6-9-2015

Smart thermostat

Matt Allen
Santa Clara University

Sam Billett
Santa Clara University

Kevin Read
Santa Clara University

SANTA CLARA UNIVERSITY

Department of Electrical Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

Matt Allen, Sam Billett, Kevin Read

ENTITLED

SMART THERMOSTAT

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE
IN
ELECTRICAL ENGINEERING

[illegible] 6/9/2015

Thesis Advisor date

[illegible] 6/10/15

Department Chair date

SMART THERMOSTAT

By

Matt Allen, Sam Billett, Kevin Read

SENIOR DESIGN PROJECT REPORT

Submitted to
the Department of Electrical Engineering

of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements
for the degree of
Bachelor of Science in Electrical Engineering

Santa Clara, California

2015

Smart Thermostat

Matthew Allen, Sam Billett, Kevin Read

Department of Electrical Engineering
Santa Clara University
2015

ABSTRACT

The current thermostat marketplace is dominated by programmable thermostats that are engineered to give the user as much control as possible. However, not all users program these thermostats optimally. Some prefer comfort over energy efficiency and will heat an empty house. Others do not care to program their thermostat and leave it at a set temperature. The Smart Thermostat that we have created is engineered to optimize the temperature for the user, using the common sense of the engineer to lower energy consumption. It uses a Raspberry Pi as a platform to implement a fuzzy logic control System. By removing the extensive control of the thermostat from the user we are able to create a more energy efficient product that also maintains the comfort level of the occupants.

Keywords: Fuzzy Logic Control, Thermostat, Energy Efficiency

Acknowledgments

The authors would like to thank the Santa Clara School of Engineering for its support in providing all equipment used in development and testing as well as funding to support our project. We would also like to help all faculty and peers who assisted us throughout the project.

They would especially like to thank Dr. Maryam Khanbaghi for all of her guidance and support with fuzzy logic and throughout the project.

Table of Contents

1.0 Introduction

1.1 Sustainability

Thermostats play an important role in the home: managing the temperature through control of the HVAC system. Efficient management of the temperature involves a balance between user comfort and energy savings, and is important for three reasons. First, conserving energy saves the user money. This is particularly important when considering the high cost of utility bills and the impact that a small change in temperature can have on energy use. For California homes, a 1.3°C (2.3°F) change in indoor temperature can decrease the energy use by about 45% (Meier 9). Second, conserving energy has a positive environmental impact by decreasing greenhouse gas emissions. Third, conserving energy lifts the strain on the power grid. Every year, energy service providers struggle to meet the growing demand during peak use periods. The prevalence of solar panels in California, while allowing electricity to be returned to the grid, has caused a phenomenon known as the duck curve. As can be seen in Figure 1 below, there is a steep increase in demand around dusk. This is the transition from low demand and high supplemental solar generation to high demand and low solar generation. With more solar panels being installed each year, the transition region is getting steeper, which puts stress on the grid. Products that can reduce the energy demand during peak periods will thus be invaluable to energy service providers. The Smart Thermostat aims to alleviate some of this stress on the grid by decreasing overall energy use spent on HVAC systems. Through the use of a modern control system, the Smart Thermostat balances comfort and efficiency better than a programmable thermostat.

