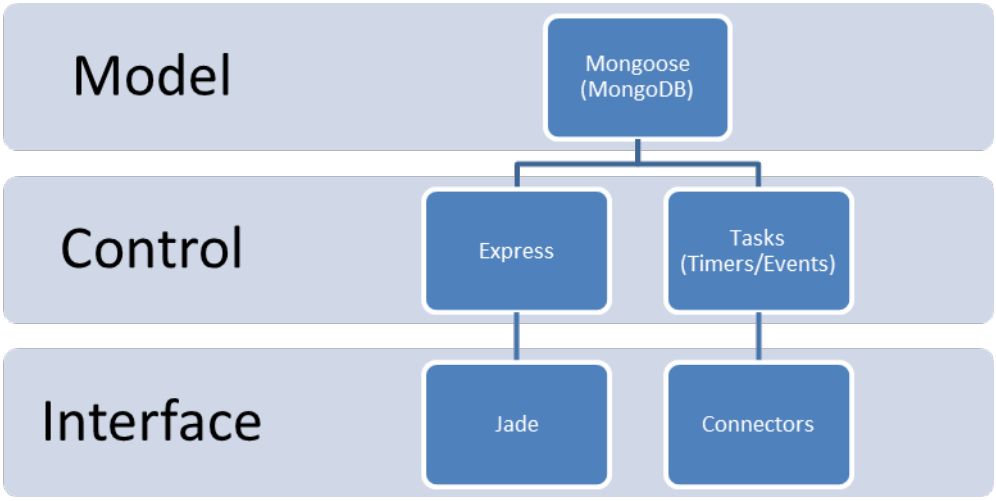

Figure 70 – Model-Control-Interface Hierarchy of the Control Server

Although not all are divided into such explicit layers, components of most complex programs can be described as model, control, and interface layers as shown in the figure above.

Layer 1: Model

The model layer of the MCI specifies a schema for the program's data. In simpler terms, the model is a description of the program's data structure. An SQL database schema is a well-known example of a model.

Layer 2: Control

A controller is an event handler. Its role in an MCI is to for a given event, pass the proper data set from the program's model to its interface. The Apache web server is an example of a complex controller that is bound to an HTTP server: it responds to HTTP requests with data that is either stored in a file or generated by a program, depending upon the server's configuration for the requested path.

Layer 3: Interface

An interface connects the program to external systems. Its responsibility in the MCI is to parse data from the controller into a format readable by the requesting system. A common example: the interface layer of most web applications is responsible for generating HTML to be transmitted to a web browser, which renders it into a user-readable graphic.

6.2.3. Language Selection

Node.js is an extension of Google's V8 JavaScript engine, which is used in their Chromium© web browser line. Although not as widely adopted as Perl or Python for server applications, JavaScript and Node.js have a distinct advantage over the two languages: Node.js has a simple and powerful event handler which leverages JavaScript's native event driven architecture. Because of its evented nature, Node.js is well suited to a software design with independent components, allowing it to handle many different tasks in a single thread. This allows developers to combine unrelated components, such as web interfaces and cron jobs, into a single program, and to share objects, such as a single database connection, between them. Additionally, Node.js includes a native packaging mechanism, allowing developers to segment projects into smaller, independently namespaced modules.

6.2.4. Software Modules

The control server was designed using several open source packages for Node.js as well as several custom packages written to connect to the project's hardware devices.

Mongoose (Model Layer)

Mongoose is an open source Node.js package developed commercially by LearnBoost for their line of classroom management software. It serves as a connector to Mongo DB (see Appendix N) as well as an advanced schema manager. A detailed description of the control server's model can be found in Appendix M.

Express (Control Layer)

Express, like Mongoose, is an open source package for Node.js. It performs application layer routing for Node.js' native Hyper-Text Transfer Protocol (HTTP) server. In simpler terms, Express specializes in mapping URLs in HTTP requests to handler functions on the web server. These handler functions use Mongoose to access the server's database and trigger a response to the requesting HTTP client.

The control server has two distinct sets of controls that are handled by Express. The first is a user interface rendered by Jade (See "Jade" Below). The second is a Representational State Transfer (ReST) API for adding, removing, updating, and accessing nodes of the server's model (See Appendix M).

Tasks (Control Layer)

Most large server applications require some tasks to be performed independently of user requests. UNIX systems include a tool called cron, which executes user-defined scripts on fixed time intervals. This

provides an extremely powerful mechanism for developers to leverage to perform periodic tasks such as garbage collection and fetching of data from remote APIs.

Node.js includes a powerful event handling system as well as JavaScript's native timer/interval primitives. The task manager is a home-grown module which combines these two features. It provides a central manager for events span multiple modules, and allows for simple management of time-interval tasks such as fetching data from remote devices, serving as a control for the server's back-end functions.

Jade (Interface Layer)

A variety of template packages exist for Node.js. Embedded JavaScript (EJS) is similar to Mason (a Perl package) in that it allows developers lace JavaScript, which is executed at request time, into plain text (usually HTML). Jade, based upon HAML (a Ruby templating package), uses a shorthand representation of HTML and allows a subset of JavaScript to be accessed with specific key words. Jade is used in this project because of its light-weight template support. Its limited subset of JavaScript is not an issue when the MCI model is implemented properly; that is, minimal data processing is performed by the interface layer.

The primary function of the control server's user interface is to provide a graphical means to modify nodes of the server's data model. For that reason, in addition to a dashboard style home page, the control server's user interface includes list, edit, and view pages for ports and devices. List pages display a flat list of every device or port available on the system. Edit and view pages display respectively a form for modifying, or a table with all information stored for a device or port. Form actions and hyper-links leverage the REST API discussed in the "Tasks" section above to modify devices and ports.

Connectors (Interface Layer)

The original control server connected to hardware devices using a routine hard-coded in the device schema, leveraging Mongoose's ability to handle custom methods for schemas. This, however, limits the functionality of the software by allowing it to connect only to a single API. The next revision of the software will farther leverage Node.js' packaging capabilities by storing device connection routines in modules, allowing users to select the correct connector when configuring the device in the web interface.

Chapter 7: Demonstration & Test Platform

This chapter will cover the implementation of the demonstration platform for this project. As an important piece of this project, the demonstration and test platform will be used to validate the prototype's effectiveness in performing the desired product requirements for this concept.

7.1. Platform Construction

This section will highlight the process taken to construct the demonstration platform to simulate a typical household multi-source heating configuration. This includes the construction of the climate zone, heat sources, and heat storage tank. Any modifications required for functionality will be noted as well.

7.1.1. Climate Zone

The climate zone is an important part of the demonstration and test platform because it will simulate the room in which the project will control the climate for. The construction of this part of the demonstration platform begins by building a rectangular box with an accessible interior. An illustration depicting the dimensions of the climate zone is shown below in Figure 71.

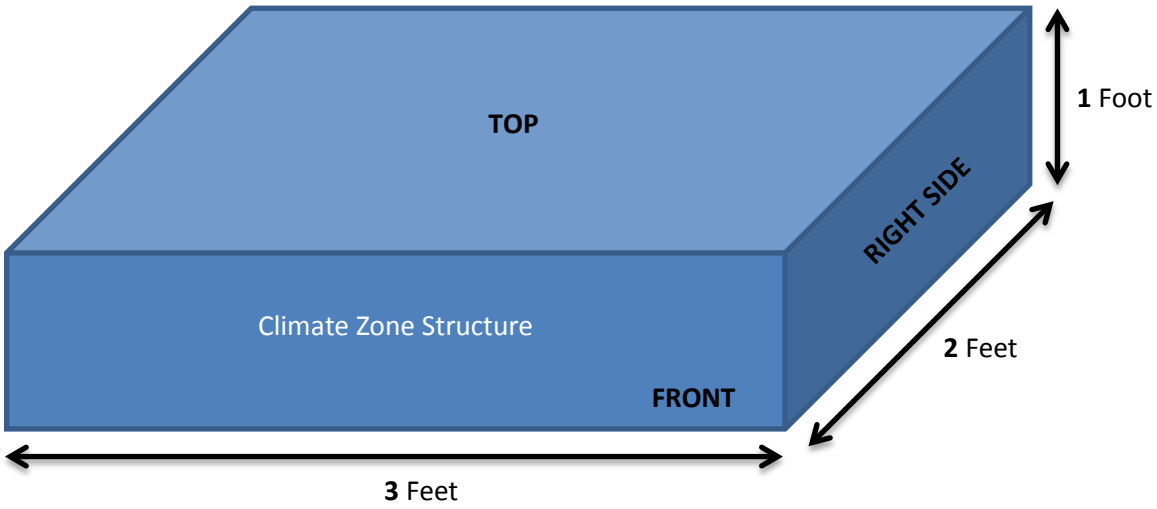

Figure 71 – Climate Zone Dimensions

The figure above depicts the approximate size and shape of the climate zone constructed for the demonstration platform. This structure will be built with a sheet of ¾ inch plywood used as the BOTTOM, LEFT and RIGHT sides, and the BACK side to form the foundation of the climate zone. The FRONT of the climate zone will contain a stationary framed-window using 1 by 2 inch trim board as the frame and an acrylic sheet as the transparent window. The TOP part will replicate the same structure of the FRONT with the exception that this part will be hinged to allow for opening and access to the interior of the climate zone box.

