



Steam Valve Controller

Final Report

F10-19

<i>Client:</i>	Iowa State University
<i>Advisors:</i>	Lee Harker Jason Boyd
<i>Team Leader:</i>	Benjamin Jusufovic
<i>Communication Liaison:</i>	Ben Cao
<i>Web Master:</i>	Curtis Mayberry
<i>System Engineer:</i>	Thinh Luong

4/27/2011

Table of Contents

Executive Summary.....	6
Acknowledgement	7
List of Figures	8
I. Background and Project Definition	13
Problem statement	13
Operating Environment	14
Intended User(s) and Intended Use(s).....	14
Assumptions and Limitations.....	15
Assumptions:.....	15
Limitations:	15
Expected End Product and Other Deliverables.....	16
Controller Box	16
Control Panel.....	16
II. Project Approach and Design.....	17
Approach Used.....	17
Design Objectives.....	19
Functional Requirements.....	19
Non-Functional Requirements.....	20
Design Constraints	20
Technical Approach Considerations and Results	20
Wireless transceiver.....	21
Microcontroller	21
LCD display:	23
Power Supply:	24
NiMH Charger:	25
Temperature sensor:.....	25
Mechanical System	26
Ethernet Transceiver:.....	31
Website Interface:	32

III.	Detailed Design and Implementation	33
	Detailed Design	33
	Control algorithm	33
	Set Point Algorithm	35
	Wireless transceiver	40
	Controller Unit Power System Detailed Design	41
	User Interface Power System Detailed Design	43
	Recharging System	43
	Microcontroller	46
	LCD Detailed Design	48
	Mechanical System	49
	Temperature Sensor	55
	Ethernet Transceiver	58
	Website Interface.....	62
	Implementation	64
	Microcontroller	64
	Wireless Transceiver	65
	LCD	66
	Motor System.....	68
	Temperature Sensor	77
	Ethernet Communication.....	80
	Enclosure.....	84
	Power Supply	84
	Buzzer.....	84
	Push Button.....	85
	Recharging Circuitry	87
	Website Interface.....	87
	Controller Algorithm	90
	PCB Board Integration.....	91
IV.	Testing Plan and Results	94
	Testing Approach Considerations	94
	Testing requirements considerations	94

Phase 1	94
Testing prior to final component selection.....	95
Testing after final component selection with Arduino	95
Testing after final component selection with that Atmel U3A0128.....	96
Exiting testing phase criteria.....	96
Phase 2	97
Security considerations.....	101
Network interface:	101
Wireless transceiver.....	101
Safety considerations.....	101
LCD:	101
Mechanical implementation:	101
Driver Circuit:	102
Temperature sensor:.....	102
Microcontroller:	102
Recommendations Regarding Project Continuation or Modification	103
Testing Results	103
Integration Testing Results	103
Functional Testing Results	105
Hardware Testing Results	107
Hardware Testing Results Continued.....	109
Software Testing Results.....	110
V. Team Information	111
Project Team Information.....	111
Client Information.....	111
Student Team Information.....	111
VI. Budget Information.....	112
Controller Box Budget.....	112
Control Panel Budget	112
Project Costs	113
VII. Closing Summary.....	114
Lessons learned.....	114

PCB Board.....	114
Microcontroller	114
Motor	114
Summary	114
References	115
Appendix A - Testing after final component selection	116
Motor driver test circuit.....	116
Magnetic shaft encoder circuit	117
Appendix B – PCB Schematic	118
Appendix C – Operational Manual.....	123
Basic Operation.....	123
Physical Setup	123
Firmware Setup.....	124
Appendix D – PCB Bill of Materials	125
Appendix E: Poster	129

Executive Summary

Iowa State University buildings built prior to the late 1960's utilize steam valves and radiators to heat rooms. These heating systems link multiple rooms through steam pipes and consist of a single valve that controls the temperature heat output through radiators within each room. The user does not have knowledge of the magnitude of revolutions needed to accommodate a desired temperature within the room and causes over and under heating throughout the rooms. Since these heat systems consist of physical controls, Facilities Planning and Management does not have remote access to the system in order to regulate temperature in times of little to no occupancy. As a result, significant energy is lost during campus night hours and school breaks, consequently increasing energy bills.

In order to solve these issues, our group has proposed to perform a system level integration in which we will design a steam valve controller unit and a user interface unit. The steam valve controller unit consist of a DC motor - used to rotate the steam valve, a microcontroller - used to run the control algorithm, a wireless transceiver - used to communicate to the user interface, and Ethernet connectivity – used to connect to the Iowa State network. The user interface consist of a LCD display – human machine interface, push buttons – used to receive user input, a temperature sensor – used to record the room temperature, a wireless transceiver - used to communicate to the steam valve controller, and a microcontroller - used to control the user interface unit. Through design, we have integrated these components into an efficient temperature control system.

The end product consists of two units: the user interface and the steam valve controller. The user interface is a wall mountable unit similar to a thermostat. It prompts for and accepts temperature values from users. The unit is also used as a means of sensing and recording current room temperature. It communicates the current room temperature and user input temperature values to the steam valve controller unit. The controller unit resembles a box structure and is situated on top of the steam valve. It runs an algorithm based on received temperature values and adjusts the steam valve appropriately through the DC motor. This system allows user friendly temperature control and monitoring.

Acknowledgement

Our team would like to thank Lee Harker and Jason Boyd for their contributions to the project. Not only are they our clients, but they also serve as valuable resources throughout the project. Instead of just providing our team with just a description of the project, they have been proactively working with us to adapt the project to meet their needs and ensure the highest chance of success.

We are thankful for the opportunity to work with such enthusiastic advisors like Lee and Jason, and acknowledge all their contributions to the project.

List of Figures

Figure 1: System Concept Sketch.....	17
Figure 2: System Level Design.....	18
Figure 3: High Level Design of the Controller Box	18
Figure 4: High level Design of the Control Panel	19
Figure 5: Xbee Wireless Transceiver	21
Figure 6: Microcontroller	21
Figure 7: LCD Display.....	23
Figure 8: Power supply transformer	24
Figure 9: Battery charger	25
Figure 10: Temperature Sensor	25
Figure 11: Current Steam Valve System.....	27
Figure 12: Electric Motor considerations.....	27
Figure 13: Motor Driver IC considerations.....	29
Figure 14: PID Control Algorithm	31
Figure 15: Ethernet Transceiver.....	31
Figure 16: Controller Design	34
Figure 17: Controller Difference Equation.....	35
Figure 18: Set point algorithm logic diagram approach 1.....	36
Figure 19: Set point algorithm logic diagram approach 2.....	39
Figure 20: Set point GX algorithm logic diagram	40
Figure 21: Schematic for the wireless transceiver	40
Figure 22: Pin connections for the wireless transceiver	41
Figure 23: Controller unit power supply diagram.....	41
Figure 24: Power supply specifications.....	42
Figure 25: User interface power supply diagram.....	43
Figure 26: User interface battery charging circuitry schematic.....	44
Figure 27: Low battery charge indicator	45
Figure 28: Battery charge circuitry Conceptual Diagram.....	45
Figure 29: Microcontroller Pin Assignment table	47
Figure 30: Microcontroller schematic.....	48
Figure 31: LCD schematic	49
Figure 32: Stationary platform CAD design.....	50
Figure 33: Gear motor specification	50
Figure 34: Shaft encoder specification.....	51
Figure 35: Gear motor specification	51
Figure 36: Mechanical Interface Interaction Diagram	52
Figure 37: Gear motor logical diagram	53
Figure 38: Temperature Error to Degrees Rotation Chart	54

Figure 39: Temperature sensor selection table	55
Figure 40: Temperature sensor pin assignment	56
Figure 41: Temperature sensor schematic and functional diagram	56
Figure 42: Microcontroller ADC for the temperature sensor	57
Figure 43: Ethernet transceiver options table	58
Figure 44: Ethernet transceiver schematic	59
Figure 45: Ethernet transceiver pin connections	60
Figure 46: LAN8700 System Block Diagram	60
Figure 47: MII Ethernet standard pin mapping table	61
Figure 48: Website interface functional diagram	62
Figure 49: Web user interface	63
Figure 50: Web admin interface	63
Figure 51: Microcontroller Schematic.....	64
Figure 52 Request Packet Format	65
Figure 53: Response Packet	65
Figure 54: LCD Schematic.....	66
Figure 55: LCD Screen	67
Figure 56: Motor Components 1.....	68
Figure 57: Motor Components 2.....	68
Figure 58: Motor Final Schematic Design	69
Figure 59: Motor Data Table	70
Figure 60: Gear Motor	71
Figure 61: Motor Encoder Port Final Schematic Design	72
Figure 62: Temperature Data Table	76
Figure 63: Temperature Sensor Schematic.....	77
Figure 64: Output Voltage Vs. Temperature Graph.....	79
Figure 65: Temperature Sensor Oven Test	79
Figure 66: Final Ethernet Schematic Design	80
Figure 67: Microcontroller pin mapping for Ethernet components	81
Figure 68: Ethernet PCB Design	82
Figure 69: Ethernet PCB Board Population	82
Figure 70: Enclosure.....	84
Figure 71: Buzzer Schematic	85
Figure 72: Push Buttons Schematic.....	86
Figure 73: Website Interface Login Page	87
Figure 74: Website Interface Login User Database.....	87
Figure 75: Website Interface Welcome Page.....	88
Figure 76: Website Interface Admin Page	89
Figure 77: Website Interface User 1 Page	89
Figure 78: PCB Board Design.....	91
Figure 79: Controller Box	92
Figure 80: Control PCB Board Population	93

Figure 81: Mechanical interface Test criteria	96
Figure 82: Hardware test plan and criteria	98
Figure 83: Software testing plan and criteria	99
Figure 84: Functional test plan and criteria	100
Figure 85: Functional Testing Results	105
Figure 86: Temperature chamber	107
Figure 87: Temperature Chamber Test Setup.....	107
Figure 88: Hardware Testing Results	107
Figure 89: Hardware Testing Results Continued.....	109
Figure 90: Software Testing Results.....	110
Figure 91: Client Information.....	111
Figure 92: Team Information	111
Figure 93: Controller Box Budget.....	112
Figure 94: Control Panel Budget	112
Figure 95: Project Costs	113
Figure 96: Motor driver test circuit.....	116
Figure 97: Shaft encoder test circuit schematic.....	117
Figure 98: PCB schematic Overview.....	118
Figure 99: PCB schematic Details.....	122
Figure 100: PCB Bill of Materials.....	128

List of Definitions

Alternating current (AC) (hardware) - The movement of electric charge periodically reverses direction.

Analog-to-digital converter (ADC) (hardware) - A device that converts a continuous quantity to a discrete digital number.

Boost Converter (hardware) - used to step up a voltage to a particular level given an input voltage by switching a transistor and using circuit elements to store energy.

Buck Converter (hardware) – A device used to step down a voltage to a particular level given an input voltage by switching a transistor and using circuit elements to store energy.

Complementary metal-oxide-semiconductor (CMOS) (hardware) - A technology for constructing integrated circuits.

Controller Box (hardware) - A module that holds all the electrical components that rotates the steam valve.

Direct current (DC) (hardware) - The unidirectional flow of electric charge.

Emitter-coupled logic (ECL) (hardware) - A logic family that achieves high speed by using an overdriven BJT differential amplifier with single-ended input.

Ethernet (hardware) - A family of frame-based computer networking technologies for local area networks (LANs).

General Purpose Input/Output (GPIO) (hardware) - An interface available on some devices to interface with external devices and peripherals.

IEEE 802.15.4 (hardware and software) - A wireless communication standard under the IEEE 802.15 standards for Wireless Personal Area Networks, currently used as a medium for a wide variety of network protocols including Xbee.

IEEE 802.3 (software) - a working group and a collection of IEEE standards produced by the working group defining the physical layer and data link layer's media access control (MAC) of wired Ethernet. This is generally a local area network technology with some wide area network applications.

LDO (hardware) – Low Dropout Regulator, a device used to achieve a given output voltage given an input voltage by burning energy.

Liquid crystal display (LCD) (hardware) - A thin, flat electronic visual display that uses the light modulating properties of liquid crystals.

Media Independent Interface (MII) - a standard interface used to connect a Fast Ethernet (i.e. 100 Mbit/s) MAC-block to a PHY chip. The MII may be used to connect the MAC to an external PHY via a

pluggable connector (see photo), or to connect a MAC chip to a PHY chip on the same printed circuit board.

NiMH – NiMH is a particular technology used in rechargeable batteries.

Plant (hardware) - The steam valve.

Power Supply Sequencing (hardware) – The start up times of the power system.

Printed Circuit Board (PCB) (hardware) - Used to mechanically support and electrically connect electronic components using conductive pathways, tracks or signal traces.

Proportional–integral–derivative controller (PID controller) (hardware) - A generic control loop feedback mechanism (controller) widely used in industrial control systems.

Reduced Media Independent Interface (RMII) - a standard that addresses the connection of Ethernet physical layer transceivers (PHY) to Ethernet switches. It reduces the number of signals/pins required for connecting to the PHY from 16 (for an MII-compliant interface) to between 6 and 10. RMII is capable of supporting 10 and 100 Mbit/s; gigabit interfaces need a wider interface.

Serial Peripheral Interface Bus (SPI) (hardware and software) - A synchronous serial data link standard named by Motorola that operates in full duplex mode.

Signal Integrity (hardware) – Rise and fall times of signals on signal lines.

Switcher (hardware) - A buck converter used to step down a voltage to a particular level given an input voltage by switching a transistor and using circuit elements to store energy.

Transceiver (hardware) - A device that has both a transmitter and a receiver combined and share common circuitry or a single housing.

Transistor–transistor logic (TTL) (hardware) - A class of digital circuits built from bipolar junction transistors (BJT) and resistors.

Universal asynchronous/synchronous receiver/transmitter (UART/USART) (hardware) - A type of "asynchronous receiver/transmitter", a piece of computer hardware that translates data between parallel and serial forms.

User Interface (hardware and software) - synonymous with Control Panel.

I. Background and Project Definition

Problem statement

Buildings at Iowa State University built prior to the late 1960's utilize steam valves and radiators to heat rooms. The controls of these heating systems consist of steam pipes and simple knobs that allow the user to adjust the flow of steam through the radiator, thus adjusting the temperature.

It is common for one steam valve to control temperature throughout several different rooms. The problem arises from the heat gradient between these rooms, created when the steam valve is adjusted too high or too low. The user does not have knowledge of the magnitude of revolutions needed to accommodate a desired temperature within the various rooms. In result, the user must constantly adjust the valve in order to accommodate a desired temperature, leading to over and under heating of the system of rooms.

Since these valves are spread across remote areas of buildings and consist only of physical controls, Facilities Planning and Management have no control over temperature ranges during times of little to no occupancy of rooms. In result, significant energy is lost during campus night hours and school breaks, consequently increasing energy bills.

Coover, the electrical and computer engineering building, utilizes this steam system. Within Coover, radiators are used to heat a multitude of adjacent rooms and steam valves are used to control temperature. In extreme cases, one steam valve controls the temperature in five different rooms through five radiators. This creates a temperature gradient across the rooms and requires continuous adjustment of the steam valve in order to accommodate the individuals within each room.

The steam pipes run along one side of the room's perimeter, sit approximately a foot off of the ground, and are routed through walls, floors, and ceilings. There exists only one steam valve control for a number of adjacent rooms. From this constraint, it is evident that there will need to be a unit placed on the valve that adjusts the valve to control the temperature in the system of rooms. Since this unit will control temperature conditions in several rooms, a device will be needed in each room that will sense each of the temperatures. The piping route through walls, floors, and ceilings restricts wired communication between temperature sensors and creates a need for a wireless communication between sensor units and the steam valve controller unit.

The most important mechanism of the project is the user interface that allows the user to control the temperature within each room. Since several rooms are governed by one steam valve and multiple individual users, a user interface will be required in each room. In that respect, the steam valve controller will accommodate an averaged input temperature. Because the pipes are low to the ground and are in remote areas, the user interface will be a separate unit from the steam valve controller. Remote override access into the system and temperature control will be provided for Facilities Planning and Management because they will be in charge of controlling room temperature throughout Coover.

Our group has formulated a general solution approach to this problem. In order to rotate the valve, we will utilize a DC motor. The motor will be connected to control circuitry that will run an algorithm based on the temperature in the various rooms and force the valve to adjust appropriately. The control circuitry will utilize wireless communication schemes in order to communicate back and forth between the temperature sensors. The controller unit - DC motor and control circuitry- will need to draw a lot of power in order to turn the valve and will therefore will require an outlet plug in. In order to provide Facilities Planning and Management with override access to set and control temperature, we will integrate an Ethernet port into the steam valve controller unit with access to the Iowa State University network.

In order to receive desired temperature inputs from users, we will utilize a separate user interface wall mountable unit. This unit will consist of a backlit LCD display and several push buttons that allow users to access menus and set desired options. Integrated into this unit will be a temperature sensor that will read the current room temperature and communicate this information to the steam valve controller unit. Another feature integrated into the user interface is a small speaker that will sound when an error or hardware malfunction occurs. Since this is a low power unit and highly digital unit, it will run off of regular AA batteries.

Operating Environment

The end product consists of two different units that will operate in two different environments. These two products will operate in offices and classrooms throughout Coover and ideally throughout various buildings at Iowa State University. The user interface will regularly operate at room temperature and will occasionally experience temperatures ranging from 50 degrees Fahrenheit to 90 degrees Fahrenheit. The unit will be situated on the wall via a mounting device and may experience a drop upon installation or an occasional bump from a passerby. It will be exposed to mildly dusty conditions and operate in an open environment. The unit will experience various forms of humidity and sunshine but will be sheltered from the weather elements such as rain and snow. Because it will be located in a busy environment, it will experience everyday use.

The controller box will be situated directly on top of the steam valve and will thus operate at temperatures ranging from 50 degrees Fahrenheit to 120 degrees Fahrenheit. It will constantly be exposed to highly dusty conditions and will operate in open environments and tight spaces. The unit will experience various forms of humidity and sunshine but will be sheltered from the weather elements such as rain and snow. It may experience the occasional bumps from nearby objects. Because the controller has to maintain the desired temperature, it will experience continuous use.

Intended User(s) and Intended Use(s)

Intended users of our product, the user interface unit, will be Iowa State University students and faculty members. The users will be young adults – early to late twenties, of various sizes, both male and

female, and will have a college education. The majority of use will come from students and professors. They will be tired from work and studies and therefore require a simple, easy to use interface. These individuals will come from varying ethnic backgrounds; therefore reducing the amount of text based commands will benefit usability.

The end product is expected to be used as a means of controlling the temperature to an appropriate level across various rooms by the individuals within them as well as Facilities Planning and Management during times of low to little occupancy. The product is used to help cut heat cost when individuals are not present and maintain a constant temperature when individuals are present. Through implementation, the end product eliminates the possibility of one individual dictating the temperature across various rooms and removes ambiguity from temperature setting. It also gives Facilities Planning and Management the power to set and regulate temperature across various rooms in which they have no current control over.

Assumptions and Limitations

Assumptions:

- Iowa State University students and faculty will use this product.
- This product will not be used outside of the United States.
- This product will be used indoors.
- Users of various ethnic backgrounds, age groups, and genders will use this product.
- This product will be used in rooms of varying sizes, sunlight exposure, and temperature sensitivities.
- The maximum number of simultaneous users for the product will be five.

Limitations:

- The end product will have to communicate wirelessly between components.
- The end product has to have Ethernet connectivity.
- Our group has limited experience with mechanical implementation – product casing and situating.
- Temperature gradient across rooms will cause constant temperature offset.
- There exists only one valve for various rooms.
- Financial budget.
- Plant restrictions (volume of steam/heat output).
- Conflicting Temperature Preferences.
- Part's lead time.
- The controller box must be able to operate on 60 Hz 120 V.
- The user interface must be operable using standard AA batteries.
- Both units must withstand dust accumulation.
- The end product needs to be completed prior to May 2011.
- Prototype testing must occur during the winter months.

Expected End Product and Other Deliverables

Prior to the end of the project in April 2011, our team will deliver one controller box, one control panel, and project design documentation.

Controller Box

The controller box will include the metal platform, the mechanical components, microcontroller, the PCB housing all the electronics, the transceiver, the Ethernet module, and the power supply. The controller box will be designed to fit and securely about the steam valve. There will be a metal bar connecting the DC motor to the steam valve allowing it to turn clockwise and counterclockwise.

The controller box will use a power supply fed by the wall electrical outlet to supply power to the controller box. The PCB board within the controller holds all the major electronics. The PCB will contain a LCD to display status information and product debugging. The transceiver will allow wireless communication between the controller box and control panels. The controller box will use the transceiver to receive information on temperature set point and room temperature measurement.

The Ethernet module will allow communication through the ISU network. The microcontroller will control all the modules on the PCB board. The microcontroller will send and receive information through the Ethernet connection to remotely configure the controller box. Our clients will have a web interface to view and configure the controller box through the Ethernet connection. Lastly, the controller box will have a buzzer that will sound in case the unit malfunctions.

Control Panel

The control panel will include an LCD display, push buttons, microcontroller, temperature sensor, buzzer, transceiver, two rechargeable NiMH AA batteries, and along with rechargeable circuitry. There will be an on board microcontroller that will handle the LCD, temperature sensor, buzzer, and transceiver. The LCD display will output current temperature of the room and the desired temperature set by the user.

The microcontroller will send the user's input data and the temperature sensor's readings to the transceiver which in turn will send that wirelessly to the controller box located on top of the steam valve. The control panel will have a buzzer that will sound in case of malfunction. The control panel will be modularized and multiple control panels can communicate to a single controller box. The range of the control panel to the controller box will be limited by the transceiver. Our clients want the range to be around 300 meters indoors. The two AA batteries will power the control panel. Lastly, the control panel will have an outer casing that allows it to be convenient attached to the wall.

II. Project Approach and Design

Approach Used

The nature of the project is of integrating a variety of components together to create a system that will accomplish our functional and non-functional requirements. The overall design of the system, shown in figure 1, gives an overview of how each sub-system contributes to meeting the requirements of the system. Our project consists of two basic sub-systems, the controller box and the control panel, outlined in figures 3 and 4 respectively. Each of these sub-systems has a number of components which have their own selection and design considerations.

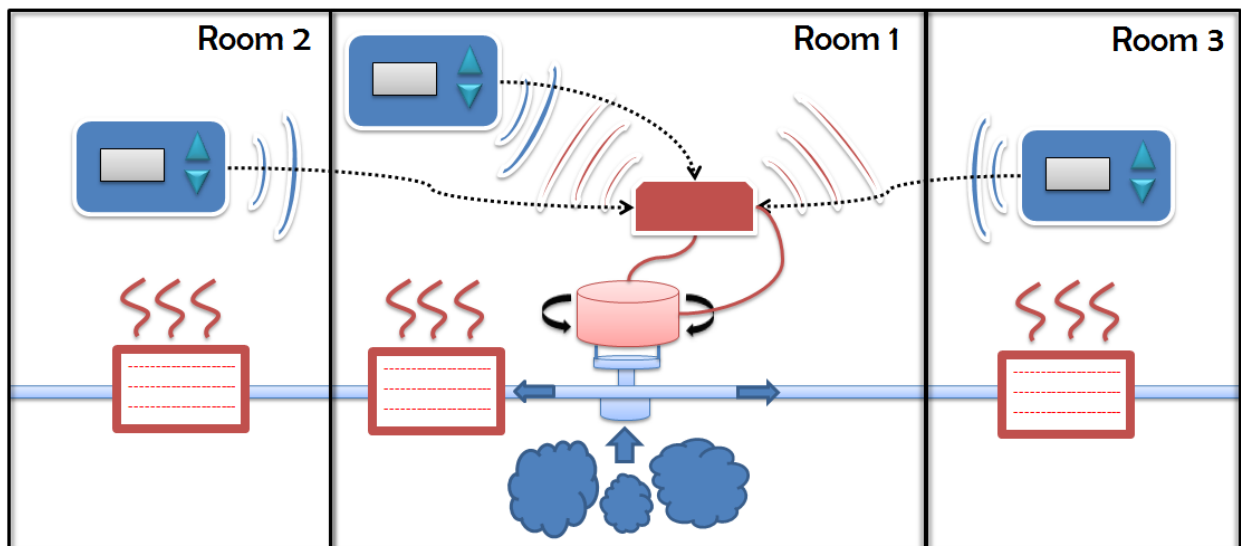


Figure 1: System Concept Sketch

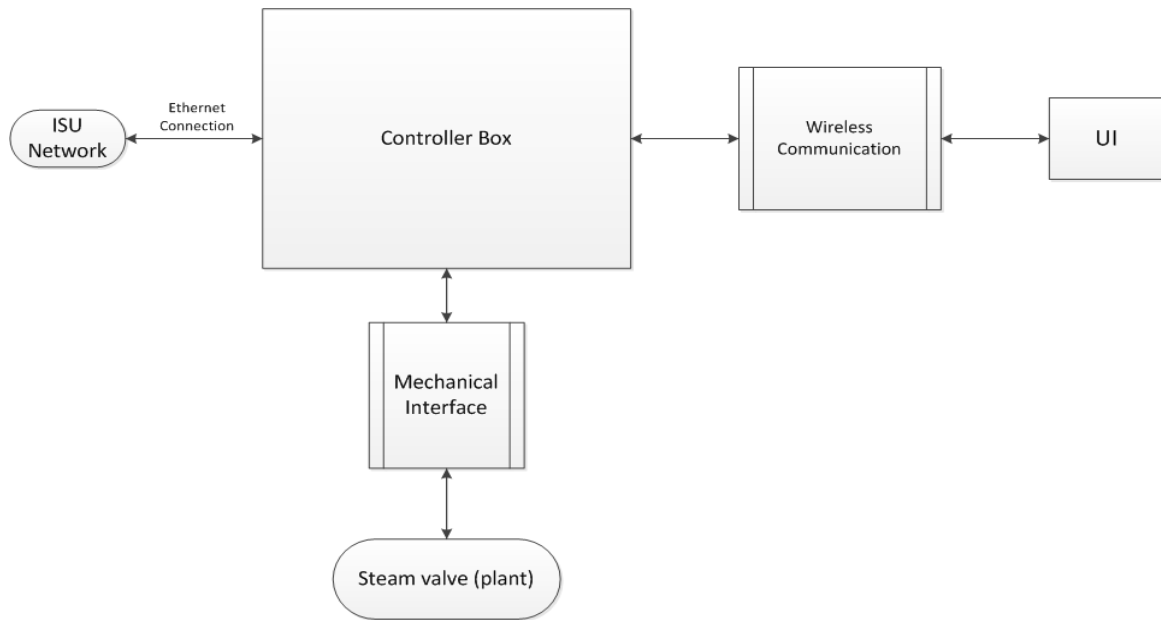


Figure 2: System Level Design

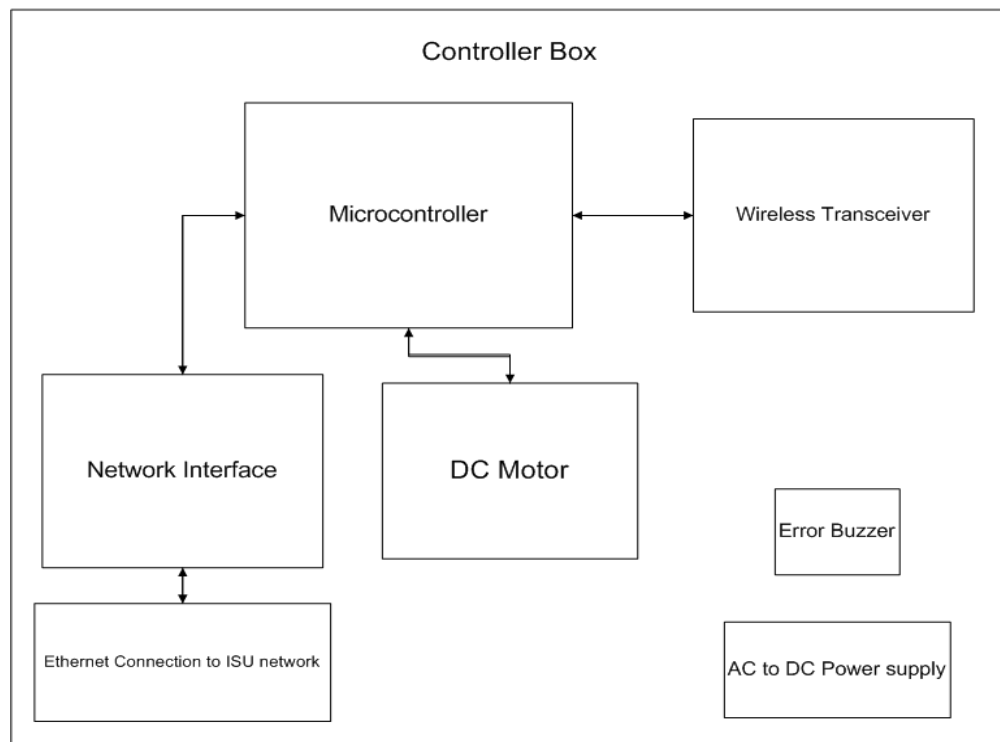


Figure 3: High Level Design of the Controller Box

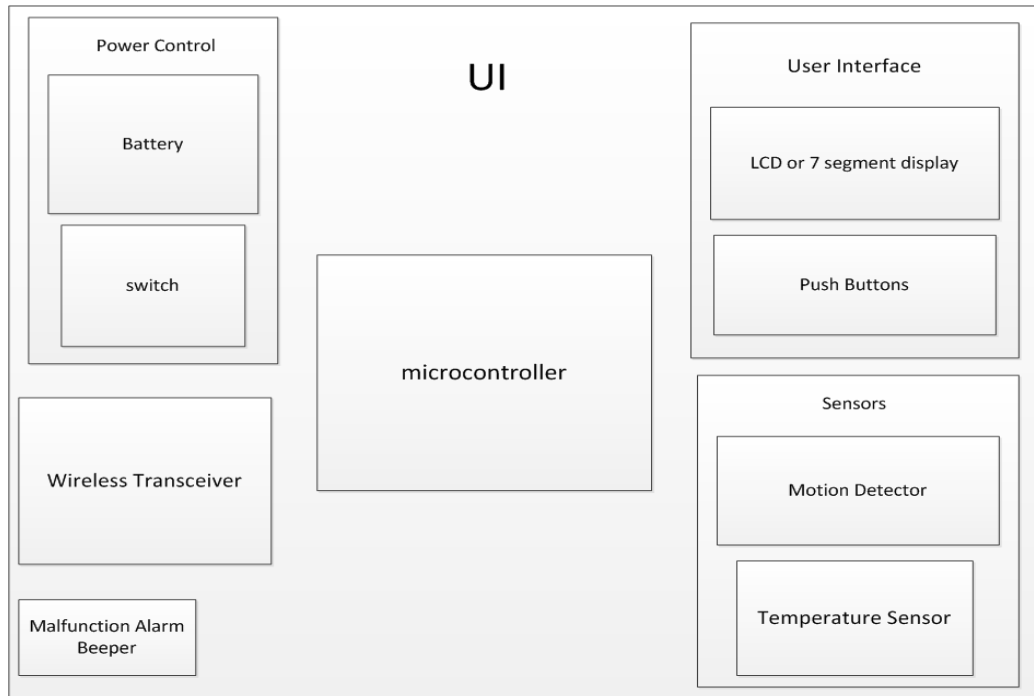


Figure 4: High level Design of the Control Panel

Design Objectives

- The controller must be able to control the temperature in up to 5 adjacent rooms using wireless communication.
- The steam valve controller must be able to communicate through the Iowa State network using Ethernet
- The unit must be capable of being detached from the steam valve without effecting the steam pipes
- Design must be capable of being mass produced and used in a variety of environments
- The interface must be user friendly

Functional Requirements

The project has functional requirements that revolve around the product's intended use as a thermostat. It has to imitate many of the capabilities of a modern heating system while utilizing existing heating methods. The functional requirements of the product include:

1. **Effectively control the temperature in the room:** The most important functional requirement is that the controller must be able to effectively control the temperature by adjusting the attached steam valve appropriately.
2. **Take occupant's preferences:** The unit must have a user interface that allows the occupant to specify a temperature preference.
3. **Multiple room control:** There are often multiple rooms that have their temperature regulated by a single steam valve. The product should be able to take preferences from each occupant to try and find the best possible valve setting.

4. **External control:** The facilities and maintenance personnel need to have a method to set limits on the temperature control.
5. **Removable mechanical interface:** The mechanical interface to the valve must be able to be easily removed. To avoid plumbing complications and to simplify installation and removal the unit must interface with the current valve knob with minimal alterations to existing infrastructure.
6. **Network interface:** The unit will have a network interface that will be able to communicate over the internet. The interface shall be able to configure the controller from a remote computer terminal.

Non-Functional Requirements

We did not receive non-functional requirements from our client. From experience with HVAC controllers, our group has a fundamental understanding of product usability and the importance of aesthetics in a successful product. With this in mind, our group has drafted non-functional requirements that will allow us to design a successful product. The portion that we will consider is the user interface unit. The following lists these requirements in no particular order:

1. Large LCD in order to improve visibility for users.
2. Graphical based LCD in order to ease usability and decrease the need for text based commands.
3. Gray, green, yellow, or blue tint of LCD to increase aesthetic appeal.
4. Large rubber base push buttons in order to ease usability.
5. Translucent push buttons to add aesthetic appeal.
6. Neutral color of user interface casing – white, off white, or gray to offset from the push buttons and LCD.

Design Constraints

1. **Minimum alteration to existing infrastructure:** The facilities planning and management would like the existing infrastructure to remain as unaltered as possible in order to preserve the integrity of the existing steam pipes. Minimum alterations of the valve knob are allowable.
2. **Limited heat output:** The controller of our design is limited by the heat output of the current steam heat distribution systems. This severely limits the ability of our controller to quickly adjust the temperature in the room.
3. **User rationality:** The extended response time of the system may also lead to irrational use by the user. The user may increase the temperature prematurely in an attempt to arrive at the desired temperature.
4. **Limited testing period:** The steam heating system is only active during the winter months. This limits our functional testing to the early spring semester.

Technical Approach Considerations and Results

Each component of the two sub-systems must be selected according to certain technical specifications and considerations. Both subsystems require a wireless transceiver and microcontroller. The other

major components for the controller box include an electric motor and network interface adapter. The other major components for the control panel include the LCD display and temperature sensor.

Wireless transceiver: The wireless transceiver is used to communicate between the controller box and the control panel. A mesh networking structure is used to allow each of the control panels to communicate with the controller box and with each other in an effort to extend the range of the system. Use of the existing Wi-Fi wireless network was considered. However the mesh network architecture eliminates any need for a separate wireless network infrastructure and is considerably more affordable. Other technologies were considered, including Wi-Fi.



Figure 5: Xbee Wireless Transceiver

A standard n network interface is 2-4 times the price of the chosen mesh network protocol, the IEEE 802.15.4 standard. This standard was chosen to ensure compatibility with the largest number of products. Although there are products with proprietary technology that expand the functionality of the wireless transceiver, the standard protocol was elected. The Xbee product by Digi has been selected as a product that meets all of our requirements, including the ease with which it can be integrated with the microcontroller using a simple serial interface, the affordability and the flexibility provided.

Other benefits of using the selected technology include low power consumption that would allow for extended battery life. It also includes the ease of expansion or reduction of the size of the system to adjust to the number of rooms connected to a single steam valve. It also has provisions for reliable packet delivery, such as retrying missed packets and acknowledgement of packets received. However the system has a very limited data rate of only 250 kbps. Although this is significantly lower than other wireless solutions such as Wi-Fi, this is more throughput than we will need in order to transmit basic temperature data and intermittent status messages.

Microcontroller: A microcontroller is needed to provide the control logic for both the controller box and the control panel. The microcontroller selection was based on its capabilities, its cost, and the level of programming difficulty. Using these considerations, our group has selected two different microcontrollers to choose from each capable of controlling and processing information from each of the devices. The following breaks up the capabilities needed for our products to function.



