March 29, 2017

Andrew H. Rawicz
School of Engineering Science
Simon Fraser University
V5A 1S6

Re: ENSC405W Design Specification – VentNet Home Heating Control System

Dear Dr. Rawicz,

Please find attached Aeolus System's design specification document for the VentNet Home Heating Control System. We at Aeolus Systems aim to create the VentNet system, a cost-effective and easy to install home heating control system that can accurately control the temperature in each room for a furnace heated home. The VentNet system will control the temperature in rooms by regulating the airflow from a home's heating ducts with motorized vents controlled by a smart thermostat and web interface.

In the design specification document, we provide details regarding the design decisions and technical solutions that will go into the VentNet system as well as an overview of the system and its modules. The appendices of the document will also explore user interface design and the test plan for the VentNet system. We believe that by thoroughly detailing the design decisions guiding the technical implementation we will have a reference for future developments when more technical decisions need to be made.

Aeolus Systems consists of 4 experienced senior computer engineering students each with industry experience and a hobby in embedded systems development. The founding members of Aeolus Systems are Paul Khuu, Jeremy Leung, James Voong, and Steven Zhou.

Thank you for reviewing our design specifications for the VentNet Home Heating Control System. If you have any further questions, please feel free to contact me by phone (778-708-8679) or email (rza32@sfu.ca).

Sincerely,

Steven Zhou

Aeolus Systems

Design Specification

VentNet

Home Heating Control System

by

James Voong

Jeremy Leung

Paul Khuu

Steven Zhou

MARCH 29, 2017

Version 1.73F

Abstract

VentNet is a combination of multiple individual smart modules formed into a cohesive system with four main features consisting of: individual zone temperature control, web application interface, the master thermostat module, and the low-power wireless communications system. To complement these features, four major components will be developed: the smart thermostat, the router module, the room sensors, and the motorized vent covers. The room sensors are integral because it provides important temperature information to the rest of the system. The smart thermostat replaces the old thermostat and allows users to set the temperature in each control zone. The information from all modules is collected by the router module which hosts a web application allowing users access to home heating data and settings via a web interface. Additionally, the application will provide users the ability to monitor their heat usage and schedule customized heating. By providing these functionalities, VentNet promotes comfort through optimal heating by minimizing the previous carelessly wasted heat.

The proof-of-concept model will consist of the following features:

- Vent Cover
- Temperature Sensor
- Master Thermostat
- Wireless Protocol and Radio

At the time of writing for this document, development of the router module is behind schedule and has been removed from the proof-of-concept stage. It has been re-targeted for the next phase.

This document describes the design specification of VentNet's proof-of-concept model, providing low-level details on the design and technical implementation for the project. Justifications for the design choices made are also included in this document. In addition, the document appendices list the test plans to be partaken by Aeolus Systems and design consideration for the UI aspects of the VentNet.

Table of Contents

List of Figures

List of Tables

Glossary

ACK	Short for acknowledge, it is a special message type use to notify a device its message has been received.
Air Vents	An opening in the home's duct network from where hot air is released into the house.
Duct	A channel allowing for the passage of air. In the context of this document, used to refer to home heating ducts that transport hot air from the furnace.
Data Collisions	The result of data being sent at the same time from different parts of the network, causing unexpected results such as data loss.
Flask	A Python web framework aimed towards keeping functionalities simple.
Furnace	A common home appliance for heating the house. It is connected to a blower and duct network to distribute the hot air.
H Bridge	An electronic circuit that enables a voltage to be applied across a load in either direction [1]
Heating Zone	Defines a specific space (set of rooms, etc.) capable of independent control over its temperature. Furnace homes often have one heating zone whilst electric and boiler homes have multiple.
Home Automation	The use of computers to control basic home functions such as heating, lighting, security, etc.
IoT	Internet of Things, the internetworking of physical devices to allow for the exchange of data.
Jinja	A Python template engine for building web pages.
JST Connector	An electrical connector typically used for battery packs.
Master device	The device in control of communications to slaves
Microcontroller	An integrated circuit dedicated to performing one specific task.
Paired Devices	Two devices that are wirelessly connected to one another to allow focussed communication.
PCB	Printed Circuit Board.

PWM	Pulse width modulation is a technique for simulating an analog result with digital means [2]
Slave device	Communicates with other elements in a network by going through a master device.
Sleep	Turning off certain functionalities on the device for a set period of time to conserve power.
Smart Devices	An electronic device capable of communicating with other devices that is able to operate interactively and autonomously to some degree.
SoC	System on Chip is an integrated circuit with many components of a standard computer (CPU, memory, etc)
SPI Bus	A serial peripheral interface bus that is typically used for communications between a microcontroller and small peripherals. [3]
Thermostat	A device capable of reading ambient air temperatures and connected to the furnace to control furnace output based on temperature setting.
WebApp	A typical client-server software application where the client interface runs in a web browser.

1 Introduction

VentNet is a robust solution for solving heating inconsistencies using a combination of sensors and smart vents. The VentNet system offers wireless vents, to induce variable heating, and smart thermostats seamlessly controlled through the VentNet web application. Our smart vents open and close according to the accurate information provided by the system's sensors and thermostats, indicating precise controls for flexibility and comfort. Through the web application, users shall have the convenience of scheduling heating habits while also being able to monitor the temperature and their usage such that their home heating experience can be further optimized. With optimized coverage of the VentNet system throughout the house, VentNet is meant to give users the freedom of not having to worry about heating inconsistencies and the ability of customizing their own heating preferences to elevate both comfort and savings.

This design specification document will outline the following:

- The hardware choices of the VentNet modules and their implications
- The software design details, resources used and the current range of functionality
- Current progress relative to requirements outlined in the requirements document and intentions for the production phase of the VentNet
- Current test plans including procedures and expected results
- UI descriptions and layouts for the production phase as well as their justifications

1.1 Scope

This document describes the details going into the VentNet system and explains how they meet the functional requirements as outlined in the Requirements Specification document. This Design Specification document will elaborate on the design considerations from implementing the requirements for a proof of concept model up to the stretch goal requirements outlined in the Requirements Specification document. However, as the primary focus of this development cycle will be the proof of concept model, only the requirements pertaining to P1 and P2 will be covered in its entirety whilst design details for lower priority requirements will be included if design planning for it has been complete. Moving forward, the design details in this document will serve as a reference for technical implementation conventions when developing for the VentNet system.

1.2 Intended Audience

This design specification document is intended for use by all members of Aeolus Systems. During the development phase, Aeolus Systems engineers shall refer to design specifications outlined here to ensure all functional requirements are met. And, during the testing phase, Aeolus Systems engineers shall refer to the test plan appendix to ensure proper functionality of the VentNet system.

2 System Overview

The VentNet system consists of four major components: the thermostat, router module, room sensors and motorized vent covers.

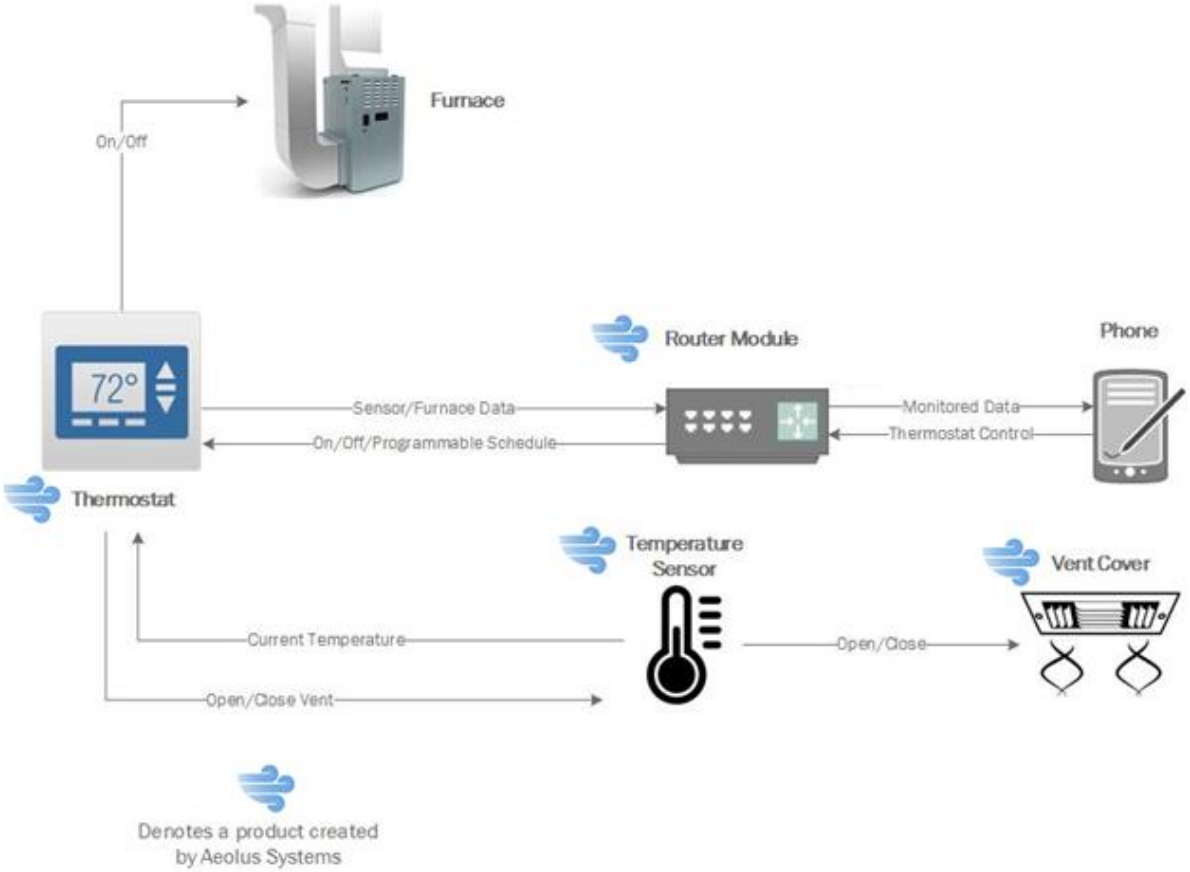

FIGURE 1: SYSTEM DIAGRAM OF THE VENTNET SYSTEM

In the VentNet system, our four modules communicate through our custom radio protocol to send data and commands to one another.