The climate zone cannot be complete without the essential components that make this box function as a simulated climate zone. This includes the hydronic heating element (also known as a radiator) such as the one shown below in Figure 72, and a supplemental heating system to function as an on-demand backup heating for the climate zone as shown below in Figure 73.

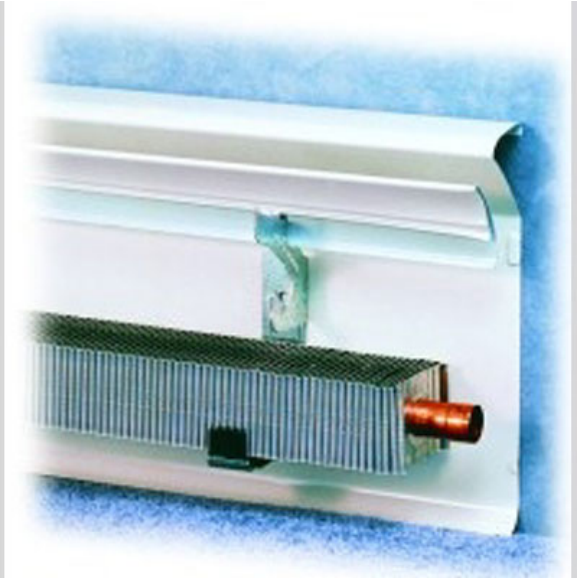

Figure 72 – Hydronic Heating Element (Radiator)

Figure 73 – Small Ceramic Heater (Supplemental)

The hydronic heating element, similar to the illustration shown above in Figure 72 will be 2 feet in length and will be installed inside towards the back of the climate zone. Two holes will be drilled to accommodate the inlet and outlet of the hydronic element. This will allow heated water to flow through the element provided from heating system outside the climate zone.

The supplemental heating system requirement will be fulfilled with the use of a small electric-based ceramic heater as shown above in Figure 73. The purpose of this component will serve as a back-up heating source when the primary heating system is not ready to deliver heat when needed. Given that this supplemental heater is electric, an additional hole is required inside the climate zone box to allow for the electrical cord to reach the outside environment. This heater will be directly interfaced with the Zone-Thermostat Module (ZTM) of the project to allow the system the ability to control this ceramic heater when needed.

7.1.2. Heat Source 1

Heat Source 1 will serve as the primary heat source of the entire heating system. This heat source will simulate the function of a typical household boiler system. The method in which this heat source will simulate a boiler will be through the use of an electric-powered hot plate burner to heat a roasting pot full of water. An inlet and outlet will be adapted to the pot to allow for the exchange of water through the pot when heat is applied. In order to control the flow of water through the roasting pot, a pump and valve configuration is required. The pump and valve are shown in the figures below.

Figure 74 – Heat Source 1 Water Pump

Figure 75 – Electrically-Controlled Water Valve

The water pump chosen for Heat Source 1 is shown above in Figure 74. This 12VDC water pump operates at approximately 4-Liters per minute, which will provide enough water flow to effectively move enough water through the roasting pot for heating to occur. This pump will be interfaced with a Source Control Module (SCM) from the project to be controlled when needed.

Along with the pump, an electrically-controlled solenoid valve is used in junction to control the flow of water through Heat Source 1. This 120V AC valve will be configured to open and allow water to flow when the pump is running. Respectfully if Heat Source 1 is not needed, then the valve will be closed to prevent water from over filling the roasting pot of the heat source and causing a catastrophic spill.

7.1.3. Heat Source 2

This heat source will serve the purpose of simulating a solar-energy collection system that many households are adapting to their traditional heating systems as “green” energy technology. This type of heating system works in parallel with the primary heating source to improve overall efficiency thus resulting is less energy consumption by the primary heat source. While an actual solar-energy collector is two large for this platform, a heating lamp is used to simulate the energy that would be provided by the sun, which is shown below in Figure 76.

Figure 76 – Heat Lamp Assembly

Figure 77 – Transmission Cooler Element

This heating lamp shown in the figure above will be used to provide infrared heat energy similar to that of the sun, but at a small enough amount for this scaled platform. In order to transfer this infrared heat energy to the water of the heating system, a transmission cooler is used as a heat-exchanger to absorb the infrared energy.

This heat source is configured such that the element is placed inside a black insulated pan for heat collection, meanwhile the heat lamp will be suspended above the element to direct the infrared energy onto the element inside the pan. The heat lamp is powered by 120V AC which will be controlled independently from the project. While the project has the capability to control the heat lamp, doing so would defeat the purpose of simulating a solar-collection system – we cannot control the sun. Vinyl tubing is used to connect to the inlet and outlet of the element so that water can flow through the element and absorb the heat energy. A 12V DC pump and 120V AC water valve is used with this heat source, similar to that of the first heat source to control the flow of water. This pump and valve will be interfaced with another Source Control Module from the project.

7.1.4. Heat Storage

The heat storage unit provides an efficient means for storing heat energy when it's not needed. This will drastically decrease the amount of energy consumed by the heat sources by eliminating the need for re-heating cold water. For example, it requires less energy overall to maintain a certain temperature of water rather than heating cold water on-demand. The storage unit also serves as an energy "saver" by storing heated water for use later rather than wasting it. To construct the heat storage unit, a 5-Gallon bucket is used as the primary containment, similarly shown below in Figure 78.

Figure 78 – 5-Gallon Sealed Plastic Bucket

Figure 79 – Fiberglass Insulation

To provide the adequate function of a heat storage unit on a smaller scale, a 5-gallon plastic bucket was used similar to the one shown above in Figure 78. This bucket contains a rubber-gasket sealable lid that snaps into place on top of the bucket to contain liquids. The proper inlet and outlet fittings were installed into the sidewall of the bucket to allow for water withdrawal and deposit. In order to prevent heat-loss from heat energy radiating from the bucket, fiberglass insulation is used by wrapping the exterior walls of the bucket. This will contain the heat energy stored inside and decrease the rate of heat-loss when hot water is being stored.

The heat storage unit will be placed between the heat sources and the climate zone in this system configuration. Each heat source has a dedicated pump and valve that will circulate water through the source and into the storage tank. An additional water pump and electronic valve is used to pump heated water from the storage tank to the climate zone when desired. For this, another SCM is required to control the heat storage unit.

7.2. Post-Build Modifications

This section will discuss the various modifications that were made to the demonstration platform after its initial construction. These modifications were required to ensure the functionality of the platform climate control system.

7.2.1. Multi-Zone Configuration

To effectively demonstrate the multi-zone adaptability of this project, the originally constructed climate zone was modified to become two equal zones within the same climate zone. This modification will help serve as a more realistic simulation to a typical household heating system. The image below is the result from the changes done on the climate zone.

Figure 80 – Climate Zone Box

As indicated in the illustration above in Figure 80, the climate zone box has been proportionately divided into two climate zones. This will provide a simulated typical household setup where two adjacent rooms are heated by the same baseboard hydronic heater. Having two climate zones in the system will allow for the project to compare each zone and provide the necessary features of a multi-zone configuration.

7.2.2. Visual Temperature Display

The best demonstration includes a strong visual component that others can see as a result of a change. The demonstration platform serves the purpose of demonstrating this climate control project's functionality. Looking back on the original purpose of this project – to control "climate" is only a feature that a person can feel, not see. Given the scale and nature of the demonstration platform, adjusting the temperature inside the simulated climate zone does not produce any visual result or confirmation.

A Solution

To approach this problem, a mini-project has been developed to be incorporated onto the demonstration platform and provide a means of visual indication of temperature inside the climate zones. To achieve this feature, a set of Red, Green, and Blue LED light strips will be used to display a temperature reference inside the climate zone. These light strips can be seen below in Figure 81.

Figure 81 – Red, Green, & Blue LED Light Strips

The LED light strips shown in the figure above will be used to illuminate the inside of each climate zone with respect to a set range of temperatures. As the temperature inside the zone reaches a certain point, the color will change to provide a visual confirmation of the project's effectiveness of controlling the climate inside the zones on the demonstration platform. In order to utilize these LED strips to provide this functionality, a circuit must be designed and constructed to drive these LEDs based on a given temperature. While this system will be adapted to the current demonstration platform, the systems data acquisition and output will be completely isolated from the climate control system itself.

Visual Display System Design

To begin designing the circuitry for this visual display system, the requirements for the functionality of this system must be defined. A systematic model will be used to describe this system and the requirements. A block diagram to illustrate this is shown below in Figure 82.