Figure 6: Microcontroller

We will be using one microcontroller to implement the functionality of the controller box as well as the user interface. Through this approach, we can save money by purchasing in bulk. The user interface unit will require a microcontroller with one SPI for the LCD interface, one ADC for the temperature sensor, a UART for the wireless communication, and 5 general purpose input output pins for the 3 push buttons, one speaker, and one potential motion detector. This microcontroller needs to operate on 3.3 V technology or better.

The steam valve controller unit will need to have one USART for the wireless communication, Ethernet port capability, PWM for the motor driver circuitry, and 3 GPIOs for a shaft encoder and a speaker. This microcontroller needs to operate on 3.3 V technology or better. Since this unit will be close to the steam pipes, the microcontroller will need to operate at around 100 degree Fahrenheit temperatures.

Given the above features and capabilities, our group has selected three microcontrollers. The first microcontroller that our group selected is produced by Atmel, model number AT32UC3A0128. There are a few benefits to using this microcontroller. This particular microcontroller has 128 KB of programming memory and operates on 3.3 V technology. It has an USB interface, an Ethernet interface, one seven channel PWM, four USARTs, two SPIs, one Analog-To-Digital converter, and an 16 bit Stereo Audio Bit stream. It meets all of our functional requirements. Also, this microcontroller has large software support libraries online that will help with overall programming. Because Iowa State University uses many Atmel products to teach students about engineering, we can save money by using the existing debuggers and compilers. However, there is a drawback to using this microcontroller. It has 144 pins and we will be using 50 to 60 % of these pins. It wastes some of the functionality but provides freedom to expand on the project during the design process without too much difficulty. Also, because it is a bigger unit, it draws more power than the average microcontroller. Nevertheless, the price is reasonable at 10 dollars per unit.

The other microcontroller that our group selected is produced by NXP Semiconductor, part number LPC1963FBD100, 551. There are also a few benefits to using this microcontroller. This particular microcontroller has 512 KB of programming memory and operates on 3.3 V technology. It has an USB interface, an Ethernet interface, one PWM, four UARTs, CAN, two I2C, one SPI, one Analog-To-Digital converter, and one Digital to Analog converter. It meets all of our functional requirements. Also, this unit has a smaller pin out of 100 pins and consumes less energy. We would use approximately 80 percent of the pins and functionality of this microcontroller. However, there are a few drawbacks to using this microcontroller. This microcontroller does not have large software support libraries online and will take a very long time to program. Because Iowa State University does not use many NXP Semiconductor products to teach students about engineering, we will have to purchase debuggers and compilers. Nevertheless, the price is reasonable at 7 dollars per unit.

Our group has decided to use the Atmel microcontroller. The reason that we decided to use this unit was strongly supported by the fact that it has a large software support library online. Even though we might be wasting functionality, we will have a lot of freedom to design and expand our system functionality in the long run. The microcontroller capabilities and functionality meets our needs for both systems. Since we will not be using some of the pins, the power consumption will be halved as that indicated on the datasheet.

LCD display: The LCD is used as an aid for our human machine interface -to communicate between the controller box and the individual. Several different LCD technologies have been considered. Our group researched LCD numeric, character, and graphic display modules. The LCD numeric display modules are extremely low power units, easy to configure, and relatively inexpensive. However, they are not user friendly, do not have a back light, and are limited to displaying only numbers. The LCD character display modules are modules are low power units, moderately simple to configure, are relatively inexpensive ~ 10 to 15 dollars, and have a back light. However, they are not user friendly, and are limited to displaying only characters. The LCD graphic display modules are modules are low power units, are moderately inexpensive ~ 20 to 22 dollars, are user friendly, have a back light, and are not limited in terms of display. However, they are moderately difficult to configure in terms of software development.



Figure 7: LCD Display

After a group discussion, we decided to use an LCD graphic display module. A super twisted pneumatic LCD was chosen in order to achieve a high contrast ratio and reduce back light intensity, thus reducing power consumption. We chose to use the DOGM128-6 Electronic Assembly family of LCDs because they have low power consumption and they offer more flexibility in graphics programming and tools to help ease the programming process. The DOGM128-6 family includes an integrated LCD driver and uses the SPI communication protocol, allowing for easy integration with a variety of microprocessors.

Another factor in our decision was the size of the display. Many of our users are weary students and faculty members. We needed an LCD that allowed for large easy to use text and graphics. Most of our electrical components function on 3.3 V technologies. The DOGM128-6 was one of the few LCDs with an appropriate size with a capacity to function with 3.3 V. Our group had to find a display that would be able to operate under varying temperature conditions. This particular LCD met our standards for operating temperature range between -20 degrees Celsius and 70 degrees Celsius.

The final consideration was availability. According to Mouser Electronics, this is one of the most widely available LCDs with a large stock. Given the size, quality, and availability, the estimated LCD cost is around 15 dollars. The DOGM128-6 family has LCDs that range from 4 to 18 dollars based on various backlighting and color schemes. This allows us to create an extremely flexible and cost effective product.

Power Supply: The power supply is used to power both the controller box as well as the user interface unit. Several different power supply technologies and designs were considered in our group's design process. The main portion of the design will consist of a AC/DC converter that steps the voltage from 120 V 60 Hz down to 12 V DC. The rest of the design will consist of the common circuit chip sets including a switcher to convert 12 V to 5 V and an LDO to drop from 5 V to 3 V. In determining the power supply technology selection, our group considered several different options.



Figure 8: Power supply transformer

Our first option was to purchase a 120 V 60 Hz to 12 V DC wall adapter and use electronic components to route and regulate the appropriate power sections of the board. Through this approach, we will already have a fairly clean 12 V DC to use to power the board. We will not have to implement any other complex circuitry. In selecting the adapter, we can select the appropriate power ratings to accommodate our system. These adapters are fairly inexpensive and are easy to implement, saving costs on expensive circuit components.

Our second approach was to purchase a 120 V 60 Hz to 20 V AC wall adapter and use a buck converter to create a stable 12 V signal to power the system. Through this approach, our team could save around 6 to 8 dollars on the adapter. However, there are a few detriments to this approach. The main component in our system is a high power motor that has the potential to draw 2 A. Due to this fact, we would lose money designing the buck converter due to expensive high power rated components. Also, the signal that we would achieve might not be as clean as the AD/DC 12 V signal.

Our third approach was to purchase an outlet plug in and design the circuitry to convert from 120 V 60 Hz to 12 V DC. There are several issues with this approach. Because our system has the potential to draw a large amount of current, the components used would have to be large and have large power ratings. It would increase the area of the PCB design by at least half the amount. This portion of circuitry would also give off a lot of heat and would require heat sinks and might become a fire hazard given that it will operate around a fairly hot steam valve. The only benefit of this system approach would be a cost savings of approximately 10 dollars.

After discussing with the group members, we have decided to implement the 120 V 60 Hz to 12 V DC power system because of all of the benefits. Through this approach, we will save money because we will not require any extra components to perform AC to DC conversion. We will also save PCB area by removing much of the bulky power section. This will increase the size of the outlet plug but will allow for a better quality of signal. Through this approach, we will already have a fairly clean 12 V DC to use to power the board. Because we will purchase the adapter, we have freedom to select the appropriate power ratings to accommodate our system. These adapters are fairly inexpensive -20 dollars- and are easy to implement. This approach is slightly more expensive – 5 dollars – but will increase the quality of our design by allowing for a more reliable system.

NiMH Charger: The NiMH charger is used to re-charge the power supply of the user interface, two AA batteries in series. Several different NiMH charger technologies were considered in our group's design process. The first is a common integrated recharging circuit called the bq2002C. It has very fast



Figure 9: Battery charger

charging capabilities, it has direct LED outputs to display charging status, it has charge termination by voltage, temperature, and time, it is a low power unit, and it is relatively inexpensive, about 2 dollars. The only detriment to this technology is that the unit will not indicate when the batteries are low. Another popular NiMH charging technology that we considered is the bq24401 charging unit. It has very fast charging capabilities, it has direct LED outputs to display charging status, it has charge termination by voltage, temperature, and time, it is a low power unit, and it has a safety check implementation that can check for damaged battery units. However, this technology also has a few detriments. The unit will not indicate when the batteries are low, it costs around 4 dollars, and it needs to be programmed.

After much discussion, our group chose to use the bq2002C because it is half the price of the bq24401 and it does not require programming. We did not want to spend more of our client's money on programmers. The bq2002C has all of the technology and capabilities that we need to perform a battery charge. Given that it cannot detect a low battery voltage, we will have to incorporate a solution in our design that tells the individual when to use the charger.

Temperature sensor: The temperature sensor is a mandatory component for the Steam Heat Controller prototype design. Below are some of the technical selection qualifications for the temperatures sensor:



Figure 10: Temperature Sensor

1. The team has decided that the temperature sensor should not be embedded on the microcontroller. This will reduce the maintenance cost since the entire microcontroller will need to be replaced if a malfunction or damage occurs to the embedded temperature sensor. By using an independent temperature sensor also eliminates possible reading errors caused by the heat produced by the microcontroller or other components for the prototype.
2. The team has decided to use a temperature sensor with a voltage output over digital output. Price is an important factor that needs to be considered when dealing with the temperature sensor. Even though temperature sensors with digital output type have wider sensing temperature range and more accuracy, its price compared to a temperature sensor with voltage output is significantly higher.

Functional Requirements

The functional requirement for the temperature sensor is as follows: be able to accurately measure the temperatures for the steam heat controlled rooms within plus or minus two Fahrenheit.

Design Constraints

- **Temperature range:** 32 degrees Fahrenheit to 100 degrees Fahrenheit. This range is the potential temperature within a steam heat controlled room.
- **Accuracy:** within ± 2 degrees Fahrenheit of the real temperature
- **Reliability:** popular user ratings (over 2500 users)
- **Availability:** readily available from its distributors (20,000 units in stock)
- **Price:** \$1-\$2

Mechanical System: There are two distinct systems that need to be analyzed and evaluated: the stationary platform and gear motor system.

Stationary platform technical approach considerations and results

Due to limitation of our team's mechanical aptitude and experiences, we worked with our clients to develop the stationary platform to house the controller box on top of the steam valve. In assessing the physical constraints surrounding the steam valve, we identified the following characteristics that were required of our stationary platform.

- Clamps will be needed to attach the stationary platform to the steam pipes. This will ensure easy attachment and detachment of the whole controller box unit.
- Extrusion aluminum will be used to form the skeleton of the stationary platform. Our clients requested the extrusion aluminum since their expertise and experience have been developed using this material.
- The mounting plate for holding the gear motor will position the gear motor vertically on top of the steam valve. This configuration will reduce additional work on implementing 90 degree angle connection between the gear motor and the steam valve. The resulting implication would be that our selection of the gear motor will be limited to gear motors that allow face mount configuration.
- The front side of the finished stationary platform should allow modifications to allow mounting the controller box unit. Due to the space constraint that the steam valve has on its backside, sides, and top, we determined that placing the controller box unit attached to the front of the stationary platform would provide flexibility for the whole unit to address different placement of the steam valve across the Coover building.

These observations were made by viewing the current steam valve placement in room 1307 of Coover. The following images were taken from that room.



Figure 11: Current Steam Valve System

With these constraints and characteristics, we worked with our clients to craft the detailed design of the final stationary platform.

Gear motor system technical approach considerations and results

The gear motor system consists of three parts: the gear motor, the shaft encoder, and the motor driver IC.

Gear motor

During our research and selection of an appropriate gear motor, we developed the following selection requirements for the gear motor.

Electric Motor	
Type of motor	DC
Operating Supply Voltage	12-24V
Torque	0.95 N m
Speed	6 rpm
Availability	5-10 years out
Functionality	CMOS/TTL compatible shaft encoder
Safety	shaft encoder & motor driver circuit
Cost	prioritize functionalities over low cost

Figure 12: Electric Motor considerations

In addition to these requirements, we had three alternative solutions for resolving our functionality need in requiring a shaft encoder. These three alternative solutions include:

Gear motor with custom shaft encoder

We would design a custom shaft encoder to complement a gear motor that lacked an integrated shaft encoder. Our approach would be to attach a saw tooth belt that would connect the shaft of the gear motor with an external shaft encoder.

The advantages of this approach include increased variety of gear motor selection, potential lower cost for selecting a shaft encoder, and availability of both gear motor and shaft encoder would remain stable for the coming years.

The disadvantages in using this approach include increase opportunities for mechanical failure due to additional work to integrate the gear motor and custom shaft encoder. Other disadvantages include increased complexity for maintain and repair, having two separate units would make it more difficult to attach or detach the units. Lastly, the time required to research, plan, implement, and evaluate the combination would endanger our project schedule.

Gear motor with attachable shaft encoder

We would purchase a gear motor capable of having an attachable shaft encoder. The addition requirement of this approach would be selecting a gear motor with a back side shaft that an external shaft encoder could fit onto.

The advantages of this approach would be a simpler integration of gear motor and attachable shaft encoder, easier maintenance and reparability of the individual units, and provide possible upgrade of either gear motor or shaft encoder.

The disadvantages of this approach include limited variety of gear motor with exposed back shaft for attachable shaft encoder, potentially higher cost, and limited variety of attach shaft encoder. Our initial research for this approach suggested low variety in component selection and would endanger the capability of our solution.

Gear motor with integrated shaft encoder

This approach requires purchasing a gear motor that has an integrated shaft encoder. The manufacturer of the gear motor would be responsible for integrating the functionality of the shaft encoder and ensuring the mechanical stability of the integrated product.

The advantages in using this approach include purchasing an integrated solution that would allow us to bypass mechanical work, simplification of our design, time savings, and lower cost savings.

The disadvantages in using the approach include limited variety of gear motors with integrated shaft encoder and high cost of replacement should one part of the integrated product fail. The initial research in the approach revealed limited selection of capable gear motors. However, the potential time savings and reduce integration complexity pushed us to consider this approach.

Selected approach

Comparing the three approaches for designing the gear motor with shaft encoder capabilities, we decided to go with the integrated solution approach. We were aware of the limited variety of selection and high cost of replacement. However, the potential time savings, cost savings, and reduced design complexity were critical advantages we needed to maintain our schedule and budget.

We were also attracted to the integrated from our testing of an available gear motor with integrated shaft encoder. The integrated solution sped up our software testing in controlling the motor and simplified the necessary circuitries.

Once we selected our technical approach to the gear motor, we had to identify the requirements for the performance of the shaft encoder.

Shaft encoder

From our chosen approach for a gear motor with integrate shaft encoder, we were now primarily concern with accuracy and technology of the integrate shaft encoder.

Shaft encoder accuracy

Upon discussion with our clients about the accuracy of the steam valve rotation, they want to see an accuracy of ± 5 degrees. The constraint will limit the selection in our search for the gear motor with integrated shaft encoder.

Shaft encoder technology

When looking at the available shaft encoder technologies, we identified two prevalent approaches: optical and magnetic.

Optical shaft encoder

Optical shaft encoders are lower cost, provide higher resolution, and are the dominant technology used in shaft encoders. However, Optical shaft encoders are prone to seal failures, mechanical shattering, and bearing failures.

Magnetic shaft encoder

Magnetic shaft encoders are less prone to seal failures and shattering. However, higher cost and lower resolution are disadvantages of using magnetic shaft encoder.

Our analysis of the two shaft encoder technology concluded that reliability is higher priority than cost and resolution. Therefore, we decided to pursue a magnetic shaft encoder to accompany our gear motor.

Once we defined the necessary shaft encoder accuracy requirement and agreed upon using magnetic shaft encoder, we proceeded to define our approach for the motor driver IC.

Motor driver IC

During our research and selection of an appropriate motor driver IC, we developed the following selection requirements, some of which were dependent upon the gear motor specification.

Motor Driver IC	
Operating Supply Voltage	dependent on selected gear motor
Accepted Input Standard	CMOS/TTL
Current Output	dependent on selected gear motor
Availability	5-10 years out
Safety	over current protection & over temperature protection
Cost	prioritize functionalities over low cost

Figure 13: Motor Driver IC considerations

With these initial requirements, we identified two groups of motor driver IC that are available on the market: application specific and general application.

Application specific

Under the application specific group, we observed functionalities and features that were specific to motor control. These ICs are well designed and documented for motor control.

The advantages of this approach include specialized enhancements designed for motor control, higher quality to handle extreme environments, and appropriate documentation on implementing motor control.

The disadvantages of this approach include high cost, over performance, and limited variety of selection.

General application

Under the general application group, we observed available ICs that had functionalities that would be modified to implement motor control. The ICs are designed for a multitude of application, but lack proper documentation for motor control.

The advantages of this approach include low cost, high variety of selection, and flexibility of implementation.

The disadvantages of this approach include limited operating environment when compared to application specific and additional design necessary to implement motor control.

Selected approach

Evaluating the two groups of available ICs for motor control implementation, we discussed our reasons for selecting ICs available under general application with our clients. General application ICs deliver the best cost effective while providing appropriate performance and functionalities. We confirmed with our clients that there were available ICs that could easily provide motor control and at low cost.

Once we had selected the approaches for the gear motor and motor driver IC, we proceeding to identify the final components selection and confirmed their ability to meet our design needs.

Control algorithm

The control algorithm used to control the valve is a PID feedback control algorithm as outlined in figure 10. This algorithm uses feedback to control the heat output of the heating system and regulate the output according to the set point. This feedback is provided by the control panel's temperature sensor. The input from 3 separate temperature sensors will be used in a weighted average to provide the feedback signal. Similarly the set point will be a weighted average of each user's preference. The algorithm is a well-established standard control algorithm that is often used in industrial control processes. The PID's controller must be tuned in order to adjust the response of the system. There must also be some mechanism to simplify the tuning process. There will be a method to profile a room in order to simplify the tuning process and potentially improve the efficiency of the temperature control using the added information. The PID controller will undergo extensive testing and revision as necessary.

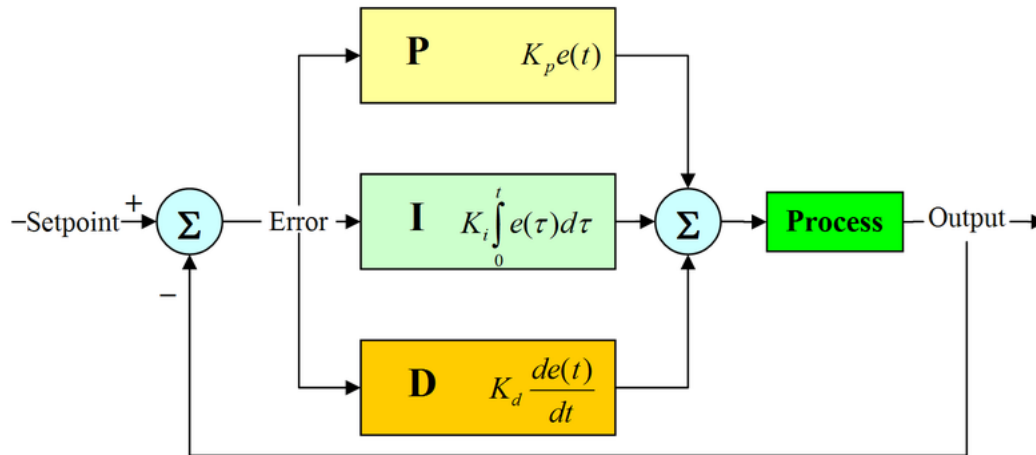


Figure 14: PID Control Algorithm

In order to calculate the set point for the controller an average of each room's preference must be taken. The set point needs to be designed as a weighted average to account for the differences between rooms.

Ethernet Transceiver: The Ethernet transceiver is a mandatory component for the Steam Heat Controller prototype design. Below are some of the technical selection qualifications for the Ethernet transceiver:



Figure 15: Ethernet Transceiver

1. The Ethernet transceiver needs to be able to easily communicate with the Steam Heat Controller's microcontroller. The microcontroller the team has picked supports both the MII and RMII interface. This means that the Ethernet transceiver needs to be able to transmit and receive data across either the MII or RMII interface.
2. The Ethernet transceiver needs to be fully compliant with the IEEE 802.3 specification, which is supported by the Steam Heat Controller's microcontroller and also the most widely used standards for the physical layer and the data link layer's media access control of wired Ethernet.

Functional Requirements

The functional requirements for the Ethernet transceiver are as follows: be able to reliably send data from the Steam Heat Controller's microcontroller to the Iowa State University's network; be able to reliably receive data from the Iowa State University's network for the Steam Heat Controller's microcontroller.

Design Constraints

- **Operational Temperature range:** The operational temperature range for the Ethernet transceiver needs to work between 32 degrees Fahrenheit and 100 degrees Fahrenheit. This range is the potential temperature within a steam heat controlled room.
- **Reliability:** popular user ratings (over 2500 users)
- **Availability:** readily available from its distributors (5,000 units in stock)
- **Price:** \$2-\$5

Website Interface: The website interface is a mandatory component for the Steam Heat Controller prototype design. Below are some of the technical selection qualifications for the website interface:

1. The website interface backend should support the platform given by the Iowa State University, which could be either Windows or Linux based server environments
2. The website interface can be done using either PHP and or Java. The advantage for both are that they are free technologies and readily available for download on the web and both contains full API support. Java is more scalable than PHP and can use JDBC to talk with MySQL database while PHP is more widely used in for web development. PHP will be used for web authentication while Java/JDBC will be used to interact with MySQL database.
3. Apache, MySQL, or Tomcat will be used for storing data from received from the Steam Heat Controller. MySQL will be used since one team member already has knowledge about MySQL thus reduces the learning curve.

Functional Requirements

The functional requirements for the website interface are as follows: be able to reliably and securely read data from the Steam Heat Controller; to be able to reliably and securely send data to the Steam Heat Controller for adjusting the steam heat valve; to be able analyze data sent from the Steam Heat Controller; to be able to have secure login for admin and users; to be able to take user input for temperature preferences.

Design Constraints

- **Server availability:** need to use the server provided by Iowa State University
- **Server space:** the limit for the server space will be capped at 1000 MB
- **Network security:** needs to securely transmit and receive data from and to the Steam Heat Controller

III. Detailed Design and Implementation

Detailed Design

Control algorithm

The PID control algorithm has a number of components. The main component is a standard PID controller that takes in one set point and gives an output that will be read by the DC motor control logic for a PWM signal to be generated. However, since the controller must support multiple rooms, there is also an averaging and logic algorithm for both the temperature sensor feedback and calculating the set point. The temperature sensor feedback is a simple average of the input temperatures. This is illustrated in the figure below:

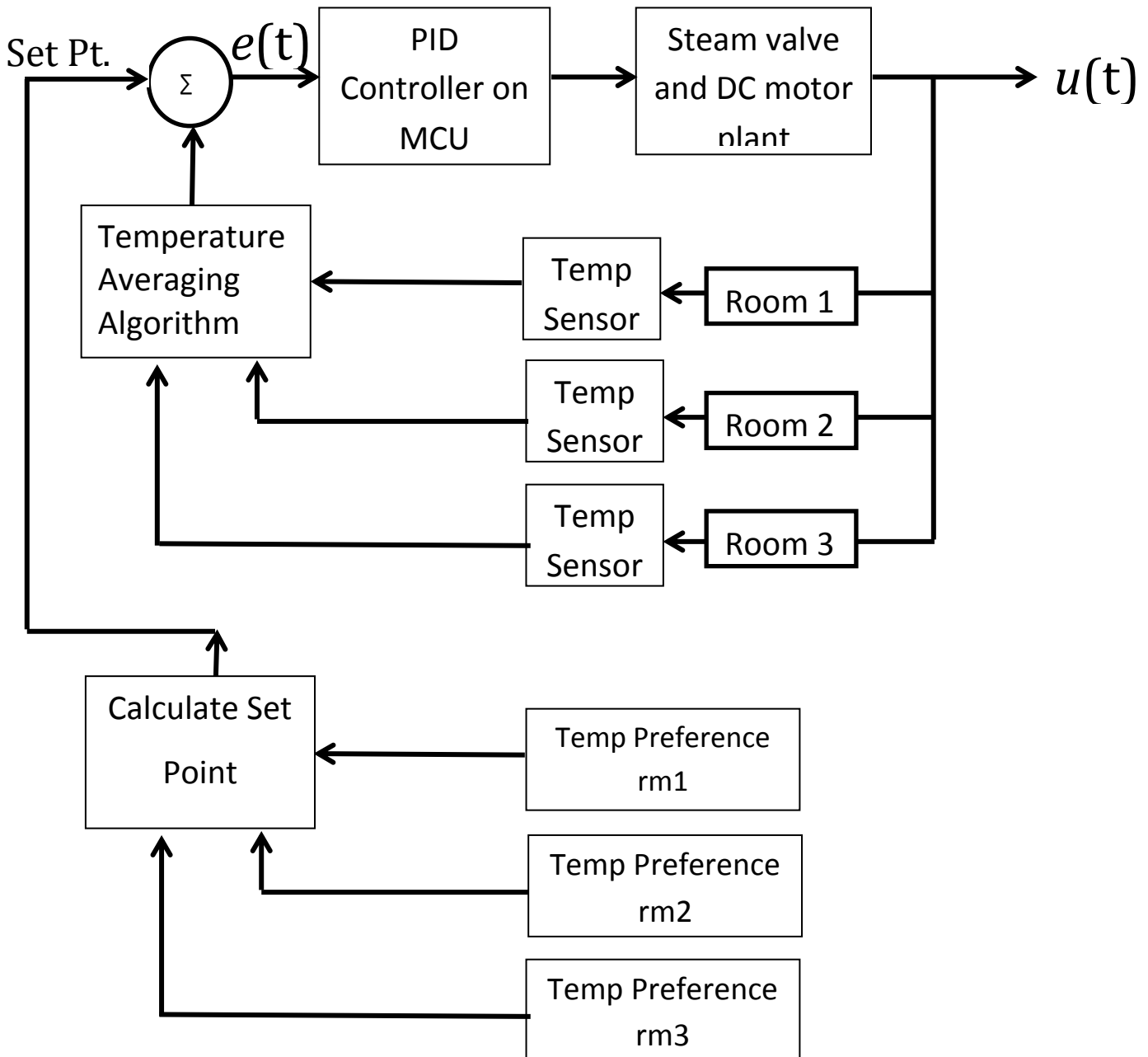


Figure 16: Controller Design

The PID control algorithm must be transformed from a continuous to a discrete algorithm in order to implement it on the microcontroller. The discrete transformation used was the Euler's approximation, which was chosen over the Tustin or bilinear transformation for its relative simplicity. The transfer function of the algorithm is given as shown in figure x, along with the difference equation that will be used to implement the algorithm in software.

$$s = \frac{1 - z^{-1}}{T_s} \quad T_s = \text{Sampling Period}$$

$$H(s) = \frac{u(s)}{e(s)} = k_p + \frac{k_I}{s} + k_D s$$

$$H(z) = k_p + \frac{k_I}{\frac{1-z^{-1}}{T_s}} + k_D \frac{1-z^{-1}}{T_s}$$

$$H(z) = k_p + \frac{k_I T_s}{1-z^{-1}} + \frac{k_D}{T_s} (1-z^{-1})$$

$$u(z) = u(z)z^{-1} + k_p(1-z^{-1})e(z) + k_I T_s e(z) + \frac{k_D}{T_s} (1-z^{-1})^2 e(z)$$

$$u(t) = u(t-1) + k_p e(t) - k_p e(t-1) + k_I T_s e(t) + \frac{k_D}{T_s} (e(t) - 2e(t-1) + e(t-2))$$

Figure 17: Controller Difference Equation

Set Point Algorithm

Approach 1

In this first approach, we make the assumption that the user will always enter a temperature between 60 and 80 degrees. We can ensure this condition by placing a lower bound of 60 and an upper bound of 80 on the microcontroller. We also make the assumption that if the temperature in room one changes by X degrees, it will also change by X degrees in rooms two and three. In order to explain our approach, several variables need to be defined:

Crmax = Room with fastest cooling rate

Crmin = Room with slowest cooling rate

TprefX = Temperature preference in room X

This approach follows a simple averaging scheme. The idea is to achieve a desired temperature in rooms 1, 2, and 3. Because all of the rooms are governed by the same steam valve and there exists a heat gradient between rooms, accommodating the exact desired temperatures within each room will be very difficult and nearly impossible. In this approach, we attempt to equally satisfy the temperature preferences through an average. Throughout the process, we will govern each individual room's temperature to ensure it is within the upper and lower bound limits.

To begin, we will profile the rooms and determine the room with the fastest cooling rate, Crmax, and the room with the slowest cooling rate, Crmin. Once determined, we will average the temperature preferences, TprefX, from all of the rooms and obtain a value between 60 and 80 degrees. We will then use this average temperature to accommodate the room with the fastest cooling rate while profiling the temperature within the other rooms. Since this room cools faster than the others, the one with the

slowest cooling rate, Cr_{min} , will heat up faster. This characteristic can potentially cause the room, Cr_{min} , to heat to an uncomfortable temperature. In the case that the temperature within Cr_{min} exceeds the value of the average temperature by an X amount of degrees, we will change our control to accommodate the averaged temperature within Cr_{min} . Eventually, someone will change their temperature preference and we will have to accommodate a new average or the room with the fastest cooling rate, Cr_{max} , will fall below the value of the averaged temperature by an X amount of degrees, in which case we would switch our control to accommodate the averaged temperature within Cr_{max} .

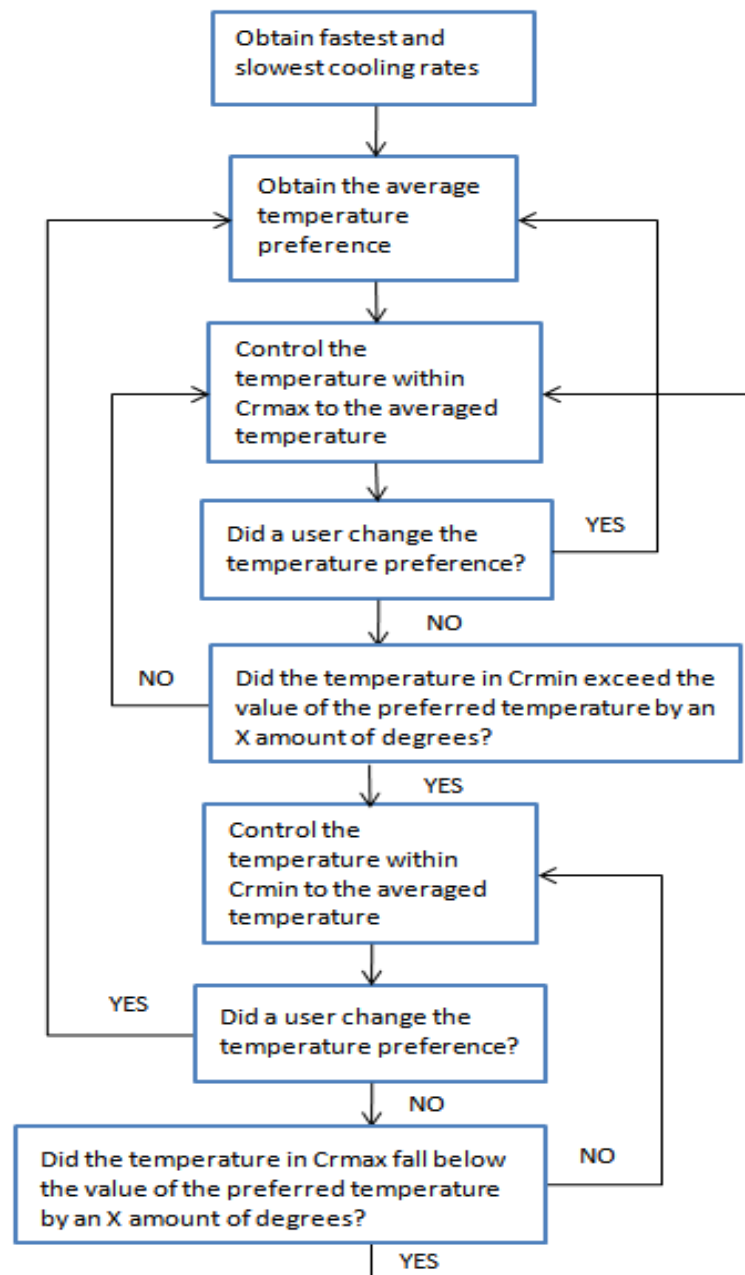


Figure 18: Set point algorithm logic diagram approach 1

Approach 2

In this second approach, we make the same assumption that the user will always enter a temperature between 60 and 80 degrees. We can ensure this condition by placing a lower bound of 60 and an upper bound of 80 on the microcontroller. We also make the assumption that if the temperature in room one changes by X degrees, it will also change by X degrees in rooms two and three. In order to explain our approach, several variables need to be defined:

Cr_{max} = Room with fastest cooling rate

Cr_{min} = Room with slowest cooling rate

T_{prefX} = Temperature preference in room X

T_{curX} = Current temperature in room X

$\Delta 12 = T_{cur2} - T_{cur1}$

$\Delta 23 = T_{cur3} - T_{cur2}$

This approach follows a logic scheme in determining temperature set point and uses temperature frequency values to accommodate a certain temperature. The idea is still to achieve a desired temperature in rooms 1, 2, and 3. In this approach, we attempt to satisfy the preference temperature by following a logical pattern and accommodating different rooms at different times. Through this approach, some individuals will receive their desired temperatures a portion of the time and the remaining individuals will receive their desired temperatures the remaining amount of time. As before, we will govern each individual room's temperature to ensure it is within the upper and lower bound limits.

To begin, we will profile the rooms and determine the room with the fastest cooling rate, Cr_{max} , and the room with the slowest cooling rate, Cr_{min} . Afterward, we will determine the temperature differences between rooms one and two, $\Delta 12$, as well as rooms two and three, $\Delta 23$. Once we know the temperature differences between rooms, we also know a range of temperatures that we can accommodate in a certain room without violating the lower and upper limits of room temperature in the adjacent rooms. Using these ideas, we formulate the following equation assuming a three room system:

$$\begin{aligned} t_{setmax} &= T_{max} + (T_{pref1} - T_{cur1}) * G1 + (T_{pref2} - T_{cur2}) * G2 + (T_{pref3} - T_{cur3}) * G3 \\ t_{setmin} &= T_{min} + (T_{pref1} - T_{cur1}) * G1 + (T_{pref2} - T_{cur2}) * G2 + (T_{pref3} - T_{cur3}) * G3 \end{aligned}$$

The above equation helps us determine a temperature set point, t_{setmax} or t_{setmin} , which the system needs to accommodate in either Cr_{max} or Cr_{min} depending on the state of the system. T_{max}/T_{min} is the temperature value in Cr_{min}/Cr_{max} at the time of setting or changing the value of t_{set} . The G signals

are Boolean value signals determined by a simple logic equation. Only one of the G signals can be high at one time – much like the output of a decoder.

The idea here is to obtain a user's desired temperature T_{prefX} and determine the difference between T_{prefX} and T_{curX} . This gives us the amount of temperature the system will change, in all adjacent rooms, if this temperature is accommodated. Now, we use the logic governing G1 to determine whether accommodating this temperature will violate the upper and lower temperature limits within any of the adjacent rooms by considering $T_{prefX} - T_{curX}$ as well as $\Delta 12$ and $\Delta 23$. If so, we set that value of GX to zero, meaning that we throw that preference out. If not, we move to the next portion of the logic algorithm, GX, and determine whether governing a desired temperature in one room will satisfy the desired temperature within adjacent rooms with $\pm Y$ degrees of error calculated from the temperature gradients $\Delta 12$ and $\Delta 23$. Whichever room's preference satisfies the most rooms, the largest frequency temperature value determined by offsetting the desired temperature by $\Delta 12$ and $\Delta 23$, we decide to accommodate the temperature preference for that particular room. In terms of GX, we set that value to one and all others to zero. Now, to accommodate the appropriate temperature, we must offset T_{curX} within Cr_{max} or Cr_{min} at the time of t_{set} by the difference of $T_{prefX} - T_{curX}$, where X is the room with a preference that satisfies the largest number of rooms, and accommodate the t_{set} temperature within Cr_{max} or Cr_{min} depending on the state of Cr_{max} and Cr_{min} .