To begin, our thermostat is the heart of our system, sending on and off signals to the furnace based on the various room temperatures. It will also send sensor and furnace data to our router module for processing and notifies the temperature sensor in rooms of changes to the user's target temperature. Our thermostat currently makes use of a Raspberry Pi development board with a radio chip connected through SPI to communicate through the system. The thermostat also makes use of a LCD touchscreen for user interaction and display purposes and is currently powered via a wall adapter.

The temperature sensor measures ambient temperature in a room and will signal the paired vent cover to open or close when the room temperature moves away from the set temperature. If the room temperature dictates that the vents are to be opened for heating, then the temperature sensor will also signal the master thermostat to turn on the furnace. Our temperature sensor is a simple box containing a microcontroller, radio, and a LED/button combo for pairing purposes.

The vent covers are each paired with a temperature sensor and open and close through signals sent by its paired temperature sensor. The vent covers mechanisms are currently pre-made by Vent-Miser and we have inserted our microcontrollers and radio to handle requests to open and close from the temperature sensor. To pair, we have also included a LED and button for the vent module.

On the advanced settings side of the system, the Router Module, connects to the local home network to host a webapp for adjusting the temperature setting in each room in addition to offering other advanced settings and monitoring features. It receives data for display from the thermostat and sends out programmed schedules to the thermostat through our radio protocol. The router module also makes use of a raspberry pi development board to host the web server with a radio chip for communicating with the system. The module will be powered through a wall outlet and connected via Ethernet to the router of the home. As of the prototype stage, the router module is not currently ready for demoing.

3 System Design

3.1 Vent Cover Design

To create our vent module, we aimed to create a vent that would be able to open and close through commands wirelessly from another module (temperature sensors) on the network. To achieve this, we required a radio to receive these requests, a microcontroller to process the request, a mechanism to open and close the vents and a button and LED for pairing purposes with the commanding module. The device would also need to be powered portably and preferably without the user having to charge the module.

Thus, we picked the following components to create our vent module:

- Vent Module with motorized opening and closing mechanism
- Feather 32u4 with 915MHz radio
- RGB LED
- Button
- 3 x AAA battery pack

The chosen development board feather 32u4 (seen in figure 2) gave us a great starting ground for the prototype phase as it is quite compact and already provides integration with the RFM69 radio module. Furthermore, the existing power solutions for the board met our criteria well, allowing us to power our module through a JST plug with 3 x AAA batteries. The remaining properties of the board can be seen below in table 1:

FIGURE 2: ADAFRUIT FEATHER [4]

TABLE 1: ADAFRUIT FEATHER 32U4 SPECIFICATIONS

Device	Adafruit Feather 32u4
Size	51mm x 23mm x 8mm
Weight	5.5 grams
Memory	32k flash, 2k RAM
SoC	ATmega32u4 @ 8MHz with 3.3V logic/power
Power Supply	Micro USB or JST
Radio Module	RFM69HCW 868/915 MHz module

For the prototype phase, we purchased a pre-existing vent module created by Vent-Miser that included a motor to control an underlying plastic that could be rotated to allow or disallow air through the vent. The motor required approximately half a second to close or open the vent and drew 400mA of current while requiring a 3V or higher voltage drop.

To open the vent cover, we would input a positive voltage across the motor but realized quickly that the development board's pulse modulated width would not be able to generate a negative voltage to close the vent cover and turn the motor in the other direction. After considering possible circuit solutions, we ultimately decided to choose a pre-existing chip to solve this problem for us as the creation of a circuit may have taken too much room inside the vent-miser and the costs were similar.

The limitations of the feather 32u4 development board are that it can provide 3.3V and a max of 500mA for a short duration. Knowing this, we selected the DRV8838 h-bridge chip (shown in figures 3 and 4) as it could function within voltages from 1.8V to 7V and provided safety measures such as protection against over-current, over-temperature and reverse voltage. The DRV8838 handles forward and reverse motion for our motor by allowing the voltage to be applied in either direction. The chip could also be easily programmed with the feather 32u4 and provided sleep functionality that could help us save power as the motor was only required in short bursts.

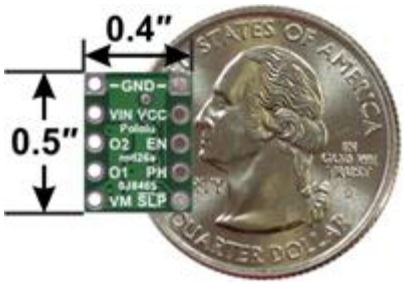

FIGURE 3: SIZE COMPARISON OF THE DRV8838 AND AN AMERICAN QUARTER[5]

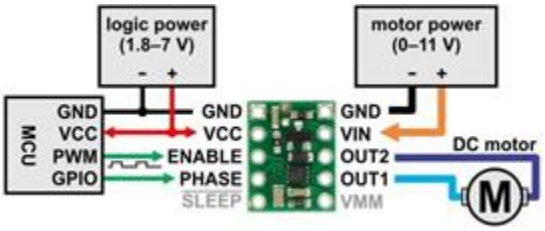

FIGURE 4: PIN LAYOUT OF THE DRV8838 [6]

With our various components, we created the circuit shown below in figure 5. A resistor was added between the outputs to reduce the 400mA pull of the motor as our microcontroller is only able to supply a current of 500mA.

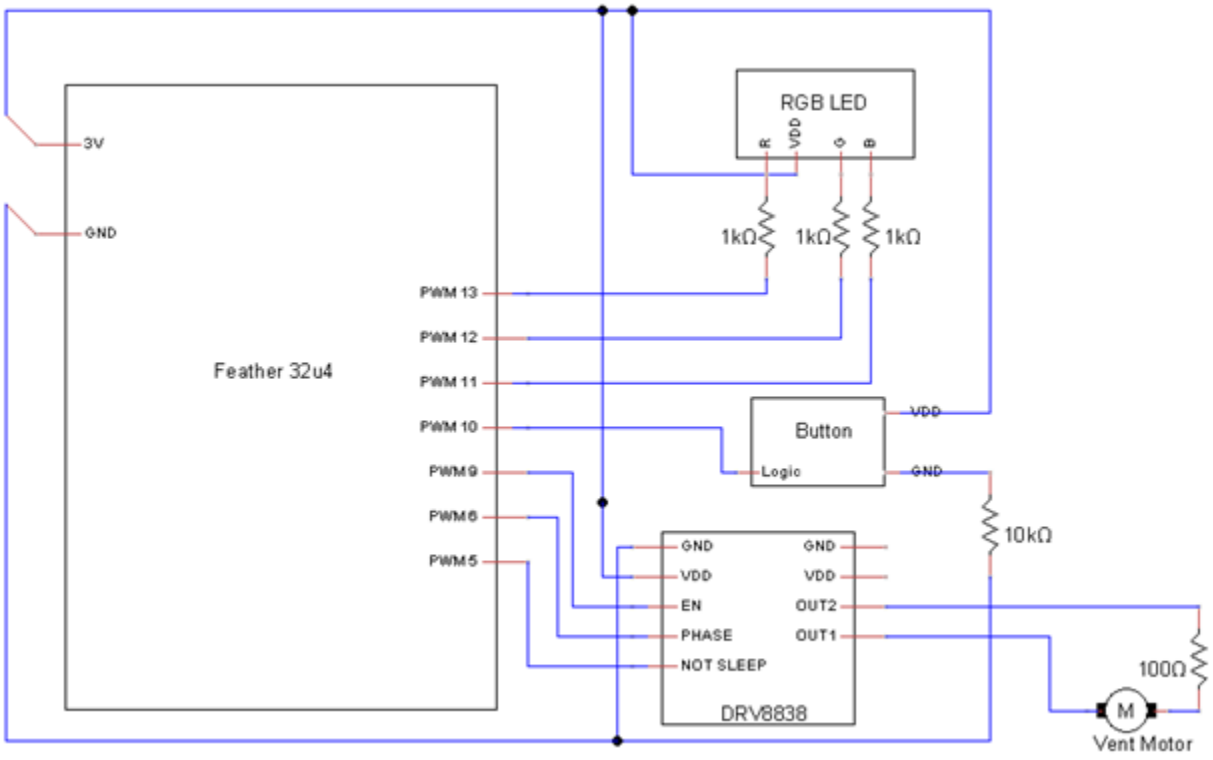

FIGURE 5: CIRCUIT DIAGRAM OF THE VENT MODULE

With the creation of the circuit and components we will require for the vent module, we will be able to create an even more compact module with a personalized controller board for the following phase knowing the exact requirements of our module. Looking at the circuit, we will require a power pin, a ground pin and 7 pins for PWM, a drastic step down from the available (and unused) pins the Feather 32u4 provides.

Regarding the current progress of the vent module, we have satisfied the various requirements set out for the prototype phase. At this time, however, the circuits are placed inside the vent covers using breadboards and are not enclosed. While not a requirement we set for the prototype, we would have liked to have completed an enclosure for aesthetic appeal and protection but instead we will prioritize this for the next phase.

For the next phase, apart from enclosures, Aeolus will investigate and create our own vent cover mechanism to allow partial cut-off of airflow. In addition, Aeolus will begin investigating and creating a personalized PCB for the vent to replace the feather 32u4 development board. With the new PCB implementation, we will also be re-evaluating the requirement we initially sought of a 2-year battery life for the vent modules and making optimizations.

3.2 Temperature Sensor Design

As with the vent module, the temperature sensor was designed simply and with the goals of being able to measure temperature, wirelessly send open and close commands to its paired vent module/s and occasionally report temperatures wirelessly to the master thermostat.

To achieve this, we selected the following components to create our temperature sensors:

- The Feather 32u4 development board with a 915 MHz radio
- RGB LED
- Button
- Temperature Sensor
- Wall adapter

For similar reasons as the vent, we used the Feather 32u4 for its compact size and dynamic array of functions for the initial prototype phase. The LED and buttons are used for pairing purposes and were chosen to be the same as the vents to provide consistency.

To measure the temperature, the Feather 32u4's radio chip has temperature measurement functionality but at an accuracy of +/- 10 C, it was simply too much of a range to consider. Looking at various models, we ultimately decided on the DS18B20 chip (shown below in figures 6 and 7) with an accuracy of +/- 0.5 C. The model takes in a voltage of 3-5V, perfect for our Feather 32u4 development board.