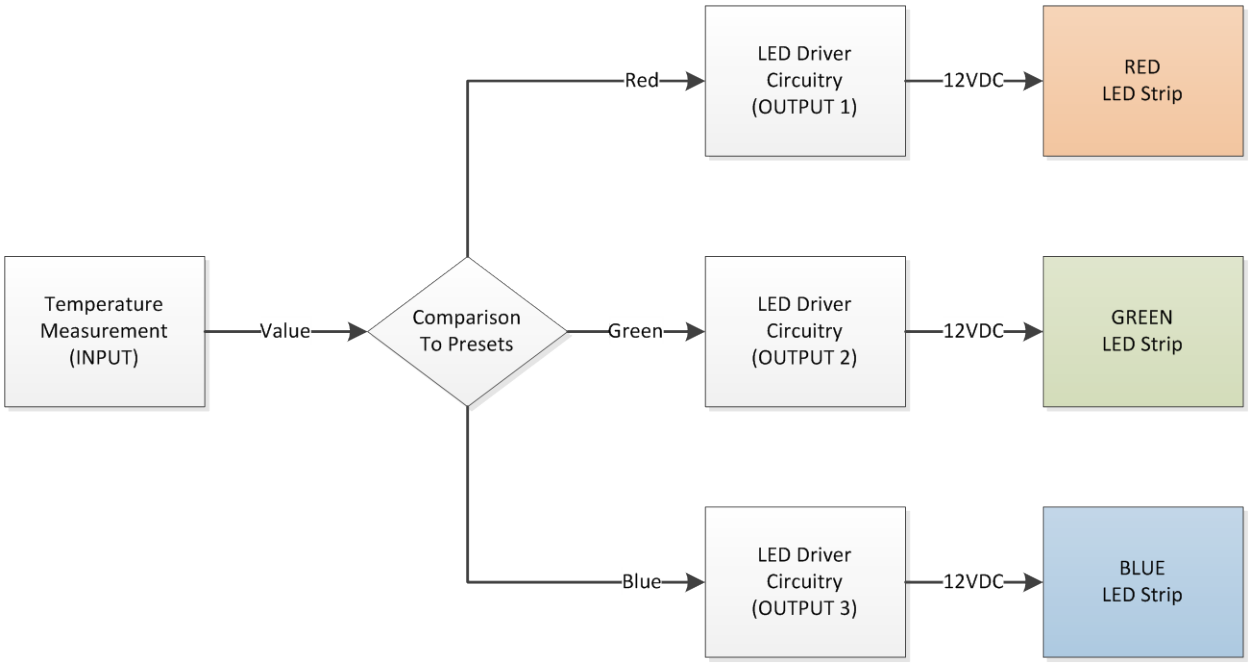

Figure 82 – Visual Display System Block Diagram

The system block diagram shown above illustrates the order of functionality for the Visual Display that will be added to the climate zone. This block diagram will be used to determine the necessary components required to implement this system.

The first part of this system begins by measuring the temperature inside of a particular climate zone. This temperature measurement will produce a value that will be taken and compared to a set of pre-defined value ranges to determine which LED color to light. Depending on the temperature value measured, the system will turn on the circuitry to drive a particular LED light strip.

Given the scale of the demonstration platform, it would be unsubstantial to maintain a realistic temperature of about 70-75 Degrees Fahrenheit. If the range of temperatures were configured realistically, then actual room temperature would prevent the climate box from falling below this threshold and the Blue LED strip would never light up! To prevent this, the desired temperature range must be significantly higher than actual room temperature.

Visual Display Implementation

In order to construct a device capable of meeting the required design specifications, the implementation of the visual display has been broken down into three main parts: input, control, and output driver.

The Input Measurement

The first part of implementing the visual display system is the input mechanism. This part is crucial to taking an accurate temperature measurement so that the correct color LED strip can be activated as desired. To achieve this feature, a Negative-Temperature Coefficient (NTC) thermistor will be used to attain a value with respect to the temperature.

For the purpose of this project, a standard precision 10K-ohm thermistor was chosen. In order to interface this thermistor such that a usable temperature-voltage value can be attained, a voltage divider circuit is used as shown below in Figure 83.

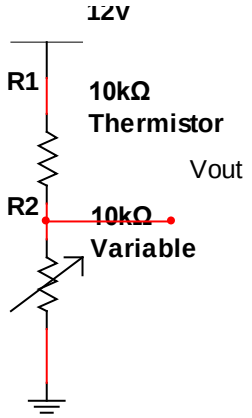

Figure 83 – Voltage Divider Circuit (Thermistor)

Data Type	Value
β-Value (Kelvin)	3977
Tolerance	0.75%

Table 1 – NTC Thermistor Values*

*NTCLE100E3 Datasheet located in Appendix X

Temperature (°C|°F)	Resistance (R1)
Room T = 25°C | 77°F	10,000 Ω (Nominal)
26.6°C | 80°F	9,356 Ω
29.4°C | 85°F	8,252 Ω
32.2°C | 90°F	7,951 Ω
35.0°C | 95°F	

Table 2 – Calculated Resistance Values for Thermistor

The circuit shown in the figure above is a simple interfacing circuit for the thermistor. In order to determine the value of V_{OUT} at a given temperature, the resistance value of the NTC thermistor must be calculated first. The equation below is used to determine these resistance values at temperature T, which are shown above in Table 2. (Note: T values in the equation are in °Kelvin).

$$R_T = R_{NOMINAL} * e^{\left(\frac{1}{T} - \frac{1}{T_R}\right) * \beta}$$

Now with the theoretical resistance calculated for the thermistor, the voltage at V_{OUT} can be determined. A variable resistor (R2) is used to offset the bias if needed to match the desired voltage output values.

The Display Driver

The purpose of the display driver is to take an input of a given voltage value, and produce an output based on that value. There are several methods that can be used to solve this requirement, but in order to maintain simplicity of this design a pre-configured integrated circuit will be used with minimal discrete components.

For the display driver circuit, a Linear DOT/BAR Display Driver: LM3914 will be used to control the circuit. The datasheet for this IC can be found in Appendix X. This display driver is specifically designed to drive an array of LEDs and is able to regulate the current for each LED with no requirement for external resistors. Unfortunately, this display driver cannot drive our LED strips directly given their current and voltage requirements; so a transistor-driver circuit will be used with the display driver, using the output of the driver as a signal-only characteristic. A schematic illustrating the circuit configuration for the display driver is shown below in Figure 84.

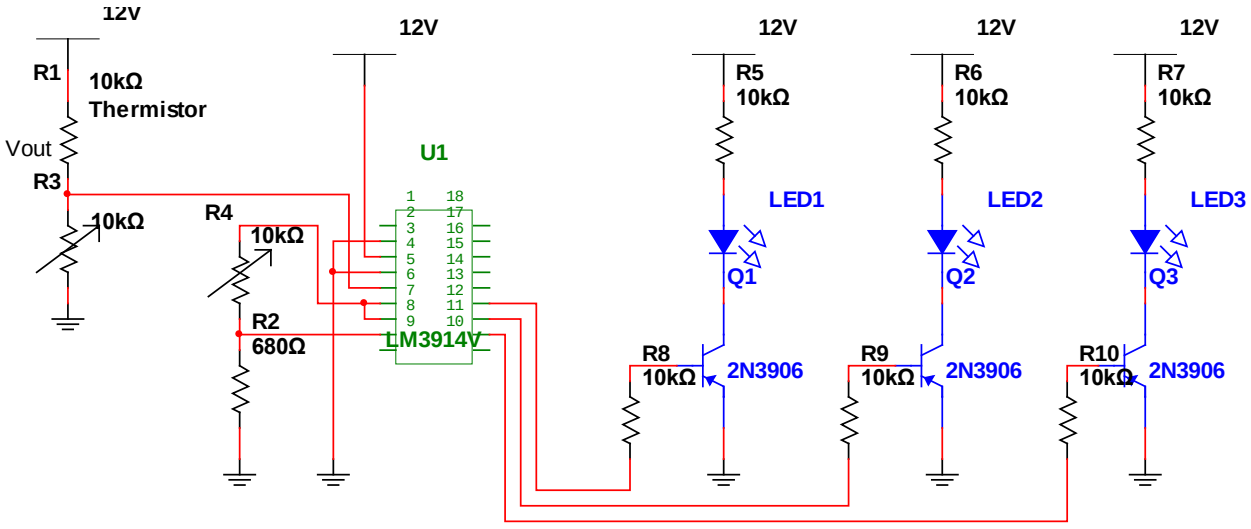

Figure 84 – Visual Display Driver Circuit

The figure shown above depicts the circuit configured to drive the LED strips for one of the two zones in the climate zone box. Along with the input voltage divider discussed earlier, the other divider in the circuit above is used to adjust the LM3914 Display Driver IC's reference value. The output side of the display driver IC are three PNP3906 Bipolar-Junction Transistors (BJTs) connected to each of the three colored LED strips (LED and resistor represent the entire strip). These BJTs are configured to turn on when their base potential is negative; i.e. when the driver IC outputs a logic LOW value.