The reason for referring to the states of Cr_{max} and Cr_{min} is because one room cools faster than the others and will always have the lowest temperature while the room with the slowest cooling rate, Cr_{min} , will heat up faster. This characteristic can potentially cause the room, Cr_{min} , to heat to an uncomfortable temperature. When we start our system, we will use the above equation for t_{setmax} , to accommodate Cr_{max} for the appropriate change in temperature while profiling the temperature within the other rooms. In the case that the temperature within Cr_{min} exceeds the value of the preferred temperature by an X amount of degrees, we will change our control to accommodate the preferred temperature within Cr_{min} based on the value of t_{setmin} . Eventually, someone will change their temperature preference and we will have to accommodate a new t_{setmax}/t_{setmin} or the room with the fastest cooling rate, Cr_{max} , will fall below the value of the preferred temperature by an X amount of degrees, in which case we would switch our control to accommodate the appropriate change in temperature within Cr_{max} to t_{setmax} .

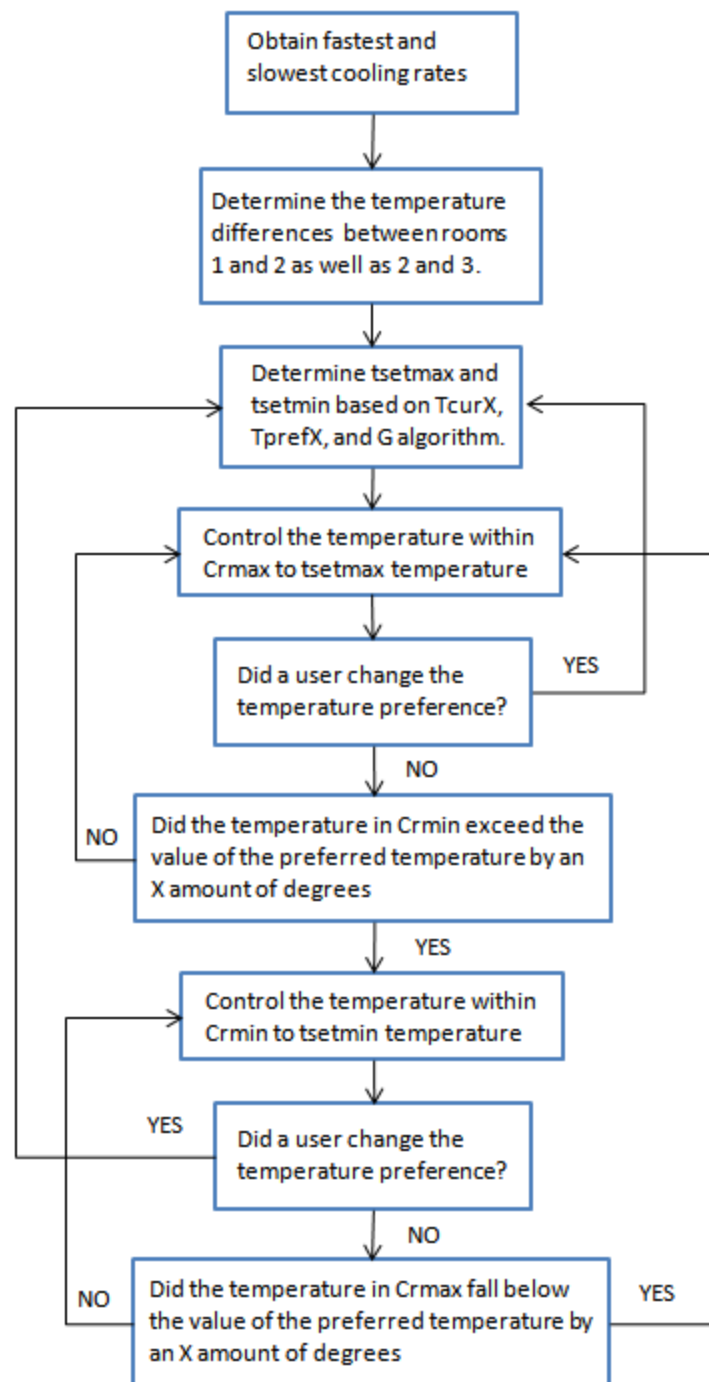


Figure 19: Set point algorithm logic diagram approach 2

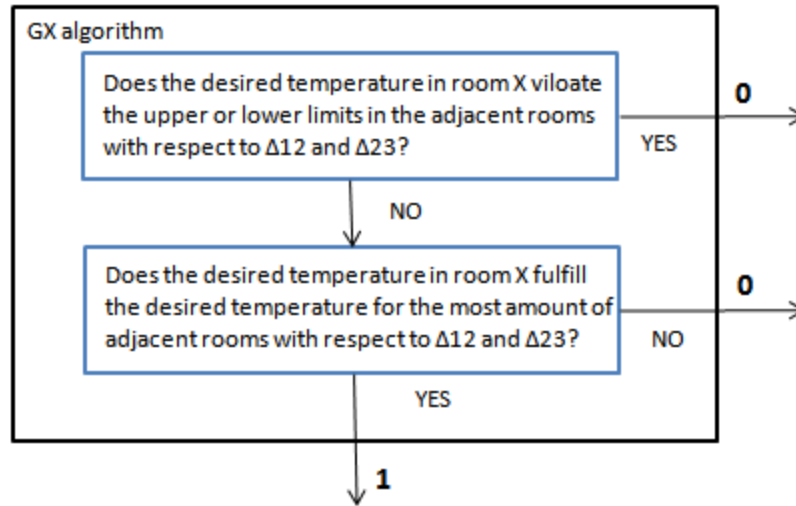


Figure 20: Set point GX algorithm logic diagram

Wireless transceiver

The wireless transceiver will be attached to the PCB using a socket for easy removal. This will allow the unit to be easily removed and programmed externally. The Xbee can then be programmed using a GUI provided by the manufacturer. The Xbee will need to have the PAN ID, channel number, unit address, and destination address set along with a number of other settings including the baud rate and turning on the API frame mode. The API frame mode was chosen over the transparent mode to ensure reliable operation.

Once the wireless transceiver has been programmed it will be inserted into its socket on the PCB. The physical connections of the transceiver are shown in the figures below.

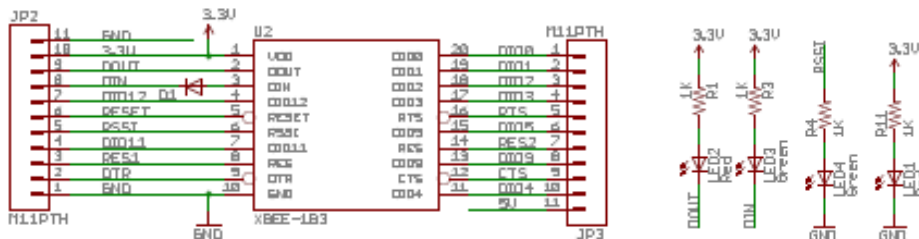


Figure 21: Schematic for the wireless transceiver

Pin Connections		
PIN	PIN name	Connection
1	VCC	3.3v power supply
2	DOUT	UART Out/LED
3	DIN/CONFIG	UART In/LED
4	D08	NC
5	RESET	GPIO
6	PWM0/RSSI	GPIO
7	PWM1	NC
8	[reserved]	NC
9	DTR/SLEEP_RQ/DI8	GPIO
10	GND	GND
11	AD4/DIO4	GPIO (optional)
12	CTS/DIO7	GPIO
13	ON/SLEEP	GPIO/LED
14	VREF	NC
15	Associate/AD5/DIO5	GPIO
16	RTS/AD6/DIO6	GPIO
17	AD3/DIO3	GPIO (optional)
18	AD2/DIO2	GPIO (optional)
19	AD1/DIO1	GPIO (optional)
20	AD0/DIO0	GPIO (optional)

Figure 22: Pin connections for the wireless transceiver

Controller Unit Power System Detailed Design

Power Supply Design

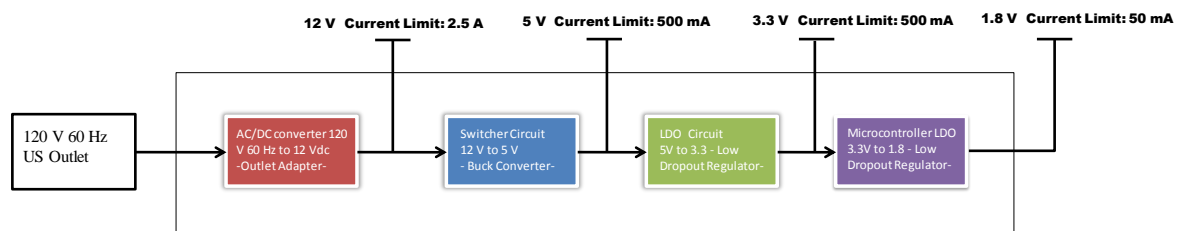


Figure 23: Controller unit power supply diagram

The above block diagram represents a high level design of the controller unit power system. The motor unit is a high power device that requires a lot of current draw. Therefore, we will power this unit using a

standard US outlet, 120 V 60 Hz. Our group has not yet implemented the power system design in our schematic because we want to assure that we know the actual power ratings of the system. As we implement our schematic design, we find the need to add other IC chips. Therefore, we do not want to under design or over design the power system until all of the appropriate components are set in place. The following table represents the power ratings of the units that we are implementing:

Power Supply Spec		
Device	Supply Voltage	Max Current Draw
LCD	3.3 V	300 uA
LCD Backlight	3.3 V	40 mA
LED Driver Circuit	3.3 V	1.5 mA
Temp Sensor	3.3 V	60 uA
Microcontroller	3.3 and 1.8	36.3 mA
DC Motor	12 V	1.3 A
Ethernet Transceiver	3.3 V	81.6 mA
Voltage Detector	3.3 V	3.5 uA
NiMH Charger	3.3 V	300 uA
Xbee Transceiver	3.3 V	250 mA

Figure 24: Power supply specifications

Using this data, our group has a good understanding of the power requirements of our system. Because the system will be powered through a standard US outlet, we are considering an AC to DC converter such as the Pihong PSC30R-120-R, rated at 12 V and 2.5 A. Because this rail is provided by an outlet, we will implement reverse diode protection circuitry to safeguard our system from lightning and will implement voltage limitation circuitry to maintain the rail voltage within a desired range. We will also use a bypass scheme to filter and reduce noise on this line. Using this rail, we will power the motor driver circuitry as well as the motor itself. On this rail, we will use a switcher circuit, buck converter, in order to achieve a 5 Vdc from the 12 Vdc signal. Because the switcher will operate at a particular frequency and will have a noisy output, we will implement various bypass schemes to reduce the noise on this line. The 5 V signal will be used power the motor driver circuitry and can be used to power the LCD as well as some IC chips on the controller unit. Since group is attempting to implement one PCB for the entire controller unit as well as the user interface, which is powered by 2 AA NiMH batteries, it is in our best interests to keep the power system design as similar as possible. Therefore, we will try to power most of our system off of the next portion of power circuitry. The next portion of the power supply will consist of an LDO, low dropout regulator, chip that will provide a 3.3 Vdc signal from the 5 Vdc input. The LDO will not produce much noise and can be handled with a few bypass capacitors. This rail will be used to power a major portion of our system and will have to have a high power rating since we are attempting to use the same power design scheme for two PCBs. The final rail of our system consists of internal 1.8 Vdc regulators within the microcontroller as well as the LCD that will power to internal IC chipsets.

Power Supply Design

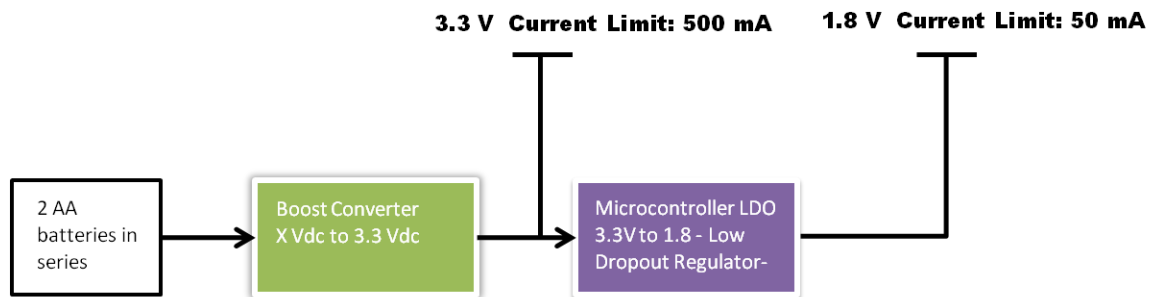


Figure 25: User interface power supply diagram

The above block diagram represents a high level design of the user interface unit power system. As explained in the previous section, our group has not yet implemented the power system design in our schematic because we want to assure that we know the actual power ratings of the system. As we implement our schematic design, we find the need to add other IC chips. Therefore, we do not want to under design or over design the power system until all of the appropriate components are set in place. We again used figure 28 to represent the power ratings of the units that we are implementing.

The power system of the user interface unit is slightly simpler to implement because it only requires two rails. The input power from the system will consist of two AA NiMH batteries in series. There will again be some form of ESD protection circuitry on the input as well as a bypass scheme to smooth out the battery voltage rail. Because batteries do not keep a constant voltage over a given amount of time, our group has decided to implement a boost converter in order to provide a constant 3.3 Vdc signal for an input that fits within a specified voltage range. Typical NiMH batteries are 1.5 V and tend to slope down to 1.2 V throughout their lifetime. Therefore, the boost converter will be designed to provide a 3.3 Vdc signal from an input range between 2 V and 3.5 V. As before, we will implement some sort of bypassing scheme due to the noise on the boost converter output pin due to the switching noise. As in the previous case, we will utilize the internal 1.8 V LDOs from the microcontroller and LCD to power internal IC chipsets in an effort to save money on another LDO implementation.

Recharging System

In an attempt to cut down on long term costs, the group designed the user interface unit to use two rechargeable NiMH AA batteries. We have implemented two forms of circuitry in order to handle the recharging functionally of the unit. The first portion of the design consists of a bq2002C chipset that we have used in order to implement a NiMH charger. This is a very simple design that checks to see whether a 5 V USB supply is present. If it is present, the unit will begin charging the two AA batteries. If

the unit takes an X amount of time to recharge the batteries, if it senses too much heat dissipation, or if the unit senses a sudden battery voltage drop off, it will stop charging the batteries and will signal that the batteries have charged. The design implementation is demonstrated below:

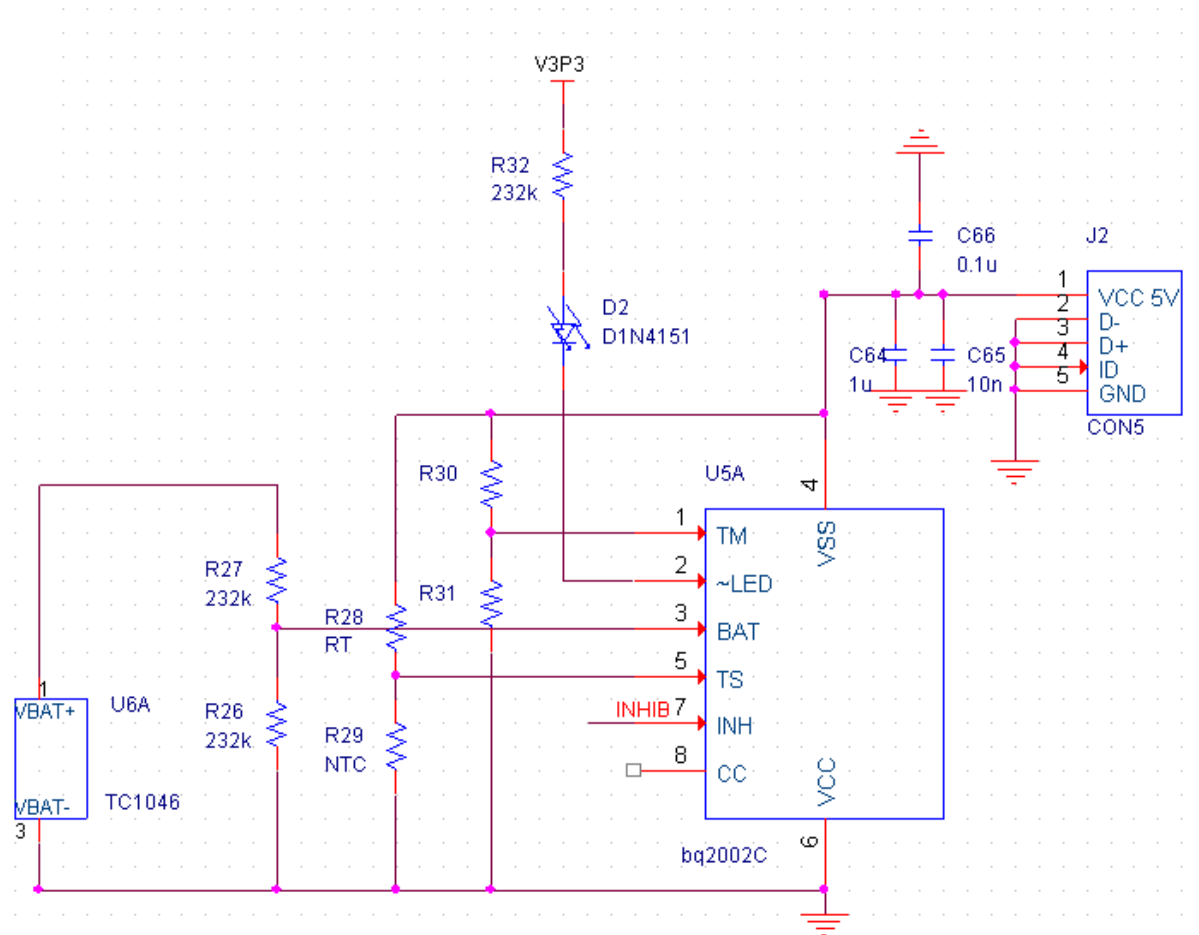


Figure 26: User interface battery charging circuitry schematic

The next portion of circuitry needed to implement this system simply indicates that the battery power unit is below a usable voltage and that the batteries need to be recharged. As of now, the system utilizes a Seiko Instruments voltage detector in order to sense the drop in battery voltage. The group will use this signal to warn the microprocessor to prompt the user to charge the unit. A beeper alarm might be implemented in order to warn the user.

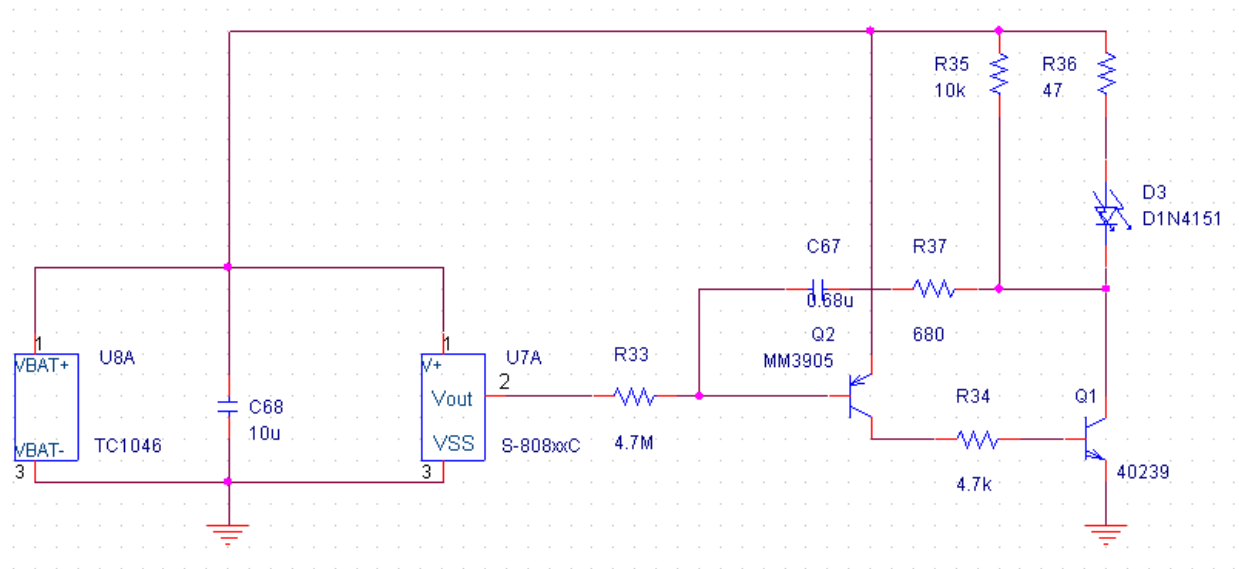


Figure 27: Low battery charge indicator

Using these two systems in sync with the microcontroller, we will implement the following functionality:

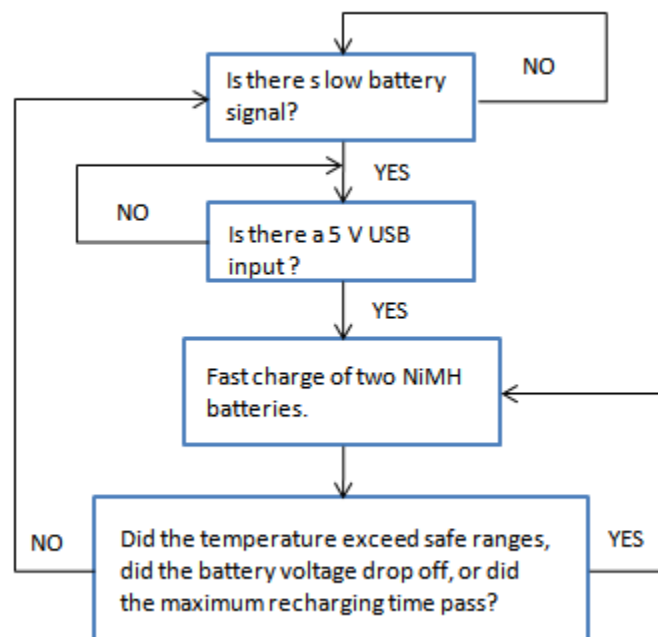


Figure 28: Battery charge circuitry Conceptual Diagram

When the unit is finished charging, the microcontroller will receive a signal from the charger and inhibit the charger from supplying current to the batteries. As the voltage drops below usable levels, the voltage detector will trigger and signal to the microcontroller that the batteries are low. In this case, the

unit will alarm the user to charge the batteries by plugging the unit into a PC via USB. Once the batteries are charged, the process will repeat.

Microcontroller

The first step in the microcontroller implementation was to mux the appropriate functionality to the proper pins. Once complete, we would use these pins to communicate to the system peripherals. Our microprocessor has 144 available pins. We are anticipating using 60 to 70 percent of these pins in this project. We are only 60 percent completed with our schematic design and are not certain what more will be implemented through the microcontroller. Our group is using SPI to control the LCD, USART to control the Xbee transceiver, ADC to receive data from the temperature sensor, the standard JTAG to program the microcontroller, PWM in order to run the DC motor, and an Ethernet MAC interface to communicate to an Ethernet transceiver and the Iowa State University network. The table below demonstrates what functionality will be mixed to what pin in the 60 percent of our implementation:

Device	Pin Number	Pin Assignment	Function	Program Default
LCD	48	PA10	SPI0 - NPCS[0]	Function A
	50	PA11	SPI0 - MISO	Function A
	53	PA12	SPI0 - MOSI	Function A
	54	PA13	SPI0 - SCK	Function A
Xbee Transceiver	25	PA00	USART0 - RXD	Function A
	27	PA01	USART0 - TXD	Function A
	30	PA02	USART0 - CLK	Function A
	32	PA03	USART0 - RTS	Function A
	34	PA04	USART0 - CTS	Function A
Temp Sensor	73	PA21	ADC - AD[0]	Function A
JTAG	128		TMS	
	129		TCK	
	130		TDO	
	131		TDI	
DC Motor	143	PB19	PWM - PWM[0]	Function A
	3	PB20	PWM - PWM[1]	Function A
Ethernet	88	PB00	MACB - TX_CLK	Function A
	90	PB01	MACB - TX_EN	Function A
	96	PB02	MACB - TXD[0]	Function A
	98	PB03	MACB - TXD[1]	Function A
	100	PB04	MACB - CRS	Function A
	102	PB05	MACB - RXD[0]	Function A
	104	PB06	MACB - RXD[1]	Function A
	106	PB07	MACB - RX_ER	Function A
	111	PB08	MACB - MDC	Function A
	113	PB09	MACB - MDIO	Function A
	115	PB10	MACB - TXD[2]	Function A
	119	PB11	MACB - TXD[3]	Function A

Figure 29: Microcontroller Pin Assignment table

The next step was to translate this information to a schematic design. We began by implementing all of the microcontroller power rails and appropriate bypass capacitors. We then labeled the appropriate pins and connected them to the appropriate peripherals, checking pin voltage and current ratings to determine whether any signal and level conversion would have to be implemented. As of now, all of the

signal lines are direct connections using in series low ohm resistors or bypass capacitors. The microcontroller schematic implementation is shown below without any peripheral connections:

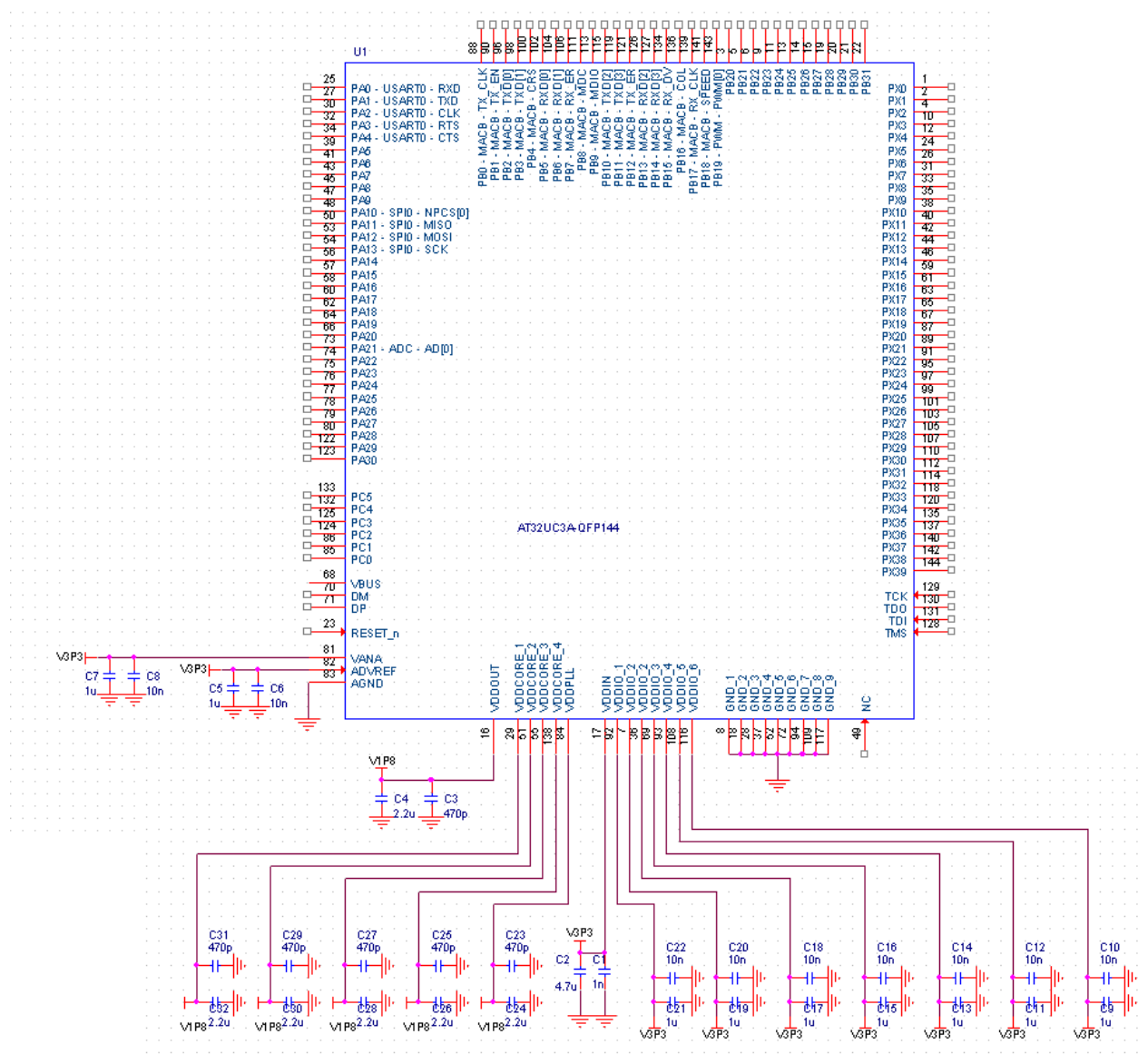


Figure 30: Microcontroller schematic

LCD Detailed Design

The LCD system design consists of two portions: the backlight design and the display design. The LCD has an internal driver that communicates commands from the microcontroller to the LCD through SPI protocol. The LCD can be run off of a single or multiple power supply. Because we are using 2 AA batteries, our group decided to implement the system using one power supply. Using the internal boost converters of the LCD, we placed capacitors in the appropriate locations in order to accommodate for a missing power supply and ensure high quality functionality. The power rail is bypassed with several capacitors in order to ensure a clean signal and high resolution for the intended viewer. The unit also

requires several bypass capacitors to be placed on a majority of lines to filter any noise that may arise from the internal boost converters.

The backlight system is implemented through a high efficiency boost converter and constant current driver. The group decided to use this approach in order to extend the quality and life of the LED backlight. This portion of the system could have as easily been implemented through series resistors and transistors connected to a few GPIO lines. However, through our approach, we are able to ensure extended life of the LED backlight through exact current draw limitations, an enable signal that helps control system transient responses to LED shut down, and control the system to ensure exact power dissipation. The LCD backlight, LCD, and driver are shown below:

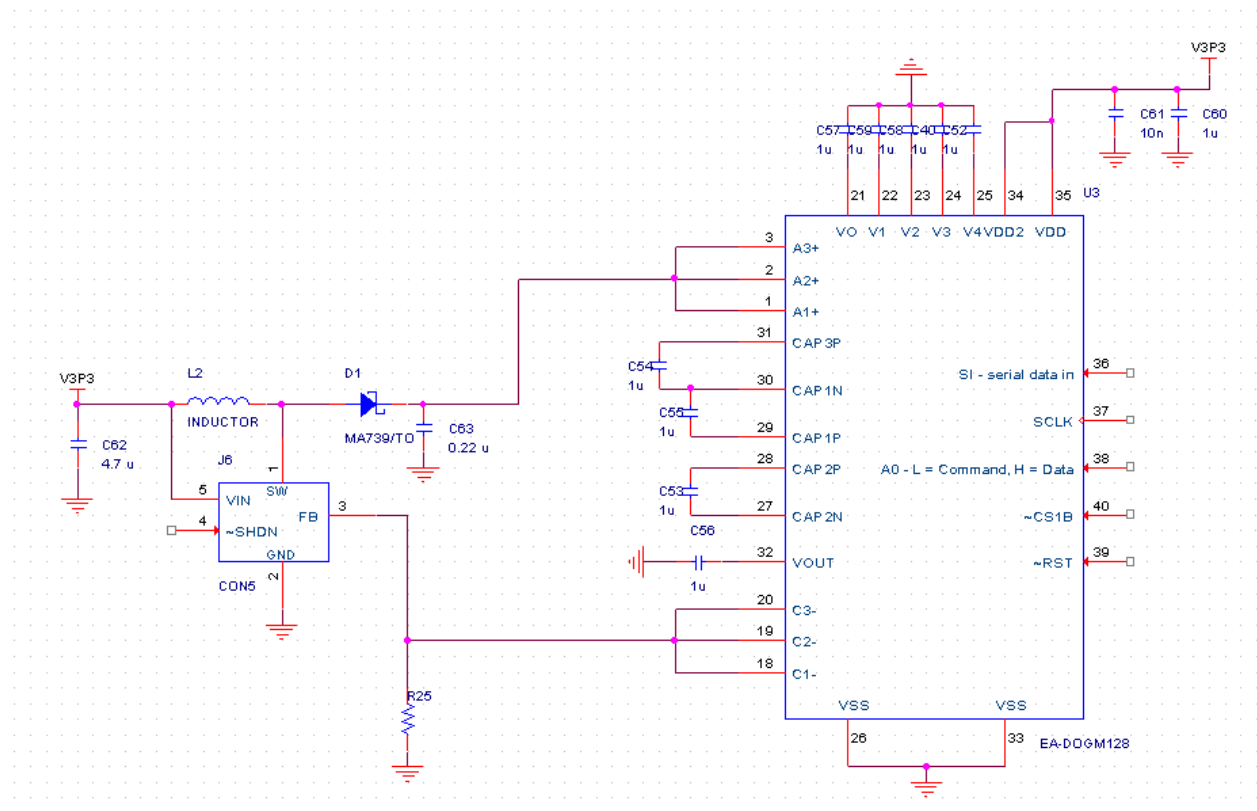


Figure 31: LCD schematic

Mechanical System

Stationary Platform

The following CAD design was done by our advisor, Lee Harker. We worked with Harker in selecting the appropriate materials and interconnect. The fabrication and machining of the finish product was performed by Harker.

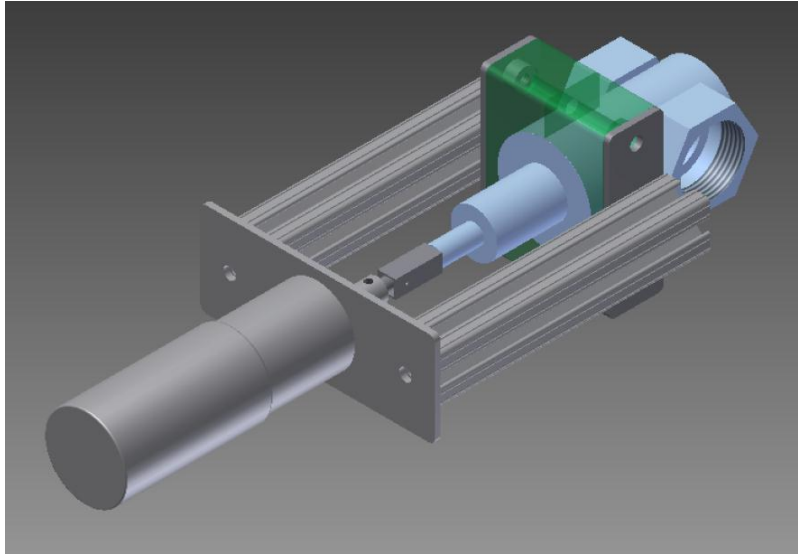


Figure 32: Stationary platform CAD design

From the CAD design, we have clamps that hold the stationary platform to the base of the steam valve. Then two extruded aluminum towers on the side to hold the gear motor mount. The gear motor will have four screws on the mount to hold the gear motor in place. Custom metal connection between the gear motor shaft and steam valve is designed by Harker. The connection will be designed to be easily detached.

With this design, we will achieve our main criteria of easy attachment/detachment and limit modification to the steam valve. With the appropriate stationary platform, we now move on to the design of the gear motor.

Gear motor

We identified the IG42 gear motor available from superdroidrobots.com to meet the minimum criteria for gear motor selection.

Operating at 24 V	Reduction Ratio	Rated Torque (N m)	Rated Speed (rpm)	Rated Current (mA)	No load Speed (rpm)	No Load Current (mA)
	1;84	1.96	75	< 2300	83	< 750

Figure 33: Gear motor specification

The IG42 gear motor operates can operate at 24V. This high operating voltage will increase the cost of implementing the power supply. However, when we looked at the speed and torque requirement of design, we determined that the gear motor would be able to deliver sufficient speed and torque when supplied with 12V. The results are that we implementation of power circuitry will have lower cost and power draw consumption will be reduced.

The accompanying magnetic shaft encoder came with the features listed below.