FIGURE 6: SIZE COMPARISON OF THE DS18B20 AND A COIN [7]

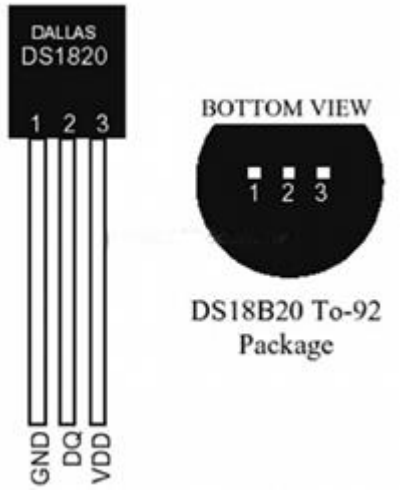

FIGURE 7: PIN LAYOUT OF THE DS18B20 [8]

The DS18B20 comes from a popular family of temperature sensors and has a well-developed library readily available making development much smooth. The chip comes with parasitic power abilities, allowing it to function with only the PWM input. Upon testing this however, we found the calibration period on first starting up was too slow relative to having the chip be fully powered using VDD.

With these considerations in mind, we constructed a similar circuit to the vent modules as shown in figure 8.

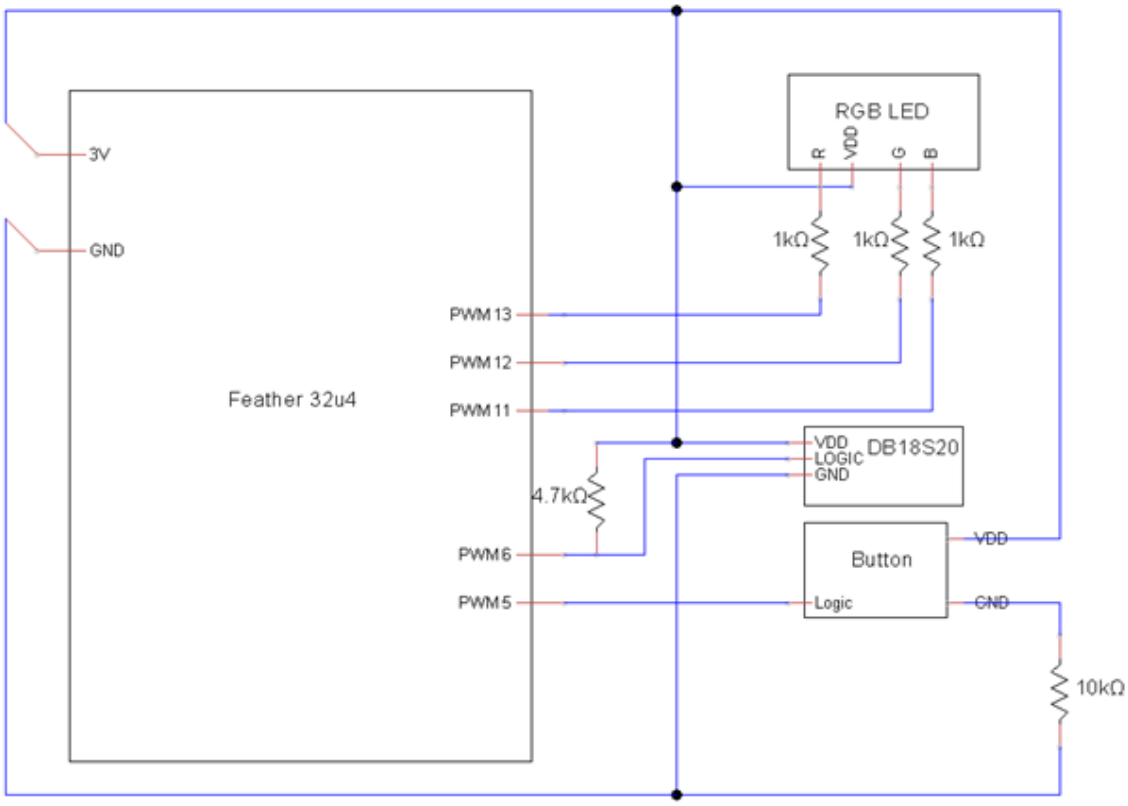

FIGURE 8: CIRCUIT DIAGRAM OF THE TEMPERATURE SENSOR

With the current progress of the temperature sensors, we have satisfied the various requirements set out for the prototype phase. The temperature sensor is currently situated on two small breadboards without an enclosure. Again, as with the vent module, while it is not a requirement we targeted for the prototype, we would have liked to have enclosures for protection and aesthetic appeal but this will have to be considered in the next phase due to time constraints. Aside from enclosures, we will consider our options in PCB creation instead of the feather development board to further downsize our temperature sensor and strictly cater to our needs. The temperature sensor also currently requires and takes up a wall outlet slot and in the next phase, we will be pursuing options to offset this by introducing a USB port capable of providing power.

3.3 Master Thermostat Design

The Master Thermostat is designed to be the heart of the VentNet system as it is the module responsible of controlling the state of the home furnace unit and it connects the Router Module and Temperature Sensors such that the communication between it can be established. This is exemplified by requirements *R1.3.14* (The thermostat shall provide sensor and furnace data to the Router Module), *R1.3.15* (The thermostat shall receive temperature data from the temperature sensors), *R1.3.16* (The thermostat shall receive commands from the Router Module and execute them), and *R1.3.17* (The thermostat shall be able to change the temperature settings stored on the temperature sensors). Additionally, the final high-level requirement for the Master Thermostat was that it needs to have a display that indicates the current temperature, statuses, and alarms [*R1.3.3*] working alongside physical buttons [*R.1.3.4*] as a means of being able to manually adjust the desired temperature for each zone of the VentNet system.

TABLE 2: RASPBERRY PI 3 SPECIFICATIONS [9]

Device	Raspberry Pi 3
Size	85mm x 56mm x 17mm
Weight	41.2 grams
Memory	1GB RAM LPDDR2
Processor	1.2GHz Quad-Core Broadcom BCM2837 64-bit ARMv8
Power Supply	5V1/2.5A Micro USB socket
WiFi	BCM43143 WiFi Chip (802.11 b/g/n Wireless LAN)
Bluetooth	Bluetooth 4.1 (Bluetooth Classic and LE)
Connectors	4 USB Ports 40 GPIO Pins Full HDMI Port Ethernet Port Micro SD Card Slot

For the proof-of-concept stage, utilizing the Raspberry Pi 3 was the perfect fit for a few important reasons. To preface this, we at Aeolus Systems desired to work with a single board computer that offered a flexible environment, a low entry cost, compatibility with physical buttons and a screen, and a system with plenty of resources available such that development of the Master Thermostat could be feasible and easy to learn within a strict schedule. With the Raspberry Pi 3 being advertised as having a strong presence in the environmental monitoring and IoT applications categories, it fulfills the previous requirements which made the decision to begin development of the Master Thermostat very easy. The Raspberry Pi 3 is a powerful machine offering plenty of built-in connectivity, such as built-in WiFi and Bluetooth, and it is packed into a non-intrusive small form-factor, as can be seen by the specifications in Table 2. Additionally, the Raspberry Pi 3 contains a powerful feature, namely the row of General Purpose Input/Output (GPIO) pins along the side of the board as shown in Figure 10. The programmable GPIO pins is the physical interface between the Raspberry Pi and the external parts required to complete the Master Thermostat. Specifically, the GPIO pins provides the capability of reading digital logic signals and outputting low-current digital logic levels. In terms of the Master Thermostat, the Raspberry Pi 3 will be utilized to push a 24V AC signal to switch the signal of the HVAC system and allows a connection between the board and the physical screen.

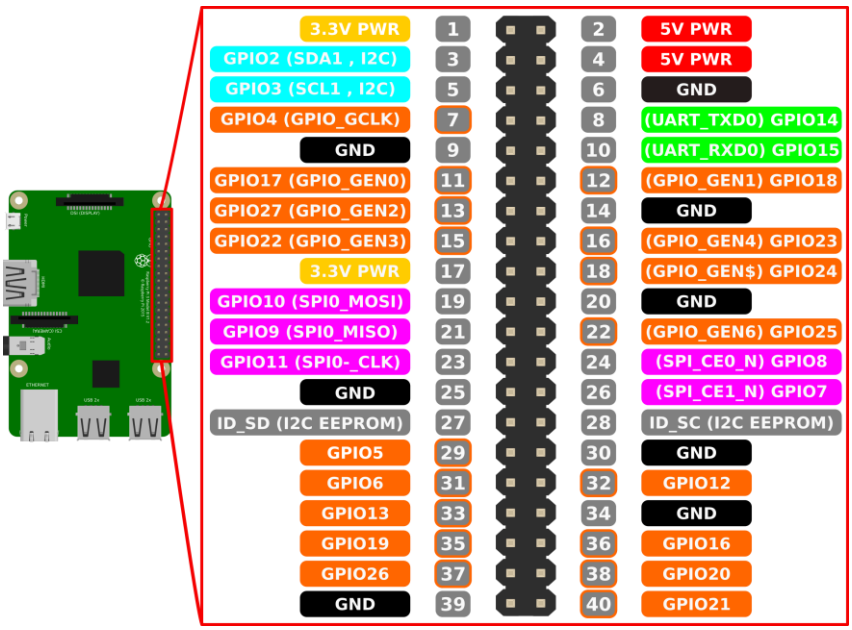

FIGURE 9: RASPBERRY PI 3 GPIO PINS DIAGRAM [10]

The Master Thermostat's job is to close the circuit between the live wire of the furnace system and the appropriate control wire. Using the Raspberry Pi 3 to push this signal and change the state of the home's furnace system, it is a requirement to have additional hardware as it is not possible for the Raspberry Pi 3 to independently push a 24V AC signal. As we can see in Figure 10, the Raspberry Pi 3 is only capable of pushing 3.3V from its GPIO pins. Therefore, the 24V AC Solid State Relay Board is a suitable candidate for transforming the Raspberry Pi's low-voltage control signals into one that is capable of switching 24V AC lines [11]. Figure 11 below is the schematic of the relay board that receives the 5v input from the Raspberry Pi 3 and transforms it using the bidirectional triode thyristor (TRIAC). The TRIAC's behaviour allows the system to act like a relay as it is designed for AC power control, providing reliable, uniform switching for full-cycle AC applications.