Visual Display Results

Upon completion of the design and construction of the visual display circuit, the circuit was installed into the climate zone box to serve its intended use – to provide a visual indication of the temperature inside the climate zones. A picture of the visual display device circuit is shown below in Figure 85.

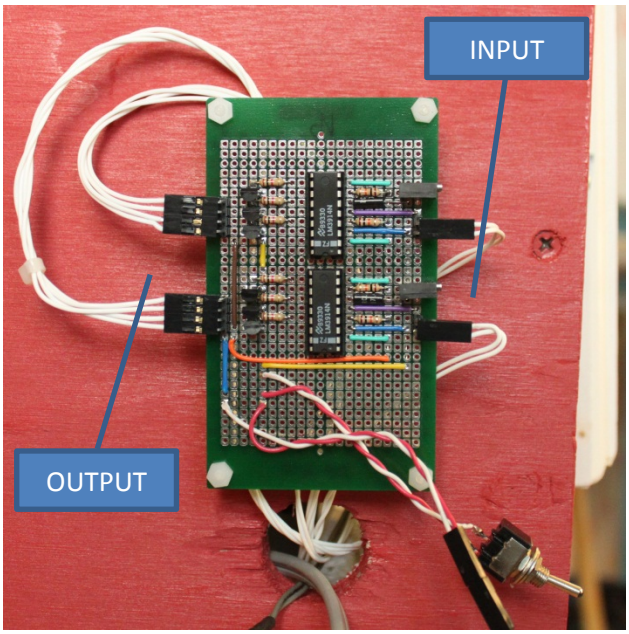

Figure 85 – The Visual Display Control Circuit Device

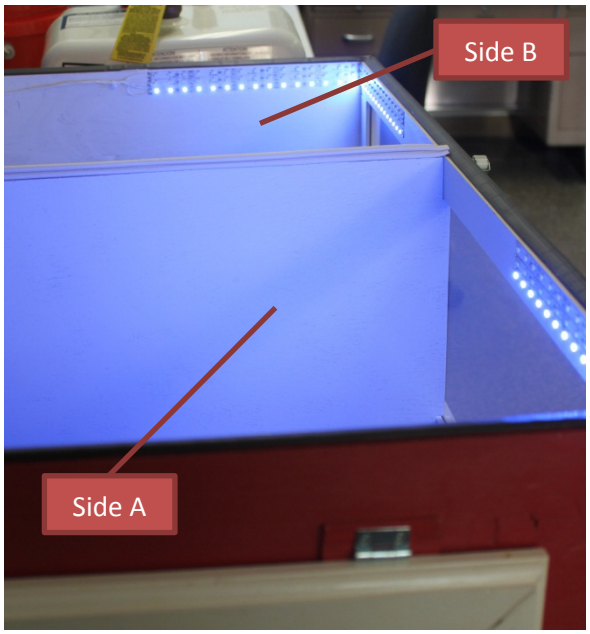

Figure 86 – Installed LED Strips in Climate Zones

The image in Figure 85 is of the installed Visual Display Control circuit that is mounted on the outside rear of the Climate Zone simulation box. The circuit has been constructed on a small-sized prototyping board, with the connected inputs indicated on the right and the output connections to the LED strips on the left. Also included in the circuit is an ON/OFF switch to shut off the entire circuit when not in use.

The image to the right in Figure 86 is of the climate zone box opened to display the LED strips that are installed – with the BLUE strip currently illuminated. This clearly indicates that the temperature inside each zone is below the desired threshold, which is to be expected with the lid of the box opened in the photo. Also indicated in this picture is the designation of each different zone: Zone A located towards the front and Zone B is located towards the back of the photograph. As described previously, the climate zone box includes the divider to simulate two separate zones – or "rooms".

7.2.3. Check Valves

One of the first problems that were identified with the demonstration platform was the absence of a check valve on the pump-side of Heat Source 1. The problem with this part of the heating system is that the configuration of Heat Source 1 is not a closed-loop like the rest of the system. Since the roasting pot is not sealed, pressure cannot be regulated in the pot. If Heat Source 1 is in idle mode (not in use) and another pump is turned on in the system, any positive pressure buildup in the system could cause a back-flow to occur through the pump of Heat Source 1. This could result in a potential catastrophe of filling the roasting pot beyond its capacity causing a spill. To prevent this problem from occurring, a check valve similar to the one shown below in Figure 87 was installed in series with the pump of Heat Source 1.

Figure 87 – Plastic Swing Check Valve

The check valve operates by only allowing water to flow in one direction. The functionality behind the check-valve's operation is based on a differential in pressure across the valve. When the valve is installed in the direction of the desired water flow, a positive pressure differential will result in the valve opening and allowing the water to flow freely. In the event that this difference in pressure becomes negative against the desired direction, the valve closes thus restricting the flow of water.

In the system, Heat Source 1 is configured by using a pump to draw hot water from the roasting pot and dump it into the heat storage unit. The valve is installed in the positive direction with the flow of the pump. In this case, when the pump is turned off, any pressure inside the rest of the system will close the check valve and prevent water from dumping backwards into the roasting pot.

7.2.4. Easier Fill/Drain Configuration

One issue that became more apparent during the testing of the demonstration platform was the difficult process in filling the system with water and draining it when needed. During initial filling of the system with water, several minor leaks occurred that required the system to be drained in order to fix the leaks properly. This became such a time-consuming process that lead to the creation of a fill and drain system that ultimately makes this process much easier. An illustration below in Figure 88 shows how this method was accomplished.

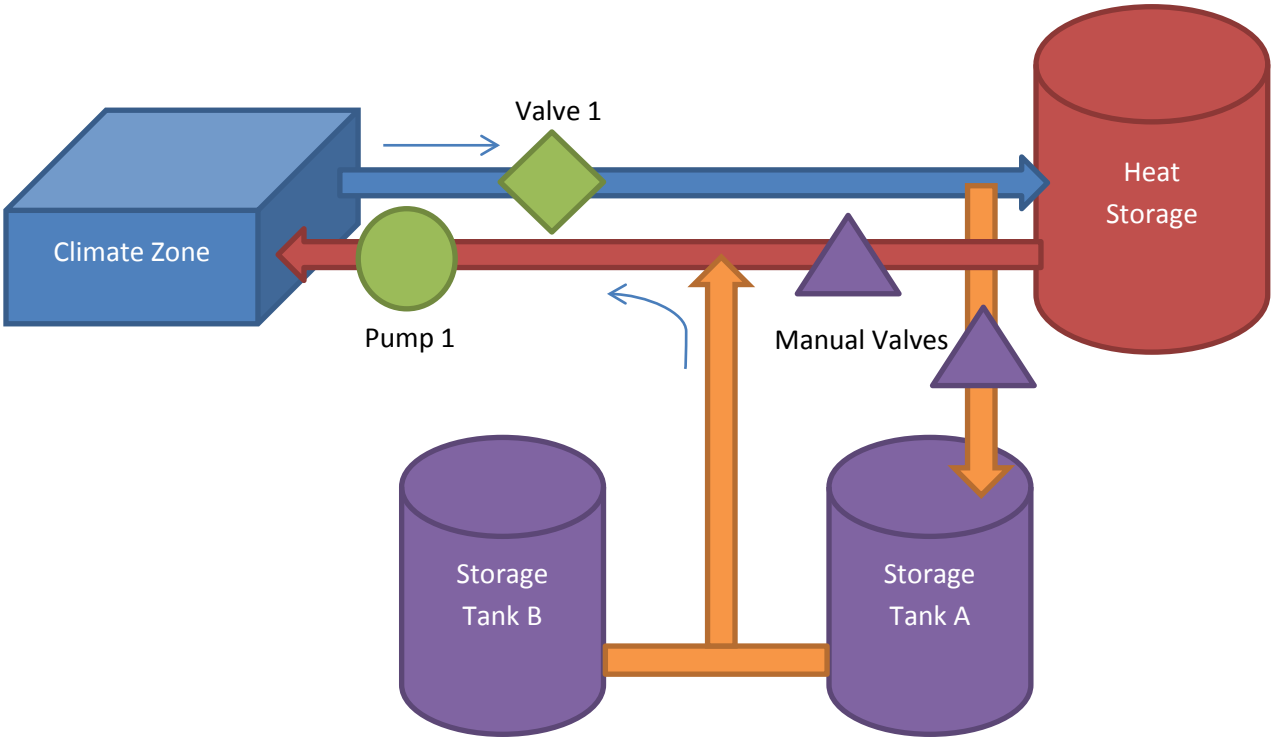

Figure 88 – Easy Fill/Empty Adaptation

The diagram depicted above is a conceptual illustration of the easy Fill/Empty adaptation made to the demonstration system. With the use of additional materials not relevant to the heating system, water can be added to fill the system when it's ready for operation as well as emptied when filled with water in order to work on the platform. The adaptation includes two storage tanks to store water for filling the system. Manual valves have been incorporated to transition the existing system to fill/drain mode. In order to fill the heating system with water, the manual valve connected to the climate zone source pipe is closed and the pump for the climate zone is used to pull water up from the storage tanks below. While this may pump water through the climate zone, the closed-loop system results in water returning through and getting dumped into the storage tank. When draining is needed, the opposite manual valve is opened and gravity pulls any water in the system down into the storage tanks.