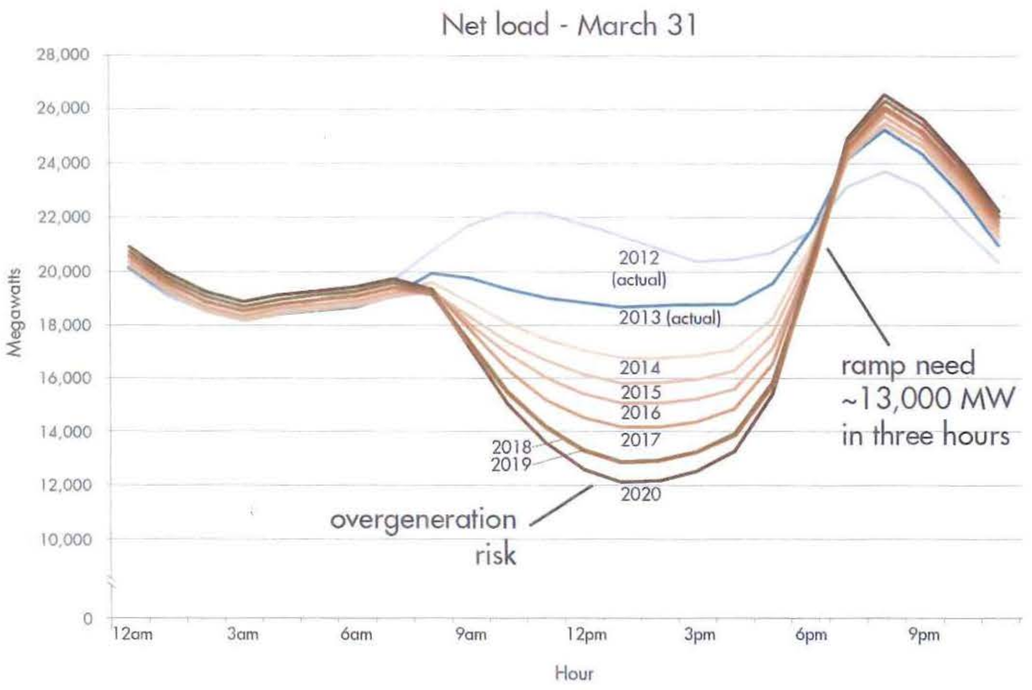


Figure 1. Showing the energy use throughout an average day for different years, including past data and projections for future data. https://www.caiso.com/Documents/FlexibleResourcesHelpRenewables_FastFacts.pdf

1.2 Issues with Traditional Thermostats

Given the importance of correct thermostat use, it may come as a surprise that many homeowners do not program their thermostats or use them in an efficient way. A number of studies have shown that people are lazy or intimidated when it comes to programming their thermostat. Of the recent buyers of HVAC equipment, the American Home Comfort Study (AHCS) reported that 56% of homeowners always program their thermostats, 32% sometimes program, 9% never program their thermostats, and 3% do not know how (Peffer 2534). Studies have also revealed that users commonly use the thermostat incorrectly. This includes setting the temperature much higher than comfortable, due to the belief that the house will heat up faster. Beyond the results of studies, there are also inherent flaws with the programmable thermostat system. For example, people's schedules change, which means there will be times when an empty house is heated or cooled, and times when the user is home and the system is not running.

1.3 Review of Smart Thermostats

The standard for new thermostats is moving towards intelligent control. These systems include more than a set point or a programmable component, but rather advanced controls such as occupancy sensing, control through mobile or computer apps, geolocation, historical learning algorithms, and more. While many products are offered today, several smart thermostats and common features stand out. One of the most recognizable smart thermostats is the Nest. The Nest includes features like auto-scheduling based on the user's historically set schedule, auto-away which utilizes occupancy sensing, wireless control through Wi-Fi and app integration, and a colorful and fashionable display which helps users interact and see how set point changes lower energy use. The Nest was one of the first thermostats that opened up the "smart thermostat" market by being user friendly, providing advanced controls, and being fashionable to put on a wall.

Other major smart thermostat products on the market include the Lyric by Honeywell and the Ecobee3 by Ecobee. While both of these products offer a typical historical scheduling learning algorithm, they also include other features that help them stand out. The Ecobee3 has a feature that allows implementation of multiple temperature sensors throughout the house. This is useful for larger houses and more advanced micro control of a houses environment. The Lyric uses a feature called geofencing that utilizes a GPS signal to recognize when a user's phone enters the home's proximity, turning on the HVAC system before the user even gets home.

2.0 Design Considerations

2.1 Solution

Given the flaws and deficiencies associated with programmable thermostats, studies which have shed light on inappropriate thermostat-user interaction, and the importance of efficiently managing house temperature, we set forth a goal to create an intelligent thermostat to address each of these criteria.

2.2 System Level Requirements

At its core, a thermostat consists of four systems: sensors, actuators, control logic, and user interface. An intelligent thermostat simply expands functionality within these four categories.

Sensors: the most important sensor is the temperature sensor, which must be located within the environment under control. Supplementary sensors, such as humidity, light, and motion have already been utilized by programmable thermostats. However, the effectiveness of these extra sensors has still depended on the user to program the thermostat and take advantage of them.

Actuators: Either mechanical or electrical, the actuators within the thermostat interact with the HVAC system to control the heating, cooling, and fans.

Control Logic: Simple programmable thermostats use a feedback loop that compares the current temperature to the setpoint temperature, and heats or cools as needed. Within this control are other mandatory features, such as hysteresis and overshoot prevention. Hysteresis of $\mp 1°F$ around target temperature prevents the HVAC system from rapidly cycling, while overshoot prevention stops the furnace early to account for residual heating.

User interface: All thermostats include displays for relevant information and buttons for user control. At a minimum, the display must show the current temperature and the setpoint temperature. The minimum requirement for buttons is two: one to increase temperature and one to decrease.

2.3 Customer Needs

With many studies revealing improper use of thermostats, there has been a surge of research responding to this issue. These papers have pointed out key issues with old designs and suggestions for new ones. These suggestions are useful for identifying features to include in our design, as well as benchmarks to evaluate it.