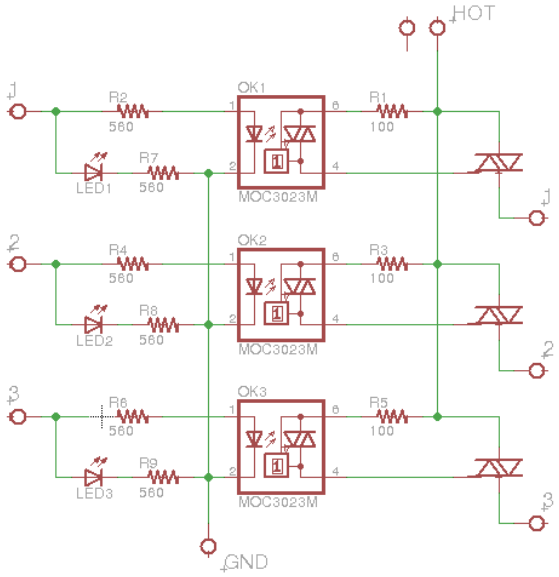

FIGURE 10: CIRCUIT DIAGRAM OF THE 24V AC SOLID STATE RELAY BOARD [11]

TABLE 3: KUMAN 3.5-INCH TFT LCD DISPLAY SPECIFICATIONS [12]

Device	Kuman 3.5-Inch TFT LCD Display
Size	55mm x 84.96mm x 2.5mm
Resolution	320 x 480
Input Voltage	2.8~3.3V
Current Draw	100 mA
Operating Temperature Threshold	60°C
Storage Temperature Threshold	70°C

As stated earlier, requirements *R1.3.3* and *R1.3.4* dictate the requirement for a physical means of manually adjusting the temperature setting of the VentNet System. Initially, our plan was to purchase a LCD screen and physical buttons as a means of fulfilling the requirement. The LCD will be used to showcase the monitoring features of the VentNet System and the buttons would manipulate the adjustable settings on the Master Thermostat. As a result, a decision needed to be made on what type of screen would be utilized according to the different trade-offs of price, ease-of-use, integration time, and risk. In addition, the requirements document also specified that "the thermostat shall be powered by battery" in *R1.3.18* indicating the trade-off of using a screen with lower power consumption. The two Raspberry Pi 3 compatible LCD screens that passed the requirements were the RGB 16x2 Character LCD and the 3.5-Inch TFT LCD Display with touch screen capabilities.

Table 4 below compares the pros and cons of the three display choices. In terms of price, the Adafruit RGB 16x2 Character LCD + Keypad Kit is priced at $19.99 USD [13] whereas the Kuman 3.5-Inch TFT LCD Display is priced at $19.29 USD [12] showing a marginal advantage to the latter. For ease-of-use, both displays have plenty of resources and online tutorials available. For integration time, the advantage is given to the 3.5-Inch TFT LCD Display because of the touch screen. By utilizing the manufactured drivers, it is simple to create a user-friendly interface with software buttons reducing the risk of not having enough physical buttons for the system. Additionally, the risk of failing physical buttons and having to order new parts is eliminated as the programmable touch screen replaces it. The only concerning tradeoff is the requirement of low power consumption where the RGB Character LCD has the distinct advantage. The RGB Character LCD has a consumption of ~10-40mA of power [13] whereas the 3.5-Inch TFT LCD has a consumption of ~100mA.

TABLE 4: RGB 16X2 CHARACTER LCD VS. 3.5-INCH TFT LCD - PROS AND CON

Raspberry Pi 3 LCD Screens	RGB 16x2 Character LCD	3.5-Inch TFT LCD
Price	✓	✓
Ease-of-Use	✓	✓
Integration Time	✗	✓
Risk	✗	✓
GPIO Pin Usage	✓	✗
Low Power Consumption	✓	✗

Although the RGB 16x2 Character LCD display is desirable for the requirement of powering the Master Thermostat through batteries, the final decision was to go with the 3.5-Inch TFT LCD. The requirement of R1.3.18 shall be pushed into the next development phase and the Master Thermostat will utilize a dedicated power supply instead, after weighing the various trade-offs so that we will not have to worry about malfunctioning parts and focus our time on development.

Thus, the Master Thermostat was constructed as a system of the following parts:

- Raspberry Pi 3
- Kuman 3.5-Inch TFT LCD Display
- 5.25V/2.4A Micro USB Switching AC Power Supply
- RFM69 Radio

For the proof-of-concept stage, the Master Thermostat requirements have been modified due to timing constraints and flexibility. As stated earlier, the requirements of a low-power LCD display and powering the Master Thermostat by battery have been shifted over to the second phase of development. This allows the usage of the Raspberry Pi 3, which offers flexibility and has a plethora of resources, and a touchscreen LCD display, which eliminates the necessity of physical buttons and offers further flexibility in terms of UI and development. For the following stages, namely the prototype and production stage, the construction of the system will be entirely revamped so that the previously adjusted requirements can be reinstated. We aim to construct a Master Thermostat using a battery-powered chip with a low power consumption display such that users will not have to worry about finding a dedicated power or constantly having to replace batteries.

3.4 Router Module and Web Application

3.4.1 Overview

The Router module contains a major part of the UI/UX of the VentNet system. Users would be able to set schedules for their vents, check temperature readings, and see the status of the system. Behind the scenes, the Router module would be listening for data to be sent from the Master Thermostat and converting the raw data into meaningful results.

3.4.2 Physical

It was decided that the Router module would run on a Raspberry Pi 2 for many of the same reasons given for the Master Thermostat. While less powerful than the Raspberry Pi 3, a Raspberry Pi 2 was available to use from one of our members who used it for hobby projects. We also decided against using a touch sensitive display contrary to our [R1.4.1] requirement because we wanted to limit the amount of user interaction with the physical modules to only the Master Thermostat while keeping the Router module out of sight.

TABLE 5: SPECIFICATIONS OF THE RASPBERRY PI FOR ROUTER MODULE [14]

Device	Raspberry Pi 2
SoC	Broadcom BCM2836
Processor	900 MHz ARM-Cortex A7
Storage	8GB
Memory	1GB RAM
Connectivity	Ethernet
Power Supply	5V/2A
Ports	4x USB 2.0 1x HDMI 1x microUSB Micro SD Card SLot
GPIO	40 pins

Thanks to the processing power of the quad core Cortex A7 in the Raspberry Pi, we can perform all the computations necessary to maintain the web server while communicating with the Master Thermostat.

The Router module is connected to the internet via Ethernet to host the web server and powered through a wall adapter. This way we can keep the Router module in a central location in the house where it can easily pair with the main thermostat module. It also removes some of our concerns about wireless interference by having one of our modules wired directly to the home network.

3.4.3 Frontend

The frontend is the webpage that users view when accessing their VentNet from their computer or mobile device. It is a basic HTML page that employs CSS for its appearance and styling and JavaScript for controlling the behaviour of different elements on the webpage.

On the webpage, users are provided with information corresponding to current temperatures and target temperatures for each registered zone on the left side. On the other side, users can make changes to their system or set new values for the different zones in their household. More about the frontend design choices are discussed in the UI Appendix.

3.4.4 Backend

Our web server is built using Flask, which is a Python web framework designed to simplify the building of web application, which is well suited for our group of fledgling web developers [15]. By providing only the core functionalities and allowing developers to customize their web applications with libraries, Flask is more lightweight and readable compared to other options. Through Flask we can tailor our web application to demonstrate basic functionality for the proof-of-concept stage.

Flask uses Jinja templates to builds its webpages [16]. It allows Flask to load content into placeholders on the webpage to display whatever is stored in the server. We use it to display the temperature values being sent from the Main Thermostat module to the Router module.

Due to the time crunch, we realized we could use the SPI to intercommunicate between the Router and Thermostat modules. Since we already use SPI to communicate between the Thermosensors and Master Thermostat, it would take a minimal amount of development time to port it over. Unfortunately, this portion of the requirements in regards to intercommunication was delayed and we are unlikely to be presenting the full functionality of the module as documented in the requirements specifications.

3.5 Wireless Protocol and Radio Design

The wireless communications system that connects the modules of the VentNet system must be capable of alleviating power consumption concerns on battery powered modules whilst also providing a robust framework for pairing and signalling. Attacking these objectives is a three-fold approach. A low power and efficient radio chip geared toward IoT applications will be used to transmit signals. The wireless communications protocol will place power hungry transmit actions onto continuously powered modules and address collisions on the network will be resolved at pairing time.

3.5.1 RFM69HCW Radio Chip

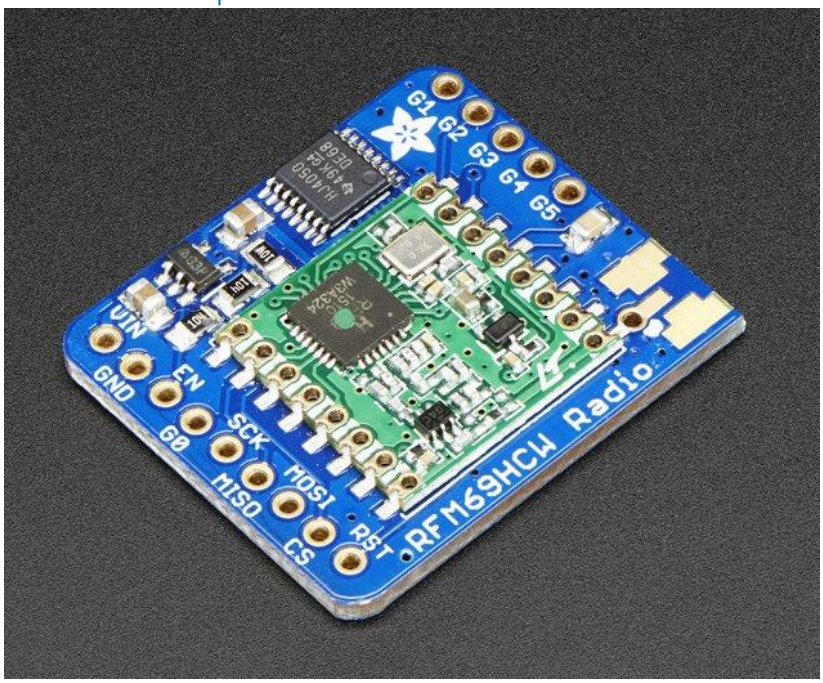

FIGURE 11: RFM69HCW RADIO WITH BREAKOUT BOARD

The radio hardware to be used will be the 915 MHz RFM69HCW from HopeRF. From a EM physics stand-point, lower frequency signals have better 'skin-depth' penetration properties. In practice, they are able to travel farther through a medium for the same amount of power. By using a sub-gigahertz signal the VentNet is able to better send signals through the walls and floors of a house to reach all the dispersed modules. Using the 915 MHz frequency also comes with the added benefits of running on the unlicensed amateur radio band in North-America which is free to use. The major downside of using a low modulating frequency is the reduced bandwidth but that is negated by the size efficient messages used in the signaling profile. Note, in the EU, the unlicensed amateur radio band is at 868 MHz and this radio can be reprogrammed in software to use this frequency.