Chapter 8: Results & Analysis

This section will cover in detail the results from the project implementation in the previous sections. This includes the Hardware, Software, and Demonstration Platform sections. These results will be analyzed and any future rendition plans will be discussed to continue this project towards becoming a marketable product.

8.1. Hardware Results

This results subsection includes an analysis of the initial design, the current state the hardware has progressed to, and the next steps for the device. After three months of hardware design and implementation, the team has learned several things about the product and the design. These concepts will be discussed in the following sections and the information used in the products future work.

8.1.1. Design Analysis

This section includes an analysis of the current design and what to change for the future. The current design uses three boards, a power and processing, a peripheral, and a user interface board. This design does not allow much flexibility in which modules are implemented in each device. The processing board also includes the AC & DC power lines, the communication, the PIC32 controller, and the ADC connections. This design was a low cost solution for implementing and testing several features initially, but it is not a sustainable long term solution.

The hardware needs to be more flexible and interchangeable. The next revision of the design will utilize a central PIC32 breakout board with board to board connectors. Other modules will be able to plug into this breakout board for control as illustrated below in Figure 89.

This allows for a central constant controller which can be used for many different individual applications based on which modules are connected. The increased modularity of the system allows for isolation of components which can increase development time and flexibility long term. This will allow for changes to specific components without affecting the entire system design. For example a different Wi-Fi communication package can be implemented at a later point without any necessary hardware changes.

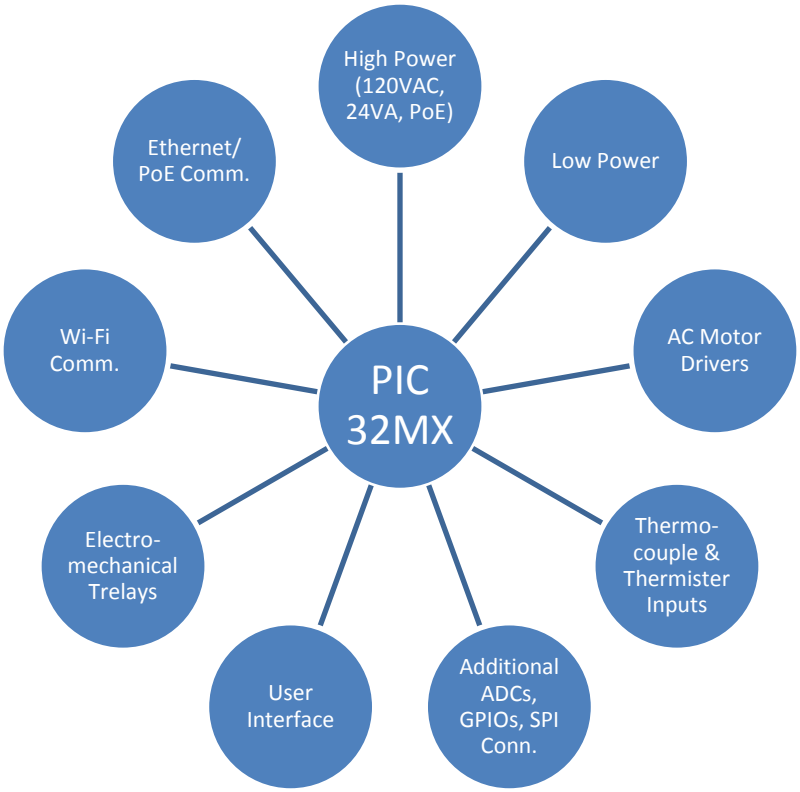

Figure 89 – Hardware Rev. 2 Concept

Another large component in the hardware design is the embedded software which controls the basic functionality of each device. The current system makes use of a free TCP/IP stack offered by Microchip. While this software works for a prototype, a more robust package is needed for commercialization. In addition, it is highly suggested a real-time operating system (RTOS) be implemented on the hardware. This will increase performance and development time. Each process becomes an individual thread which run in parallel and controlled by the RTOS.

8.1.2. Current State

Following completion of the initial hardware design analysis, the team proceeded forward with development. This included a re-design of the PIC32 core components. The design focused on producing a robust breakout platform for the PIC32. With the intention of it becoming the fundamental basis for the central control board in the final design. The design took into account the lessons learned about PIC32 stability from the initial design and used a simple signal breakout pattern. The main concept being that using jumper wires, other modules could be tested and perfected.

The hardware design is simple and robust. It features a centrally placed PIC32 controller with all necessary passive components (bypass capacitors, resistors, and oscillators) within 3mm of their

corresponding pins. Three 10uF bulk capacitors have also been added to increase power supply stability. The problems identified in the initial hardware design are all corrected in this design. The hardware includes a 3”x3” four layer PCB with a 100 .1” spaced male pins for all of the device control signals Figure 90. A panel including three individual PCBs purchased from Advanced Circuits for $66 (Appendix Y).

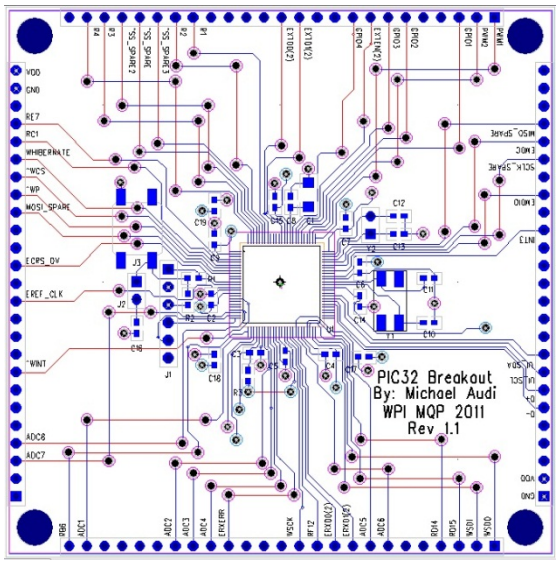

Figure 90 – PIC32 Breakout Board Layout

The resulting boards, after a successful build and test are significantly more robust and stable than the Microchip offered starter kits. Software development has started in implementing a functional TCP/IP stack with Wi-Fi on the new boards. The boards drive relays and read temperature data using the thermocouple amplifiers detailed previously.

8.1.3. Next Steps

The hardware design still requires future development. The next main focus points for the hardware are to, prototype a functional system using the new hardware, implement a RTOS, implement hardware modules, and finally write and test module interface software. These points can be implemented partially in parallel. However, for the most part they should be completed in the order listed. The new hardware must be interfaced and a prototype demonstrating the core features of the system be created. Next a RTOS including a commercial TCP/IP stack is to be implemented. In parallel to the RTOS, hardware design and development of the flexible modules for peripheral interfacing will be developed. As these hardware modules are built and tested, software development to control all the different aspects of the peripherals will start. These next steps will help the design approach a final product ready for the market.

8.2. Software Results

This section will discuss the results of the software implementation for this project. The current state of the control server is the result of several iterations of models and user interfaces. The following sections will discuss these previous versions, and propose several improvements for future versions.

8.2.1. Past Iterations and Improvements

Much effort has been spent trying to optimize the flow of the user interface. This is for the dual purposes of usability with a conventional web browser and future portability to a mobile browser interface. Shown below is a screenshot of the user interface in Figure 91.

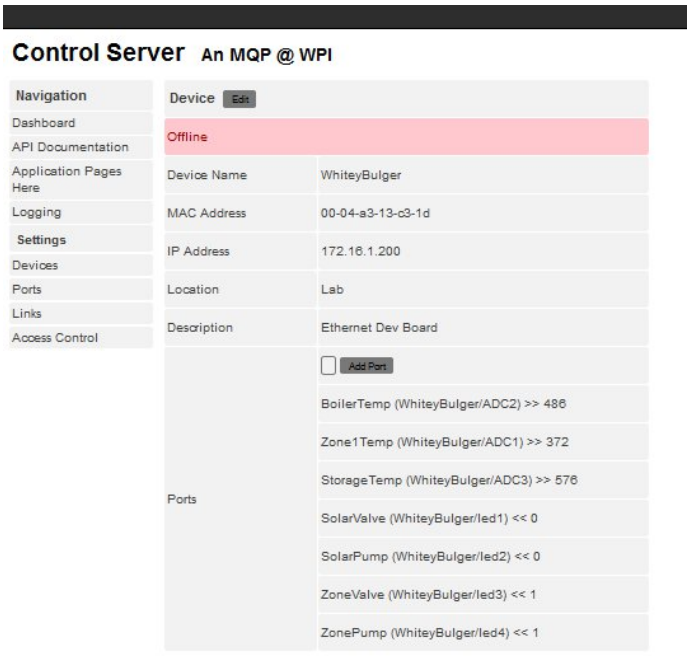

Figure 91 – A View Interface, Displaying a Device

The original interface requires that users move into an edit mode to modify all properties of a node in the server's model. This proved to be cumbersome if the user needed to make changes to multiple nodes. The simple solution was to alter the layout of the interface to reduce the number of "clicks" a user was required to make to complete a task.