Encoder pulse per rev	Gear Motor Reduction Ratio	Total pulse per rev (after gear reduction)
4	84	336

Figure 34: Shaft encoder specification

Although the magnetic shaft encoder only count 4 pulse per revolution, after applying the gear motor reduction ratio, the encoder is able to achieve 336 counts per revolution of the gear motor shaft. This comes how to be

$$\left(360 \frac{\text{degrees}}{\text{rev}}\right) * \left(\frac{1}{336} \frac{\text{rev}}{\text{pulse}}\right) = 1.071 \frac{\text{degrees}}{\text{pulse}}$$

There is roughly a one-to-one correlation between degrees and pulses. To resolve the fractional degrees lost per pulse, we will add back the missing 24 pulses per each complete revolution. This will also be prorated according to the degrees rotated.

$$\text{degree rotation} = \text{degree rotation desired} * \left(1 + \frac{24}{360}\right)$$

This approach will help in improving the accuracy, but daily calibration will be needed to ensure consistent operation.

Bill of Materials

The following BOM detail is for implementing the mechanical system.

Item	Qty	Part
1	1	IG42 gear motor
2	1	L298N motor driver
3	4	diode
4	2	100uF capacitor
5	2	1Kohm resistor
6	1	2 ohm sense resistor

Figure 35: Gear motor specification

These parts can reference under schematic under Appendix A.

Functional Diagram

The following functional diagram shows the input and output relationship between modules interacting with the mechanical system.

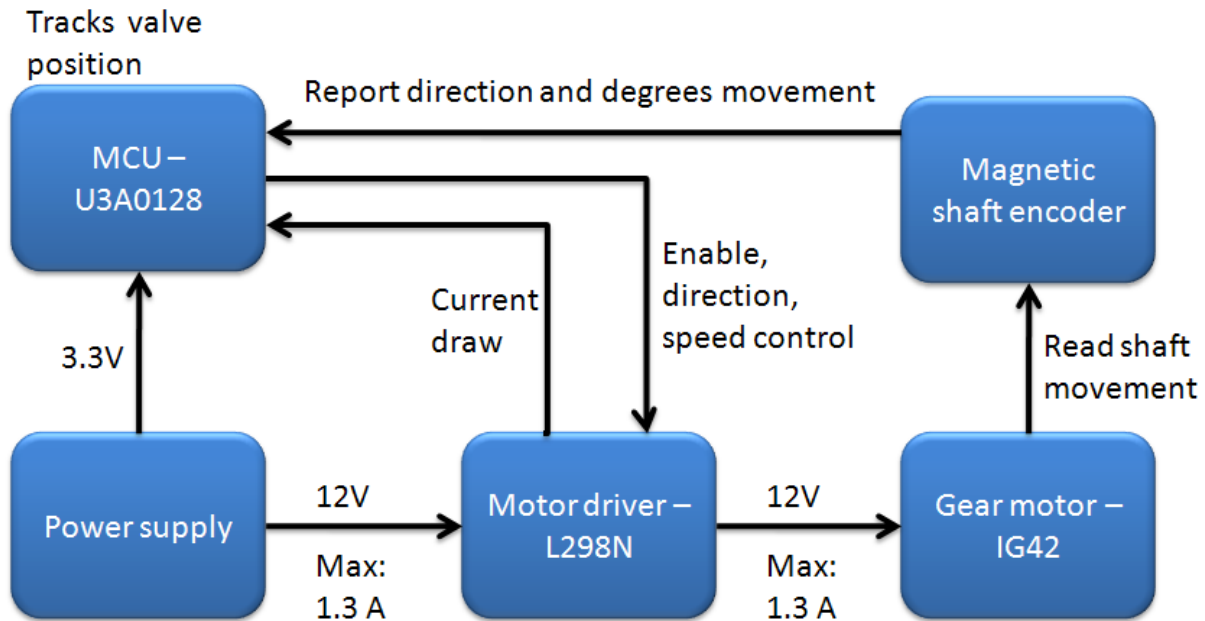


Figure 36: Mechanical Interface Interaction Diagram

Beginning with the power supply, it will supply 12V and a max current of 1.3A to the motor driver. The MCU generates the enable, direction, and speed through TTL and PWM signals to control the motor driver circuit. With the inputs, the motor driver is able to provide 12V and max of 1.3A to the gear motor and also provide current consumption feedback to the MCU. The rotation of the gear motor will be read by the integrated magnetic shaft encoder and that information is fed back to the MCU for processing.

With this functional system in place, the MCU will be able to control and track the rotation and position of the steam valve.

Logical Diagram

We have two logical implementation of the gear system software. Since the gear motor does not instantaneously stop when we command it to stop, the extra coasting of the gear motor will decrease our accuracy control.

To address this problem, we measured the extra coasting in degrees that the gear motor would have when fed with an instantaneous pulse. We then set the threshold value to be three times the extra coasting angular degrees. This will determine which set of speed the gear motor would use. The following two diagrams show two different method of rotating the steam valve.

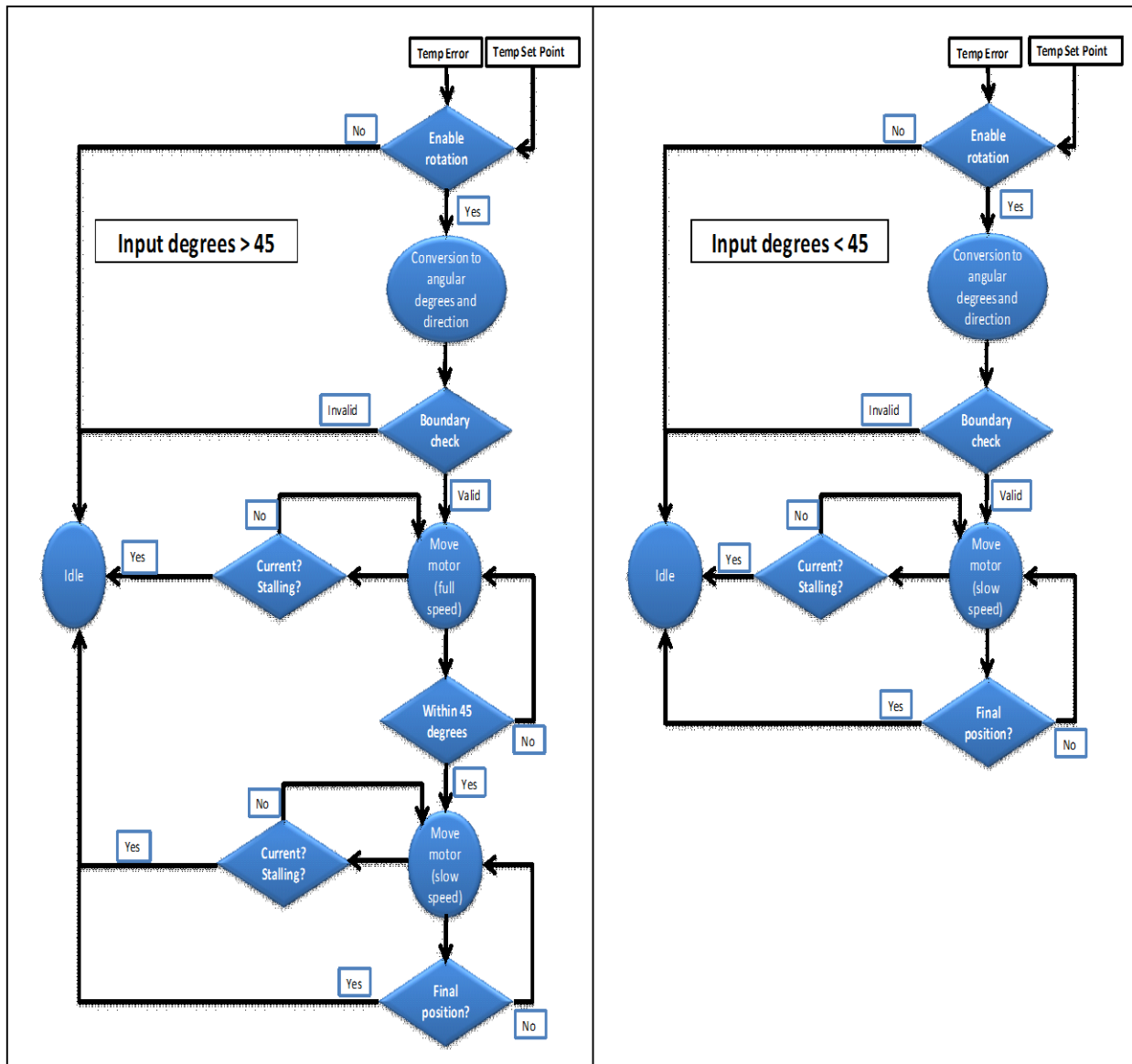


Figure 37: Gear motor logical diagram

Looking at the logical flow for input degrees greater than 45 degrees; we begin by receiving the inputs temperature error value and temperature set point from the controls system. We will enable rotation request only if a new temperature set point has been provided.

The next step after enabling the rotation process, we will convert the temperature error to a corresponding rotation that would reduce the temperature error to zero. In order to perform this conversion, we will create the follow table where we could correlate temperature error to angular degrees rotation.

Temperature Error (degrees Fahrenheit)	Rotation (angular degrees)
-20	Max limit: -1800
-19	.
-18	.
.	.
.	.
.	.
0	0
.	.
.	.
.	.
19	.
20	Max limit: 1800

Figure 38: Temperature Error to Degrees Rotation Chart

For clarification, we assume the lower and upper limit of allowed temperature to be 60 and 80 degrees Fahrenheit respectively. This will limit our temperature error to be from -20 to 20 degrees Fahrenheit. The max limit for allowed rotation is 1800 angular degrees. This comes from the fact that the steam valve needs 5 full revolutions from fully close to fully open.

Since this table will only be applicable to one temperature, we will have to create 20 tables to accommodate the temperature range from 60 - 80 degrees Fahrenheit. We will generate these tables once automation of the mechanical and MCU is complete.

Now that we have the angular degrees to be rotated, we proceed to validate that the rotation will not exceed the fully close and fully open position. If the new position will exceed the boundary, the gear motor will idle and inform the MCU that the rotation cannot be perform. If the new position is valid, the gear motor will rotate at full speed towards the new position.

As the gear motor rotates, over current and stalling check will ensure the gear motor is actually moving. Should those checks go off, the gear motor will stop and the MCU will be informed of the situation. The next check is to slow down the gear motor once it is within 45 angular degrees of the final position. Again, the gear motor is subject to over current and stalling check as it rotates. The last check will confirm that the gear motor has rotated the steam valve to the desired position.

The other logical diagram is used when the rotation is less than 45 angular degrees. The only difference will be the gear motor will operate at slow speed from the beginning. This ensures that small angular turns will not have coasting problems at high speed.

This completes our detail design of the mechanical system. The gear motor description, BOM, functional diagram, and logical diagram will ensure the mechanical system operate as required for the project.

Temperature Sensor

Using the selection qualifications and design constraints, we were able to narrow down to the following selections:

Digi-Key Part Number	TC1046VNBTRCT-ND	Price Break	Unit Price	Extended Price
Quantity Available	18,497	1	0.70	0.70
Manufacturer	Microchip Technology	10	0.59	5.90
Manufacturer Part Number	TC1046VNBTR	25	0.49	12.25
Description	IC TEMP-VOLT CONV PREC SOT23B	100	0.45	45.00
Lead Free Status / RoHS Status	Lead free / RoHS Compliant			

All prices are in US dollars.

Digi-Key Part Number	MCP9700A-E/TO-ND	Price Break	Unit Price	Extended Price
Quantity Available	10,407	1	0.34	0.34
Manufacturer	Microchip Technology	10	0.31	3.10
Manufacturer Part Number	MCP9700A-E/TO	25	0.26	6.50
Description	IC SENSOR THERMAL 2.30 TO-92-3	100	0.24	24.00
Lead Free Status / RoHS Status	Lead free / RoHS Compliant			

Digi-Key Part Number	TMP36GRTZ-REEL7CT-ND	Price Break	Unit Price	Extended Price
Quantity Available	18,956 Note	1	1.35	1.35
Manufacturer	Analog Devices Inc.	25	1.13	28.13
Manufacturer Part Number	TMP36GRTZ-REEL7	100	0.90	90.00
Description	IC SENSOR TEMP 2.7/5.5V SOT23-5	250	0.68	168.75
		500	0.63	315.00
Lead Free Status / RoHS Status	Lead free / RoHS Compliant	1,000	0.60	598.50

Figure 39: Temperature sensor selection table

In the end, we picked the High Precision Temperature-to-Voltage Converter TC1046VNBTRCT-ND. This is the ideal temperature sensor for our project. It has an ideal supply voltage range of 2.7 V to 4.4 V, wide temperature measurement Range: -40 degrees Fahrenheit to + 257 degrees Fahrenheit. It has a cheap unit price of \$0.59 at the quantity of 10, and has almost 20,000 units available for order. Though the price when compared to the MCP9700A-E/TO-ND is slightly more, the TC1046VNBTRCT-ND's overall reliability, availability, and user rating is higher than MCP9700A-E/TO-ND. TC1046VNBTRCT-ND's unit price is about half of the price of TMP36GRTZ-REEL7CT-ND and when the two temperature sensor's features, reliability, and user ratings are compared showed very similar ratings. Though the TMP36GRTZ-REEL7CT-ND has a wider operation range from -40 degrees Fahrenheit to + 302 degrees Fahrenheit, both temperature sensors satisfy the required operation range for the project of 14 degrees Fahrenheit to 100 degrees Fahrenheit.

The team will be using pin 17 (ADC) from the microcontroller

Pin Number	Pin Assignment	Function	Program Default
73	PA21	ADC - AD[0]	Function A

Figure 40: Temperature sensor pin assignment

Below is the (Left) Schematic design for the temperature sensor. The Vout of the temperature sensor will be mapped to the microcontroller's pin 73 as shown in the functional block diagram below (right).

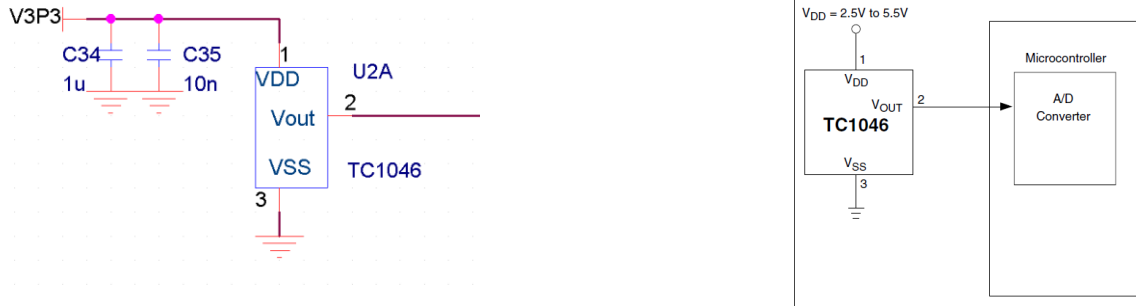


Figure 41: Temperature sensor schematic and functional diagram

Address	Peripheral Name	Bus
0xFFFF1C00	USART2 Universal Synchronous Asynchronous Receiver Transmitter - USART2	PBA
0xFFFF2000	USART3 Universal Synchronous Asynchronous Receiver Transmitter - USART3	PBA
0xFFFF2400	SPI0 Serial Peripheral Interface - SPI0	PBA
0xFFFF2800	SPI1 Serial Peripheral Interface - SPI1	PBA
0xFFFF2C00	TWI Two Wire Interface - TWI	PBA
0xFFFF3000	PWM Pulse Width Modulation Controller - PWM	PBA
0xFFFF3400	SSC Synchronous Serial Controller - SSC	PBA
0xFFFF3800	TC Timer/Counter - TC	PBA
0xFFFF3C00	ADC Analog To Digital Converter - ADC	PBA

The Peripheral Address shown to the left shows the address value of 0xFFFF3C00 for the ADC. This will be used for the temperature sensor.

The ADC will be using is based on a Successive Approximation Register (SAR) 10-bit Analog-to-Digital Converter (ADC). We will be using the ADC_NB_CHANNELS-to-1 analog multiplexer to convert the analog signals from the temperature sensors to digital signals from the ADC_NB_CHANNELS analog lines as shown below.

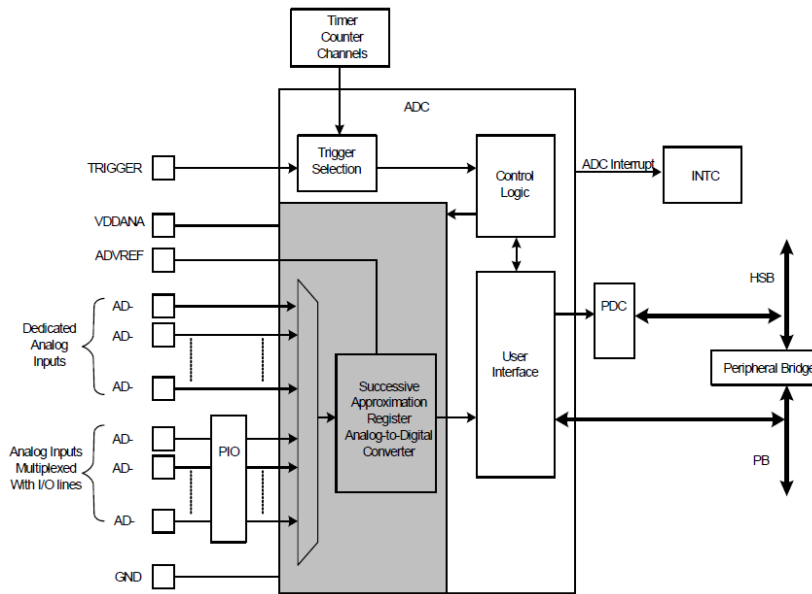


Figure 42: Microcontroller ADC for the temperature sensor

Ethernet Transceiver

Using the selection qualifications and design constraints, we were able to narrow down to the following selections:

Product Info.			
Mouser Part #:	886-LAN8700C-AEZG	777-CS8952-IQZ	886-LAN8187-JT
Mfr.'s Part #:	LAN8700C-AEZG	CS8952-IQZ	LAN8187-JT
Manufacturer:	SMSC	Cirrus Logic	SMSC
Description:	Ethernet ICs IC w/HP Auto-MDIX Embedded Ethernet	Ethernet ICs IC 100BASE-TX and 10BASE-T Transceiver	Ethernet ICs HiPerfrm Ethernet PHY
Data Sheet:	Data Sheet	Data Sheet	Data Sheet
Specifications			
Data Rate:	10 Mbps or 100 Mbps	10 Mbps or 100 Mbps	10 Mbps or 100 Mbps
Ethernet Connection Type:	10 Base-T, 100 Base-TX	-	10 Base-T, 100 Base-TX
Package / Case:	QFN-36	TQFP-100	TQFP-64
Product:	Ethernet Transceivers	Ethernet Transceivers	Ethernet Transceivers
Type:	MII/RMII Ethernet Transceiver	Single Chip PHY Transceiver	MII/RMII Ethernet Transceiver
Mounting Style:	SMD/SMT	SMD/SMT	SMD/SMT
Supply Current (Max):	39 mA, 81.6 mA	+/- 10 mA	39 mA, 81.6 mA
Supply Voltage (Max):	3.3 V	6 V	3.3 V
Supply Voltage (Min):	1.8 V	- 0.3 V	1.8 V
Maximum Operating Temperature:	+ 70 C	+ 70 C	+ 70 C
Minimum Operating Temperature:	0 C	0 C	0 C
Number of Transceivers:	1	1	1
Product Category:	Ethernet ICs	Ethernet ICs	Ethernet ICs
Standard Supported:	802.3ab	802.3ab, 100BASE-FX or 100BASE-TX or 10BASE-T	802.3ab
Packaging:	-	-	Bulk
Ordering Information			
Stock:	4,346 Can Ship Immediately,	720 Can Ship Immediately,	Non-Stocked
Factory Lead-Time:	2 Weeks	5 Weeks	Request Delivery Quote
Pricing:	1: \$2.28 25: \$1.36 100: \$1.18 250: \$1.12 500: \$1.04	1: \$17.58 25: \$15.81 100: \$12.30 250: \$10.54 500: \$9.49	620: \$1.43

Figure 43: Ethernet transceiver options table

886-LAN8700C-AEZG was picked because of its high availability and cost. All three Ethernet transceivers have met the technical approach qualifications but only 886-LAN8700C-AEZG and 777-CS8952-IQZ are currently available and can be purchased individually. 886-LAN8700C-AEZG has over 4,000 in stock compared to the 700 in stock for 777-CS8952-IQZ and the price for 886-LAN8700C-AEZG is only \$2.28 compared to 777-CS8952-IQZ's \$17.58.

The details design of the Ethernet transceiver below shows all of its registers and components mappings. The Ethernet transceiver will only be placed in the microcontroller attached to the room with the steam valve. It will take the data of other rooms and transmit all of them to the Iowa State University network.

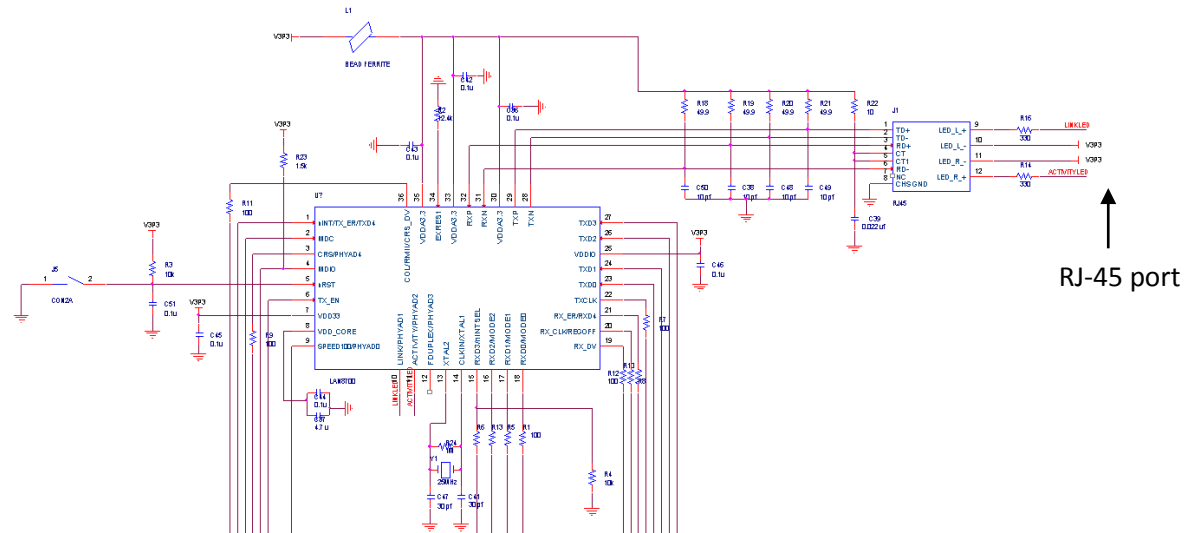


Figure 44: Ethernet transceiver schematic

The Ethernet transceiver will be using the microcontroller's pins as shown below.

Pin Number	Pin Assignment	Function	Program Default
88	PB00	MACB - TX_CLK	Function A
90	PB01	MACB - TX_EN	Function A
96	PB02	MACB - TXD[0]	Function A
98	PB03	MACB - TXD[1]	Function A
100	PB04	MACB - CRS	Function A
102	PB05	MACB - RXD[0]	Function A
104	PB06	MACB - RXD[1]	Function A
106	PB07	MACB - RX_ER	Function A
111	PB08	MACB - MDC	Function A
113	PB09	MACB - MDIO	Function A
115	PB10	MACB - TXD[2]	Function A
119	PB11	MACB - TXD[3]	Function A
121	PB12	MACB - TX_ER	Function A
126	PB13	MACB - RXD[2]	Function A
127	PB14	MACB - RXD[3]	Function A
134	PB15	MACB - RX_DV	Function A
136	PB16	MACB - COL	Function A
139	PB17	MACB - RX_CLK	Function A
141	PB18	MACB - SPEED	Function A

Figure 45: Ethernet transceiver pin connections

The diagram below shows the LAN8700 System Block Diagram where the MII interface will be mapped to the Ethernet controller from the microcontroller. We have decided to use MII over RMII because MII contains more functionalities thus have more possibilities for future expandabilities.

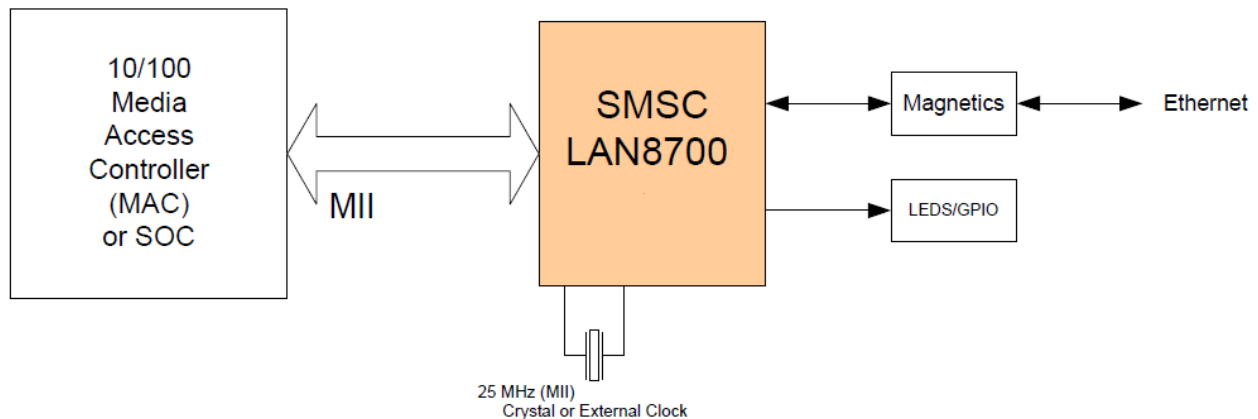


Figure 46: LAN8700 System Block Diagram

The team will use the pins shown below to implement the MII interface.

Pin Name	MII	RMII
ETXCK_EREFC	ETXCK: Transmit Clock	EREFC: Reference Clock
ECRS	ECRS: Carrier Sense	
ECOL	ECOL: Collision Detect	
ERXD	ERXD: Data Valid	ECRSD: Carrier Sense/Data Valid
ERX0 - ERX3	ERX0 - ERX3: 4-bit Receive Data	ERX0 - ERX1: 2-bit Receive Data
ERXER	ERXER: Receive Error	ERXER: Receive Error
ERXCK	ERXCK: Receive Clock	
ETXEN	ETXEN: Transmit Enable	ETXEN: Transmit Enable
ETX0-ETX3	ETX0 - ETX3: 4-bit Transmit Data	ETX0 - ETX1: 2-bit Transmit Data
ETXER	ETXER: Transmit Error	

Figure 47: MII Ethernet standard pin mapping table

The DMA controller will be using the Ethernet transceiver to perform six types of operation on the bus. In order of priority, these are:

1. Receive buffer manager write
2. Receive buffer manager read
3. Transmit data DMA read
4. Receive data DMA write
5. Transmit buffer manager read
6. Transmit buffer manager write

Website Interface

The block diagram below shows the relationship between the website interface and the steam heat valve controller system of three rooms. Note that only the room with the steam heat valve is able to communicate with the Ethernet interface. The Ethernet interface then talks with the website interface.

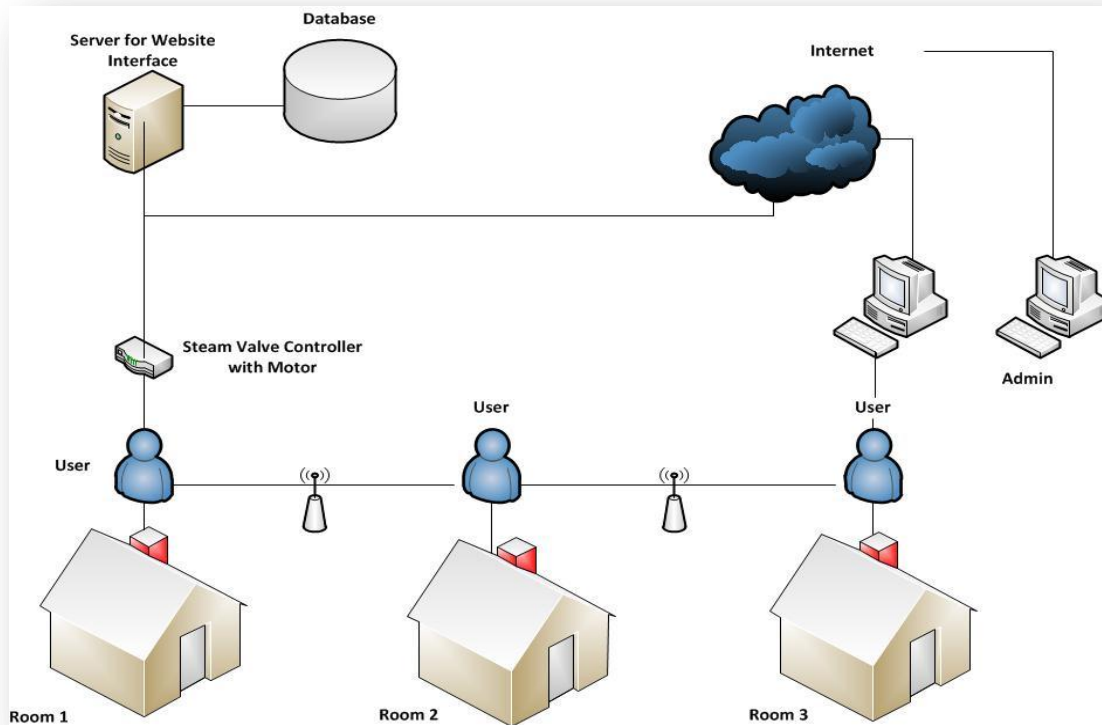


Figure 48: Website interface functional diagram

The design below shows the overall layout of the home page of the website interface for a three room system for an average user.

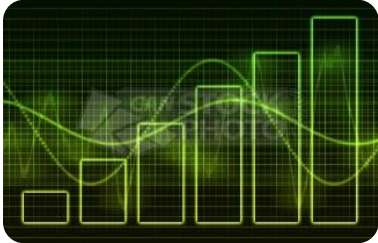
Steam Valve Controller Web Interface

Current Temperature: **78 F**

Desired Temperature:

Current System Status: **OK**

Data History and Graphs



Room 1

Room 2

Room 3

Hello Ben!

Feedback Form

Figure 49: Web user interface

The design below shows the overall layout of the home page of the website interface for a three room system for the admin.

Steam Valve Controller Web Interface

Current Temperature: **78 F**

Desired Temperature:

Current System Status: **OK**

Room1 Users

Ben

John

Julie

Room1 Users

Data Analysis

Room 1

Room 2

Room 3

Hello Admin!

This system is really easy to use, I feel very comfortable sitting in the room.

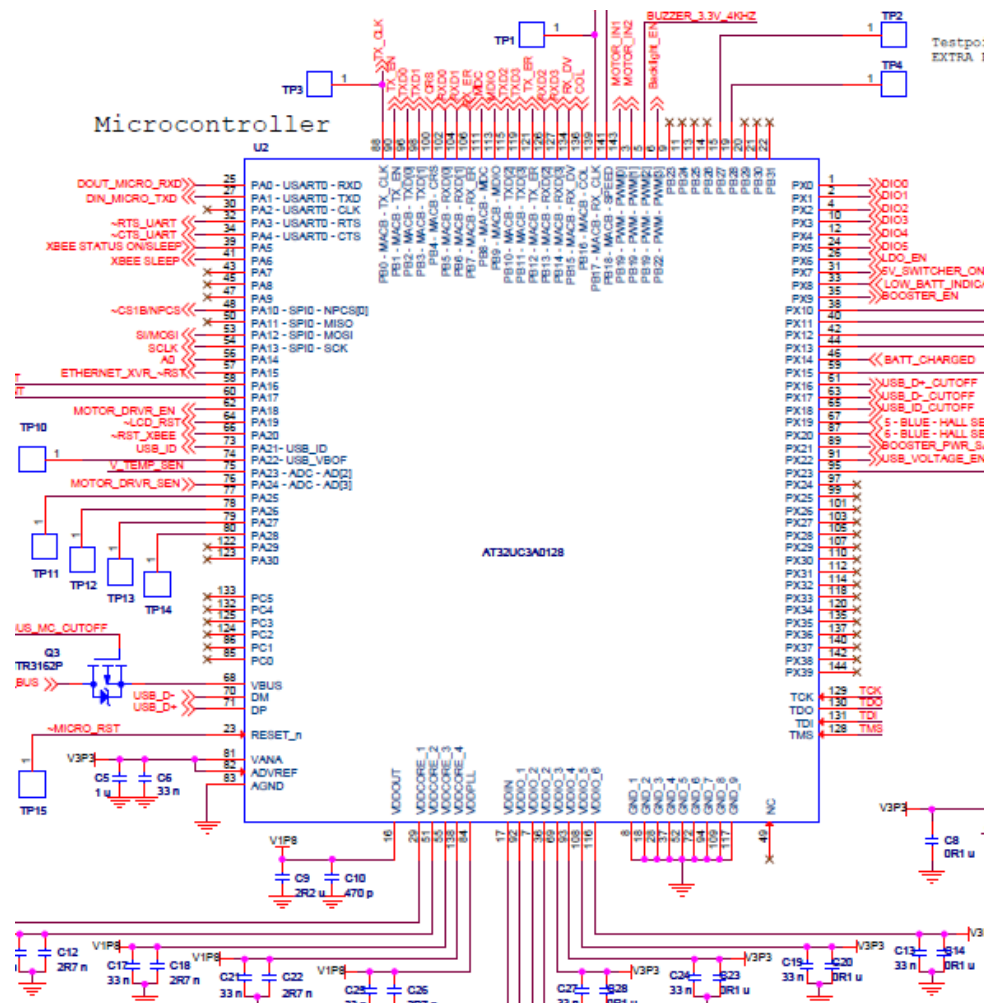
Figure 50: Web admin interface

PHP will be used to handle the front end of the website interface such as user authentication and the graphical user interface. The PHP code will then execute a script call with calls the Java JDBC code. The Java JDBC template code is used to connect to the database, which stores all the information about the steam control system and information about the users and admin.

Implementation

Microcontroller

The microcontroller final schematic is shown below. The microcontroller interacts directly with most major components used for the controller box, control panel, and motor circuitry. The different components that the microcontroller interacts with will be covered in detail during the sections below.



Wireless Transceiver

Wireless Transceiver Implementation

During the integration of the wireless transceiver software interface there were several modifications made in order to improve the system. Although the implementation of the wireless transceiver stayed roughly the same, the actual communication protocol for the wireless transceiver had several major changes. These changes were all implemented during software integration as improvements to the initial system design in order to meet the needs of the system. The new structure uses the controller box as the communication hub that requests information from each of the control panels. The request is sent out by transmitting the number of the requested room. Then the room replies with a 4 byte packet while the control box waits for a reply. The packet includes the room number of the replying user interface, the current temperature, the room's requested set point, and finally the transmission complete delimiter for bytes zero to three respectively. The request and reply packets are outlined below:

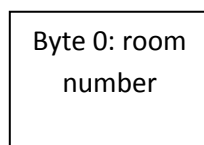


Figure 52 Request Packet Format

Request Packet Format: Sent from the controller box to all Xbee, with one specific room number sent out to be requested from

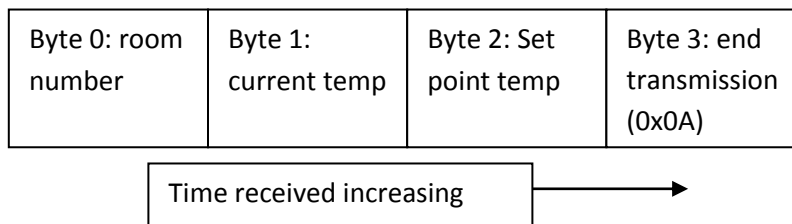


Figure 53: Response Packet

Response packet format: Sent from the controller panel to the controller box in response to a request

After having sent a request, the control box will continue to wait a second before continuing on to the next request.

In order to transmit these data packets to and from the microcontroller, a USART is used to communicate with the wireless communication module. In order to receive the data an interrupt is used on the X-Bee that triggers on an incoming byte on the USART. The byte is then stored, along with the rest of the message in a FIFO buffer. The buffer is then parsed in a call in the main program loop. The function then takes the incoming data and stores it in a data structure that stores the current

Both the LCD display and the LCD backlight run on the 3.3V power line. There are multiple uses of capacitors for the LCD to properly operate. There are six important control signals in the LCD circuitry: Backlight_EN, SIMOSI, SCLK, A0, CSIBNPCS, and LCD_RST.