In his book *Addicted to Energy: A Venture Capitalist's Perspective on How to Save our Economy and our Climate*, author Elton Sherwin set forth 12 characteristics needed in the thermostat of the future:

1. No programming requirements. They will just have an up arrow and a down arrow to change temperature. Smart thermostats will learn what you like.
2. Half degree increment adjustability.
3. An occupancy sensor to detect when a room is empty.
4. An infrared sensor to detect the temperature of the whole room.
5. A light sensor to detect when a room is dark.
6. A humidity sensor, and perhaps even a CO2 sensor.
7. A radio receiver for a remote thermometer to detect temperature and motion in another room.
8. Internet connection to get the weather forecast and various other pieces of useful data from the electric utility.
9. PC connectivity to send detailed usage data to building owners.
10. A remote emergency off switch to enable electric utilities to turn off air conditioners in a crisis.
11. Twenty-four months of memory. Long-term memory in thermostats is helpful for finding problems in home heating and air-conditioning systems.
12. Superior comfort. They will learn the rhythms of a house and have fewer temperature swings.

2.4 Objectives

Considering these customer needs and system level requirements, we have two primary objectives, one design related and one device related. First, to use a modern control system capable of intelligently managing the temperature setpoint without requiring the user to program their schedule. "Intelligently manage" means the design will rely on a variety of sensors to detect house occupancy, much like a regular programmable thermostat. However, we will explore the potential benefits of the lesser-used CO_2 sensor. Second, the hardware system will be a proof of concept, serving as a platform to test out the control system and gauge its effectiveness. There are aspects of a thermostat which will be ignored in our design, including size, power issues, and specialized controls for sophisticated HVAC systems. All of these issues are crucial factors for creating a product, but will not be addressed within the scope of our project.

3.0 System Overview

The Smart Thermostat is comprised of several sensors, a control algorithm, and a platform to integrate these two things. We are using a Raspberry Pi to read the information off of our sensors and run a fuzzy logic control system with that information. The output of the control system is then sent to a set of relays that change the HVAC system as the control system dictates.

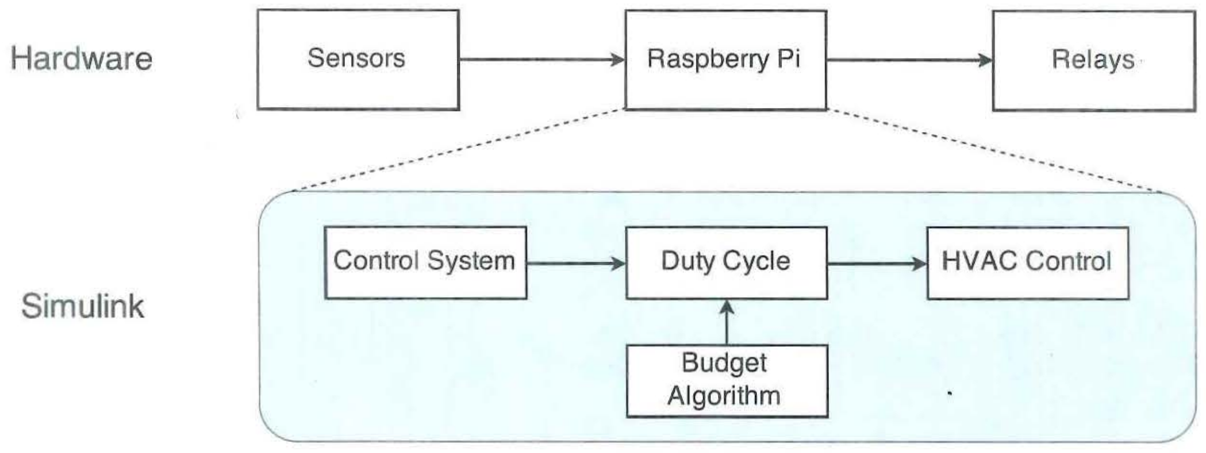


Figure 2. A Block Diagram representing our system. The Raspberry Pi is running a control system that incorporates information from the sensors and creates the HVAC control signals.