The power consumption of the radio has been tested to use only 53mW during listening, whilst transmitting can consume up to 429mW [17]. These figures appear high but by implementing deep sleep intervals and automatic gain control the power consumption can be reduced greatly. More details regarding deep sleep and automatic gain control can be found in the signalling profile section and AGC section.

For this iteration of the development cycle we are targeting a proof of concept model as the goal. In the proof-of-concept model the radio chip will be attached by ribbon cable to the Raspberry Pi whilst it will come built onto the Feather Development board. The primary specifications of the RFM69HCW radio module are listed below:

TABLE 6: KEY RFM69HCW SPECIFICATIONS [17]

Size (with breakout)	16mm x 16mm x 1.4mm (25.3mm x 29.3mm x 3.5mm)
Input power (with breakout)	1.8V - 3.6V (3V - 5V)
Max Tx power output	+20dBm / 100mW
Tx power use (at +20dBm)	130mA / 429mW
Rx power use	16mA / 52.8mW
Rx sensitivity	-120dBm
Sleep power use	0.1µA / 0.33µW
Max bitrate	300 kb/s
Packet engine	66-byte FIFO, CRC-16
Encryption	AES-128
Operating temperature	-55°C to +115°C

3.5.2 Network Organization

During normal operation, the modules of the VentNet system talk to each other on a custom wireless protocol. This ensures communications are optimized to be energy efficient in a home environment with many walls and floors.

Owing to the use of the RFM69HCW radio module, we will be using the LowPowerLab RFM69 library [18] and its Python port. This library operates the radio and interfaces with our module firmware but it has some defined capabilities. Radio messages are sent on any one of 256 network addresses to any one of 256 node addresses in the network. These messages may be encrypted by a 128bit encryption key but are limited to 61 bytes to allow for timely processing on the RFM69 hardware. Working with the addressing and encryption scheme of the library the VentNet wireless network is organised in the manner shown in figure 12 to ensure no address collisions occur between modules in a home or between different homes.

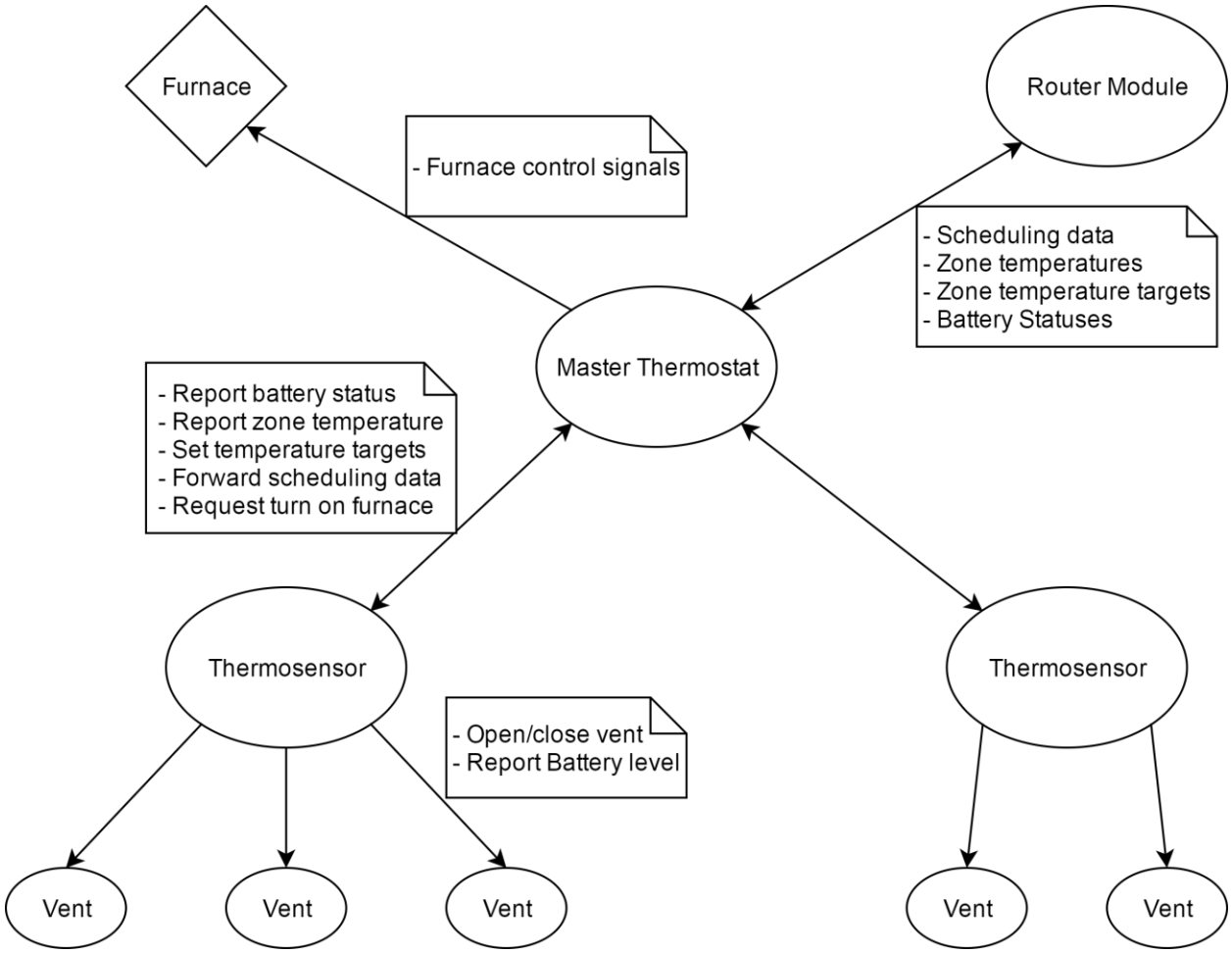

FIGURE 12: SIGNALS PASSED DURING NORMAL OPERATION OF VENTNET

In implementing the signalling profile of the wireless protocol, we organise the modules into a hub and spoke network with the Master Thermostat and Thermosensors acting as communications hubs. Each hub (master) establishes its own unique network number to talk to modules on its spokes (slaves). The allocation of network numbers (IDs) is handled during the pairing phase between the Master Thermostat and Thermosensor to ensure unique network IDs are used by each master. The node IDs on masters on its own network is always 0 whilst node IDs for slaves are decided by the master during pairing between master and slave. The VentNet systems of different homes are prevented from encroaching on each other by the unique encryption keys used in their communications derived from the serial number of the controlling Master Thermostat.

Of the modules in the VentNet system the Thermosensor is the only device that acts as both a master and a slave as it is in the middle tier of the network. When communicating with vents the Thermosensor sets its network ID to that of its own network and its node ID to the default master node ID of 0; when communicating with the Master Thermostat it sets its network ID to that of the Master Thermostat and its node ID to the one given to it by its Master Thermostat. This way, the Master Thermostat need not handle the vents directly as transmitting commands to each vent is power intensive and the Thermosensor being a wall-powered device is better suited for the task.

The Router Module, whilst being a slave device in the network handles many of the same settings that the Master Thermostat can. As the host of the web interface for our network the Router module is the access point for many advanced settings and the network could just as easily have been designed around it as the primary hub. However, for the proof of concept model the network will be organised as described above with the Master Thermostat as the primary hub. Going forward, the network design may change to reflect greater optimization for power efficiency by moving the primary hub to the Router Module; or, kept as is to allow for the advanced features provided by the presence of a Router Module to be optional when building a VentNet system.

3.5.3 Pairing Procedure

Catering to the fanout present in the network organization the pairing procedure ensures that all network and node addresses are unique to avoid data collisions. This necessitates a strict pairing procedure where devices must be paired in order:

1. **Master Thermostat is paired with Thermosensors**

   The Master Thermostat's serial number is used as the encryption key to avoid crosstalk between VentNet systems in different houses. The Thermosensors being the only other master type devices in the network will receive the unique network IDs they are to use for their networks from the Master Thermostat. This way, the Master Thermostat can organise unique addresses for all the Thermosensors. To keep things organized, the unique network IDs doled out to the Thermosensors will also be their node IDs when acting as slaves on the Master Thermostat's net.

2. **Master Thermostat is paired with Router Module**

   The Router Module must be able to track the temperatures and settings in each heating zone so it must also know which Thermosensors are in the home. By pairing the Router Module after the Thermosensors this allows for the Master Thermostat to pass on the list of heating zones in a house during pairing. This aspect of the design could change in future iterations to allow for a different pairing order.

3. **Thermosensors are paired with Vents**

   Only after a Thermosensor has been paired with a master thermostat can it accept pairing requests from Vents. Without a unique network ID and encryption key, there would be no way to avoid data collisions.

The LowPowerLab RFM69 library provides functions to send message with or without ACK. ACK is short for acknowledge and allows for senders to be notified that their message was received. The radio driver library used by VentNet also allows for the receiver to decide whether send back an ACK. This facilitates conditional checks on received messages to ensure valid messages are received before acting on them. It also allows for eavesdropping without being noticed. Utilizing the flexibility afforded by the driver, our software takes the following steps when pairing mode is initiated:

1. On each module, a button and RGB LED will be used as the UI for pairing. When held down for a few seconds the pairing button will bring the device into a pairing mode and make the LED blink blue for the duration of pairing.
2. The device accesses a predefined pairing network (ID 255) using a predefined node ID depending on its device type and a predefined public encryption key.