The SIMOSI, SCLK, and CSIBNPCS signals are SPI interface signal in order the SPI operation to work correct. The Backlight_EN and LCD_RST handle the backlighting control and software reset of the LCD. Lastly, the A0 signal carries commands and data to the LCD.

Software

The software implementation of the LCD consists of many functions that perform the data storage and data transmission to the actually LCD. There are three high LCD function that used by the main programming to use the LCD.

```
void initialSetupLCD(void) ;
```

```
void update_LCD_current_temp(unsigned char curTemp) ;
```

```
void update_LCD_set_pt(unsigned char setPt) ;
```

The function initialSetupLCD() enables the appropriate used by LCD circuitry, resets the LCD hardware, and set up the LCD template. The resulting output is shown below.



Figure 55: LCD Screen

In arriving at the above template, we wrote many smaller LCD functions to handle particular objects in the template.

The function update_LCD_current_temp() updates the LCD screen with new values for the current temperature. It handles only the current temperature zone on the LCD.

The function update_LCD_set_pt() updates the LCD screen with new values for the set point temperature. It handles only the set point temperature zone on the LCD.

Motor System

Hardware

The hardware within the motor system consists of three components: platform, motor driver IC, and gear motor.

Platform

The two important functions of the platform include mounting the gear motor vertically, attaching the platform to the base of the steam valve.

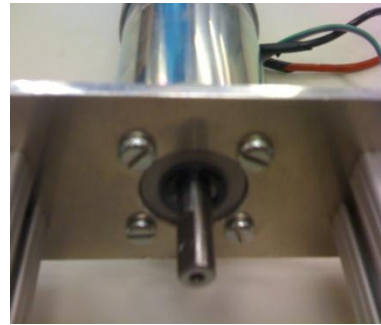
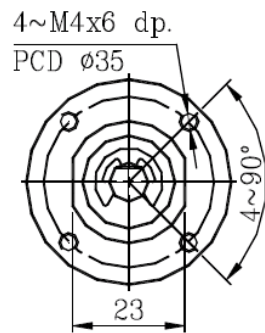


Figure 56: Motor Components 1

Within the datasheet of the gear motor, the positions of the mounting holes are laid out in the above diagram. Our advisor, Lee Harker, designed and manufactured the platform to vertically mount the gear motor. The result is achieved using four slotted screws as seen in the above Figure.

In addition to mounting the gear motor, the platform has to be capable of attaching and detaching from the steam valve with minimum effort. Again our advisor, Lee Harker, designed and manufactured the mechanism for securing the platform to the base of the steam valve to form a steady foundation.

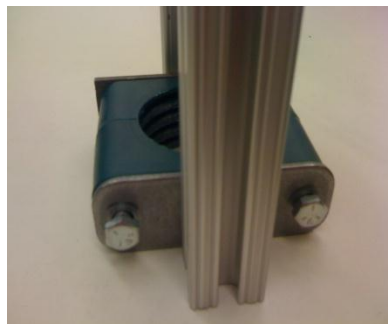


Figure 57: Motor Components 2

The mechanism uses two long hex cap screws to tighten the plastic structure around the base of the steam valve. This approach forms a solid foundation for the platform and provides ease of installation and removal.

Motor Driver

The motor driver circuitry allows the microcontroller to control the rotation and direction of the gear motor. The schematic shown below highlight the inputs, outputs, and power control in the motor circuitry.

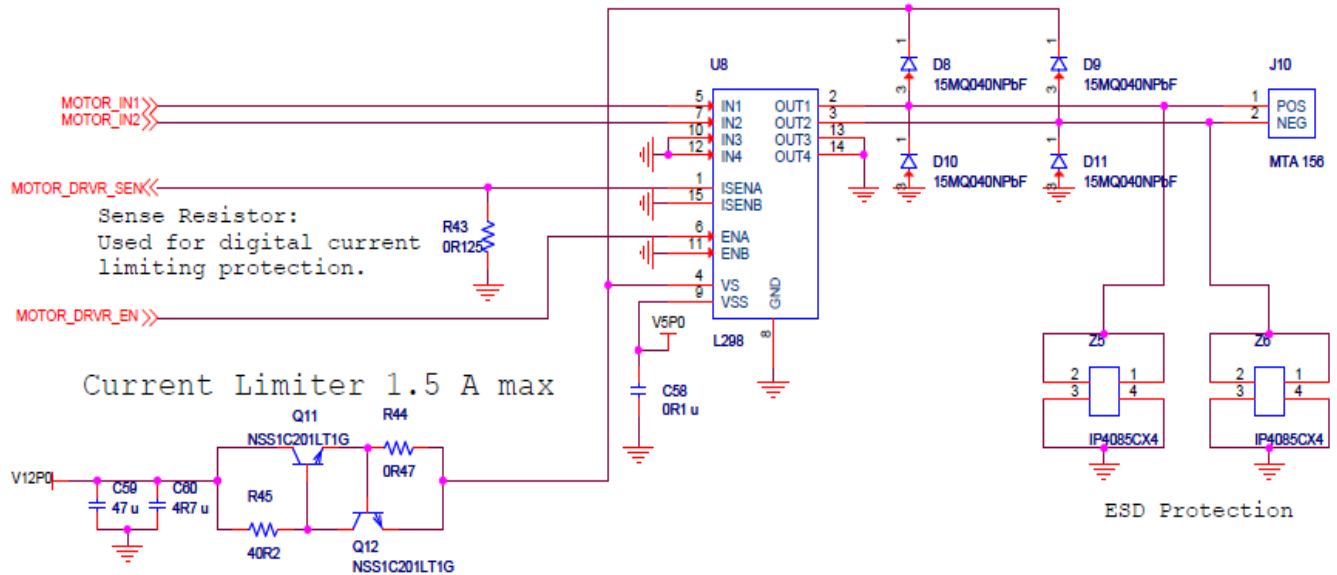


Figure 58: Motor Final Schematic Design

The L298N motor driver IC has two sets of identical control circuitry for control up to two gear motor. We only use the first control circuitry in controlling the gear motor. The second control circuitry is grounded and the respective pins are shorted to ground.

The motor driver circuitry uses two different power lines, 5V and 12V. The 5V power line is use for powering the logical transistors in the L298N. The 12V power line is use for power the gear motor. For the 12V power line, we implemented a current limiter to limit the current draw to less than 1.5 Amps. This is to ensure that should any technical errors were to happen, the power to the gear motor would be cut off preventing motor burn out.

From our torque testing of the steam valve, we determined that the maximum required torque in turning the steam valve was approximately $1.0N \cdot m$. With this information, we measured the current drawn by the gear motor under $1.0N \cdot m$ torque using a 12V power supply. We recorded a maximum reading of 1.2 Amps from the gear motor. We then provide a safety buffer of 0.3 Amps. The final current limiting threshold was set at 1.5 Amps.

There are three input signals from the microcontroller to the motor driver: MOTOR_DRV_R_EN, MOTOR_IN1, and MOTOR_IN2. All signals are active high. Both MOTOR_IN1 and MOTOR_IN2 are PWM signals. MOTOR_DRV_R_EN is a TTL signal. The following chart documents the outputs of the motor driver circuitry given the three input signals.

Inputs			Function	Steam Valve
	MOTOR_IN1	MOTOR_IN2		
MOTOR_DRV_R_EN = H	H	L	Forward	Closing
	L	H	Reverse	Opening
MOTOR_DRV_R_EN = L	H	H	Fast Motor Stop	X
	X	X	Motor Stop	X
	H = HIGH			
	L = LOW			
	X = DON'T CARE			

Figure 59: Motor Data Table

Upon acting on the input signals, pins OUT1 and OUT2 of the L298N drives the gear motor to rotate in the desired direction. In addition, the L298N device outputs the same amount of current drawn by the gear motor on pin ISEN. At pin ISEN, a sense resistor of 0.125 ohms lowers the voltage at that node. The voltage is then read by the microcontroller as MOTOR_DRV_R_SEN. The MOTOR_DRV_R_SEN routes to the ADC channel 3 or pin number 76 on the microcontroller hardware. In using ADC channel 3, the microcontroller uses the 3.3V reference to convert the analog voltage signal to a digital value.

Diodes D8 - D11 form a fly-back diode structure. The structure drives any excess power generated by the rotation of the gear motor towards the 12V power line thereby preventing any damages to the L298N device.

Lastly, the ESD protection is implemented on the OUT1 and OUT2 pins that power the gear motor. This is to ensure that ESD will not damage the L298N device.

Gear Motor

There are three implementation aspects for the gear motor: connector hub, encoder, and molex connector.

First is the connector hub. As the gear motor shaft have different form than the steam valve knob, there needed to be a custom made hub connector to attach the two shafts. Our advisor, Lee Harker, design and manufactured this custom hub connector to bridge the connection.

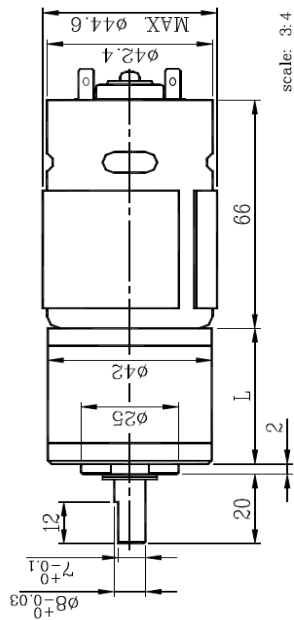


Figure 60: Gear Motor

The result was three sections to the hub connector. The first section is the aluminum connector piece that downsizes the shaft diameter from approximately 7mm to 6mm. The second section is the dark rectangular connector that fits over the steam valve knob and fits the 6mm shaft of the aluminum connector piece. Lastly, a strip of metal is used to lock the aluminum connector and dark rectangular connector together. This metal strip completes the hub connector, allowing the rotation of the gear motor shaft to rotate the steam valve knob.

The metal strip also serves a safety measure. In the case where the gear motor applies excessive force in rotating the steam valve, the metal strip will break under pressure and disconnect the aluminum connector from the dark rectangular connector. This will allow the gear motor shaft to turn freely and not twist off the steam valve knob.

The second component of the gear motor is the shaft encoder. The encoder is integrated on the back of the gear motor and has its own circuitry. The following schematic below shows the implementation of the encoder and how it communicates back to the microcontroller.

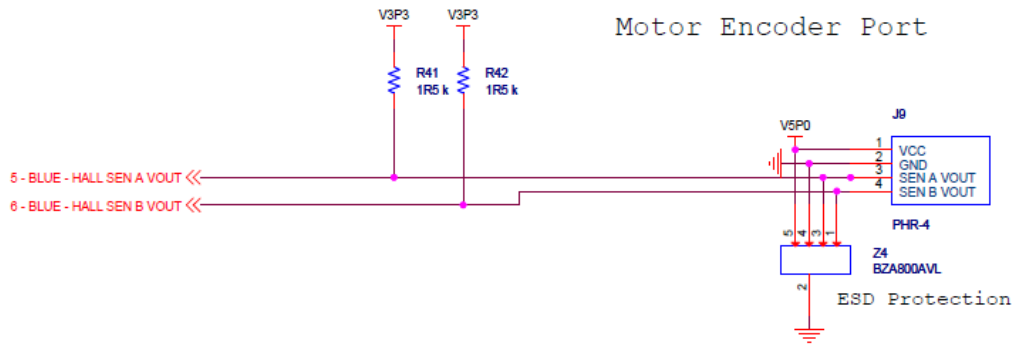


Figure 61: Motor Encoder Port Final Schematic Design

The encoder is powered by a 5V power line. The outputs of the encoder, Channel A and Channel B, are open drain outputs. Both channels are connected by a 1.5K ohms resistor to the 3.3V power line. The 3.3V line is used because the microcontroller can only read signal between 0V and 3.3V as gpio inputs. Channel A routes to pin 67 and Channel B routes to pin 87 on the microcontroller.

Due to the encoder not having enough resolution for 1 pulse per degree, there will be theoretically 24 missing pulses for each complete revolution of the gear motor. In addition to this discrepancy, the movement of the shaft of the gear motor lags the rotation of the motor connected to the encoder. This is because the gear ratio between the motor and the shaft incurs some time delay. This delay will result in further inaccuracies of tracking the valve position.

For this reason, we manually tested the gear motor and calibrated the count of the encoder with observed 360 degree rotation. We arrived at a calibration of a factor of 10% additional count is needed per each complete 360 degree rotation.

Lastly, the molex connectors for the gear motor power lines and the encoder ports are common molex connectors that are placed on the controller box PCB.

Software

There are three groups of functions written for the motor system. The groups include configuration functions, basic functions, and positioning functions.

Configuration Functions

The first group of functions is used for starting up critical components on the microcontroller and configuration appropriately prior to use.

```
extern static void readChanA(void)
```

```
extern void initInterrupt(void)
```

```
extern void startPWM(unsigned int CHANNEL_ID, unsigned int PWM_PIN, unsigned int PWM_FUNCTION, int PWM_DUTYCYCLE);
```

```
extern void stopPWM(unsigned int CHANNEL_ID, unsigned int PWM_PIN, unsigned
int PWM_FUNCTION);

extern void initMotor(void);
```

The function readChanA() is a routine that get executed whenever there is an interruption flag on the Channel A connected to the encoder port. Its basic function is to check for the desired target position and immediately stop the gear motor when the desire target position has been reach. Its other function is to increment or decrement the valve position variable corresponding to the direction of the rotation.

The function initInterrupt() initializes the interrupt for Channel A of the encoder on pin 67 of the microcontroller. This sets up the interrupt to catch the falling edge of Channel A. In addition, there is specific set up for AVR32_GPIO_IRQ_10 to enable the interrupt controller group that includes pin 67.

The function startPWM() initializes the given PWM channel with the desired operating frequency and desired duty cycle. The operating frequency set for PWM operation is set to 45Hz. This is achieved by using the microcontroller's internal RC clock of 115.2KHz, along with a prescaler of 256, and channel period of 10 units.

$$\text{Input waveform frequency} = (115.2\text{KHz}) * \left(\frac{1}{256}\right) * \left(\frac{1}{10}\right) = 45\text{Hz}$$

The PWM channel mode is left aligned and the polarity is high. This means that the default output of the PWM channel is low. For the motor system, there are two PWM channels used as input signals into the motor driver circuitry. MOTOR_IN1 and MOTOR_IN2 are PWM inputs. They are connected to corresponding PWM channel 0 and PWM channel 1.

The function stopPWM() initializes the given PWM channel with 0V output. Again, the stopPWM() has the same clock configuration and output configuration as startPWM(), but the duty cycle of the PWM channel is set to zero.

The function initMotor() initializes all the input and output signals in the motor system. It sets the MOTOR_DRV_R_EN to ground. It sets the MOTOR_IN1 and MOTOR_IN2 as PWM with duty cycle of zero. It then sets HALL_SENSOR_A and HALL_SENSOR_B as input pins with pull up resistors. The function initMotor() then calls initInterrupt() to set up the HALL_SENSOR_A interrupt. Lastly, there is an optional function calibrateMotor() that could be called within the initMotor() function. The function calibrateMotor() is discussed under the basic functions group.

Basic Functions

The second group of functions is use for basic rotation of the gear motor, overcurrent detection, and returning the current position of the steam valve.

```
extern int readCurrent(void)

extern bool checkOverCurrent(void)

extern int rotateMotor(double degree, int dir);
```

```
extern int getValvePosition(void);

extern void resetValvePosition(void);

extern bool calibrateMotor(void);
```

The function `readCurrent()` returns the current read from ISENAs from the motor driver L298N device. This function calls the `readADC()` function to convert the analog voltage signal at ADC channel 2 to a digital value. With the digital value, `readCurrent()` transforms the value back into a voltage reading and divides that over the sense resistor value to determine the current drawn by the gear motor.

$$\text{current reading} = \frac{ADC_{\text{digitalvalue}} * ADC_{\text{resolution}}}{Resistor_{\text{sense}}}, \quad ADC_{\text{resolution}} = \frac{3.3V}{1024} = 3.22mV \text{ per unit}, \\ Resistor_{\text{sense}} = 0.125 \text{ ohms}$$

Due to the switching behavior induced by the PWM input signals. The current draw by the gear motor will also have an AC frequency component. To remove this AC frequency component on the ISENAs line, we put the node through a low pass filter before reading the voltage reading at the ADC channel 2 pin of the microcontroller.

The function `checkOverCurrent()` returns the Boolean value true if the gear motor draws excessive current or false if the gear motor draws below a set threshold current. To determine this threshold current, we performed live testing on the steam valve. We monitored the current draw of the gear motor at the extreme ends of the steam valve. From our observations, we recorded approximately 550 mA drawn by the gear motor before it completely closes or opens the steam valve. In the `checkOverCurrent()` function, we first check to see if the current draw exceeds 500 mA. If the first exceeds 500 mA, we perform another current reading to see if it exceeds the 550 mA threshold. This set of testing allows us to remove noise from the reading and ensure we can accurately detect over current.

The function `rotateMotor()` takes into two parameters: degrees of desired rotation, and direction of the desired rotation. The function first determines which gear motor speed to use relative to the degree of rotation. If the rotation is larger than 180 degrees, the duty cycle of the PWM for MOTOR_IN1 and MOTOR_IN2 is set to be 100%. If the rotation is less than or equal to 180 degrees, the duty cycle used is 80%. The next step in `rotateMotor()` is to calculate the final valve position given the degree and direction. This is used to determine when the steam valve has reached its desired destination. The direction parameter has 0 as closing of the steam valve and 1 as opening of the steam valve. The function then checks to see if the final valve position is within the steam valve range of rotation. This step also includes setting the correct signals for MOTOR_IN1 and MOTOR_IN2 to configure the gear motor rotation with the desired direction.

Afterwards, the function `rotateMotor()` pulls the MOTOR_DRV_EN high to enable the gear motor rotation. The buzzer circuitry will also be turned off. This action will enable a high pitch buzzer indicating the rotation of the gear motor. The function then waits in a while loop. The while loop waits for the rotation complete flag that is set by the interrupt routine `readChanA()` when the steam valve has reached its final position. Inside the while loop, the function continuously calls the `checkOverCurrent()`

function and will pull the MOTOR_DRV_R_EN signal low in the case of over current. Upon receiving the rotation complete flag, the function breaks out of the while loop, resets the motor control signals, clear all the flags, turns off the buzzer circuitry, and exits the function.

The function getValvePosition() returns the current position of the steam valve. This function is mostly used for debugging within the main program.

The function resetValvePosition() performs a closing rotation on the steam valve. The proper control signals for implementing this rotation are set up by the function. Upon starting the rotation, the function waits in a while loop until the valve position reaches zero or there is an over current detection. After breaking out of the while, the function resets the control signals and reset valve position to zero.

The function calibrateMotor() is used for determining the rotating range of the steam valve. As our system will be implemented on many different steam valves, there is a significant need to automatically determine the rotation boundaries for the system. The function relies on the over current detection function to identify the fully closed position and fully opened position. The function first sets the steam valve to fully closed position and sets the valve position variable to 0. Then function then rotates the steam valve to the fully opened position using over current detection as an indicator. The function then sets the fully opened position to be equal to the current valve position. This will serve as the fully opened boundary of the steam boundary. To minimize the risk of the steam valve getting stuck at the fully opened position, we lowered the fully opened boundary position by 10%. This is to ensure the safety of the entire system because a large of heat endangers the system if the steam valve is stuck fully opened. Upon determining the range of rotation of the steam valve, the function returns the steam valve position back to 0. If there is over current detection before the steam valve reaches 0, the function will use this new position as the fully closed position and set the variable valve position to 0.

Positioning Functions

The third group of functions is use for positioning of the steam valve according to a set point and current temperature reading.

```
extern void setValvePosition(int newPosition);  
  
extern void setStableTemp(int stableTemp);  
  
extern void setTemp(int tempSetPoint, int currentTemp);
```

The function setValvePosition() takes in the desired valve position and rotates the steam valve to that valve position. The function calculates a modified valve position that takes into the calibration factor that addresses the inaccuracies of the encoder. With this modified valve position, the function determines the direction of the rotation and calls the rotateMotor() to perform the needed low level control signals.

The function setStableTemp() translate the desired stable temperature to a particular valve position. The function uses a switch statement to correlation temperature to a valve position, by calling the setValvePosition() function. The table below illustrates the translation.

Desired Temperature (Degrees Fahrenheit)	Valve Position
< 60	fully opened
60	0
61	0
62	0
63	0
64	0
65	360
66	450
67	540
68	630
69	720
70	810
71	900
72	990
73	1080
74	1080
75	1080
76	1080
77	1080
78	1080
79	1080
80	1080
> 80	0

Figure 62: Temperature Data Table

During our testing the steam valve in the multi-disciplinary lab of Coover 1316, we discovered that the shutting off the steam valve does not mean completely stopping the heat transfer within the pipes and the room. In addition to this effect, our testing period was in mid-March, when outside temperature had risen above 40 degrees Fahrenheit. Because of the outside temperature environment, our temperature-position translation table will need to be updated with new data during the next winter season.

The function `setTemp()` is controller algorithm function for choosing the appropriate temperature setting and valve position. The function takes in two parameters: current temperature, and set point temperature. The function first determines the upper and lower limits of the current temperature. This is done by adding and subtracting five degrees from the current temperature. If the set point is higher than the upper limit, the steam valve is turned to its opened position for maximum heat output. If the set point is lower than the lower limit, the steam valve is turned to its closed position for minimum heat output. The function then checks to see if the temperature difference between the current temperature and set point is one degree, the steam valve is turned to its closed position. This is because the steam

trapped in the vents will still heat the room even when it is fully closed. Lastly, the default option is to call the `setStableTemp()` function, which assigns a valve position to specific set point temperature.

In addition, the function `setTemp()` will track the current temperature reading and ensure that it is between the allowed 60 – 80 degrees Fahrenheit. If the current temperature reading is below 60 degrees, the steam valve is turned to the fully opened position. If the current temperature reading is higher than 80 degrees, the steam valve is turned to the fully closed position.

Encoder Calibration

As noted in the hardware of the encoder, there are inherent errors in the encoder readings due to missing pulses and the inertia lag between the motor shaft and actual motor because of the gear reduction. To reduce this encoder error, we perform an encoder calibration to mathematically adjust for additional pulses for more accurate turn. We performed this calibration by using a protractor to measure a 360 degrees rotation under different calibration factors. We performed this calibration through trial and error to arrive at the 1.10 calibration factor. We use this calibration factor by calculating the new pulse requirement for a particular rotation.

$$Degree_{calibrated} = 1.10 * Degree_{notcalibrated}$$

Steam Valve Calibration

To accommodate various steam valve their respective range of rotation, we designed the controller box to physically test the rotation range of any particular steam valve. We perform this calibration by using the `calibrateMotor()` function. The process of this calibration is documented under the previous software discussion. In addition to the software, the steam valve calibration must be performed under supervision. There is possibility of rusting of the steam valve that could give inaccurate testing of the rotational range. Supervision during the calibration is therefore needed to ensure that the test is completed correctly.

Temperature Sensor

Hardware

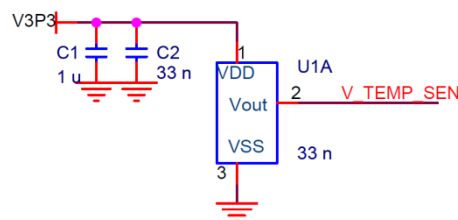


Figure 63: Temperature Sensor Schematic

The temperature sensor hardware consists of two capacitors connected to the 3.3V power line that feeds into the temperature sensor. The capacitors are there to ensure constant power into the temperature sensor. Besides that the 3.3V power line connection, the temperature sensor output routes straight to the microcontroller ADC channel 2 pin. On the microcontroller, the ADC channel 2 pin refers to pin PA23 in software or pin 75 on hardware.

In using the temperature sensor with the microcontroller, there is also a requirement to supply the microcontroller ADVref pin with 3.3V. The voltage is necessary in order for the microcontroller to convert the analog voltage level to a digital value relative to the 3.3V. The conversion is performed using 10-bit.

Software

The software requirement for using the temperature sensor consists of three important function calls.

```
extern signed short readADC(unsigned int ADC_CHANNEL, unsigned int
ADC_PIN, unsigned int ADC_FUNCTION);

extern int readTemp(void);

extern int cel2Fahr(double celsius);
```

Function readADC() returns the voltage value at a particular ADC channel pin. Function readTemp() specifically calls readADC() to read the voltage level at ADC channel 2.

The function cel2Fahr() is a simple math processing function that convert Celsius readings to Fahrenheit.

Lastly, the function readTemp() calls readADC() to read the digital value from the temperature sensor, convert the digital value into a Celsius reading, and calls cel2Fahr() to convert Celsius unit to Fahrenheit. The function readTemp() performs the following calculations.

$$temp_{volt} = ADC_{digitalvalue} * ADC_{resolution}, \quad ADC = \frac{3.3V}{1024} = 3.22mV \text{ per unit},$$

$$temp_{celsius} = \frac{(temp_{volt} - temp_{sensorbias})}{temp_{sensorslope}}$$

Within the readTemp() function, there are two parameters that need to be set in order for the conversion from the digital value to a Celsius temperature reading. One is the temperature sensor bias and the other is the temperature sensor slope.

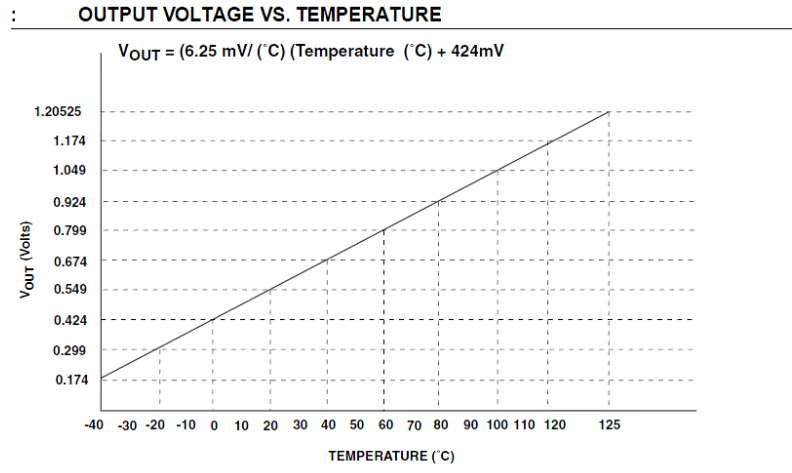


Figure 64: Output Voltage Vs. Temperature Graph

The datasheet for the temperature sensor notes the expected values for the two parameters. Due to manufacturing variability and different operating environment, we developed a calibration test to confirm and update the relationship between the output voltage of the temperature and the temperature in the environment.

Calibration

We tested the temperature sensor in a controlled environment using a temperature oven and monitored the voltage output of the temperature sensor. We recorded the results of the temperature sensor over a large temperature range, graphed the results, and extrapolated the temperature sensor zero degrees Celsius bias and slope parameter.

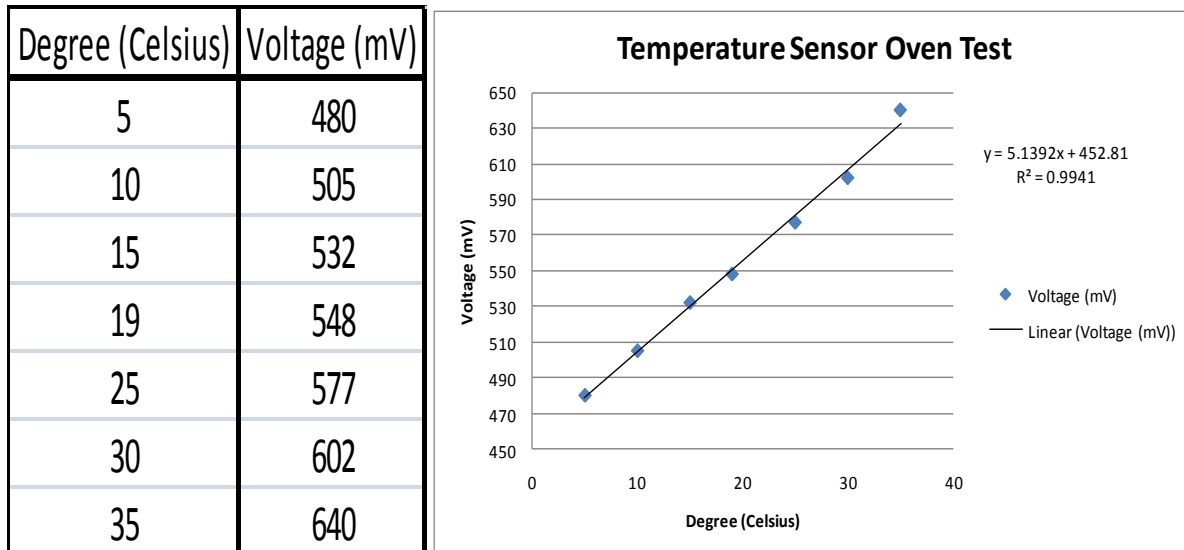


Figure 65: Temperature Sensor Oven Test

Upon completing the calibration testing, we confirmed that there are minor discrepancies between the datasheet and real world parameter values for the temperature sensor. Although our results differed from the datasheet, we plan to use the parameter values given in the datasheet. This is because the datasheet values may have gone strenuous testing and verification over the lifetime of the temperature sensor. Therefore, the datasheet should be more accurate than our calibration over the long term.

We plan to use the calibration method in the future to monitor the accuracy of the temperature sensor. This is important given that our system is very sensitive to temperature reading. After completion of the PCB board, we discovered that there is 7 degrees offset between the temperature sensor reading and the actual temperature. We programmed the software to account for this bias offset.

Ethernet Communication

The final implementation for Ethernet communication used the Media Independent Interface (MII) and the Ethernet transceiver chip SMSC LAN8700. However, due to hardware difficulties and time limitations, the implementation functionalities cannot be fully verified.

Hardware

The schematic for the Ethernet components shown below is implemented to the PCB design:

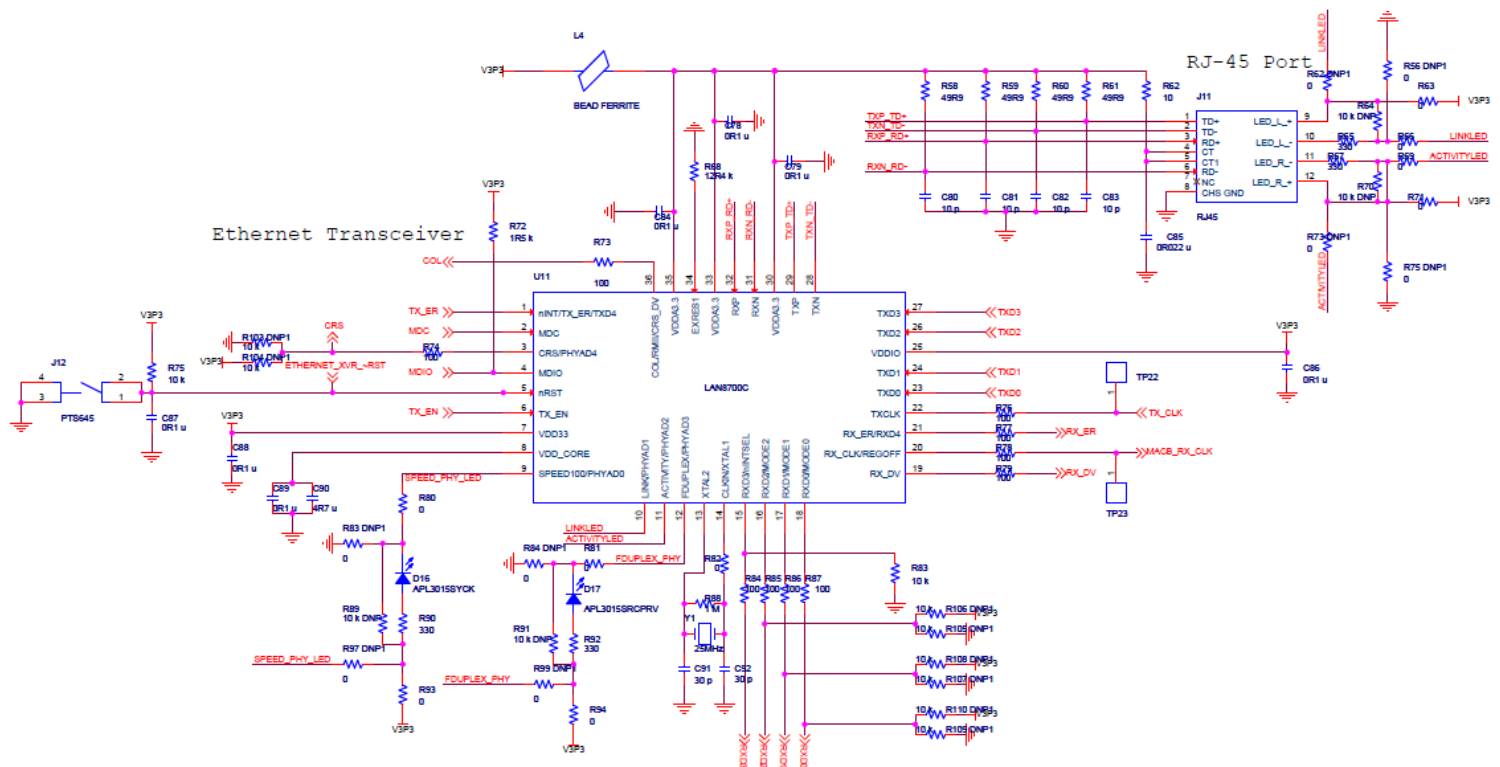


Figure 66: Final Ethernet Schematic Design

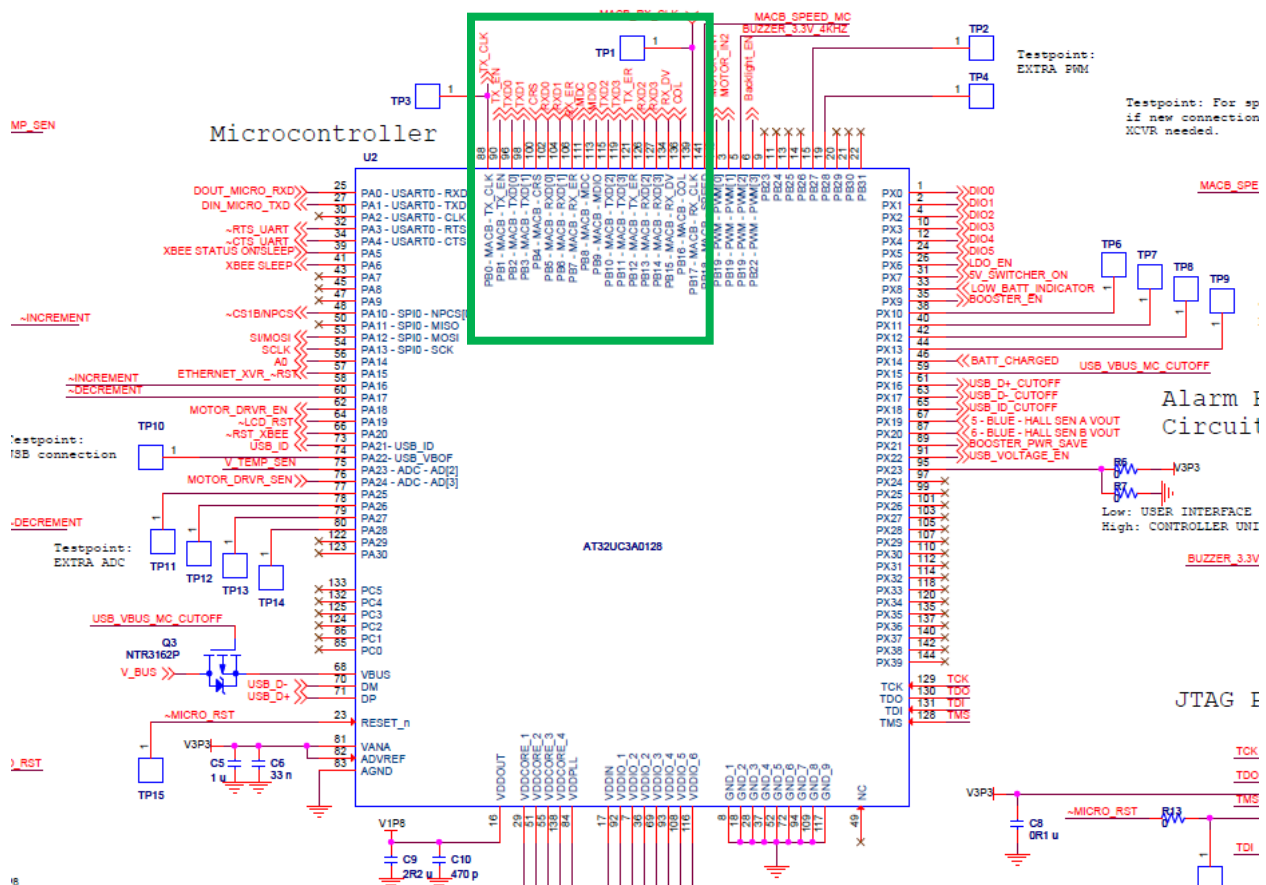


Figure 67: Microcontroller pin mapping for Ethernet components

As highlighted above in green, the Ethernet components are mapped out with the microcontroller pins corresponding to the MII implementation. The parts highlighted by the green boxes shows in Figure 65 below shows where the Ethernet components are placed in relationship to the PCB board layout. The RJ45 port is located between area 16 and 15.