The linking structure of the server provides its core functionality, that is, to use an aggregation of inputs to generate a signal for each output. The original server design only allowed ports designated as "Input" to be used in calculations. This, again, proved cumbersome if more than one output shared the

same driving function. To remedy this, the server was altered to allow all ports to behave as inputs. This behavior allows the cascading to output signals through more than one port.

8.2.2. Current State

The current state of the control server is stable and operational. It is able to perform calculations to drive outputs, and provides a simple interface for users to modify its behavior. The server is designed in a modular fashion, with code for various tasks grouped logically in files and folders. Additionally, the code provides facilities to easily integrate new features.

8.2.3. Future Improvements

Being a software product, there is room for improvement in virtually every aspect of the control server; however, as the objective of this project is to produce a working product, improvements will be limited to several facets. The first focuses upon the server's ability to connect to external systems. By improving the server's "device connector " facilities, the control server can be used to aggregate data from not only the project's hardware devices, but virtually any service with a network API, including weather services and fuel suppliers.

This leads to the second improvement target: a plugin packaging system. This will allow future development of the server to be developed in a highly modular fashion. The plugin system will require a set of management functions to import plugins at startup, and will leverage the server's existing API for model and control management and access. Plugins will include any necessary model schemas, controls, and panes for the user interface as well as custom code required by the plugin.

The final improvement is in regard to the architecture of the user interface. That is, to separate the interface from the data API, and to move as much of the rendering as possible to the browser side by leveraging JavaScript and jQuery. This practice is becoming common in high-end web applications such as Twitter and Google's Search page because it removes a large part of the processing load for user requests from web servers, making the service faster and more reliable.

8.3. Demonstration Platform Results

This section will detail the results from the demonstration platform implementation. This includes analysis of these results, including any problems that have occurred. Any future rendition plans and recommendations will be included in this section as well.

8.3.1. Design Analysis

To recap the demonstration setup, the platform simulates a household heating system with multiple heat sources and multiple climate zones. The purpose of this demonstration platform was to provide a typical heating configuration such that this project's concept could easily adapt to and control efficiently and effectively. A picture of the demonstration platform setup is shown below in Figure 92.

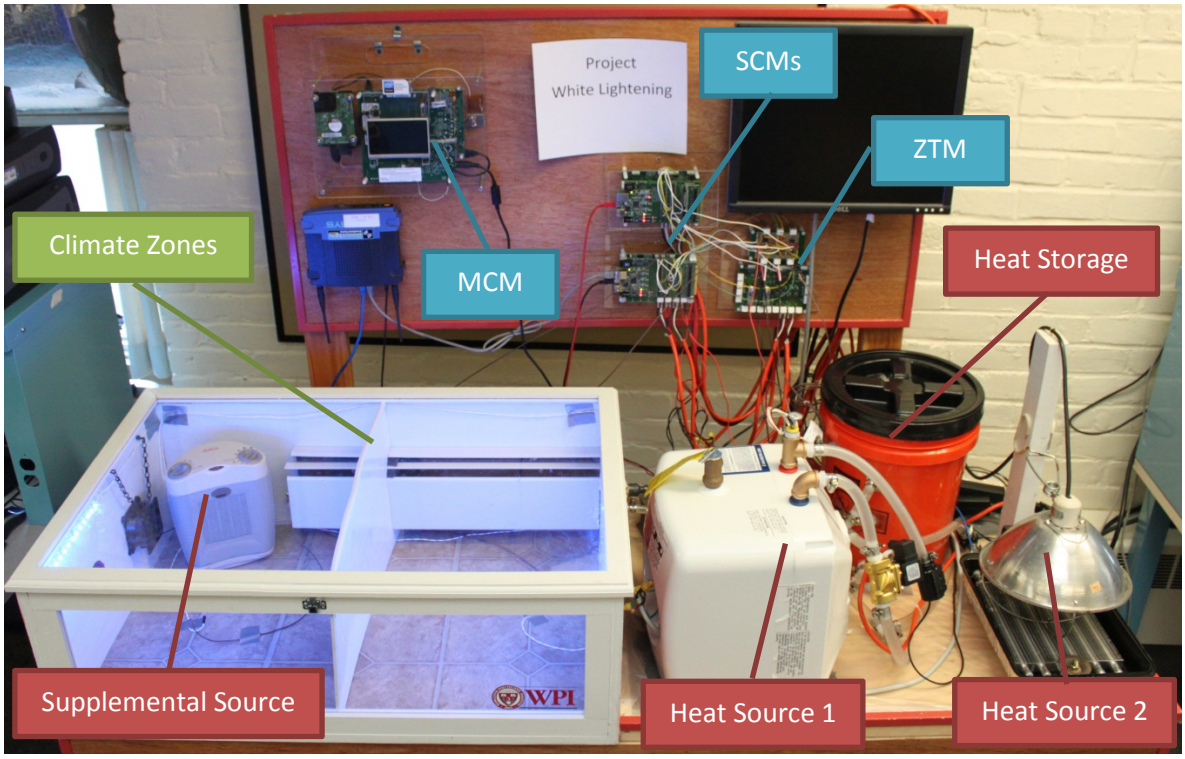

Figure 92 – Demonstration Platform Results

As shown in the figure above, the demonstration platform consists of two heat sources: a simulated hot-water boiler (Heat Source 1) and a simulated solar collector (Heat Source 2). These sources are fed into a primary storage tank (Heat Storage). From the storage, the heated water is fed into the climate zone box's hydronic heater. A zone-specific backup heat source (Supplemental Source) is included in one of the two zones. This entire system is controlled by the Master Control Module (MCM), Source Control Modules (SCMs) and Zone-Thermostat Module (ZTM) shown mounted above the demonstration.

8.3.2. Defining Concept Functionality

The primary purpose of this demonstration platform was to simulate an example multi-source household configuration. The objective of this project was to utilize this platform to demonstrate how this climate control system could control a heating system more efficiently than typical household climate control methods (i.e. Thermostats). To highlight this concept functionality in this document, two scenarios will be demonstrated below with and without the Home Energy Automation Technology (H.E.A.T.) system.

Introduce a Common Problem – The Draft

One common problem that is found in many households with multiple rooms is that a draft can cause a room or several rooms to experience a drop in temperature compared the rest of the house. The image below in Figure 93 shows how the climate zone box was modified to introduce such a problem using an exhaust fan.

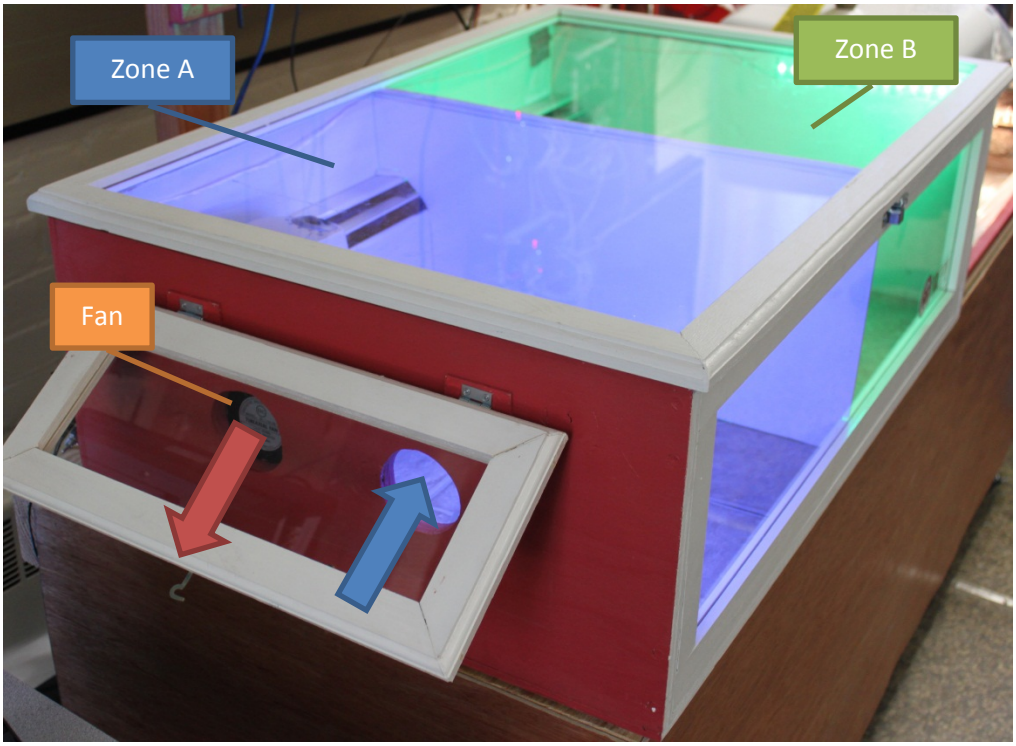

Figure 93 – Creating a Draft: Window with Exhaust Fan

The figure above illustrates how a draft can cause a difference in temperature between the two zones, with Zone A dropping below the desired temperature setting indicated by the blue illumination. However, this draft does not affect Zone B, which remains at the desired temperature setting for the climate box.