   The Thermosensor is unique in that it is both a master and a slave so when selecting its predefined pairing node ID, it will choose between two different IDs depending on if it has been paired to a Master Thermostat or not. This decision will also affect whether it is considered a master or a slave during pairing.

3. To ensure there are not multiple devices of the same type on the pairing network at the same time, master devices will first ping the node address for its device type. If there is a response, it means a master is in the active listening state described in the next step. The device that sent the ping exits pairing mode with a failure.

   Slave devices attempt to eavesdrop (read messages but not send back ACK) on the node ID of its targeted master device. If a message arrives within a second announcing a slave on the network with the same type then this slave will exit pairing mode with a failure.

   If pairing mode is exited, the LED will blink red for a few seconds before subsiding. Otherwise if the device has reasonably ascertained there are no similar device types to its own on the network it proceeds to the next step.

4. Master type devices will attempt to active listen whilst slave devices will actively broadcast their device type to their targeted master device type node ID. For a master type device, if a ping arrives from another device of the same type then an ACK will be sent to the other device notifying it that the pairing slot is not vacant.
5. If two devices that can pair with each other are in their pairing states, then eventually (within a couple milliseconds) the master will catch the slave's signal and read its device type. If the message and device type is deemed to be valid then the master will then send an ACK to the slave.
6. Now that both devices have detected each other's presence and the master knows what kind of slave is requesting pairing, it will send a message back to the slave's predefined node ID containing the master's network ID, the node ID for the slave, and the 128 bit encryption key. This message is encoded as a 144 bit message with the first 8 bits denoting the network ID, the next 8 bits denoting the node ID, and the remaining 128 bits as the encryption key.

   Note that for the Thermosensor the network ID and encryption key are given to it by the Master Thermostat during pairing with the Thermosensor's network ID being its node ID on the Master Thermostat's network.

7. When the slave receives the message from the Master with its networking info it sends back an ACK.
8. Now that both devices have shared networking info the data is saved in flash memory and the devices reconfigure their radios to use the new network ID, node ID, and encryption key.
9. Before exiting the pairing mode the master generates a short random color sequence with 4 blinks of 4 possible colors. It sends this sequence to the slave using the new network and node IDs to test its connection. Upon receiving an ACK from the slave both devices will blink the color sequence on the RGB led to display to the user that the correct devices have been paired.

3.5.4 Signalling Profile

During normal operation, a set of specific messages are used when communicating between modules to ensure they can be verified by the receiver. The message set will use the first 8 bits of a packet to encode the message ID. The radio driver automatically encodes the node ID of the sender into the message and allows for it to be read separately by the receiver. These properties are accounted for in the message set described in Table 7:

TABLE 7: MESSAGES USED DURING REGULAR OPERATION

ID	Command	Sender	Receiver	Period	Encoding
0	General message	Any	Any	Async	{[char character]}…
1	Ping	Any	Any	Async	[uint8_t target device ID or no ID]
2	Request time	Any	Any	Async	Nothing

3	Send time	Any	Any	Async	[uint8_t temp. target][uint4_t day of week][uint4_t hour][uint8_t minute]
4	Report vent battery level	Vent	Thermosensor	10 m / on boot	[uint8_t battery level]
5	Vent open/close	Thermosensor	Vent	Async	[uint8_t % to open to]
6	Report zone temp.	Thermosensor	Master Thermostat	60 s / on boot	[uint8_t temp.]
7	Furnace on/off	Thermosensor	Master Thermostat	Async	[bool on/off]
8	Report lowest vent battery level	Thermosensor	Master Thermostat	10 m / on boot	[uint8_t battery level]
9	Set temp. target	Master Thermostat	Thermosensor	Async	[uint8_t temp. target]
10	Set schedule array	Master Thermostat	Thermosensor	Async	[uint8_t # of temp. targets] {[uint8_t temp. target][uint4_t day of week][uint4_t hour][uint8_t minute]}…
11	Use schedule on/off flag	Master Thermostat	Thermosensor	Async	[bool on/off]
12	Report zone temp. array	Master Thermostat	Router Module	60 s	[uint8_t # of zones] {[uint8_t zone ID][uint8_t temp.]}…
13	Report lowest zone battery array	Master Thermostat	Router Module	10 m	[uint8_t # of zones] {[uint8_t zone ID][uint8_t lowest battery level in zone]}…
14	Set temp. target	Router Module	Master Thermostat	Async	[uint8_t zone ID][uint8_t temp. target]
15	Set schedule array	Router Module	Master Thermostat	Async	[uint8_t zone ID][uint8_t # of temp. targets]{[uint8_t temp. target][uint4_t day of week][uint4_t hour][uint8_t minute]}…
16	Use schedule on/off flag	Router Module	Master Thermostat	Async	[uint8_t zone ID][bool on/off]

As the Vents and Master Thermostats are in a low power sleep mode most of the time, messages may need to be resent until it is acknowledged. The resend interval is 250 ms whilst the number of times to retry will be 120; this leads to a 30 second timeout period for sending before it is considered as failed.

On the receiving side, a device waking from sleep to receive messages will only turn on its radio for 550 ms to catch messages. If a message is caught, verified, acknowledged, and processed, the radio remains in listening mode for another 550 ms to catch more messages. This repeats until all valid messages are processed.

In phase 2 development we hope to tune the timeout period for sending and receiving for specific messages and device types to further increase energy efficiency.

3.5.5 Automatic Gain Control

At full transmit power the radio is using 429 mW; this is especially concerning when transmitting on battery power devices. The default RFM69 library from LowPowerLab allows for 32 levels of output amplification. By modifying the acknowledge function in the library to include the signal strength at which the message was received the firmware will be able to tune to transmission gain toward a quiet but acceptable level. The sample library for doing this has been started by Thomas Studwell but further modifications are necessary to account for all the sources of interference in a home environment.

3.5.6 Requirements fulfillment

Overall, the P1 requirements necessary for the proof of concept demo are met with the current design. Hardware, network organization, pairing, and signals were all designed to fulfill network requirements necessary for the VentNet system. Table 8 below summarizes the wireless protocol requirements and their state of incorporation into the design:

TABLE 8: REQUIREMENT FULFILLMENT IN CURRENT DESIGN

Requirement	Fulfillment	Details
R2.5.1 – P1	yes	RFM69HCW with breakout is smaller than the required 3cm x 3cm
R2.5.2 – P1	yes	Radio antenna will be a flexible wire wrapped around inside of device enclosure
R2.5.3 – P1	yes	Operating temperature range of RFM69HCW exceeds suspected environment temperature range of -10°C to 60°C
R1.5.1 – P1	yes	Network supports multiple devices via the addressing system from the LowPowerLab RFM69 library and our pairing procedure that avoids address collision
R1.5.2 – P1	yes	RFM69HCW is capable of broadcasting on the 915MHz amateur radio band
R1.5.3 – P1	Further investigation needed	Pairing procedure allocates addresses to avoid data collisions but channel congestion may still occur. Further investigation with many many radios is needed to verify if congestion exists
R1.5.4 – P1	yes	Current set of messages in signalling profile supports all core messages that need be sent
R1.5.5 – P1	yes	SendWithRetry (request ACK) is used for all messages in the signalling profile to ensure complete transmission
R1.5.6 – P1	yes	Pairing procedure allows for reliable pairing right now
R1.5.7 – P3	Partial implemented	Network data is saved to flash after pairing but is currently not loaded on bootup
R1.5.8 – P2	yes	RFM69HCW has a sleep mode which draws sub micro watts
R1.5.9 – P1	yes	RFM69HCW breakout interfaces via SPI
R1.5.10 – P2	yes	Current high radio power is capable of near 100% transmit rates however automatic gain tuning may lower this transmit rate to conserve power

4 Conclusion

The design specification provides a low-level understanding of the design choices and implementations used in the creation of the VentNet proof-of-concept and the solutions to the requirements previously established. In particular, hardware choices were justified and in-depth software details such as an extensive look into the pairing protocol were discussed. The document also outlines the goals for the prototype phase and test plans followed by Aeolus Systems to ensure product quality, safety and reliability. In contrast, the UI appendix features a high-level look at the design elements that were specifically chosen to provide a pleasant and smooth experience to potential end-users.

Our goal is to address the heating issues revolving around forced-air furnaces and provide an easy solution for homeowners that would promote a more comfortable home environment. To accomplish this, the development and design of the VentNet took place across five main components: the vent covers, temperature sensors, the master thermostat, the router module and the wireless protocol before an integration period to bring the system together. At the time of this document the VentNet is in the middle of the integration stage and will feature four of the five main modules for the proof-of-concept presentation on April 5th, 2017.

Test Plan Appendix

Test Case – Vent Power Functionality

Actions/Steps	Expected Result
1. Insert 3xAAA batteries to the vent and switch the battery pack on	• LED will blink green if the battery is 25%-100% or red if the battery is below 25% for 3 seconds to indicate it has been turned on. • LED then stops blinking
2. Turn off the battery pack switch	• Vent circuit turns off within 3 seconds
3. Turn on and leave the vent module on for a week	• Vent module should still be powered at the end of the week
4. Have the vent module on and hot air blowing towards the vent	• The microcontroller of the vent module should not overheat • The plastics inside the vent module should not overheat
5. Have hot air blowing towards the vent and the vent closed	• The temperature should maintain its current level in the room
6. Using the microcontroller, open or close the vent cover	• Motor should be turned on for an instance and the vent cover should open or close
7. While the vent module is powered, press the button down for two seconds	• The LED should blink blue for 30 seconds.