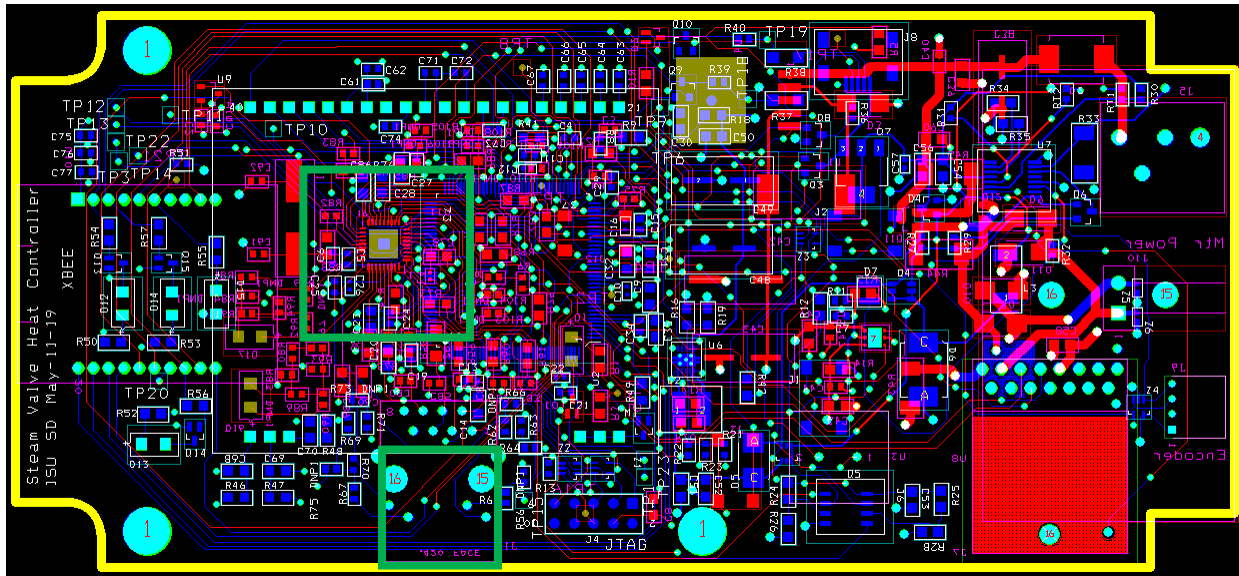


Figure 68: Ethernet PCB Design

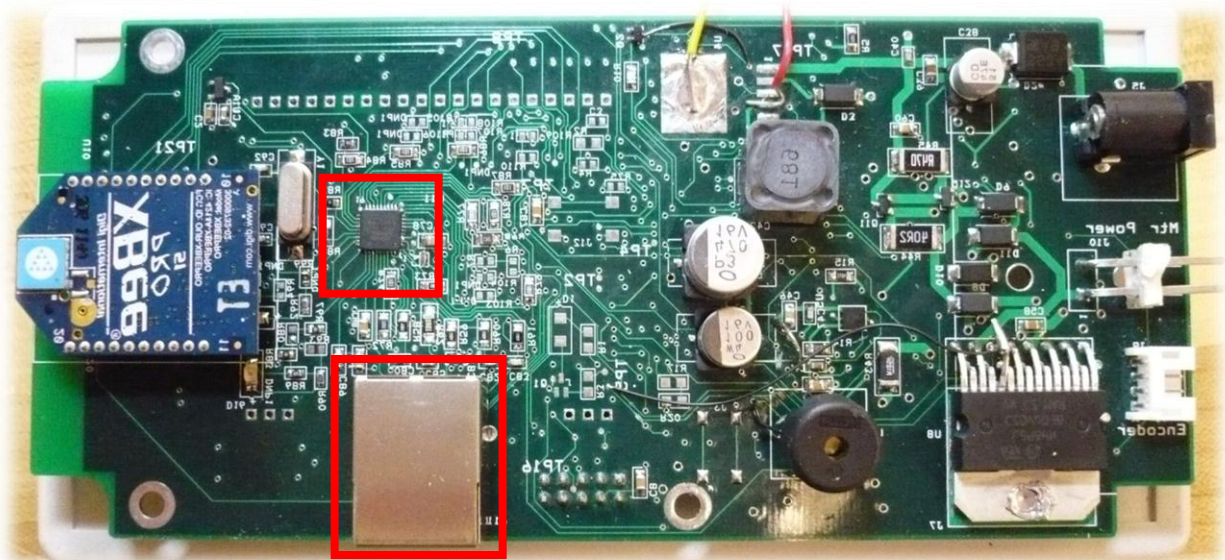


Figure 69: Ethernet PCB Board Population

The parts highlighted by the red boxes shows in Figure 68: Ethernet PCB Design and Figure 69: Ethernet PCB Board Population above shows where the Ethernet components are placed in relationship to the actual PCB controller box during board population. In Figure 69: Ethernet PCB Board Population, the RJ45 port is highlighted inside the bigger red box.

Software

To work with MII Ethernet standard, we changed the pin mapping for MACB_GIO_MAP to the following:

```
static const gpio_map_t MACB_GPIO_MAP = {
    {AVR32_MACB_TX_CLK_0_PIN, AVR32_MACB_TX_CLK_0_FUNCTION},
    {AVR32_MACB_CRS_0_PIN,  AVR32_MACB_CRS_0_FUNCTION },
    {AVR32_MACB_COL_0_PIN,  AVR32_MACB_COL_0_FUNCTION },
    {AVR32_MACB_MDC_0_PIN,  AVR32_MACB_MDC_0_FUNCTION },
    {AVR32_MACB_MDIO_0_PIN, AVR32_MACB_MDIO_0_FUNCTION },
    {AVR32_MACB_RX_DV_0_PIN, AVR32_MACB_RX_DV_0_FUNCTION },
    {AVR32_MACB_RXD_0_PIN,  AVR32_MACB_RXD_0_FUNCTION },
    {AVR32_MACB_RXD_1_PIN,  AVR32_MACB_RXD_1_FUNCTION },
    {AVR32_MACB_RXD_2_PIN,  AVR32_MACB_RXD_2_FUNCTION },
    {AVR32_MACB_RXD_3_PIN,  AVR32_MACB_RXD_3_FUNCTION },
    {AVR32_MACB_RX_ER_0_PIN, AVR32_MACB_RX_ER_0_FUNCTION },
    {AVR32_MACB_RX_CLK_0_PIN, AVR32_MACB_RX_CLK_0_FUNCTION},
    {AVR32_MACB_TX_EN_0_PIN, AVR32_MACB_TX_EN_0_FUNCTION},
    {AVR32_MACB_TXD_0_PIN,  AVR32_MACB_TXD_0_FUNCTION },
    {AVR32_MACB_TXD_1_PIN,  AVR32_MACB_TXD_1_FUNCTION },
    {AVR32_MACB_TXD_2_PIN,  AVR32_MACB_TXD_2_FUNCTION },
    {AVR32_MACB_TXD_3_PIN,  AVR32_MACB_TXD_3_FUNCTION },
    {AVR32_MACB_TX_ER_0_PIN, AVR32_MACB_TX_ER_0_FUNCTION },
};
```

The three functions listed below were implemented for Ethernet interface and website interface communication:

```
// send current room data via socket packets to the website interface
packets macb_send_socket_packet(macb_packet_s *pkt)
```

```
// get room temperature requests from the website interface
packets macb_receive_packet(macb_packet_s *pkt))
```

```
// mainly used for testing on internet connectivity
void macb_send_ping_response(macb_packet_t *pkt)
```

Many other function calls and methods are involved, such as void start_pll(void), which starts up a PLL at 2.5 MHz for the 10Base-T Transmit on MII.

Issues

In order to transmit packets between the Ethernet interface and the website interface, the microcontroller needs to support the operational clock rate of 2.5 MHz. This is the required rate in which the 10Mbps serial data stream operates for the SMSC LAN8700 chip on MII. It was assumed that the AT32UC3A0128 MCU was capable of operating up to 12MHz from its internal clock. However, this is not the case. In order to fix this, we need to have an external crystal that supports 2.5 MHz connected

to the MCU on the PCB board. By the time this issue was discovered the amount of time left was very limited and the Ethernet functionalities were not able to be tested on the PCB board.

Enclosure

The enclosure used is shown below. The green box indicates where the LCD is visible to the user once the enclosure is properly cut. The two red circles indicate where the push bottoms are placed. The controller box and control panel PCB will be placed inside the enclosures.

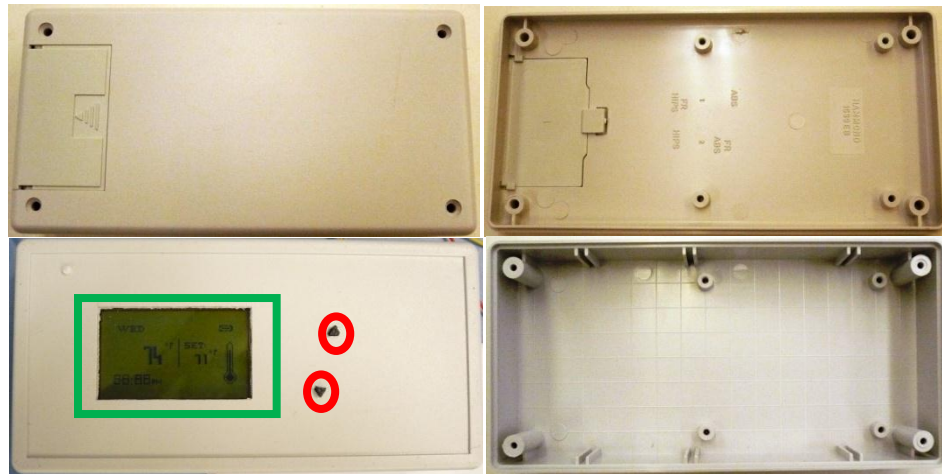


Figure 70: Enclosure

Power Supply

The power supply implemented on the final PCB includes the following:

- 120 Volt 60 Hertz to 12 Volt AC to DC Switching power supply : Motor
- 12 Volt to 5 Volt AC to DC Switching power supply: Motor Driver Logic Circuitry
- 5 Volt to 3.3 Volt LDO: Low Drop Regulator: all other component
- 3.3 Volt to 1.8 Volt Low Drop Regulator internal microcontroller supply basic functionalities

Buzzer

Hardware

The buzzer circuitry is used by both the controller box and control panel. The schematic documents the buzzer circuitry and actual implementation on the PCB board.

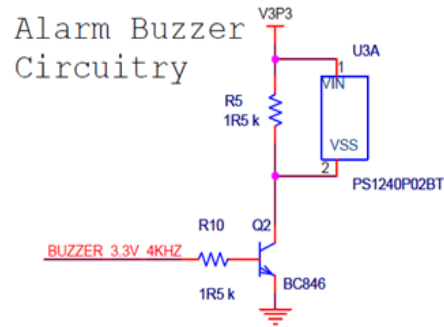


Figure 71: Buzzer Schematic

The buzzer circuitry consists of using a parallel 1.5 kohms resistor with the buzzer to prevent overcurrent through the buzzer. The buzzer control signal from the microcontroller is PWM channel 2, or pin 5 on the microcontroller IC. The frequency used by the PWM channel is approximately 3.84 kHz.

$$\text{Buzzer waveform frequency} = (115.2\text{KHz}) * \left(\frac{1}{30}\right) = 3.84 \text{ kHz}$$

Software

There are only two functions used for control of the buzzer.

```
extern void startBuzzer(void);
```

```
extern void stopBuzzer(void);
```

The function startBuzzer() initializes the PWM channel 2 controlling the buzzer and output the 3.84 kHz square wave. Calling this function will make the buzzer vibrate and create the desired audio noise.

The function stopBuzzer() stops the waveform output of PWM channel 2 controlling the buzzer. Calling this function will cause the buzzer to stop vibrating.

Push Button

Hardware

The push button circuitry consists of voltage divider resistors and capacitor. The circuitry keeps 3.3V high at the node connected to the push button pins on the microcontroller.

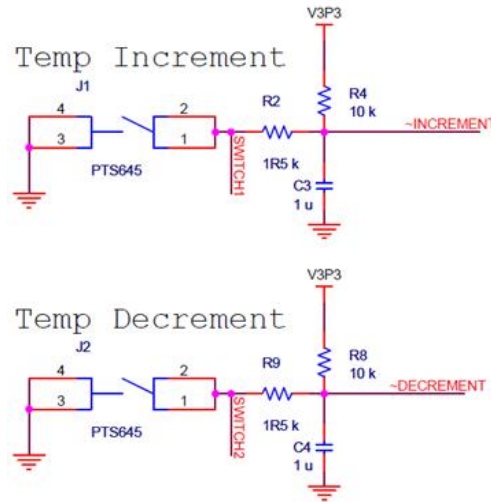


Figure 72: Push Buttons Schematic

The two gpio pins used for the push buttons are pins 58 and 60 on the microcontroller IC. Pressing the push buttons will short the microcontroller node to ground thereby register a valid press.

Software

There are two functions used in initializing the push button and monitoring the set point value.

```
int UI_setpt;

extern void init_Push_Button(void);

static void pushButton(void)

extern int get_setPt(void);
```

The variable UI_setpt keeps track of the most current set point value. Any press performed on the push button will either decrement or increment the UI_setpt value by the respective push button.

The function init_Push_Button() initializes pins 58 and 60 as gpio pull-up pins so the microcontroller can accurately read the input signals. The function will also assign the interrupt register to pins 58 and 60. If the interrupt controller reads falling edge on either pins, the microcontroller will call the interrupt routine pushButton().

The interrupt routine pushbutton() first identifies which push button was pressed. Pin 58 corresponds to the increment push button. Pin 60 corresponds to the decrement push button. By reading the interrupt flag for each pin, we either increment or decrement the UI_setpt variable. We also limit the range of UI_setpt between 60 and 80.

The function get_setPt() returns the UI_setpt value for use at the top level of the control panel program.

Recharging Circuitry

The recharging circuitry contains:

- Safety temperature shut off
- Quick recharging circuitry
- Uses outer switching circuitry to supply the appropriate current, and uses USB.

Due to the time limitations, the recharging circuitry was not populated in the PCB.

Website Interface

The website interface used Spry Validation Control, which generated Client-side validation messages when the username or password is left empty:

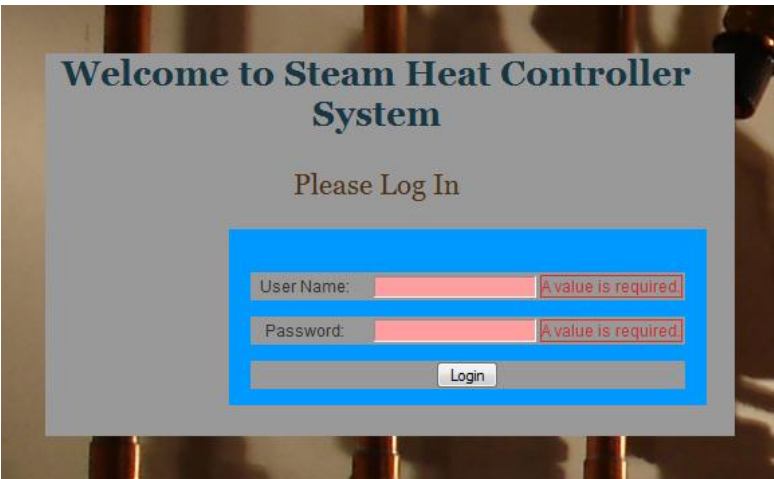


Figure 73: Website Interface Login Page

Once the user name and password are entered by the user, data entry forms are submitted and their values are compared with the values stored inside the back end database. The initial username, password, and access level fields are as set in the table shown below:

Table interface showing user database records with pagination and sorting options.

Options: Show: 30 row(s) starting from record # 0 in horizontal mode and repeat headers after 100 cells. Sort by key: None.

	username	password	Accesslevel
<input type="checkbox"/>	webAdmin	webPassw0rd	0
<input type="checkbox"/>	webUserRoom1	webUserPassw0rd1	1
<input type="checkbox"/>	webUserRoom2	webUserPassw0rd2	2
<input type="checkbox"/>	webUserRoom3	webUserPassw0rd3	3
<input type="checkbox"/>	webUserRoom4	webUserPassw0rd4	4
<input type="checkbox"/>	webUserRoom5	webUserPassw0rd5	5

Check All / Uncheck All With selected: [edit icon] [delete icon]

Options: Show: 30 row(s) starting from record # 0 in horizontal mode and repeat headers after 100 cells.

Figure 74: Website Interface Login User Database

The database currently has a unique access level for each room users and also the admin. The current website interface supports up to five different rooms and is able to expand as needed. The web interface has successfully implemented a secure login functionality that prevents malicious users from setting undesired temperature settings. The current access levels contains unique numeric values that serve as the key for each room users and administrator.

The website interface is currently running in WAMP Server locally and can be put to remote via Iowa State University server if server space is allocated.

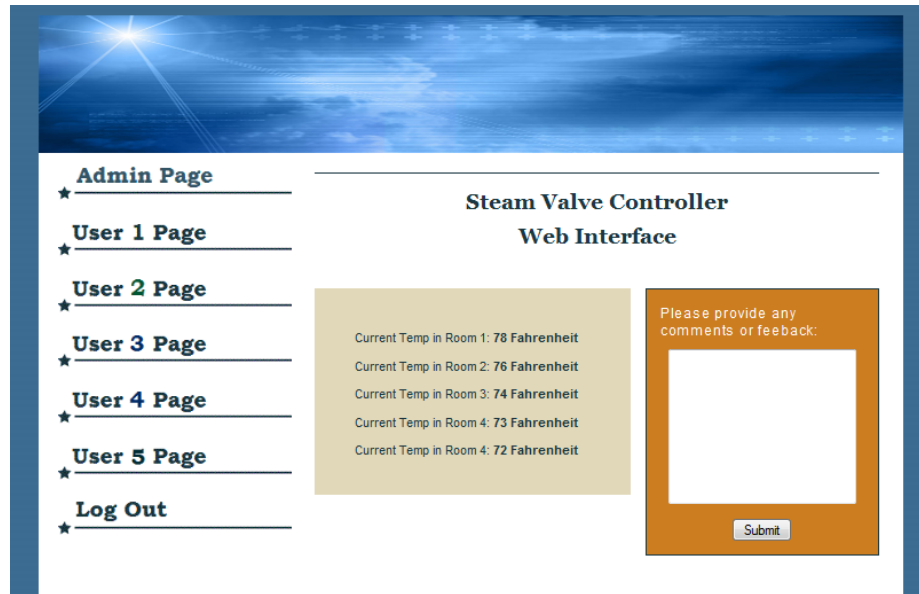


Figure 75: Website Interface Welcome Page

The figure above shows the welcome page for the website interface. Once a valid user (any user from any rooms or the admin) logs in via the login screen they are directed to the welcome screen. The welcome screen contains a section for the current user to provide feedbacks or comments to the administrator or the lab coordinator.



Figure 76: Website Interface Admin Page

The admin has the ability to click on the “Admin Page” to the left but does not have any privilege to access any user pages. The admin has the ability to request for any temperature preferences for any rooms and also change the room user password and user name. There is a log out option that allows the current user to log out.



Figure 77: Website Interface User 1 Page

The figure above shows what a typical room user would see. As noticed, the user can only set their room preferences and also get to see the current temperatures in all rooms.

Limitations

The website interface is currently limited because of the Ethernet communication. It is currently unable to communicate with the Ethernet communication from the controller box. The current temperature readings are arbitrarily set and also the requests packets once the user clicks on “send” is not sent to the controller box. PHP Socket is in place to handle the Ethernet communication, however, due to the issues with the Ethernet components, the full functionalities of the website interface cannot be verified.

Controller Algorithm

Even though the team initially implemented the PID controller algorithm, due to the long time constant, the algorithm had to be modified. The team implemented a simpler weighted average algorithm that assigns different weights to different rooms, where the middle room takes the most weight since the middle room is the average for a linear temperature distribution room system. Below the controller algorithm written in C:

```
for(panel_i=0; panel_i < UI_count; panel_i++) {  
  
    switch(UI_count) {  
  
        case(1):  
            weight = 1;  
            break;  
        case(2):  
            switch(panel_i) {  
                case (0): weight = .4; break;  
                case (1): weight = .6; break;  
            }  
            break;  
        case(3):  
            switch(panel_i) {  
                case (0): weight = .3; break;  
                case (1): weight = .4; break;  
                case (2): weight = .3; break;  
            }  
            break;  
        case(4):  
            switch(panel_i) {  
                case (0): weight = .2; break;  
                case (1): weight = .3; break;  
                case (2): weight = .3; break;  
                case (3): weight = .2; break;  
            }  
            break;  
        case(5):  
            switch(panel_i) {  
                case (0): weight = .15; break;  
                case (1): weight = .2; break;  
                case (2): weight = .3; break;  
                case (3): weight = .2; break;  
                case (4): weight = .15; break;  
            }  
            break;  
    }  
}
```

```

    }

    currentTemp_sum = currentTemp_sum + data_measurement-
>curr_temp_measurement_UI[panel_i];
    setPt_sum = setPt_sum + weight*data_measurement-
>set_pt_measurement_UI[panel_i];
}

```

PCB Board Integration

PCB

The PCB is done in a four layer board and includes all the components needed for the controller box and control panel. Each major component is populated onto the PCB board and its functionalities are verified before the next major component is populated on the PCB board.

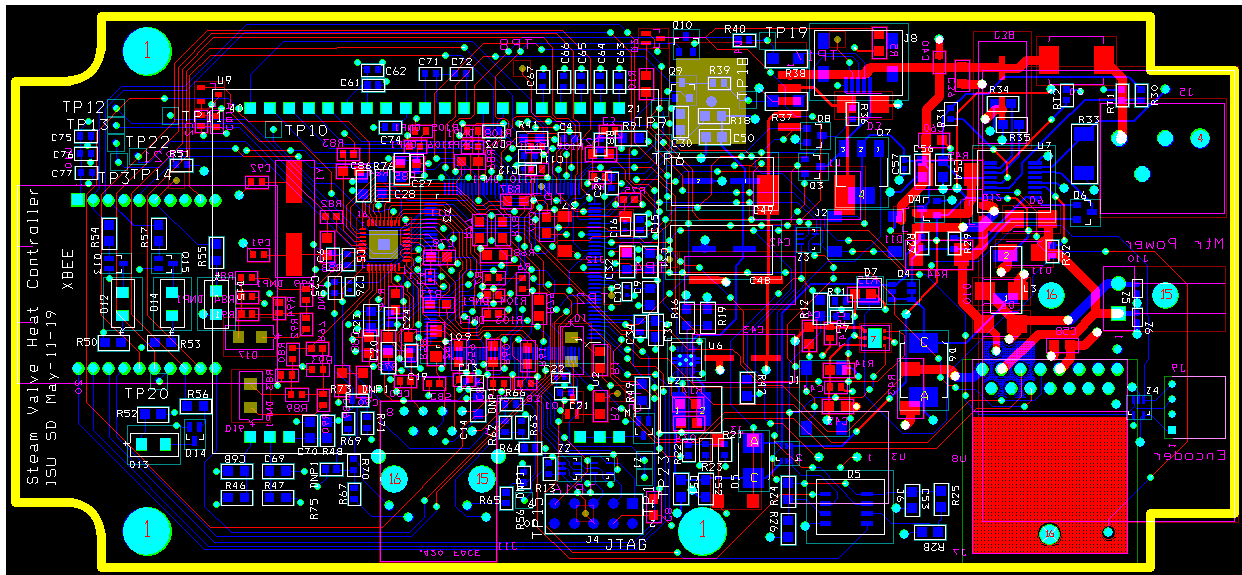


Figure 78: PCB Board Design

Hardware

Software

92

Control Panel

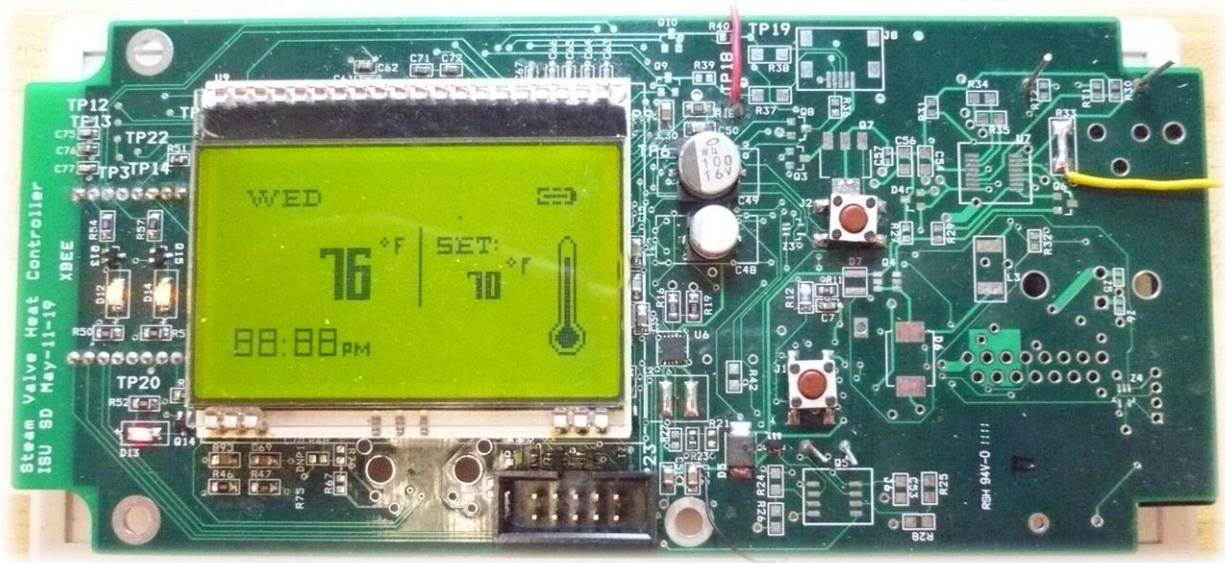


Figure 80: Control PCB Board Population

Hardware

The hardware of the control panel consists of the enclosure and populated PCB (microcontroller, temperature sensor, LCD, power supply circuitry, wireless transceiver, and buzzer).

Software

The main program of the control panel will be to output information through the LCD for users to view.

IV. Testing Plan and Results

Testing Approach Considerations

Testing requirements considerations

Our testing approach will be broken into two phases. In the first phase each of the components will be tested individually to ensure they meet their own individual requirements. This phase will also assist in developing our understanding of each component. After this phase our integration plan may need to be altered as we understand each component's functionality better. Then in the second phase the entire integrated system will be tested including functional and non-functional tests. This will have to take place during the winter months and will be a vital link in the success of the project. A PID (Proportional Integral Derivative) feedback control algorithm will be used to actually control the temperature of the rooms by adjusting the steam valve. This type of controller will require significant adjustments to the controller's parameters (coefficients) for the controller to be effective.

Phase 1

Testing requirements and procedures for each component are outlined below:

Wireless transceiver: There are a number of tests that must be performed. The unit must be tested to ensure basic functionality, range, and reliability. The most basic test would be to test basic functionality to ensure the unit can properly transfer data bytes. Then the transceiver's range must be tested. This must be done in the product's intended environment. A variety of environmental factors must be tested, including having a transceiver that needs to reliably communicate up to 3 rooms away. Incorporated into the range tests will be a test of reliability, with reliability being measured as the percentage of the transmitted bytes that were received by each transceiver as a function of the range.

LCD display: There is a multitude of tests that must be performed. The first tests to be performed should ensure basic functionality and reliability. The most basic test would be to test basic functionality to ensure the unit can properly display data and graphics. Then, the LCD backlight should be tested. This should be done in the product's intended environment across various lighting conditions and temperature ranges. This testing will allow us to set the appropriate contrast and brightness and allow for minimum power consumption. Pixel reliability needs to be measured in order to make sure that the LCD is displaying the transmitted data properly.

Temperature Sensor: In order to test the accuracy of the temperature sensor, the team will compare its readings with temperature readings from reliable readings (thermometer). Data recordings from both the reliable reading and the readings from the temperature sensor will be recorded overnight and analyzed for accuracy. The team will also measure the temperature sensor's accuracy at 32 degrees Fahrenheit and 100 degrees Fahrenheit

Mechanical interface: Our testing approach to implementing the mechanical system began prior to selecting the individual parts. We were fortunate enough to have on hand parts that we could test and develop the software prior to selecting our final components.

Testing prior to final component selection

With the following components available from the Ecpe resource department, we were able to write test code for mechanical functionality.

- Arduino - microcontroller
- Pittman GM9000 - gear motor
- HEDS-9100 - optical shaft encoder
- LMD18200T - motor driver IC

As a result of our early testing, we were able to learn the following objectives that would aide in testing and integrating the final selected components.

- functional code available to be ported to selected microcontroller
- experience with circuit layout for testing
- aide in selecting approaches for mechanical system

The opportunity to work with available material was helpful in deciding appropriate technologies, approaches, and requirements for the final component selection. Once we had the selected components on hand, we proceeded to test them using software coded in the initial testing.

Testing after final component selection with Arduino

With the following components, we are currently testing the components and ensuring the software coding will implement the correct mechanical functionalities.

- Arduino - microcontroller
- IG42 - gear motor
- Integrated magnetic shaft encoder
- L298N - motor driver IC

Notice that we are still using the Arduino to code for and test the functionality of the gear motor, shaft encoder, and motor driver IC. This is because we want to have a working mechanical system in place prior to January 1st of 2011 so that we could begin temperature profiling of the rooms. Since our selection of the microcontroller will be later on in the design process and complexities of the final selected microcontroller would require additional time to code for, we will use the Arduino to provide preliminary mechanical functionality while we bring up the final coding on the selected microcontroller.

We used the test circuits, available in Appendix A. With the testing, we were to confirm motor control accurate to ± 10 degrees, therefore further investigation will be need to increase accuracy of the motor control.

Our current progress is to improve the accuracy of the motor control while bringing up the testing using our final selected microcontroller, the Atmel U3A0128.

Testing after final component selection with that Atmel U3A0128

Testing of the selected mechanical components using the Atmel U3A0128 will employ the same test circuitry shown in Appendix A. However, we will port the Arduino code to the Atmel U3A0128 prior to mounting the mechanical in place and performing a final testing.

Exiting testing phase criteria

The criteria that need to be met before exiting the testing phase of the mechanical system include:

Testing Criteria	Satisfactory (Y/N)	Comments
Directional tracking		
Speed control		
Accuracy		within 5 degrees
Torque requirement		greater than 0.95 N m
Debugging interface		report position, direction, current
Recalibration		identify fully closed position
Overcurrent detection		stop operation when current > 1.3 A
Stalling detection		stop operation when motor report stalling
Mechanical stability		operate continuously for 1 week
Electrical stability		operate continuously for 1 week
Operational stability		operate continuously for 1 week

Figure 81: Mechanical interface Test criteria

The mechanical system undergoes extraneous testing due to safety implications. We want to ensure that the mechanical system will not endanger the steam valve and the Coover building as a whole. We will evaluate the final mechanical system to the testing criteria above and additional testing as we progress in its development.

Ethernet Transceiver: In order to test the functionality of the Ethernet transceiver, the team will first transmit a 1 byte data packet across the Iowa State University's network to the Steam Heat Controller. The team will verify the data integrity by displaying the data transmitted on the LCD. The team will then transmit a 1 byte data packet from the Steam Heat Controller to Iowa State's network and verify the data integrity by printing the data to the web interface. Larger size data will be used to test data integrity once the 1 byte data packet test is successful.

Website Interface: To test the website interface, the team will try to authenticate the website interface as admin and user and try to respectively set temperature preferences for the steam valve controlled rooms. The team will also be implementing series of tests to make ensure that reliability and integrity of the data received.

Microcontroller: There are a number of tests that must be performed. The unit must be tested to ensure basic functionality and reliability. The most basic test would be to test pin functionality to ensure that the units perform the appropriate functions and have good signal integrity. The voltage rails need to be tested in order to ensure low ringing on the lines. The microcontroller's operation needs to be tested at a wide range of temperatures in order to assure a reliable product. The signal communication lines need to be tested for reliability, with reliability being measured as the percentage of the transmitted bytes that were received correctly.

Control algorithm: The control algorithm must undergo extensive testing and revision in the final stages of the project. The control algorithm is a software implementation that is dependent on the hardware configuration and hence must be part of the last stages of development. However as a critical component to the success or failure of the project there must be sufficient time allocated to the testing, tuning and revision of the control algorithm.

The testing of the control algorithm will need to include live testing where the system is ran and monitored over a series of tests under different environmental standards and operating conditions. Testing will begin with a single room that is controlled using a single control panel and controller box. The testing will then be expanded to include multiple rooms with multiple user interfaces.