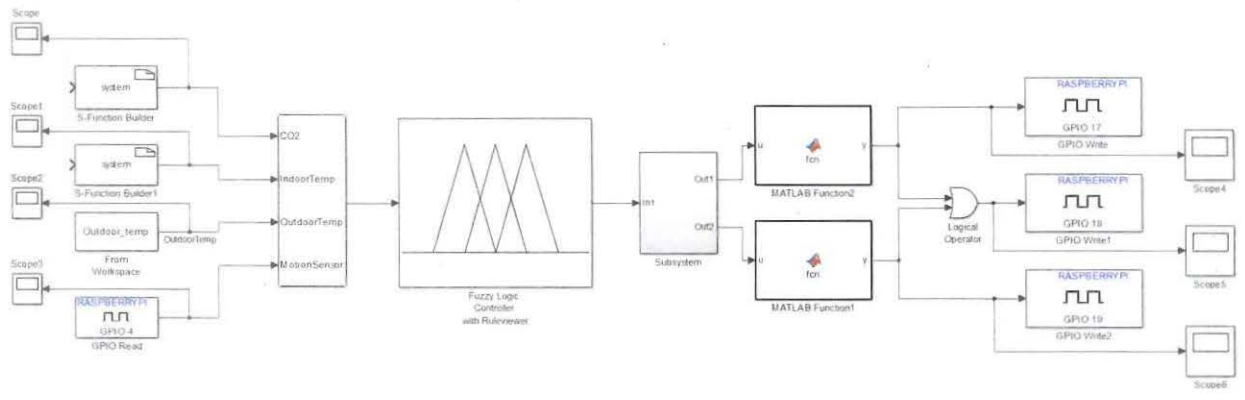


Figure 3. An overview of the whole system as laid out in Simulink. The inputs on the left are read from the Raspberry Pi and the outputs on the right are written to the Raspberry Pi. The fuzzy logic controller is represented in the middle.

3.1 Occupancy Detection

The Smart Thermostat will use a Carbon Dioxide sensor to detect the occupancy of the environment that the thermostat is controlling. According to previous research, when a human enters a room, the Carbon Dioxide level increases significantly, rising to about 600 ppm if one person is in the room and even higher with multiple people. This Carbon Dioxide level will drop once the room becomes empty. While the Carbon Dioxide level is high the thermostat will control the room as if it is occupied and while the level is low, representing an empty room, the thermostat will control the room as if it is empty. Beyond the Carbon Dioxide sensor the thermostat also implements a motion sensor as a failsafe. If the Carbon Dioxide sensor reads a value that matches the 'empty' threshold but the motion sensor detects movement then the thermostat will control the room as if occupied despite the Carbon Dioxide level.

4.0 Fuzzy Logic Control

The control system is built with fuzzy logic. This is a type of control system that uses rules to define the output in any situation. These rules are created with the common sense of the engineer and are combined through the fuzzy inference system of Matlab and imported directly to Simulink. The rules are not completely granular, however. They are given different weights for different inputs and the output is a weighted average of the individual outputs of each rule. This output is calculated using the centroid method, shown below in Figure 4.

$$C_m = \frac{\sum_{i=1}^{m} \mu_z(x_i) x_i}{\sum_{i=1}^{m} \mu_z(x_i)}$$

Figure 4. Centroid method for fuzzy logic control.

A fuzzy logic control system can be created in Matlab using the fuzzy logic toolbox. This toolbox takes care of the coding for the engineer, allowing the engineer to focus on the design itself.

The toolbox permits the engineer to easily manipulate the membership functions for the inputs and outputs, and create the rules that govern the system as a whole.

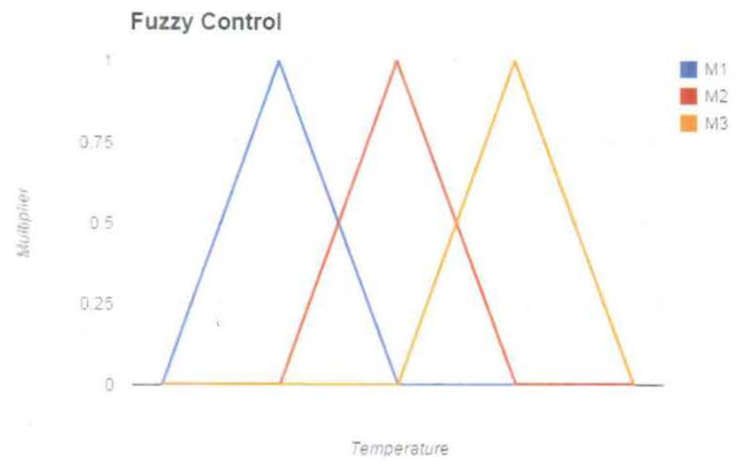
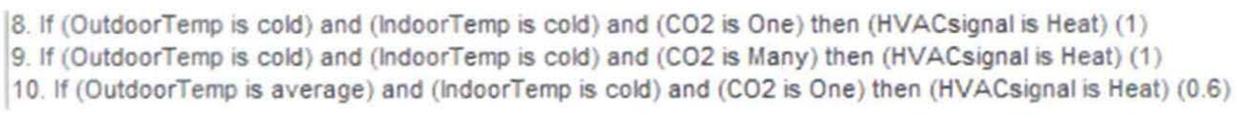


Figure 5. A representation of membership functions. The x axis is the input temperature and the y axis is the weight given to the membership function in the fuzzy inference system. The different triangles represent different membership functions, the above might represent cold, average, and hot.