Without H.E.A.T. – Typical Control Scenario

Now with the problem that was introduced by creating a draft in one of the two zones, the temperature has fallen below the desired threshold in Zone A as illustrated in Figure 93. With this scenario, if the main thermostat was located in Zone B where the temperature remains as desired, then Zone A would always remain colder because the heating system would remain OFF. However, if the thermostat was located in Zone A, then the house's heating system would be activated given the fall in temperature. Unfortunately, this causes another problem to arise. Take the figure shown below:

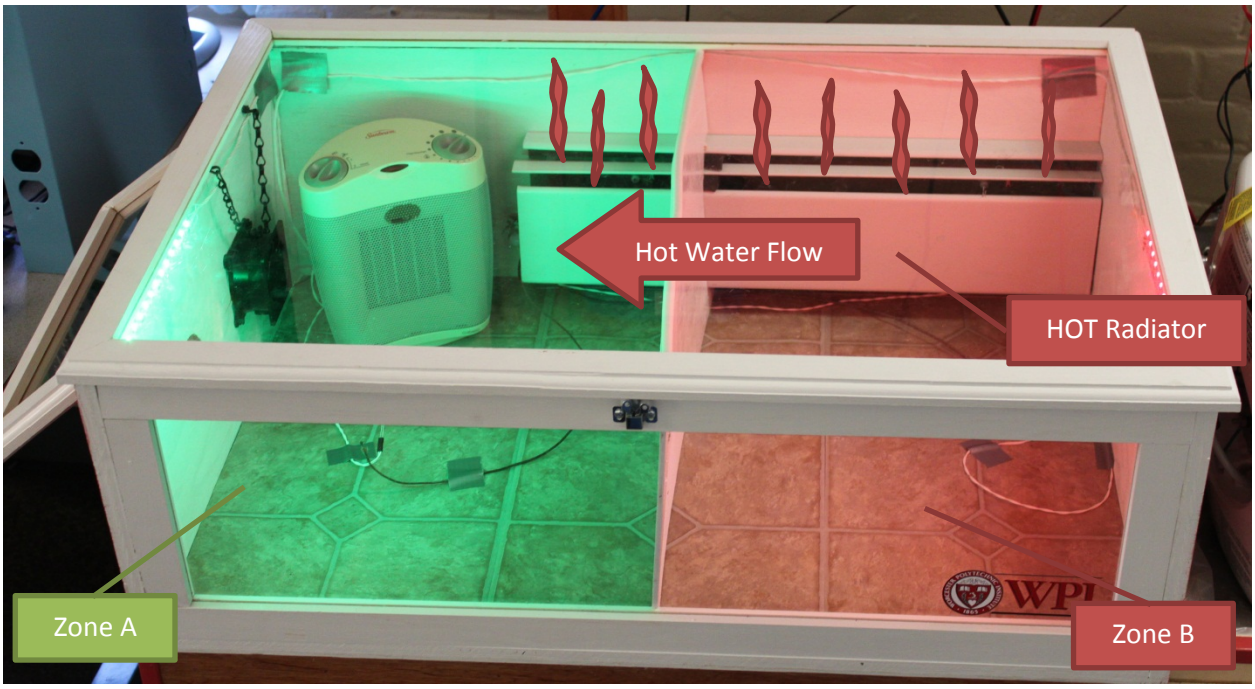

Figure 94 – Simulating a Standard Climate Control System

With the call for heat from Zone A in Figure 94, the radiator becomes heated and radiates heat within the "house" as designed. Zone A eventually reaches the desired temperature with the draft induced on the side wall. However, another problem is caused with this type of climate control method. Many household hydronic heating systems typically have baseboard radiators that run through multiple zones (as simulated above). While Zone A reaches its desired temperature, Zone B suffers from over-heating since the radiator runs through Zone A and B.

This type of climate control is very inefficient. This hydronic heating system is simulating that of a hydronic boiler setup, which can consume either #2 heating oil or natural gas. This is a waste of resources just to heat up one zone when the remainder of the house is already at the desired temperature.

With H.E.A.T. – An Efficient Control Scenario

Introducing the Heat Energy Automation Technology system, this climate control setup can drastically improve the overall efficiency of a household heating system. Building off of the previous scenario where a draft was introduced into one of the climate zones (as shown in Figure 93), the H.E.A.T. system would handle this problem differently. The figure shown below depicts how this control system would handle the temperature difference problem.

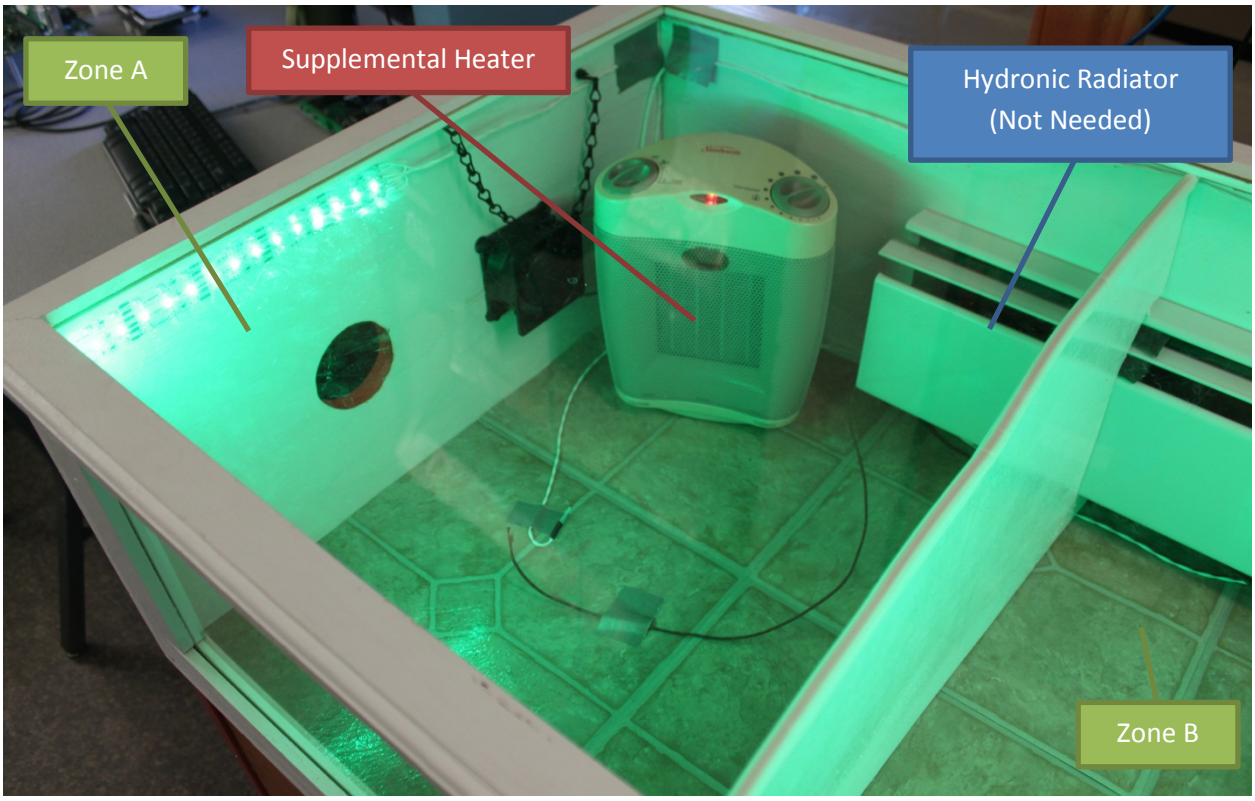

Figure 95 – The H.E.A.T. System is the Solution

Shown above in Figure 95, the H.E.A.T. system is able to provide a solution to the draft problem in an efficient and effective manner. Here is how the system handled the problem:

Given the draft introduced in Zone A, the drop in temperature was "noticed" by the H.E.A.T. system (similar to a thermostat). However, with the multi-zone configuration, the system was able to identify that Zone B was at the desired temperature. Given this situation, the system is able to determine the most efficient method for getting heat to Zone A. In this configuration, Zone A is equipped with a supplemental heater. By activating this heat source, the system is able to bring Zone A back to the desired temperature without firing up the boiler and wasting more-costly resources.