Temperature Sensor: In order to test the accuracy of the temperature sensor, the team will compare its readings with temperature readings from reliable readings (thermometer). Data recordings from both the reliable reading and the readings from the temperature sensor will be recorded overnight and analyzed for accuracy. The team will also measure the temperature sensor's accuracy at 32 degrees Fahrenheit and 100 degrees Fahrenheit

Phase 2

The phase two tests are divided into three main categories, hardware, software and functional testing. The testing requirements and procedures for each component are outlined below:

Hardware Testing Approach Considerations:

The hardware will be tested upon receiving and populating the first prototype of the PCB board. It will be tested in the senior design lab using standard bench equipment such as the oscilloscope, multimeter, and power supplies. The following portions of hardware will be tested along with particular subsections indication the appropriate tests involved:

Section	Satisfactory Y/N	Notes	Comments
Power:			
Test voltage levels, voltage ripple, voltage stability, and voltage behavior due to transients of all power supplies over their full loads	Y	3.3 V booster circuit needs re-route in order	Max voltage ripple is 50 mV. All voltages within 5% of target value
Test the limits on inputs to all switchers and LDOs	Y	3.3 V booster circuit needs re-route in order	LDO limit: 10 V. 5 V Switcher Limit: 20 V
Test typ. current draw for all major power rails	Y		Controller Box 3.3 V: 90 mA max 5.0 V: 30 mA max 12.0 V: 2.64 A max Control Panel: 3.3 V: 200 mA max
Test typ. noise levels on all major IC power rails	Y		Typical voltage ripple on IC power pins is approximately 25 mV.
Test the limits of current limiter circuit	Y		Current Limit: 2.64 A
Test reverse protection circuitry	Y		Reverse protection tested up to 30 V. Satisfactory result.
Ensure proper power supply sequencing	N/A	3.3 V booster circuit needs re-route in order to test	Controller Box: 12 V sequenced prior to 5V sequenced prior to 3.3 V
Microprocessor:			
Test high speed signal lines for signal integrity:			
SPI signal integrity CD lines	Y		Typical rise time < 30 ns
USART signal integrity on transceiver lines	Y		Typical rise time < 30 ns
Ethernet signal integrity on the transceiver lines	Y		N/A
PWM lines on motor driver lines	Y		Typical rise time < 30 ns
Test system reset functionality	N/A		Reset functionality only resets the mP clocks and not the system
Test microprocessor can be JTAGged and programmed as needed	Y		JTAG and programming fully functional
Test microprocessor can communicate to LCD, Xbee transceiver, Ethernet	Y		Proper handshakes between mP and peripherals. 100 %
Test RJ-45 and USB port lines for proper level conversion	N/A	Need to add 12 MHz	Using the current PCB, we could not test without an external
Test LED systems to determine if operating in safe current limits	Y		LED circuitry has appropriate current draw and operating limits.
Ensure proper clock start up sequence	Y		mP clock starts up accordingly. Only one clock used currently.
Ethernet Transceiver:			
Test high speed signal lines for signal integrity:	N/A	Need to add 12 MHz external crystal to mP	functionality, need to add a 12 MHz mP.
Ethernet signal integrity on the microprocessor lines	N/A	Need to add 12 MHz external crystal to mP	functionality, need to add a 12 MHz mP.
Ethernet signal integrity on the RJ45 lines	N/A	Need to add 12 MHz external crystal to mP	functionality, need to add a 12 MHz mP.
Test system Reset functionality	N/A	Need to add 12 MHz external crystal to mP	functionality, need to add a 12 MHz mP.
Test transceiver can communicate to Iowa State University network and vice versa	N/A	Need to add 12 MHz external crystal to mP	functionality, need to add a 12 MHz mP.
Test Ethernet Transceiver communication to microprocessor	N/A	Need to add 12 MHz external crystal to mP	The design is a prototype. In order to obtain Ethernet functionality, need to
Push Buttons:			
Test for noise, bounce, and transient levels during switching for slider	Y		No switch bounce due to filtering design.
Test for proper signal rise/fall times during switching for slider switches and	Y		Maximum rise/fall time 3 ms due to rc time constant.
Unit casings:			
ESD testing to determine the quality of design	Y		ESD does not seem to penetrate the casing. Does not affect performance due to
Temperature sensitive testing in order to determine endurance	Y		Does not deform in the operating temperature range of 40 to
LCD:			
Test under temperature limits to determine quality dynamics and	Y		Tested from 20 to 100 degrees Fahrenheit. Contrast and quality
Test under various lighting conditions to determine proper	Y		Decided to use medium strength to save energy. Sufficient in
Temperature sensor:			
Test accuracy of the temperature sensor in various environmental conditions	Y		Tested in hot, cold, humid, and dusty conditions. Always achieve +/- 1 degree Fahrenheit
Test limits of operational temperature	Y		Tested between 40 to 100 Fahrenheit. Achieved +/- 1 degree
DC Motor:			
Test operation under extensive temperature range	Y		Tested in hot, cold, humid, and dusty conditions. Temperature range is 40 to 120 degrees Fahrenheit.
Test output torque and make certain that it meets required specification	Y		Tested typical torque of steam valves around Coover and set the motor torque output to
Test quality of performance using different driver circuits	Y		Tested three driver circuits, obtained similar performance and used the least expensive

Figure 82: Hardware test plan and criteria

Testing will be accurately determined through thorough implementation and careful observation of system behavior. Upon completing a test, the results will be documented along with information pertaining to whether further action needs to be performed in order to fix that particular portion of the system. Benjamin Jusufovic and Curtis Mayberry will perform the hardware testing. One will test a section and the other will verify the results through another implementation of the procedures.

Software Testing

LCD, Temperature, and Microcontroller				
Test Case		Pass	Fail	Comments
1.	Display all characters supported by the LCD			
2.	Verify correct display results for all components			
	a. Correctly displays current temperature readings			
	b. Warning for low battery			
3.	LCD display updates accordingly after user input			
4.	LCD displays desired output after system reboot			
Ethernet Components				
Test Case		Pass	Fail	Comments
1.	Receive buffer manager write of 1 byte data			
2.	Receive buffer manager read of 1 byte data			
3.	Transmit data DMA read of 1 byte data			
4.	Receive data DMA write of 1 byte data			
5.	Transmit buffer manager read of 1 byte data			
6.	Transmit buffer manager write			
Website Interface				
Test Case		Pass	Fail	Comments
1.	Password authentication for average user			
2.	Password authentication for admin			
3.	Administrator’s ability to add, remove, or modify user			
4.	Administrator’s ability to overwrite temperature preferences			
5.	Verify that JDBC correctly communicates with database			
6.	Web interface correctly displays the correct temperature for all rooms			
7.	Verify that web interface correctly communicates with the steam valve microcontroller			
8.	Verify that the user request is processed and sent to the steam valve microcontroller			
9.	The status of the overall system is displayed correctly			
10.	The user feedback is archived displayed for the admin			
11.	The web interface correctly displays the data interpretation from the data collected from the microcontroller			
12.	The database correctly stores the data sent from the microcontroller			
13.	The web interface correctly retrieves data after system failure			
14.	The web interface displays appropriate message after invalid user requests			

Figure 83: Software testing plan and criteria

Functional Testing

The functional testing plan was developed to test the system as a whole. A great variety of tests were developed in order to verify the functionality of the unit. The test results of the system provide final verification of the project's success.

Functional Testing		
Basic Functionality		
Test	Success (P/F)	Criteria
temperature set pt. is higher than the current temp		valve opens more
temperature set pt. is lower than the current temp		valve closes more
set temperature preference and then check to see if set pt. is reached		valve is close (within 20% of turn range) to being closed
Set all preferences to 68°		temperature is set to within 2° c after
Stand 3 ft. away from the LCD and read the temperature		LCD is visible, can be read from 3 feet away
Attempt go below the lower temperature limit of 60°f		UI should stay at 60°f
Attempt go above the upper temperature limit of 80°f		UI should stay at 80°f
Valve Calibration		
Test	Success (P/F)	Criteria
Increment temperature preference by one °F starting from 60 to 80.		valve should open slightly more each time the temperature preference goes up
Decrement temperature preference by one °F starting from 80 to 60.		valve should close slightly more each time the temperature preference goes down
Range Tests		
Test	Success (P/F)	Criteria
Send one message to the transceiver at each range between 0 walls to 4 walls		Range of up to 300 ft. with up to 4 walls between the UI and the control box
Send 100 messages to farthest range (300 ft., 4 walls)		Have 95% of sent messages received successfully
Response Test		
Test	Success (P/F)	Criteria
User sets temperature to 78°F from 72°F		The room should reach approximately 78°F in two hours
User sets temperature to 72°F from 78°F		The room should reach approximately 72°F in two hours
Equalization and weighting Test		
Test	Success (P/F)	Criteria
Start from temperature set point of Room1: 75, Room2: 78, Room3: 83		The room should reach approximately 78°F depending on states
Start from temperature set point of Room1: 60, Room2: 75, Room3: 85		The room should reach approximately 75°F depending on states
Environmental and extreme Tests		
Test	Success (P/F)	Criteria
Put the steam valve control system in the oven at 100°F		Controller box functions after running for 10 hrs. in 100° F
Put the steam valve control system in the fridge at 100°F		Controller box functions after running for 10 hrs. in 10°F Temperature
Set all preferences to maximum		The temperature is set to within 2° F
Set all preferences to minimum		The temperature is set to within 2° F
Power and Charging		
Test	Success (P/F)	Criteria
Plug the controller into the USB when the battery level is at 50%		The unit indicates charging state
Drain the controller battery level to less than 10%		The buzzer sounds indicating low battery state
Plug the controller into the USB when the battery level is at 100%		The unit will indicate charging complete state
Monitored extended use testing		
Test	Success (P/F)	Criteria
Run system for 1 week, monitoring the temperature		The temperature is set within one time constant
Run system for 1 week, monitoring the LCD quality		The contrast of the LCD should remain the same, the LED backlighting should be at medium brightness
Press push buttons over 300 times		The incrementing and decrementing of the temperature should be functional

Figure 84: Functional test plan and criteria

Security considerations

The major security considerations to be addressed are the wireless interface and the web interface. Both of these are vulnerable to attack by outside sources. Below are the security considerations for each:

Network interface: There are two main security vulnerabilities in the network interface. First, the traffic going from the internet interface to the Iowa State network might be prone to attacks. Anyone from the room will have access to the physical Ethernet connection interface connecting to the Iowa State network and alter the data sent to the web or instructions sent from the web. A possible solution is to lock the Ethernet connection between the microcontroller and Iowa State network from the users in the room.

The wireless traffic might be prone to attacks. The data broadcasted uses a common protocol and can be easily picked up by surrounding devices using sniffing programs. This gives attackers the ability to perform attacks to the network interface using false authentication and change the temperature of the room. A possible solution is to encrypt data being broadcasted, and create an authentication password.

Wireless transceiver: The Xbee includes built-in security features. These security features include the addressing method used by the transceiver. There are over 65,000 unique addresses that can be set for each transceiver. The source and destination of each message is addressed. The system also has the ability to encrypt the sent data using 128-bit AES (Advanced Encryption Standard). This will help secure information sent over the wireless serial link between the control panel and the controller box. Even though the Xbee does an excellent job of securing data when configured properly, the system is designed so that the data being transmitted wirelessly is not very sensitive to intrusion. The two pieces of data that will be sent through the transmitter are the current temperature of the room to be used to complete the feedback loop of the control algorithm and the user's preference.

Safety considerations

Safety is always a major concern for any product. Our system has safety concerns related to a number of components:

LCD: The LCD consists of a glass plate structure. It needs to be properly insulated from the casing of the user interface unit in order to assure minimal stress and decrease the possibility of shatter. On the two sides of the LCD, protective plastic coverings are placed in order to protect against damage during shipping. These must be removed prior to design integration in order to the chance of reduce circuit malfunction and overheating.

Mechanical implementation: Safety is a real concern for the mechanical implementation. Our clients are concern about fire hazards should there be any malfunctions with the motor. One method to reduce this safety concern is by use of the shaft encoder. Because the shaft encoder relays information about the movement of the motor, any unresponsive movement from the

motor after issuing a movement command would indicate failure in the motor. This will prevent burn out of the motor and in turn decrease the potential for fire hazards.

The other method of reducing potential fire hazards is to choose an accompanying motor driver circuit with over current protection or over temperature protection, preventing burn out of the motor. Our clients are actively concern about the safety of the product and are willing to spend additional funding to provide both solutions. We are required to have a shaft encoder to use with the DC motor and a motor driver circuit with over current protection and over temperature protection.

Driver Circuit: The driver circuit will be critical in providing the safety mechanism for the DC motor. The over temperature protection available in some driver circuits would be a plus in ensuring our DC motor avoids burn out. Over current protection will be in addition to the over temperature protection and provide further mitigation of burn out events. We will try to find a motor driver circuit that provides both features.

Temperature sensor: The Temperature sensor could malfunction due to incorrect reading or broken sensor. This can lead to huge room temperature fluctuations, which can cause extreme heating or cooling. Possible solutions to this issue are to set a low and high boundary for the motor control unit (15 degrees Celsius to 30 degrees Celsius). Another solution is to have a secondary temperature sensor that would become primary temperature sensor once an error is detected

Microcontroller: The only safety consideration is microprocessor overheating. If the microprocessor becomes too hot, it may burn individuals, cause damage to the unit, and even cause a fire. This can be avoided by smart design. Our group is experienced enough with hardware to design around these issues. We will pay great attention to detail and follow through with the manufacturer recommendations.

Recommendations Regarding Project Continuation or Modification

We recommend continuing our project as planned. Our current approach satisfies both the functional and nonfunctional requirements of the project. The estimated cost of the system has remained reasonable and the development costs have remained within budget. The team is on schedule and implementation efforts have begun. Further development efforts are planned for winter break in order to prepare the prototype for testing during the winter months.

Testing Results

Integration Testing Results

STK600

The STK600 was the platform for us to perform integration testing of the many components prior to our completion of the PCB design. We tested both the hardware and software by using the STK600 and that saved us time once we began transition our integration on the PCB boards.

Microcontroller

The microcontroller was the first component we had to get working on the STK600 development board. After some help from the Atmel support team, our advisors, and debugging time, we were able to unlock the microcontroller, program the microcontroller through the on board JTAG connection, initializing an internal time counter, and perform simple gpio operations.

Push Button

After the controller, we began testing the interrupt function necessary for many of the components. We chose the push button because the STK600 had on board push button available for us to test. This integration of the push button on the STK600 served the software basis on the final PCB design.

Motor System

The motor system was a larger system to integration onto the STK600 platform. There were three section of the motor system we had to integrate: the motor driver control, the gear motor encoder, and the motor driver current sense.

In getting the motor driver control circuitry to work, we were able to confirm PWM operation and determined the appropriate duty cycle for the speed of the gear motor. Our testing revealed that 100% duty cycle would be sufficient in rotating the steam valve.

After control of the motor driver was accomplished, we began integrating the encoder input signals back into the microcontroller. We achieved the integration of the encoder and were able to accurately position the gear motor to any desired position. Our work became the foundation for when we transition to the PCB design.

Lastly, the motor driver current sense allowed us to integrate the ADC function of the microcontroller with the rest of the components. We were able to confirm the operation of the microcontroller's ADC and accurately convert the analog reading to digital stored in our program.

Once the motor system was integrated onto the STK600 platform, we were able to test more advance position functions. These included `resetValvePosition()` and `calibrateMotor()` that allows the controller box system to autonomous determine the range characteristics of the steam valve.

Temperature Sensor

The temperature sensor integration was smooth given our previous experience with using the ADC functionality for the motor driver current sense.

LCD

The integration of the LCD was also smooth given that the pins used for the LCD were mostly GPIO and specified SPI interface pins. The LCD worked as expected with the STK600 along with the rest of the previously integrated components.

Wireless Transceiver

The wireless transceiver was much more difficult to set up, integrate, and test. We used an additional wireless transceiver on our work laptop to monitor communication to and from the wireless transceiver controlled by the STK600. After much work debugging the software code to keep the wireless transceiver working with the other components, we completed the basic functionality of sending and receiving data on the STK600.

Other

One of the most important tools in aiding us during the integration process was USART communication through the RS232 protocol. We were able to enable and set up communication between the STK600 and our work laptop to monitor internal data and processes inside the microcontroller.

Functional Testing Results

Functional Testing		
Basic Functionality		
Test	Success (P/F)	Criteria
temperature set pt. is higher than the current temp	Y	valve opens more
temperature set pt. is lower than the current temp	Y	valve closes more
set temperature preference and then check to see if set pt. is reached	F	valve is close (within 20% of turn range) to being closed
Set all preferences to 68°	Y	temperature is set to within 2° c after
Stand 3 ft. away from the LCD and read the temperature	Y	LCD is visible, can be read from 3 feet away
Attempt go below the lower temperature limit of 60°f	Y	UI should stay at 60°f
Attempt go above the upper temperature limit of 80°f	Y	UI should stay at 80°f
Valve Calibration		
Test	Success (P/F)	Criteria
Increment temperature preference by one °F starting from 60 to 80.	Y	valve should open slightly more each time the temperature preference goes up
Decrement temperature preference by one °F starting from 80 to 60.	Y	valve should close slightly more each time the temperature preference goes down
Range Tests		
Test	Success (P/F)	Criteria
Send one message to the transceiver at each range between 0 walls to 4 walls	Y	Range of up to 300 ft. with up to 4 walls between the UI and the control box
Send 100 messages to farthest range (300 ft., 4 walls)	Y	Have 95% of sent messages received successfully
Response Test		
Test	Success (P/F)	Criteria
User sets temperature to 78°F from 72°F	F	The room should reach approximately 78°F in two hours
User sets temperature to 72°F from 78°F	F	The room should reach approximately 72°F in two hours
Equalization and weighting Test		
Test	Success (P/F)	Criteria
Start from temperature set point of Room1: 75, Room2: 78, Room3: 83	F	The room should reach approximately 78°F depending on states
Start from temperature set point of Room1: 60, Room2: 75, Room3: 85	F	The room should reach approximately 75°F depending on states
Environmental and extreme Tests		
Test	Success (P/F)	Criteria
Put the steam valve control system in the oven at 100° F	F	Controller box functions after running for 10 hrs. in 100° F
Put the steam valve control system in the fridge at 100° F	F	Controller box functions after running for 10 hrs. in 10° F Temperature
Set all preferences to maximum	F	The temperature is set to within 2° F
Set all preferences to minimum	F	The temperature is set to within 2° F
Power and Charging		
Test	Success (P/F)	Criteria
Plug the controller into the USB when the battery level is at 50%	F	The unit indicates charging state
Drain the controller battery level to less than 10%	F	The buzzer sounds indicating low battery state
Plug the controller into the USB when the battery level is at 100%	F	The unit will indicate charging complete state
Monitored extended use testing		
Test	Success (P/F)	Criteria
Run system for 1 week, monitoring the temperature	F	The temperature is set within one time constant
Run system for 1 week, monitoring the LCD quality	F	The contrast of the LCD should remain the same, the LED backlighting should be at medium brightness
Press push buttons over 300 times	F	The incrementing and decrementing of the temperature should be functional

Figure 85: Functional Testing Results

The functional testing was run with the motor attached to the valve. A number of the tests were performed with the control panel inside a temperature chamber. The temperature chamber can be seen in the figure below.



Figure 86: Temperature chamber

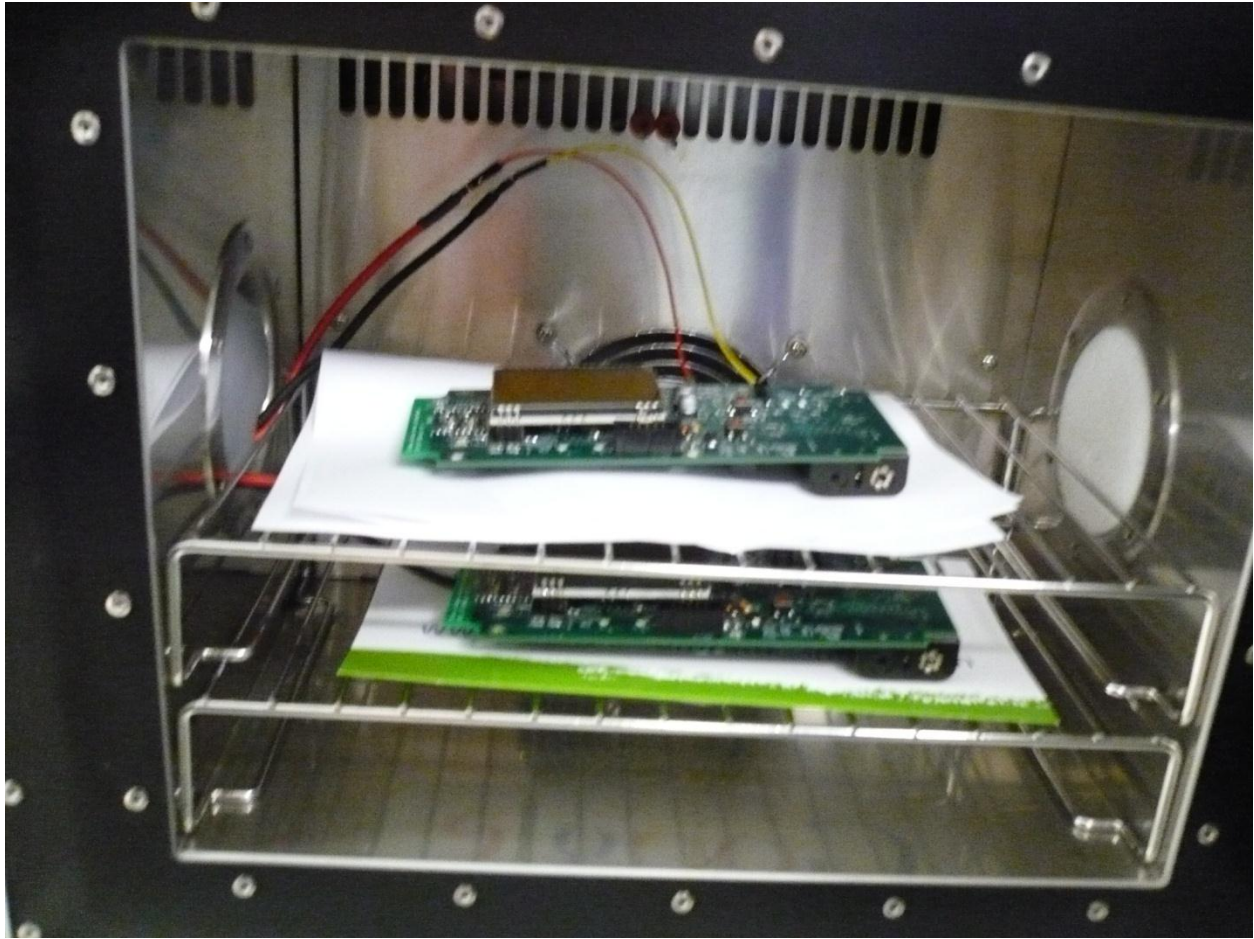


Figure 87: Temperature Chamber Test Setup

Hardware Testing Results

Testing Criteria	Satisfactory (Y/N)	Comments
Directional tracking	Y	
Speed control	Y	
Accuracy	Y	within 5 degrees
Torque requirement	Y	greater than 0.95 N m
Debugging interface	Y	report position, direction, current
Recalibration	N	identify fully closed position
Overcurrent detection	N	stop operation when current > 1.3 A
Stalling detection	N	stop operation when motor report stalling
Mechanical stability	N/A	operate continuously for 1 week
Electrical stability	N/A	operate continuously for 1 week
Operational stability	N/A	operate continuously for 1 week

Figure 88: Hardware Testing Results

Hardware Testing Results Continued

Section	Satisfactory Y/N	Notes	Comments
Power:			
Test voltage levels, voltage ripple, voltage stability, and voltage behavior due to transients of all power supplies over their full loads	Y	3.3 V booster circuit needs re-route in order to test Used 3.3 V	Max voltage ripple is 50 mV. All voltages within 5% of target value
Test the limits on inputs to all switchers and LDOs	Y	3.3 V booster circuit needs re-route in order to test Used 3.3 V	LDO limit: 10 V. 5 V Switcher Limit: 20 V
Test typ. current draw for all major power rails	Y		Controller Box 3.3 V: 90 mA max 5.0 V: 30 mA max 12.0 V: 2.64 A max Control Panel: 3.3 V: 200 mA max
Test typ. noise levels on all major IC power rails	Y		Typical voltage ripple on IC power pins is approximately 25 mV.
Test the limits of current limiter circuit	Y		Current Limit: 2.64 A
Test reverse protection circuitry	Y		Reverse protection tested up to 30 V.
Ensure proper power supply sequencing	N/A	3.3 V booster circuit needs re-route in order to test Used 3.3 V	Controller Box: 12 V sequenced prior to 5V sequenced prior to 3.3 V Control Panel: N/A
Microprocessor:			
Test high speed signal lines for signal integrity:			
SPI signal integrity CD lines	Y		Typical rise time < 30 ns
USART signal integrity on transceiver lines	Y		Typical rise time < 30 ns
Ethernet signal integrity on the transceiver lines	Y		N/A
PWM lines on motor driver lines	Y		Typical rise time < 30 ns
Test system reset functionality	N/A		Reset functionality only resets the mP clocks and not the system
Test microprocessor can be JTAGged and programmed as needed	Y		JTAG and programming fully functional
Test microprocessor can communicate to LCD, Xbee transceiver, Ethernet Transceiver, and motor driver circuit - proper handshake	Y		Proper handshakes between mP and peripherals. 100 % communication.
Test RJ-45 and USB port lines for proper level conversion	N/A	Need to add 12 MHz external	Using the current PCB, we could not test without an external crystal.
Test LED systems to determine if operating in safe current limits	Y		LED circuitry has appropriate current draw and operating limits.
Ensure proper clock start up sequence	Y		mP clock starts up accordingly. Only one clock used currently.
Ethernet Transceiver:			
Test high speed signal lines for signal integrity:	N/A	Need to add 12 MHz external crystal to mP	The design is a prototype. In order to obtain Ethernet functionality, need to add a 12 MHz mP.
Ethernet signal integrity on the microprocessor lines	N/A	Need to add 12 MHz external crystal to mP	The design is a prototype. In order to obtain Ethernet functionality, need to add a 12 MHz mP.
Ethernet signal integrity on the RJ45 lines	N/A	Need to add 12 MHz external crystal to mP	The design is a prototype. In order to obtain Ethernet functionality, need to add a 12 MHz mP.
Test system Reset functionality	N/A	Need to add 12 MHz external crystal to mP	The design is a prototype. In order to obtain Ethernet functionality, need to add a 12 MHz mP.
Test transceiver can communicate to Iowa State University network and vice versa	N/A	Need to add 12 MHz external crystal to mP	The design is a prototype. In order to obtain Ethernet functionality, need to add a 12 MHz mP.
Test Ethernet Transceiver communication to microprocessor	N/A	Need to add 12 MHz external crystal to mP	The design is a prototype. In order to obtain Ethernet functionality, need to add a 12 MHz mP.

Figure 89: Hardware Testing Results Continued

Software Testing Results

Software Testing

LCD, Temperature, and Microcontroller				
Test Case		Pass	Fail	Comments
1.	Display all characters supported by the LCD	Pass		
2.	Verify correct display results for all components	Pass		
	c. Correctly displays current temperature readings	Pass		
	d. Warning for low battery	N/A		Was not tested
3.	LCD display updates accordingly after user input	Pass		
4.	LCD displays desired output after system reboot	Pass		

Website Interface				
Test Case		Pass	Fail	Comments
1.	Password authentication for average user	Pass		
2.	Password authentication for admin	Pass		
3.	Administrator's ability to add, remove, or modify user	N/A		Was not tested
4.	Administrator's ability to overwrite temperature preferences	N/A		Was not tested
5.	Verify that JDBC correctly communicates with database	Pass		
6.	The status of the overall system is displayed correctly	Pass		
7.	The user feedback is archived displayed for the admin	Pass		
8.	The web interface correctly retrieves data after system failure	Pass		
9.	The web interface displays appropriate message after invalid user requests	Pass		

Figure 90: Software Testing Results

V. Team Information

Project Team Information

Should there be any questions or interested clients about our project, please use the following project team information to get in contact with us.

Client Information

Client's Name	Iowa State University Electrical and Computer Engineering (ISU ECPE)	
Contact Info	Primary: Lee Harker	Backup: Jason Boyd
Address	1341 Coover Ames, IA 50011-3060	
work phone #	515-294-3247	515-294-1256
Fax #	515-294-8432	
Email	leharker@iastate.edu	jaboyn@iastate.edu

Figure 91: Client Information

Student Team Information

Team Member's Name	Ben Jusufovic	Thinh Luong	Curtis Mayberry	Ben Cao
Major	Electrical Engineer	Electrical Engineer	Electrical Engineer	Computer Engineer
Address	4733 Toronto St. Unit 309 Ames, IA 50014	3522 Lincoln Way Unit 69 Ames, IA 50014	3108 Coover Ames, IA 50011	2213 Frederiksen Ct. Ames, IA 50010
work phone #	515-577-7515	515-724-8530	515-451-5625	402-707-9957
Email	bj1@iastate.edu	thinhvluong@yahoo.com	curtisma@iastate.edu	becao@iastate.edu

Team website: <http://seniord.ece.iastate.edu/may1119/index.htm>

Figure 92: Team Information

VI. Budget Information

Controller Box Budget

The table below contains the budge details for the Controller Box:

Controller Box					
MODULE	ITEM DESCRIPTION	VENDOR	COUNT	PRICE/UNIT	PRICE
DC motor	IG42 24v gear motor	MPJA	1	\$ 59.99	\$ 59.99
Motor driver circuit	L298N	Digikey	1	\$ 4.78	\$ 4.78
Metal platform	Extruded aluminum	80/20	1	\$ 15.00	\$ 15.00
Transceiver	Digi Xbee PRO XBee	Digi	1	\$ 30.00	\$ 30.00
AC to DC power supply	120 V 60 Hz to 12 V DC wall converter		1	\$ 19.99	\$ 19.99
Miscellaneous components	resistors, capacitors, inductors	ECPE	1	\$ 10.00	\$ 10.00
Microcontroller	Atmel AT32UC3A0128	Mouser	1	\$ 10.00	\$ 10.00
Malfunction alarm beeper			1	\$ 1.00	\$ 1.00
LCD and LCD backlight	DOGM128-6 Electronic Assembly	Mouser	1	\$ 22.00	\$ 22.00
Ethernet driver	886-LAN8700C-AEZG	Mouser	1	\$ 2.28	\$ 2.28
PCB			1	\$ 8.00	\$ 8.00
TOTAL					\$ 183.04

Figure 93: Controller Box Budget

Control Panel Budget

The table below contains the budge details for the control panel:

Control Panel					
MODULE	ITEM DESCRIPTION	VENDOR	COUNT	PRICE/UNIT	PRICE
Temperature sensor	IC TEMP-VOLT CONV PREC SOT23B	Digikey	1	\$ 0.70	\$ 0.70
LCD and LCD backlight	DOGM128-6 Electronic Assembly	Mouser	1	\$ 22.00	\$ 22.00
Transceiver	Digi Xbee PRO XBee	Digi	1	\$ 30.00	\$ 30.00
Miscellaneous components	resistors, capacitors, inductors	ECPE	1	\$ 10.00	\$ 10.00
Microcontroller	Atmel AT32UC3A0128	Mouser	1	\$ 10.00	\$ 10.00
Malfunction alarm beeper			1	\$ 1.00	\$ 1.00
Battery	2 rechargeable NiMH AA batteries		1	\$ 4.00	\$ 4.00
USB battery recharging circuit	Maxim DS2712E	Mouser	1	\$ 5.13	\$ 5.13
PCB			1	\$ 8.00	\$ 8.00
TOTAL					\$ 90.83

Figure 94: Control Panel Budget

Project Costs

The table below contains the budget for the overall project cost:

Project Costs					
MODULE	ITEM DESCRIPTION	VENDOR	COUNT	PRICE/UNIT	PRICE
Controller panel		various	3	\$ 90.83	\$ 272.49
Controller box		various	1	\$ 183.04	\$ 183.04
Microcontroller development kit		Atmel	1	\$ 265.00	\$ 265.00
Xbee development supplies		various	2	\$ 25.00	\$ 50.00
Subtotal					\$ 770.53
LABOR COST			HOURS	PAY/HOUR	PRICE
Ben Jusufovic	Based on \$60000 /year estimate from Career Services		250	\$ 28.84	\$ 7,210.00
Ben Cao			250	\$ 28.84	\$ 7,210.00
Curtis Mayberry			250	\$ 28.84	\$ 7,210.00
Thinh Luong			250	\$ 28.84	\$ 7,210.00
Subtotal					\$ 28,840.00

Figure 95: Project Costs

VII. Closing Summary

Lessons learned

If the team was given the project with the same requirements again, the team will make the following adjustment and modifications:

PCB Board

- Tie all enable pins to input voltage instead of microcontroller
- Cut enable pin traces/lines to microcontroller
- Rework motor driver 12 Volt plan and have more clearance between pins to prevent short
- Rework 3.3 volt booster routing
 - Have capacitors very close to the voltage input pin
 - Have bypass capacitors extremely close to the output voltage pin
 - Via voltage output line as close as possible to booster circuit
- Take more careful selection for inductor selection for low parasitic capacitance, and resistive characteristics.

Microcontroller

- Less pins and unnecessary functionalities
- Supports clock rate of 25 MHZ
- Contains internal IC for Ethernet communications
- Contains well documented example codes and drives

Motor

- Need low pass filter for the current sense readings
- Check for voltage offset created by the 12V line on the motor driver IC
- Choose gear motor with shaft encoder that has index channel for accuracy

Summary

Iowa State University buildings built prior to the late 1960's utilize steam valves and radiators to heat rooms. These heating systems link multiple rooms through steam pipes and consist of a single valve that controls the temperature heat output through radiators within each room. The user does not have knowledge of the magnitude of revolutions needed to accommodate a desired temperature within the room and causes over and under heating throughout the rooms. Since these heat systems consist of physical controls, Facilities Planning and Management does not have remote access to the system in order to regulate temperature in times of little to no occupancy. As a result, significant energy is lost during campus night hours and school breaks, consequently increasing energy bills.

In order to solve these issues, our group has proposed to perform a system level integration in which we have designed a steam valve controller unit and a user interface unit. The steam valve controller unit consists of a DC motor - used to rotate the steam valve, a microcontroller - used to run the control

algorithm, a wireless transceiver - used to communicate to the user interface, and Ethernet connectivity – used to connect to the Iowa State network. The user interface consists of a LCD display – human machine interface, push buttons – used to receive user input, a temperature sensor – used to record the room temperature, a wireless transceiver - used to communicate to the steam valve controller, and a microcontroller - used to control the user interface unit. Through design, we have integrated these components into an efficient temperature control system.

The end product consists of two units: the control panels (user interface) and the controller box (steam valve controller). The user interface is a wall mountable unit similar to a thermostat. It prompts for and accepts temperature values from users. The unit is used as a means of sensing and recording current room temperature. The controller unit resembles a box structure and is situated right next to the steam valve. It runs a weighted algorithm based on received temperature values and adjust the steam valve appropriately through the DC motor. This system allows user friendly temperature control and monitoring.

Though the team had missed the major window for a real systems testing, the team have conducted various integration testing, system testing, hardware testing, and software testing.

References

Boley, Brian L. "Overview Tutorial of Electric Motor Types." Odd Parts. Web. 23 Sept. 2010. <<http://www.oddparts.com/acsi/motortut.htm>>.

"ELECTRONIC ASSEMBLY : LCD DOG Series, Flexible, Flat and Colorful." *ELECTRONIC ASSEMBLY : making Things Easier, Serial Graphic Displays with Built-in Intelligence TFT Display, Touch panel LCD Module, Dotmatrix, Graphic Module*. Web. 1 Oct. 2010. <<http://www.lcd-module.com/products/dog.html>>.

"Logic Level." *Wikipedia, the Free Encyclopedia*. 21 June 2006. Web. 24 Sept. 2010. <http://en.wikipedia.org/wiki/Logic_level>.

"PID Controller." *Wikipedia, the Free Encyclopedia*. Web. 23 Sept. 2010. <http://en.wikipedia.org/wiki/PID_controller>.

"XBee® & XBee-PRO® 802.15.4 OEM RF Modules - Digi International." *Digi International - Making Wireless M2M Easy*. Web. 25 Sept. 2010. <<http://www.digi.com/products/wireless/point-multipoint/xbee-series1-module.jsp#overview>>.

Appendix A - Testing after final component selection

Motor driver test circuit

The following test circuit was recommended in the L298N datasheet for a bi-directional motor control circuit. We will construct the circuit and it will form the basis for our final circuit implementation.