The Smart Thermostat is built using a fuzzy logic control system that incorporates the indoor temperature, outdoor temperature, and occupation of a room. Each separate input is classified into membership functions that define whether for example a temperature is considered hot, average, or cold. Rules are then created that define what the controller should do in each situation, i.e. if the room is hot then turn on the AC. These rules use IF AND THEN logic.



```
8. If (OutdoorTemp is cold) and (IndoorTemp is cold) and (CO2 is One) then (HVACsignal is Heat) (1)
9. If (OutdoorTemp is cold) and (IndoorTemp is cold) and (CO2 is Many) then (HVACsignal is Heat) (1)
10. If (OutdoorTemp is average) and (IndoorTemp is cold) and (CO2 is One) then (HVACsignal is Heat) (0.6)
```

Figure 6. An example of the rules that govern a fuzzy logic control system and the IF AND THEN logic used.

5.0 Hardware

Part	Notes
Raspberry Pi	- Credit-card sized computer - 900MHz quad-core ARM Cortex-A7 CPU - 1 GB RAM
Temperature Sensor	- One for indoor temperature, one for outdoor - Accuracy to 1.0°F - Small size, low cost
CO_2 Sensor	- Can detect levels above 400 ppm - Bought with a chip because requires 7V
Motion Sensor	- On or off, 3.3V or 0V respectively - Range of up to 7m
Solid State Relay	- Can handle 24VAC and up to 8A of current
Magnetic Relay	- Older systems use line voltage of 120V or 240V to power thermostat - This relay can handle up to 30A of current at 240V

6.0 Simulink

6.1 Raspberry Pi Setup for Simulink

Before running simulink models on the Raspberry Pi, software must be downloaded to both the computer and the Micro SD card. The tutorial for doing so was accessed from the MathWorks site (http://www.mathworks.com/hardware-support/raspberry-pi-simulink.html).

The three key items to download

- *Mathworks Software*: Matlab and Simulink Release 2013a or later (2015a was used for this project).
- *Compiler (for Mathworks Software)*: The Simulink model must be converted into an executable file in order to run on the Pi. Microsoft Visual Studio 2013 was used for this project, the professional version of which was downloaded for free through Microsoft DreamSpark (https://www.dreamspark.com). In order to check which compiler is being used, type the command >>mex -setup
- *Simulink Support Package for Raspberry Pi Hardware*: Within Matlab, go to Add-ons -> get hardware support packages -> install from internet -> Raspberry Pi. This will launches an install wizard which downloaded necessary 3rd party software as well as a Raspberry Wheezy firmware image onto the SD card.

6.2 Simulink Model

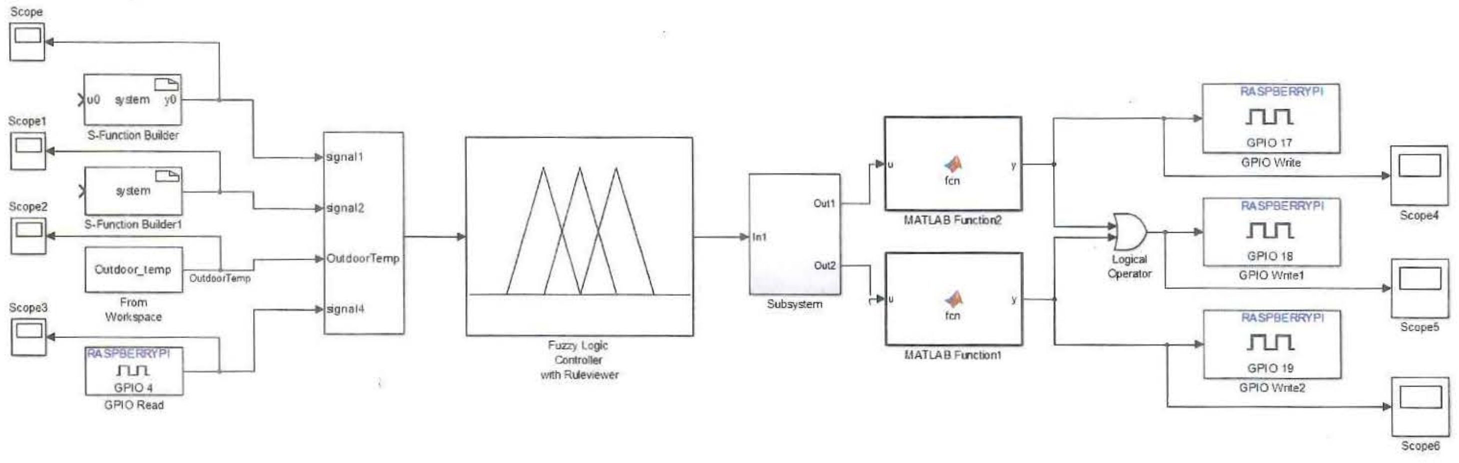


Figure 7. An overview of the whole system as laid out in Simulink. The inputs on the left are read from the Raspberry Pi and the outputs on the right are written to the Raspberry Pi. The fuzzy logic controller is represented in the middle.