Test Case – Temperature Sensor Individual Functionality

Actions/Steps	Expected Result
1. Connect a micro USB power adapter to the temperature sensor module	• LED will blink green if the battery is 25%-100% or red if the battery is below 25% for 3 seconds to indicate it has been turned on. • LED then stops blinking
2. Turn on and leave the temperature sensor on for 3 seconds	• The temperature sensor should be able to report a temperature that is +/- 0.5C of the current temperature

Test Case – Vent module and Temperature Sensor interaction

Actions/Steps	Expected Result
1. Connect a prepaired set of vent and thermosensor to the Arduino serial monitor	• Both devices are now reporting on the serial terminal
2. Use serial terminal to set target temperature of thermosensor to ambient temperature	• Serial monitor should show that thermosensor registers this change
3. Use fingers to hold thermosensor probe tightly	• Vent should close once temperature change is registered • Serial monitor should report message transmission and ACK
4. Place ice pack on thermosensor probe	• Vent should open once temperature change is registered • Serial monitor should report message transmission and ACK

Test Case – Master Thermostat Functionality

Actions/Steps	Expected Result
1. Connect a micro USB power adapter to the Master Thermostat	• The LCD display backlight shall light up and display a boot screen with the company logo • When booting is completed, the LCD display shall display the main screen showing the current temperature and the desired temperature
2. Increase the desired temperature by tapping the up arrow	• The displayed desired temperature shall increment by 0.5
3. Decrease the desired temperature by tapping the down arrow	• After tapping the down arrow, the displayed desired temperature shall decrement by 0.5
4. Tap the power button	• The LCD display shall display an exit screen • The LCD display backlight shall turn off

Test Case – Device pairing

Actions/Steps	Expected Result
1. Connect an unpaired set of vent and thermosensor to the Arduino serial monitor	• On both devices LED blinks green for 3 seconds then subsides
2. Press the pairing button on both devices	• LED blinks blue during pairing • When pairing is finished the same random sequence of 3 blinks occurs on both devices
3. Use serial terminal to read the network ID and node ID used by both devices	• The master device should now be using node ID 0 • The slave device should be using the net ID of the master • The slave device should have a node ID that is the lower available node ID on the master's network

Test Case – Set Temperature from Master Thermostat

Actions/Steps	Expected Result
1. Power on a prepaired set of thermosensor and master thermostat	• Pairing procedure concludes successfully
2. Connect thermosensor to the Arduino serial monitor	• Thermosensor now reporting on the serial terminal
3. From touch GUI on master thermostat set the target temperature of the thermosensor to a new value	• After the master thermostat enters its transmission period the command will be sent to the thermosensor • Arduino serial terminal will show the thermosensor registering this new temperature

UI Appendix

Introduction

The user interface is the means in which users can interact with a system. In terms of design, the user interface is vital for determining the success of a product. Even if the product fulfills all of its detailed requirements but has a poor user interface, the product will not reach its full potential and fall short of reaching success. Any interface too complex and unintuitive will push away users compared to designing an interface that is simple and effective, which would increase both the usability of the product and the user experience. However, having an amazing user interface does not necessarily mean that the system will succeed as an imbalance of great user interface and poor functionality will still lead to failure. In the UI appendix, the document begins with an analysis of the users of VentNet. It will outline their backgrounds and experience with previous systems and how it will translate to using VentNet. Secondly, a technical analysis of the VentNet system in regards to the "Seven Elements of UI Interaction" will detail out the user interface design of the VentNet system and the individual modules. User interface engineering standards related to the VentNet design at this proof-of-concept stage will then be outlined and the document will end with sections speaking about analytical and empirical usability testing.

User Analysis

As the goal of this development cycle is the production of a proof of concept model the targeted user for this version of the VentNet will be an experienced Aeolus Systems engineer. In its current state, knowledge regarding how to interface with the firmware through a terminal and work around potential bugs will be necessary to operate the demo device. However, going forward we will be developing a consumer oriented user experience for the VentNet in the second development cycle (ENSC440W). To establish the ground work for that we begin by analyzing the targeted users for the VentNet.

The production version of the VentNet System is designed to be an upgrade upon regular thermostat controlled furnace heated homes and so its Master Thermostat UI will cater to everyone that can use a regular thermostat. We expect the user of regular thermostats to be average people capable of deciding on a temperature for their room, discerning between the few labeled and shaped buttons on the Master Thermostat to navigate a menu with depth of 2, and set the temperature with the buttons. Overall this is not a significantly different task from operating a normal thermostat and should be suitable for people that have encountered a thermostat previously.

As the VentNet's user interface also includes a small website for accessing data and changing the settings, a more advanced user with experience using the internet will be necessary to operate the settings from the connected website. This kind of user will be capable of reaching the website via a written URL, navigating a simple webpage, and using buttons on the webpage to change settings in a visual layout not unlike that of a regular thermostat. In this day and age, most people in first world countries have been exposed to the internet and would know how to operate controls on a website in a simple manner.

The most advanced and technically challenging aspect of the VentNet system for a user will be its installation. To install the VentNet system the user must be willing to turn off their furnace and unscrew the wires connected to their current thermostat. Many homeowners are uncomfortable to work with electrical elements of their home and so this task may fall to a professional contractor, electrician, or just someone else willing. The remaining installation steps involve pairing the devices and will be akin to Bluetooth pairing. The user will need to read an instruction manual and hold a pairing button on two devices. As the installation process involves some technical challenge we are designing the VentNet to be installable by people who can use a screw driver, follow instructions from a manual, and is willing to do the task.

In summary, different aspects of the VentNet system is designed for different kinds of users. The Master Thermostat is designed to be very similar to a regular thermostat and is geared toward the average person with experience using a regular thermostat. The website will be kept simple but a user with experience using a web browser to navigate a website will be the targeted user. Installation is the most challenging task when considering the VentNet system but should be accomplishable by those who can use simple tools, read the instruction manual, and is willing to do the task.

Vent Cover

The vent cover has a button, battery pack and LED exposed to the user. For the prototype phase, a user has the affordability of switching the power of the vent module on and off through a switch on the battery pack and pairing with a temperature sensor. The button element is used for pairing purposes with a temperature sensor and the LED provides feedback to the user.

To help the user follow along on, we designed the LED and button to behave in the following manner:

1) Hold down the button for 2 seconds to initiate the pairing process
2) The LED begins blinking blue while the microcontroller attempts to pair
3) If the device is not paired 30 seconds after pairing is initiated, the LED stops blinking
4) If the device is paired, the LED will blink a color sequence at the same time as the paired device

To show the battery level and provide feedback to the user that the vent module has powered on, the LED will either blink green or red for 3 seconds depending on its current battery level when it is turned on. In addition, a user may choose to press the button to get the current battery level.

At this time, no enclosures are available to hide the circuitry inside the vents and would not comply with engineering standards as a market device. We plan to address this for the next phase and by buying or 3D printing a cover so the user will only be exposed to the battery pack, button and LED. To provide safety for the prototype, we will be covering the h-bridge as the currents flowing may be harmful if a user is exposed.

Temperature Sensors

The temperature sensor will only have a button and LED exposed to the user. As with the vents, we followed a similar procedure when pairing for consistency of discoverability across our devices:

1) Hold down the button for 2 seconds to initiate the pairing process
2) The LED begins blinking blue while the microcontroller attempts to pair
3) If the device is not paired after 30 seconds, the LED stops blinking
4) If the device is paired, the LED will blink a color sequence at the same time as the paired device

The LED will also provide a short blinking sequence to indicate that it is currently powered and can be toggled on through the button by a user to check whether the module is properly powered.

While not currently ready, we plan to create an enclosure for the temperature sensor in the next phase. The likely approach we will take is to use 3D printing to create a plastic box enclosure for our temperature sensor with a hole for the micro USB power slot. Depending on changes to the temperature sensor design the enclosure size may be reduced in the next phase.

LED Allocation

To facilitate pairing and visually interfacing with the user on devices that do not have a screen an RGB LED has been installed on all modules. Table 9 which lists the functions of each color can be found below:

TABLE 9: LED COLOR ALLOCATION

Color	Function
Red	• Less than 15% battery remaining (Vent only) • Device has not been paired
Green	• 15% to 100% Battery (Vent Only) • Regular operation
Blue	• Used to show device is currently searching for a pairing partner
Purple	• Used for successful pairing verification sequence only
Orange	• Used for successful pairing verification sequence only
Yellow	• Used for successful pairing verification sequence only
White	• Used for successful pairing verification sequence only

The colors were chosen in an attempt align with everyday conceptual models, red for low battery, green for high battery and blue for pairing (often associated with Bluetooth). The final step in pairing involves testing the connection by sending a random sequence of 4 colors from one device to the other and playing out the sequence on the LED to verify that the two correct devices have indeed been paired. This verification uses 4 colors reserved specifically for this step so as to not confuse the user by using colors associated with regular operation or pairing.

Master Thermostat

In order to utilize the proof-of-concept Master Thermostat, the touchscreen LCD and micro USB port must be exposed so that there can be an interface for the user with the system. The micro USB port shall be used to power the device and the touchscreen LCD shall be used to read the temperature monitoring information and adjust the desired temperature for each zone. In regards to the "Seven Elements of UI Interaction", this section will go through discoverability, feedback, conceptual models, affordances, signifiers, mappings, and constraints. Discoverability can be defined as how easy it is to figure out how to use an object through interaction. By limiting the means of interacting with the Master Thermostat, the discoverability of the system increases because of the system constraints and how it eliminates the capability of all other interactions. Additionally, the software of the Master Thermostat is based off a library called pygame. Pygame is a simple open-source python programming language library that features interactive multimedia objects and has plenty of resources. As such, we used this library to develop a touch-friendly application by permitting large and easily readable software buttons for the user. Not only does the simple and easy to read software interface offer discoverability, the usage of different colors for button state changes and including app navigation information on the menu bolsters the effect of strong feedback.

The conceptual model is defined as how the user understands a system to work and is important for giving the user a sense of control. For our targeted audience, all of them should know what a thermostat is and how to work with it, e.g. being able to turn up the heat. Because of this previous experience, there is a conceptual model in place for their physical thermostat that our design had to be tailored towards. For those who have experience with smart phones or any touch screen device, there is also the conceptual model of interacting with these devices without any physical feedback, e.g. pushing a button. For the Master Thermostat, the design of the menu and how it displays both the current ambient temperature and desired temperature alongside the two software buttons that control the desired temperature offers an easily understandable conceptual model on how to increase the temperature of the home. Additionally, this feature also showcases our design for having strong feedback and best-mapping practices as the controls of the temperature are directly found on the same screen as where the desired temperature is displayed. Finally, the small details such as using intuitive icons and the color of them can strongly impact the user experience. For example, the icon used to turn up the heat is a red-orange up arrow while the icon used to turn down the heat is a blue down arrow juxtaposing the difference between hot and cold.