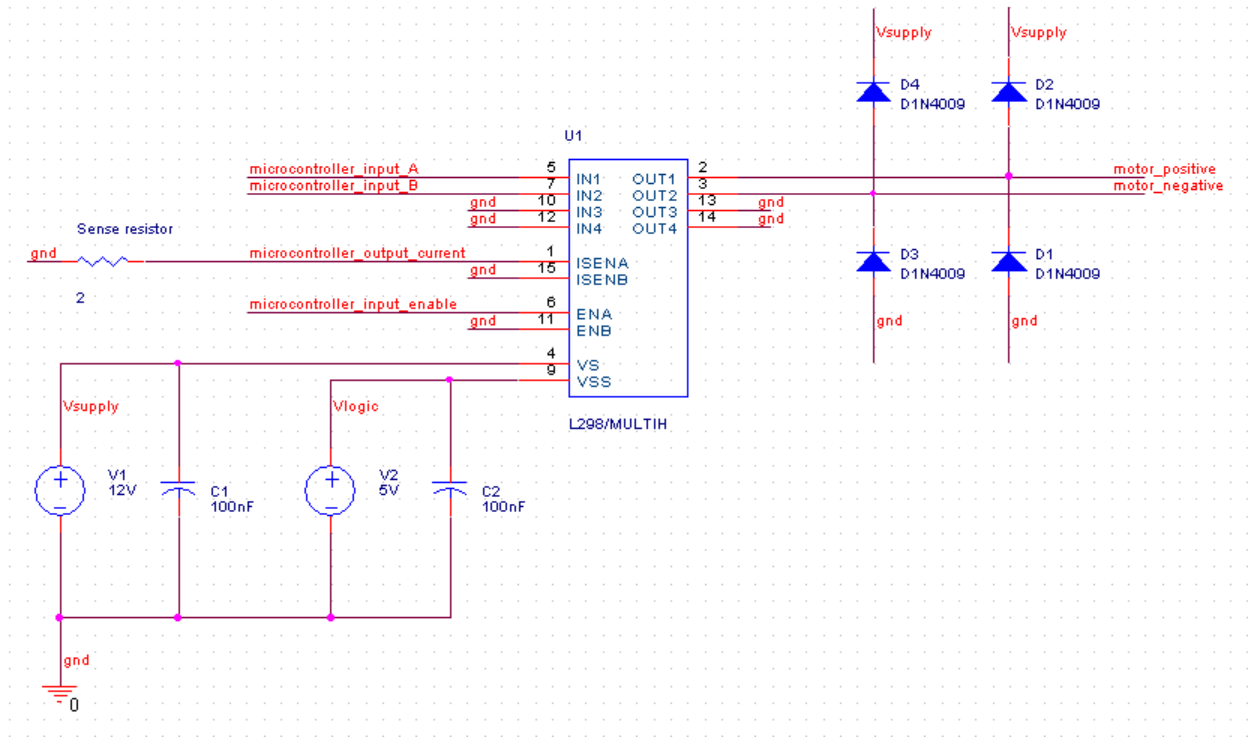


Figure 96: Motor driver test circuit

Note the fly back diodes configurations at the output. This is to prevent power generated by the rotating gear motor from going back into the L298N and damaging the device.

Magnetic shaft encoder circuit

The following test circuit was used to correctly receive the shaft encoder output.

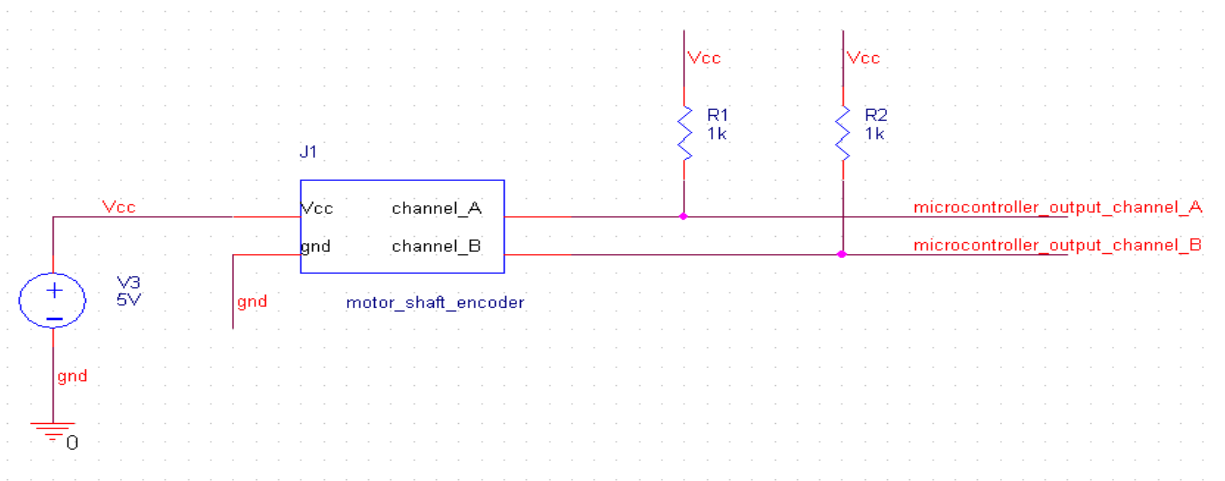


Figure 97: Shaft encoder test circuit schematic

Appendix B – PCB Schematic

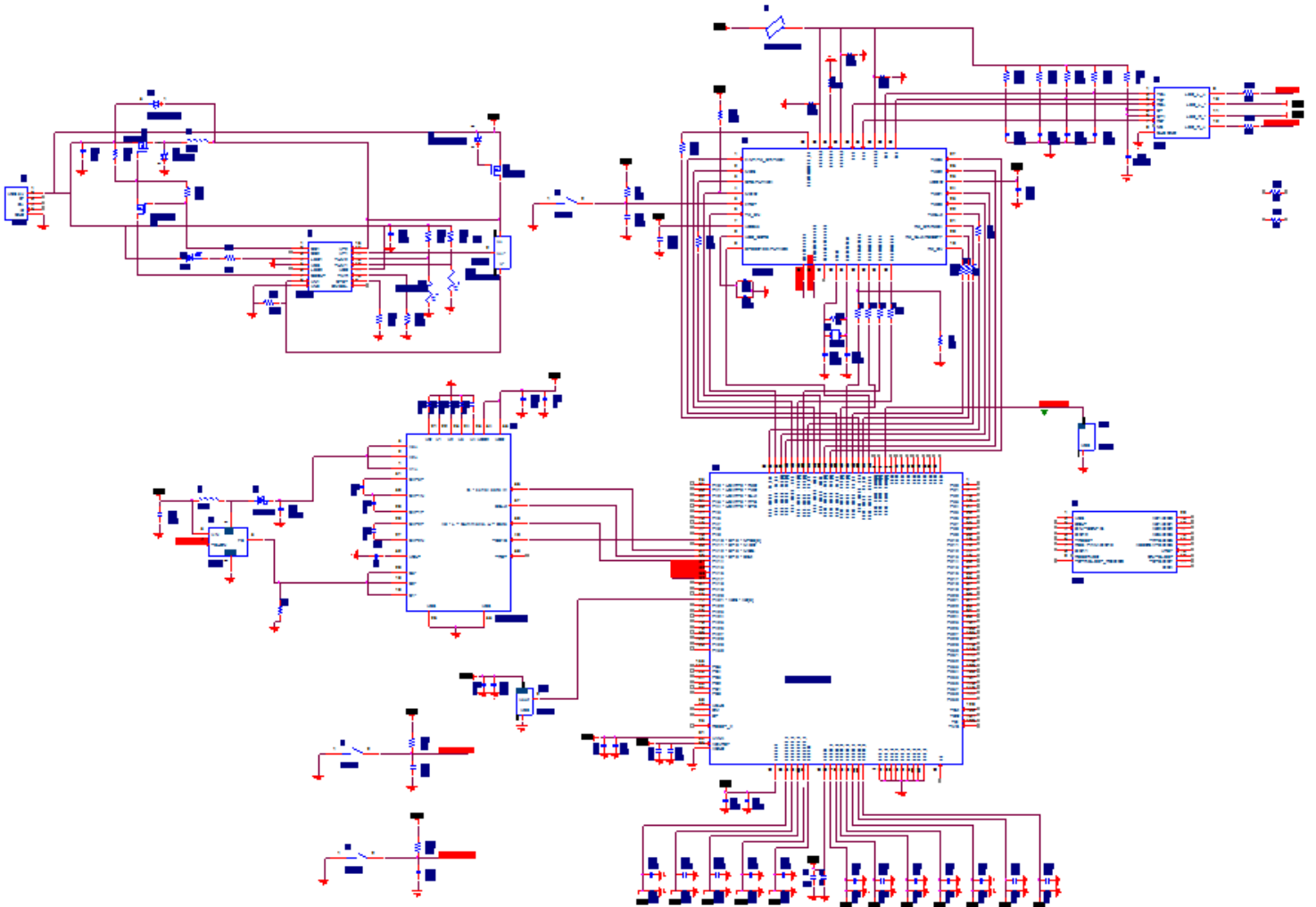
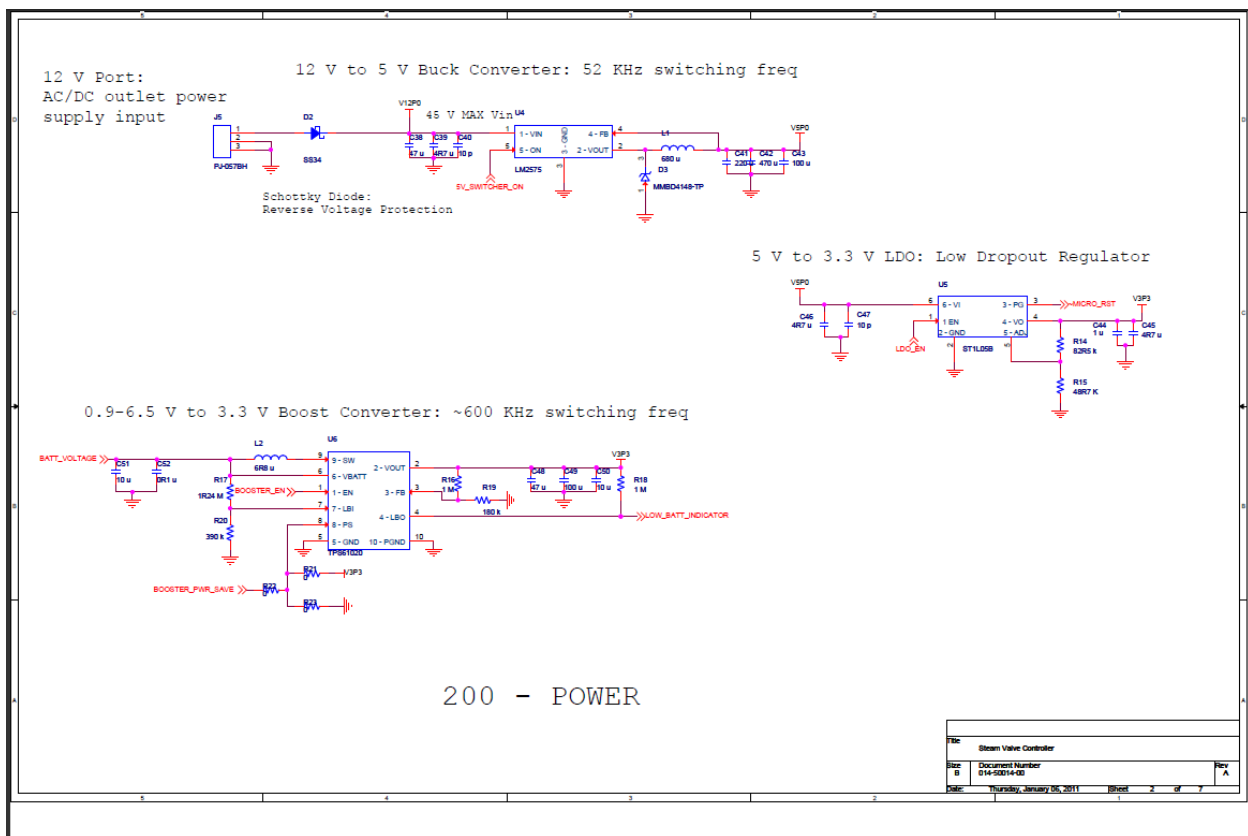
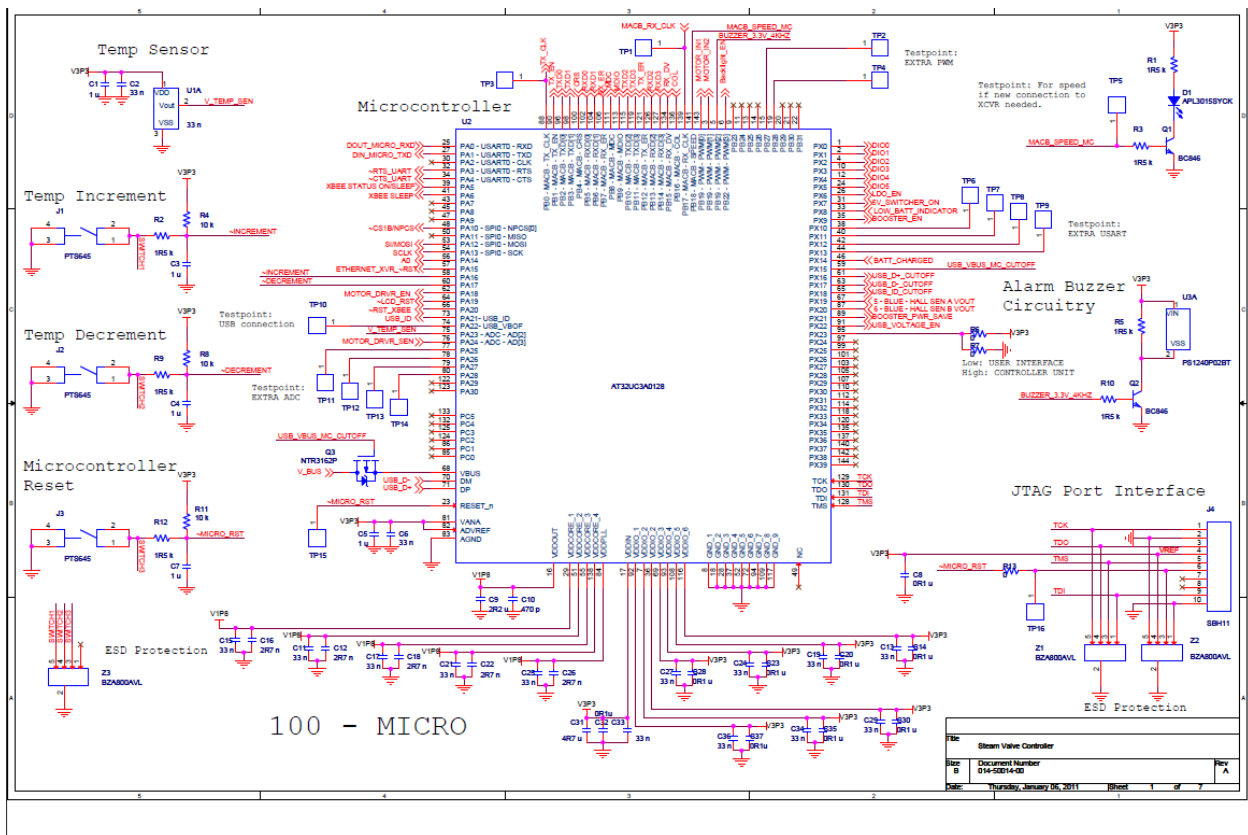
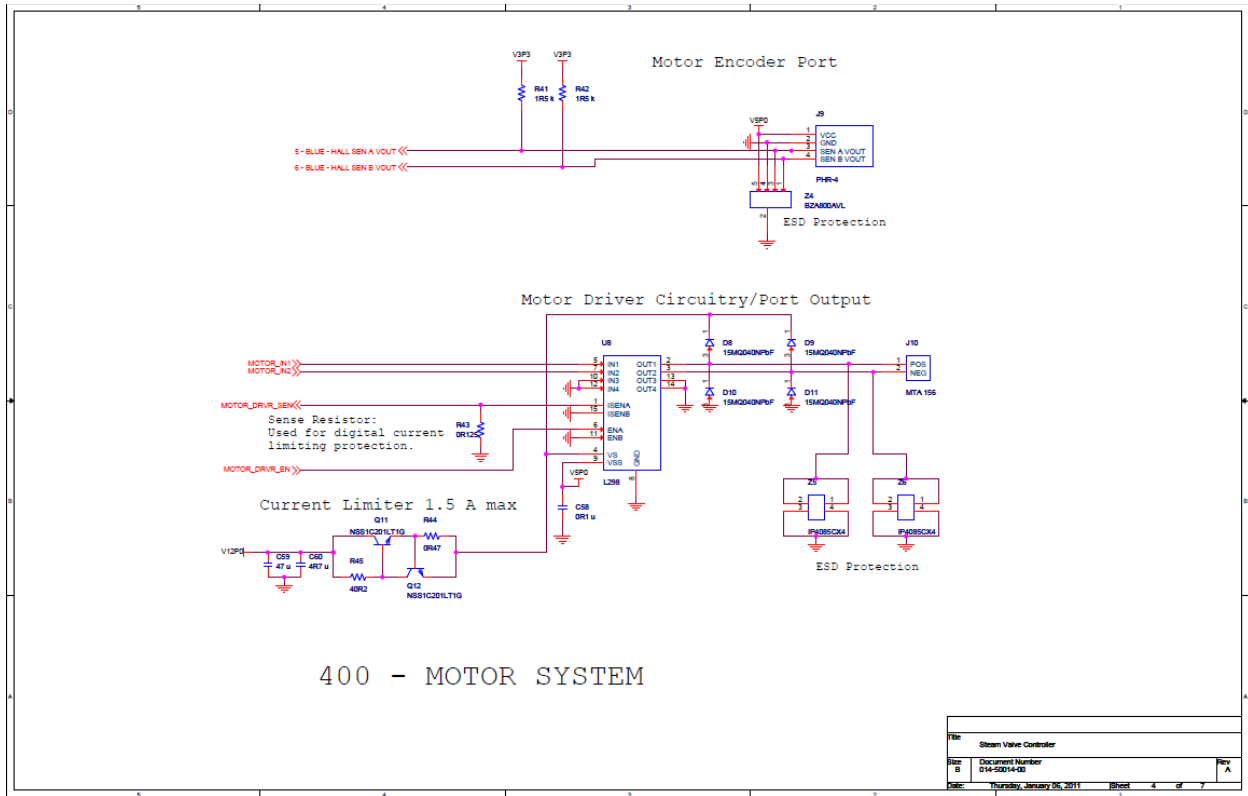
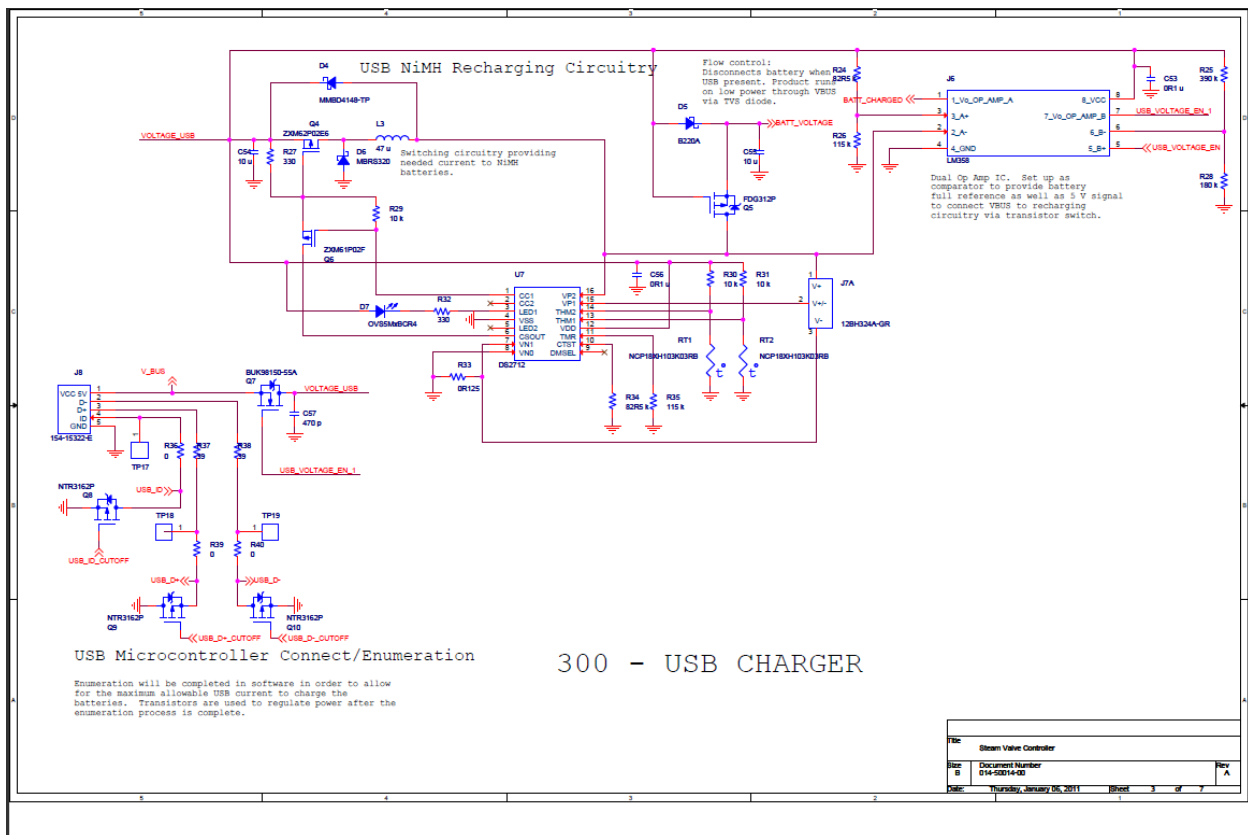


Figure 98: PCB schematic Overview





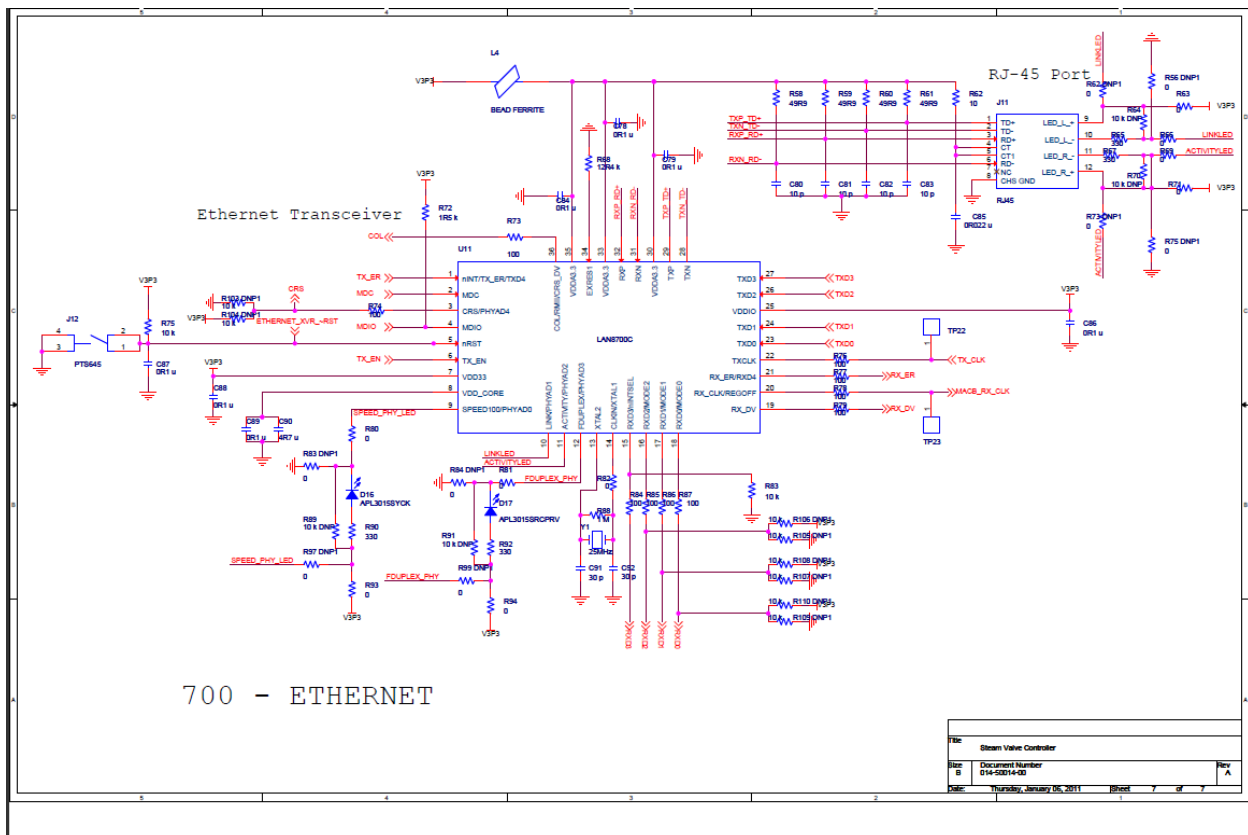


Figure 99: PCB schematic Details

Appendix C – Operational Manual

Basic Operation

When the system first starts up, the LCD will clear the pixelated screen. The initial values for the current temperature reading and set point will be 00. After a few seconds, the temperature reading will change to that of the room. The LCD backlight will be on and will remain on until the set point is changed from the 00 degree position. Afterward, followed by a pushbutton press, the LCD backlighting will turn off after every 3 seconds.

In order to set the current temperature, the user needs to use the increment and decrement buttons on the screen. To increase the temperature, use the upward pointing arrow. To decrease the temperature, use the downward pointing arrow. The set point on the LCD will change based on set point and the backlight will turn on, after three seconds the backlighting will turn off.

If the motor is stuck due to valve rust or malfunctions and spins to the end of valve rotation, the system will sound an alarm and will beep in order to let the user know to fix the valve. The buzzer will also sound whenever the motor is turning, this is normal.

Physical Setup

The setup for the current system is detailed below:

1. Setup mechanical interface
2. Setup the controller box
3. Setup the control panel

To setup the mechanical interface:

- First unscrew the screw holding the steam valve cap
- Remove the steam valve cap
- Fit the motor system platform over the steam valve knob
- Lay the shaft connect over the steam valve knob
- Tighten the two screws to secure the motor system platform to the base of the steam valve
- Attach the 2 line power male molex connector of the motor system to the 2 line power female molex connector on the controller box
- Attach the 4 line encoder male molex connector of the motor system to the 4 line encoder female molex connector on the controller box

To setup the controller box:

- Make sure that a constant power source of 5 volts is supplied
- Plug in the 12 volt power source
- Upon start up the valve will turn 360° clockwise and then 360° counter-clockwise
- The buzzer will sound whenever the motor is turning

To setup the control panel:

- Make sure that a constant power source of 3.3 volts is supplied
- Use the push buttons to adjust the desired temperatures

Firmware Setup

Prior to the installation of the system a few parameters in the firmware must be programmed in order to customize the system to the specific room configuration and the type of unit being programmed. The number of units in the system, the unit's ID number and the type of unit must be customized. The number of units is the number of control panels or rooms in the system. The unit ID is a number that is unique to each control panel connected to a valve. Finally the type of unit selects whether a control panel or control box is being programmed. All parameters are clearly marked as "Need to set" at the top of the main code and the top of the Xbee library source files.

Appendix D – PCB Bill of Materials

Bill Of Materials					
Distributor	Distributor PN	Distributor price	Part Reference	Tech	Value
Mouser	80-C0603C102J5G	0.05	C1	NPO	1n
Mouser	81-GRM219R61C475KE5D	0.05	C2	X7R	4.7u
Mouser	81-GCM1555C1H471JA6D	0.05	C3	NPO	470p
Mouser	81-GRM21R70J225KA01L	0.16	C4	X7R	2.2u
Mouser	81-GRM18R60J105KA01J	0.04	C5	X7R	1u
Mouser	81-GRM155R71H103KA8D	0.02	C6	X7R	10n
Mouser	81-GRM18R60J105KA01J	0.04	C7	X7R	1u
Mouser	81-GRM155R71H103KA8D	0.02	C8	X7R	10n
Mouser	81-GRM18R60J105KA01J	0.04	C9	X7R	1u
Mouser	81-GRM155R71H103KA8D	0.02	C10	X7R	10n
Mouser	81-GRM18R60J105KA01J	0.04	C11	X7R	1u
Mouser	81-GRM155R71H103KA8D	0.02	C12	X7R	10n
Mouser	81-GRM18R60J105KA01J	0.04	C13	X7R	1u
Mouser	81-GRM155R71H103KA8D	0.02	C14	X7R	10n
Mouser	81-GRM18R60J105KA01J	0.04	C15	X7R	1u
Mouser	81-GRM155R71H103KA8D	0.02	C16	X7R	10n
Mouser	81-GRM18R60J105KA01J	0.04	C17	X7R	1u
Mouser	81-GRM155R71H103KA8D	0.02	C18	X7R	10n
Mouser	81-GRM18R60J105KA01J	0.04	C19	X7R	1u
Mouser	81-GRM155R71H103KA8D	0.02	C20	X7R	10n
Mouser	81-GRM18R60J105KA01J	0.04	C21	X7R	1u
Mouser	81-GRM155R71H103KA8D	0.02	C22	X7R	10n
Mouser	81-GCM1555C1H471JA6D	0.05	C23	NPO	470p
Mouser	81-GRM21R70J225KA01L	0.16	C24	X7R	2.2u
Mouser	81-GCM1555C1H471JA6D	0.05	C25	NPO	470p
Mouser	81-GRM21R70J225KA01L	0.16	C26	X7R	2.2u
Mouser	81-GCM1555C1H471JA6D	0.05	C27	NPO	470p
Mouser	81-GRM21R70J225KA01L	0.16	C28	X7R	2.2u
Mouser	81-GCM1555C1H471JA6D	0.05	C29	NPO	470p

Mouser	81-GRM21R70J225KA01L	0.16	C30	X7R	2.2u
Mouser	81-GCM1555C1H471JA6D	0.05	C31	NPO	470p
Mouser	81-GRM21R70J225KA01L	0.16	C32	X7R	2.2u
Mouser	81-GRM18R60J105KA01J	0.04	C34	X7R	1u
Mouser	81-GRM155R71H103KA8D	0.02	C35	X7R	10n
Mouser	80-C0805C104K3R7210	0.02	C36	X7R	0.1u
Mouser	81-GRM219R61C475KE5D	0.05	C37	X7R	4.7 u
Mouser	81-GRM185C2A100JA01D	0.04	C38	X7R	10 pf
Mouser	81-GRM18B11H223KA01D	0.04	C39	X7R	0.022 uf
Mouser	81-GRM18R60J105KA01J	0.04	C40	X7R	1u
Mouser	81-GRM39C300J50	0.04	C41	X7R	30 pf
Mouser	80-C0805C104K3R7210	0.02	C42	X7R	0.1u
Mouser	80-C0805C104K3R7210	0.02	C43	X7R	0.1u
Mouser	80-C0805C104K3R7210	0.02	C44	X7R	0.1u
Mouser	80-C0805C104K3R7210	0.02	C45	X7R	0.1u
Mouser	80-C0805C104K3R7210	0.02	C46	X7R	0.1u
Mouser	81-GRM39C300J50	0.04	C47	X7R	30 pf
Mouser	81-GRM185C2A100JA01D	0.04	C48	X7R	10 pf
Mouser	81-GRM185C2A100JA01D	0.04	C49	X7R	10 pf
Mouser	81-GRM185C2A100JA01D	0.04	C50	X7R	10 pf
Mouser	80-C0805C104K3R7210	0.02	C51	X7R	0.1u
Mouser	81-GRM18R60J105KA01J	0.04	C52	X7R	1u
Mouser	81-GRM18R60J105KA01J	0.04	C53	X7R	1u
Mouser	81-GRM18R60J105KA01J	0.04	C54	X7R	1u
Mouser	81-GRM18R60J105KA01J	0.04	C55	X7R	1u
Mouser	81-GRM18R60J105KA01J	0.04	C56	X7R	1u
Mouser	81-GRM18R60J105KA01J	0.04	C57	X7R	1u
Mouser	81-GRM18R60J105KA01J	0.04	C58	X7R	1u
Mouser	81-GRM18R60J105KA01J	0.04	C59	X7R	1u
Mouser	81-GRM18R60J105KA01J	0.04	C60	X7R	1u
Mouser	81-GRM155R71H103KA8D	0.02	C61	X7R	10n
Mouser	81-GRM219R61C475KE5D	0.05	C62	X7R	4.7 u
Mouser	80-C0805C224K3R	0.11	C63	X7R	0.22 u
Mouser	80-C0805C104K3R7210	0.02	C64	X7R	0.1u
Mouser	81-GRM21BR61E106KA3L	0.15	C65	X7R	10u
Mouser	80-C0805C104K3R7210	0.02	C66	X7R	0.1u
Mouser	80-C0805C104K3R7210	0.02	C67	X7R	0.1u
Mouser	833-DL4151-TP	0.05	D1		MA739/TO
Mouser	828-OVS5MYBCR4	0.35	D2		LED

Mouser	512-MBRS320	0.44	D3		MBRS320
Mouser	833-MMBD4148-TP	0.08	D4		MMBD4148-TP
Mouser	625-P6SMB220A-E3	0.37	D5		P6SMB220A TVS
Mouser	673-J00-0061NL	3.9	J1		RJ45
Mouser	154-15322-E	0.51	J2		CON5
Mouser	506-FSM2JSMATR	0.43	J5		CON2A
Mouser	698-CAT4238TDG	1.6	J6		CON5
Mouser	700-DS2712E	5.13	J7		CON16
Mouser	506-FSM2JSMATR	0.43	J8		CON2A
Mouser	506-FSM2JSMATR	0.43	J9		CON2A
Mouser	815-ACML-0402H-750-T	0.04	L1		BEAD FERRITE
Mouser	652-SDR0703-470KL	0.53	L2		47uH
Mouser	652-SDR0703-470KL	0.53	L3		47uH
Mouser	522-ZXM61P02FTA	0.45	Q4		ZXM61P02
Mouser	522-ZXM62P02E6TA	1.14	Q5		ZXM62P02
Mouser	512-FDG312P	0.43	Q6		FDG312P
Mouser	301-100-RC	0.03	R1		100
Mouser	292-12.4K-RC	0.04	R2		12.4k
Mouser	304-10K-RC	0.04	R3		10k
Mouser	304-10K-RC	0.04	R4		10k
Mouser	301-100-RC	0.03	R5		100
Mouser	301-100-RC	0.03	R6		100
Mouser	301-100-RC	0.03	R7		100
Mouser	301-100-RC	0.03	R8		100
Mouser	301-100-RC	0.03	R9		100
Mouser	301-100-RC	0.03	R10		100
Mouser	301-100-RC	0.03	R11		100
Mouser	301-100-RC	0.03	R12		100
Mouser	301-100-RC	0.03	R13		100
Mouser	301-330-RC	0.03	R14		330
Mouser	301-330-RC	0.03	R15		330
Mouser	301-330-RC	0.03	R16		330
Mouser	301-330-RC	0.03	R17		330
Mouser	304-49.9-RC	0.04	R18		49.9
Mouser	304-49.9-RC	0.04	R19		49.9
Mouser	304-49.9-RC	0.04	R20		49.9
Mouser	304-49.9-RC	0.04	R21		49.9
Mouser	302-10-RC	0.04	R22		10
Mouser	304-10K-RC	0.04	R23		1.5k
Mouser	292-1.0M-RC	0.04	R24		1M
Mouser	71-CPF15R0000FHE14	0.37	R25		5

Mouser	71-WSL2010R1250FEB	0.81	R26		0.125
Mouser	304-10K-RC	0.04	R27		10k
Mouser	304-10K-RC	0.04	R28		10k
Mouser	304-10K-RC	0.04	R29		10k
Mouser	304-10K-RC	0.04	R30		10k
Mouser	292-75K-RC	0.04	R31		75k
Mouser	652-CR0603-JW-104GLF	0.02	R32		100k
Mouser	652-CR0603-JW-271GLF	0.02	R33		270
Mouser	304-10K-RC	0.04	R34		10k
Mouser	301-330-RC	0.03	R35		330
Mouser	81-NCP18XH103K03RB	0.14	RT1		THERMISTOR
Mouser	81-NCP18XH103K03RB	0.14	RT2		THERMISTOR
Mouser	556-AT32UC3A0128ALUT	10.18	U1		AT32UC3A-QFP144
Digi Key	TC1046VNBTRCT-ND	0.7	U2A		TC1046
Mouser	790-EADOGM128E6	18	U3		EA-DOGM128
Digi Key	XBP24-ACI-001	32	U4		XBEE
Mouser	12BH324A-GR	0.57	U6A		TC1046
Mouser	810-PS1240P02BT	0.39	U7A		Buzzer
Mouser	886-LAN8700C-AEZG	2.28	U8		LAN8700
Mouser	815-ABLS-25.0M-F-T	0.43	Y1		25MHz

Figure 100: PCB Bill of Materials

Appendix E: Poster