1. Input blocks: There are four input blocks, one for each sensor. The input block labeled "GPIO Read" comes from the Simulink support package for raspberry pi. The pin is specified in the block options, and is able to read the digital output of the motion sensor. Two custom s-function blocks were needed to communicate with the CO_2 sensor and the temperature sensor. The GPIO blocks could not be used because the temperature sensor needed the pin to be pulled down in order to output the temperature data.
2. Mux: The mux allows four output lines to be connected to the input of the fuzzy logic controller.
3. Fuzzy logic controller: This block encapsulates the fuzzy logic system discussed earlier.
4. Duty cycle bucketization: This block subsystem contains a set of switches to prevent rapidly pulsing the compressor. If a value is too close to 0, it will turn on the furnace for very short time, so a 0 is passed through. If a value is too close to 1, it will keep the furnace on and then turn it off for a very short time before turning it on again. I 1 is then passed instead to keep the furnace on for the whole cycle. The same logic is applied for the compressor.
5. Custom Matlab functions: These functions take a value from the previous block and turn it into a signal. A specific length of time is hardcoded into the function, which is the

period of the duty cycle signal. The block outputs a high for a length of time specified by the fraction of the period, then outputs a low for the remaining fraction of the period.

6. Write blocks: The GPIO write blocks are also part of the support package library. They allow a digital signal to be output on the specified pin.

6.3 Simulink Results

HVAC output as a function of outdoor temperature and indoor temperature

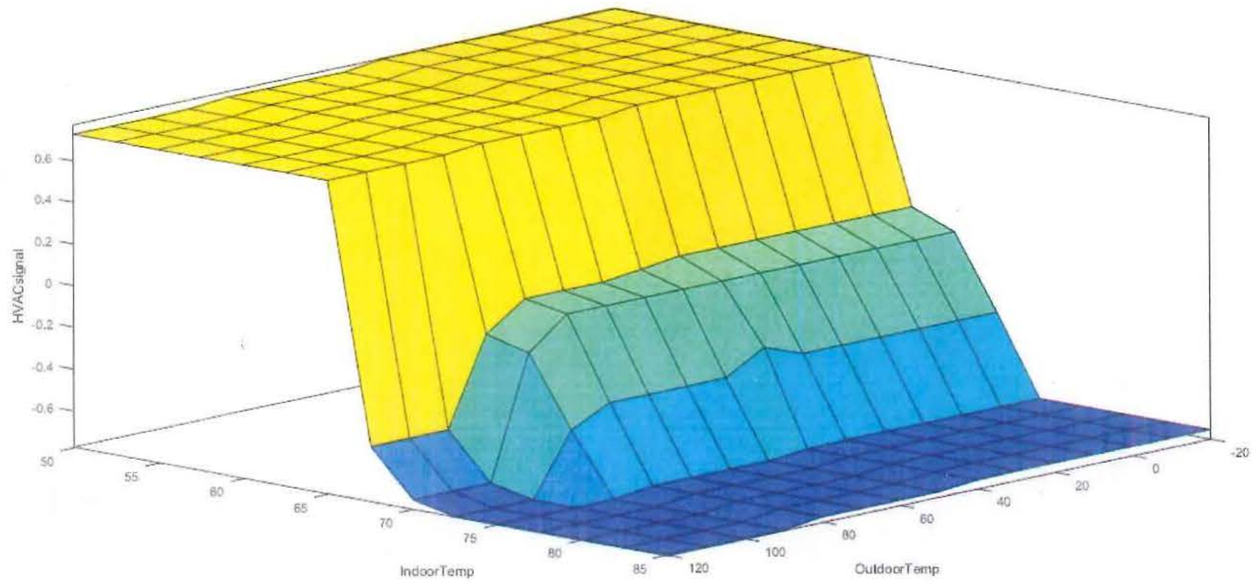


Figure 8. Output of the fuzzy logic control as a function of the indoor temperature and outdoor temperature. Yellow represents heating and blue represents AC while the light blue/green color represents the system remaining off.