Chapter 9: Conclusion

The primary objective of this project was to identify an important problem in home climate control and develop a viable product concept that will provide an ideal solution to this problem. Throughout the duration of this Major Qualifying Project, many obstacles were overcome to achieve the objectives set forth with this project. The completion of this project resulted in a functional product prototype that was capable of demonstrating the product solution concept.

First, background research was conducted on various common household heating systems and typical climate control methods. The problem in this topic area was identified as an absence in unified climate source communication and control. A few products were identified as providing a similar solution to this problem, but neither product directly focused on multi-source climate control. This became the foundation for this project's concept.

The project concept was defined as a computerized climate control system that provides the following capabilities:

- Ability to adapt to multiple climate sources of various types
- Ability to control these climate sources simultaneously
- Ability to maintain the most efficient method of controlling these sources effectively

With designing a product concept that incorporates these three major characteristics, a viable and effective product can be implemented to solve this problematic area.

For project design and implementation, the development was broken down into three major sections: Hardware, Software, and the Demonstration & Test Platform. Each section is critical to the success of this project and the functionality of the product prototype. The hardware portion of this project includes the physical embedded system components that had to be designed and constructed so that the software developed could function properly on the hardware. The software portion included developing the communications, Inputs & Outputs, and climate control methods for the entire project. The demonstration platform was designed and constructed to test the functionality of the prototype. This platform included a scaled-down multi-source multi-zone household heating system that would enable configuration, testing, and demonstration of the working prototype.

Appendix A: Control Server Documents

The following documents are in reference to the Control Server part of this report.

Document A-1: Control Server Model

Shown below is a layout of the model for the Control Server.

Key:

- **[descriptor]** Array
- **(type)** Data Type
- **Blue** Mongoose Schema Object
- **Green** Scalar Value
- **Red** Mongoose DB-Reference

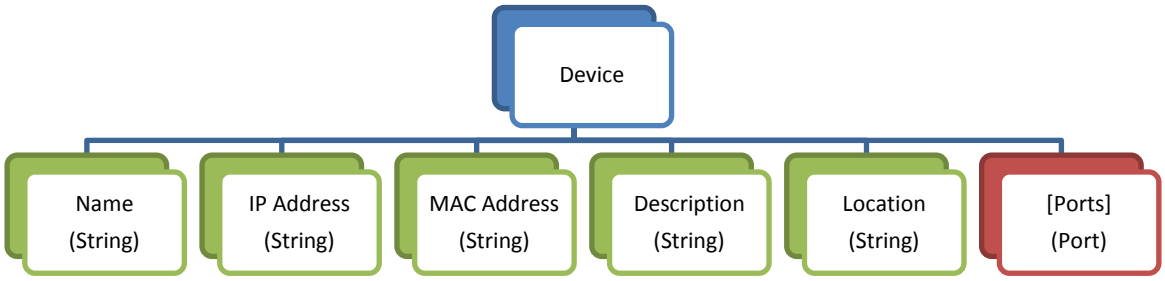

Figure 96 – Device Model

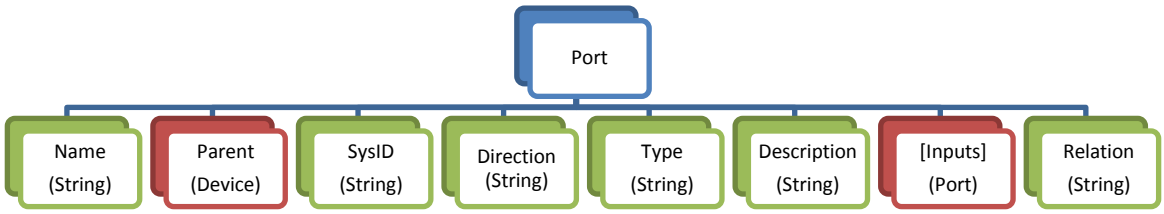

Figure 97 – Port Model

Why? Model Rational

The intention of the control server is to provide a flat environment to perform advanced calculations with data from multiple network sources. To accomplish this, a duality between devices, as physical entities, and ports, as virtual entities has been established. Devices, in their physical sense, have inputs and outputs, and an IP-based means to manipulate them. The device model (Figure 96) allows the server to fetch data from an external system and store data points in ports referenced by the device, as well as write updated data back to that system.

The port model (Figure 97) acknowledges that ports are bound to a parent device, but stores ports in a separate collection, rather than as a child object in the device model. This allows ports to be accessed in a flat fashion to reduce the processing and time required to access port object by its identifier from anywhere in the system. Using a set of references (a feature of Mongoose and Mongo DB; see Appendix N), ports can directly access their parent device, and likewise, devices can directly access their list of child ports.

Ports operate in two modes: input or output. Input ports are responsible for performing any conversion, filtering, and smoothing required producing a usable signal. Output ports calculate their values from the combination of input port values and a function to relate those values.

Document A-2: Mongo DB and NoSQL

Mongoose is a NoSQL database engine. NoSQL is a relatively new class of database which provides storage with varying degrees of schema definition. Memcache, for instance, is a database engine which provides key-value pair storage. This is a useful model for web page session storage, in which a user's session data must be accessed with a session identifier, and need only be stored for a short amount of time. Mongo DB specializes in storing JSON-serialized data. Documents (containing JavaScript-style objects) are stored in collections in a database.

Like an SQL engine, MongoDB uses databases to aggregate data for an application; however, data is not stored in rows in tables, but in collections of objects. The database takes no action to check the contents of objects being added to collections, leaving this task up to the software developer. The result is a smaller, faster database engine.

In contrast, schema-centric databases such as MySQL store data in tables. Such a database provides a rigid, platform independent schema in which tables are defined with a fixed number of columns of fixed data sizes. APIs for numerous languages need only provide communication and error handling capabilities to developers, who must define schemas for their applications with SQL queries. The downside is that a single database server, which may need to serve many applications, will need to validate every write operation with the corresponding table's schema and return an error message on failure. Recently, as the processing power of application servers have increased, these operations, which require many disk operations, have proven to be disproportional performance bottle necks.

Appendix B: Source Code Documents

The following documents contain source code developed for this project.

Document B-1: AC Motor Driver MATLAB Code

The following code was written to simulate the AC Motor speed controller in MATLAB:

```
%AC motor speed controller matlab simulator
%By: Michael Audi

%Function GetMvals accepts a target speed (a percentage of the total rms)
% a resolution (number of calculations), the nominal amplitude & frequency,
% the PWM frequency and the number of cycles to be simulated. The function
% returns the output RMS and the matrix of output values to the motor. The
% function also plots the AC input sine wave, the target sine wave, and the
% output values sent to the motor.

function [RMSout,ACout] = GetMvals(speed,res,nomamp,freq,PWMF,cycles)

targ = ((speed./100)*nomamp); %desired amplitude
res = res*cycles; %total resolution
[lookup,dim,t] = GenLookup(nomamp,freq,PWMF,cycles); %AC input matrix
ACin = lookup; %local variable for AC input matrix
Size = dim; %local variable for size of time matrix
[lookup,dim,t] = GenLookup(targ,freq,PWMF,cycles); %AC target matrix
ACtarg = lookup; %local variable for AC target matrix
Step = round((Size./res)); %step size for output calculations

%variable declarations & initializations
ACout = 0;
RMSout = 0;
count = 1;
i = 1;
P = 2;

%loop until you reach the end of the time matrix
while (P < Size) && (count <= res)
    NP = P+Step; %next position

    if(NP>Size) %check for outside matix dimensions
        NP=Size; %set to top of range
    end
    %sine wave increasing
    if ACtarg(NP)>ACtarg(P)
        %loop until the next closest value in the ac input to the ac targ
        while (ACin(i)<ACtarg(P)) && (i<Size)
            i = i+1;
        end
    %sine wave decreasing
    else
        %loop until the next closest value in the ac input to the ac targ
        while ACin(i)>ACtarg(P)
            i = i+1;
```

```
        end
    end
    %store AC output values
    ACout(count) = ACin(i);
    ACtime(count) = i*(1./PWMF);
    %next position
    count = count+1;
    P = NP;
end

Size = size(ACout,2); %size of AC output matrix

%RMS Calculation rms=sqrt((1/n)*acout(1)^2+acout(2)^2+....acout(n)^2)
for i = 1:Size
    RMSout = RMSout + (ACout(i))^2;
end
RMSout = sqrt(RMSout/Size);

plot(t,ACin,t,ACtarg); %plot input and target sine waves
hold on;
plot(ACtime,ACout,'o') %plot AC output values
hold off;

%Function calculates a matrix of size Frequency (Freq)/PWM Frequency (PWMF)
%The matrix contains the data for equation Amp*sine(w*t) where w = 2*pi*Freq

function [lookup, dim, t] = GenLookup(Amp, Freq, PWMF,cycles)
x = 1./PWMF; %period of PWM
y = 1./Freq; %period of wave
t = 0:x:y*cycles; %time matrix
w = 2*pi*Freq; %radians per second for sine wave
lookup = Amp*sin(w*t); %sine wave
dim = size(t,2); %size of time matrix
```