Web Application

The web application is the face of the VentNet. We want users to navigate the webpage with ease and convey information to them as clearly and concisely as possible. To achieve this, we made careful design choices to decrease the clutter in the UI of our application.

First, we kept all our content on one page to keep navigation simple. The statistical data handled by the web server is kept all on one side, while the scheduler and mode toggles are kept on the other side. It is segregated so that the user does not have to search for the information they need, but will be in the same place every time. When the user needs to check some statistics, or make custom changes, they only need to focus on one half of the page at a time.

FIGURE 13. INPUT FIELD FOR INPUTTING A NEW TEMPERATURE

For each target temperature placeholder, we have beneath it an input field for updating the target temperature. The size of the input field box is limited as a natural constraint for the number of characters allowed for a new temperature. The text field is mapped in such a way so that the user's eyes would not have to search across the screen. Next to the input field, we have mapped the submission button. Upon pressing the button, a confirmation will appear so the user can review their changes. This adds some security and helps lower the risk of user error.

FIGURE 14. LARGE BRIGHT TOGGLE BUTTON, BEFORE AND AFTER CLICKING

Since the web page is so simple, borders around text give users the idea that they are interactable objects, making their discoverability high. Also, the buttons on our web application are designed to change colours when hovered over to signify that they can be pressed. The toggle buttons specifically have a white bar on their side as a signifier to indicate that they have the affordance of moving back and forth like a switch.

Engineering Standards

While this iteration is only a proof-of-concept and getting our design to work takes the highest priority, adhering to industry standards early on will save us from headaches when we want to push for a marketable product. Below is a list of standards that could pertain to the VentNet.

TABLE 10: LIST OF POTENTIAL STANDARDS RELATED TO VENTNET

ISO 9241-151	Guidance on World Wide Web user interfaces
ISO 9241-171	Guidance on software accessibility
ISO 9241-210	Human-centred design for interactive systems
ISO 11581	Icon symbols and functions
ISO 14915	Software ergonomics for multimedia user interfaces
ISO 20282	Ease of operation of everyday products

Analytical Usability Testing

Analytical usability testing is a verification test for the user interface of a system and is performed by the developers. For the VentNet system, the analytical usability testing was performed in increments where it began with individual inspection of the modules and then the system as a whole. With individual observations without any communication in-between, honest and unbiased opinions could be shared during discussions leading to effective meetings and validation of certain design choices. Additionally, the independent testing eliminates the risk of missing critical design flaws as the test was repeated for each developer. The strength of this test is that it is easy to do and requires little preparation time, unlike empirical testing. As each developer is involved at each stage of development, constant feedback is provided such that each feature continuously iterates until the end of the development cycle. Additionally, it is inexpensive in both time and cost as instead of spending time and planning public user testing, like in empirical usability tests, the developers are capable of providing input at all times resulting in lower costs and more time developing the system. The last advantage of this type of testing is that all the developers have stronger background knowledge of the system compared to the targeted users. As such, the developers will provide more effective criticisms and have an easier time identifying problems.

For this proof-of-concept stage, a larger focus has been placed on the core functionality of the system rather than the user interface and user experience. As such, Aeolus Systems targets the following development cycle (prototype and production stage) to improve the user interface and experience. We plan to incorporate tests involving the performance of the application such as boot time and latency, tests involving the application and menu structure, tests involving visibility and display contrast, and more.

Empirical Usability Testing

With the current proof of concept demo requiring in-depth knowledge on the VentNet system to operate, we have not done much user testing of the UI yet. However, development in the next iteration will see the VentNet system be designed for consumer use. The user tests we plan to run on the VentNet system will focus on the learnability of functions and controls, efficiency of UI layout, and user errors.

As the VentNet is a replacement for a regular thermostat system we anticipate the average person should be able to pick-up quickly on the operating schemes. To verify this, we will have average people with no technical knowledge or training on the VentNet attempt to set temperature in multiple zones. From the high discoverability of modern digital furnace controls, we realize this is a core responsibility of a thermostat and should require almost no familiarity with our product. Another test case with no fore knowledge will be to set the temperature, and make a schedule on the webapp relying only on the user to make use of the clear affordances on the webapp interface to guide them. By using test cases where users are minimally prepared we will gauge the ease of use and learnability of our UI, ensuring it can offer a seamless transition in ease of user interaction from the old thermostat.

As with any replacement of a dumb device with a smart system, complexity increases and increases the risk of user error. Seeking to minimize this outcome, we will design tests for minimally prepared users where a sequence of many tasks needs to be completed on our system and gauge the accuracy of the outcome. For example, a test would involve pairing, setting the temperature in multiple zones, making schedules for multiple zones, then recording some temperature data. If the user can successfully complete the task without getting lost when navigating the UI, we will consider our design a success.

More test cases and methods of user testing may be developed in the future to combat UI design deficiencies but for the moment we anticipate using user testing to cover these usability aspects. Analytical testing will be used to cover testing of encountering software bugs and entering stuck states. We currently do not specifically test for user satisfaction since the next product iteration will use a simple but rough UI and 'polishing' of the UI will likely occur in the third product development cycle.

Conclusion

The user interface design appendix gives an overview of our design choices that we made based on our audience. We took careful consideration of our users in designing the UI of the VentNet. Since our system could find its place in many homes, we tried to cater to users who are less tech-savvy, while also leaving room for more advanced users to make customizations. The physical modules mainly consisting of blocks, with a power switch, a button, and a set of LEDs, which are almost as simple as modern thermostats. Using common conventions for colors, like red for low battery, blue for searching for a device, the bright LEDs make the setup process as painless as possible, while also finding use as status indicators for each module. Furthermore, the web application was designed to be simplistic and easy to navigate.

We mainly did in-house testing for our proof of concept version. For our next development phase, we plan to do more empirical usability testing to polish our design. Custom enclosures for modules will also be made in the next development phase. Although the proof of concept phase will not involve much interaction with potential real users, we now have a clear idea of what we are capable of and will plan accordingly for development during the prototype phase.

References

[1] "H bridge", *En.wikipedia.org*, 2017. [Online]. Available: https://en.wikipedia.org/wiki/H_bridge. [Accessed: 30- Mar- 2017].

[2] "Arduino - PWM", *Arduino.cc*, 2017. [Online]. Available: https://www.arduino.cc/en/Tutorial/PWM. [Accessed: 30- Mar- 2017].

[3] "Serial Peripheral Interface (SPI) - learn.sparkfun.com", *Learn.sparkfun.com*, 2017. [Online]. Available: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi. [Accessed: 18- Feb- 2017].

[4] "Feather 32u4 ", *Cdn-shop.adafruit.com*, 2017. [Online]. Available: https://cdn-shop.adafruit.com/970x728/3078-00.jpg. [Accessed: 30- Mar- 2017].

[5] "DRV8838", *A.pololu-files.com*, 2017. [Online]. Available: https://a.pololu-files.com/picture/0J5751.600x480.jpg?99af3fba112cbef297645d73c7b3d8b4. [Accessed: 30- Mar- 2017].

[6] "DRV8838", *A.pololu-files.com*, 2017. [Online]. Available: https://a.pololu-files.com/picture/0J5753.600x480.jpg?50c496833e2581e265001cafd9085ba4. [Accessed: 30- Mar- 2017].

[7] "DS18B20", *Cdn.sparkfun.com*, 2017. [Online]. Available: https://cdn.sparkfun.com//assets/parts/1/8/7/00245-1.jpg. [Accessed: 30- Mar- 2017].

[8] "DS18B20", *Yourduino.com*, 2017. [Online]. Available: http://www.yourduino.com/sunshop/images/products/large_134_DS18B20-1.jpg. [Accessed: 30- Mar- 2017].

[9] "Raspberry Pi 3", 2017. [Online]. Available: • https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Accessed: 31- Mar- 2017].

[10] "Raspberry Pi GPIO Pin Layout with Pi", openclipart, 2017. [Online]. Available: https://openclipart.org/detail/264608/raspberry-pi-gpio-pin-layout-with-pi. [Accessed: 31- Mar- 2017].

[11] "24V AC Solid State Relay Board", makeatronics, 2013. [Online]. Available: http://makeatronics.blogspot.ca/2013/06/24v-ac-solid-state-relay-board.html. [Accessed: 31- Mar- 2017].

[12] "Kuman 3.5 Inch TFT LCD Display 480x320 RGB Pixels Touch Screen Monitor for Raspberry Pi 3 2 Model B B+ A+ A Module SPI Interface with Touch Pen SC06", Kumantech.com, 2017. [Online]. Available: http://kumantech.com/kuman-35-inch-tft-lcd-display-480x320-rgb-pixels-touch-screen-monitor-for-raspberry-pi-3-2-model-b-b-a-a-module-spi-interface-with-touch-pen-sc06_p0014.html. [Accessed: 31- Mar- 2017].

[13] "Adafruit Blue&White 16x2 LCD+Keypad Kit for Raspberry Pi ID: 1115 - $19.95 : Adafruit Industries, Unique & fun DIY electronics and kits", Adafruit.com, 2017. [Online]. Available: https://www.adafruit.com/product/1115. [Accessed: 31- Mar- 2017].

[14] "Raspberry Pi 2 Model B", *Raspberry Pi*, 2017. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/. [Accessed: 29- Mar- 2017]

[15] "Foreword — Flask Documentation (0.12)", *Flask.pocoo.org*, 2017. [Online]. Available: http://flask.pocoo.org/docs/0.12/foreword/. [Accessed: 29- Mar- 2017].

[16] "Template Designer Documentation — Jinja2 Documentation (2.9)", *Jinja.pocoo.org*, 2017. [Online]. Available: http://jinja.pocoo.org/docs/2.9/templates/. [Accessed: 29- Mar- 2017].

[17] HopeRF, "RFM69HCW ISM TRANSCEIVER MODULE DATASHEET v1.1 ," HopeRF. [Online]. Available: http://www.hoperf.com/upload/rf/RFM69HCW-V1.1.pdf. [Accessed: 28-Mar-2017].

[18] F. Rusu, "LowPowerLab/RFM69," GitHub, 08-Feb-2017. [Online]. Available: https://github.com/LowPowerLab/RFM69. [Accessed: 30-Mar-2017].

[19] "Standards", Iso.org, 2017. [Online]. Available: https://www.iso.org/standards.html. [Accessed: 30- Mar- 2017].