HVAC output as a function of indoor temperature and CO_2

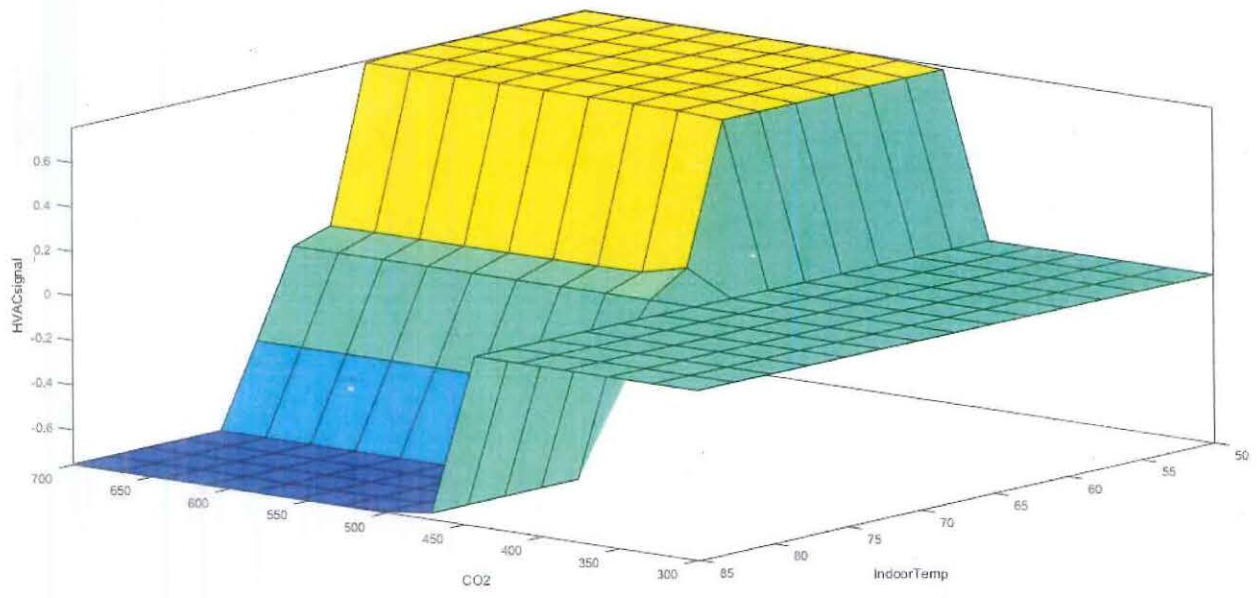


Figure 9. Output of the fuzzy logic control as a function of the indoor temperature and the CO_2 level. Yellow represents heating and blue represents AC while the light blue/green color represents the system remaining off.

7.0 Testing and Analysis

7.1 Temperature Testing

To test the hardware, we put the device in simulated environments and measure how the various sensors react. We graph the output of the temperature sensor in a room temperature environment, a heated environment, and a very cold environment. To test a hot temperature, we place the sensor in our hands. To test the cold temperature, we place the entire device in the freezer and record the temperature as it drops in response.

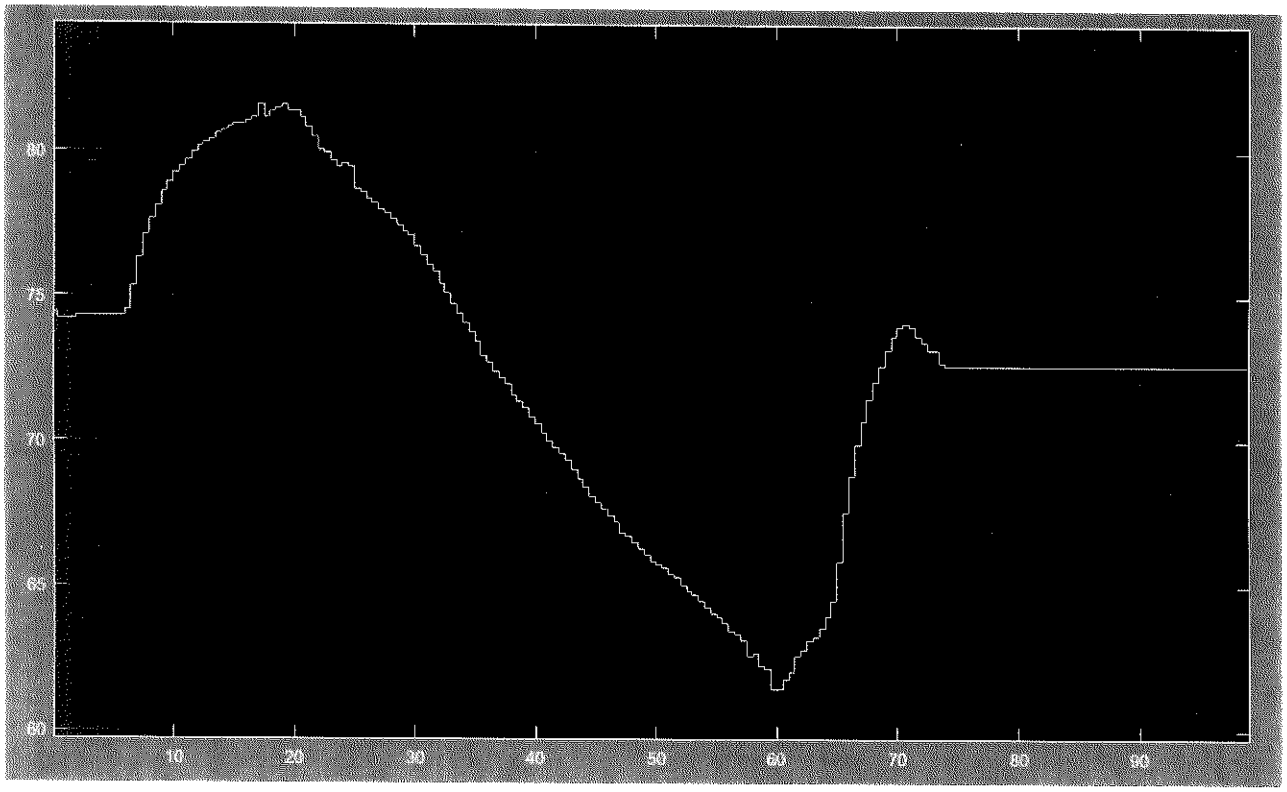


Figure 10. A graph of the temperature (degrees Fahrenheit) measured by our temperature sensor over time (seconds).

7.2 CO_2Testing

To test the CO_2 sensor, we first allow the sensor to warm up. The sensor must be at a certain temperature before it becomes accurate, so it must stay on during use. We let it sit for a time in an unoccupied room to get a baseline reading. The atmospheric concentration of CO_2 in Santa Clara is roughly 390 parts per million. With our sensors minimum accuracy being 400 parts per million, our baseline value appears flat. We then move it into a small room our team is working in. Because we had been occupying the room previously, the concentration of CO_2 was already elevated. We allow the CO_2 sensor to peak around 1200 parts per million before removing it from the room. This allows us to view the sensitivity of the sensor and get data for our fuzzy logic simulations.

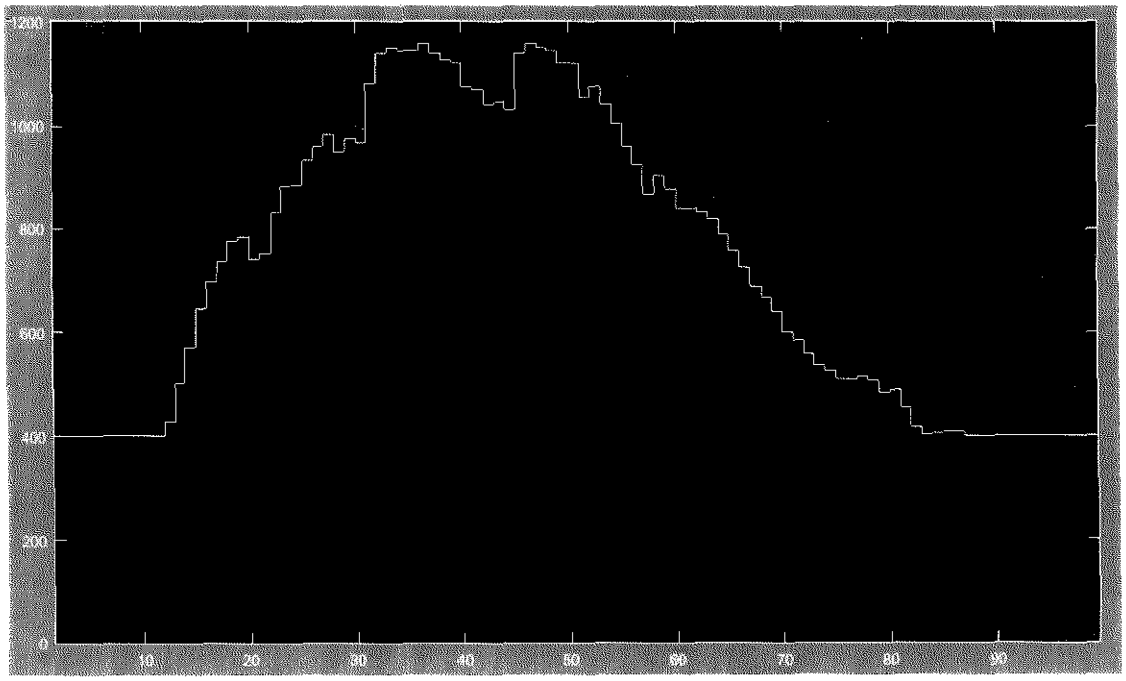


Figure 11. A graph of the CO_2 level (ppm) over time (seconds) as measured by our CO_2 sensor.

7.3 Fuzzy Logic Control Testing

To test the fuzzy logic control system we connected test inputs into the control system in Simulink and observed the output. Because the product is not ready to integrate into a house it is not feasible to test the system with real inputs because the system does not have control over the outputs and so the inputs would be difficult to work with: room temperature inside temperature, roughly flat CO_2 and uncontrollable outdoor temperature. Instead we used the following sets of simulated inputs and observed the outputs.

Outside	85
Inside	72.9
CO_2	480
Output	-0.541

Figure 12. A set of inputs given to the fuzzy logic system and the resulting output.

These inputs simulate an environment with warm indoor and outdoor temperatures and a person present. The output is a duty cycle of the HVAC system. The negative sign represents that this signal will be sent to the AC system.

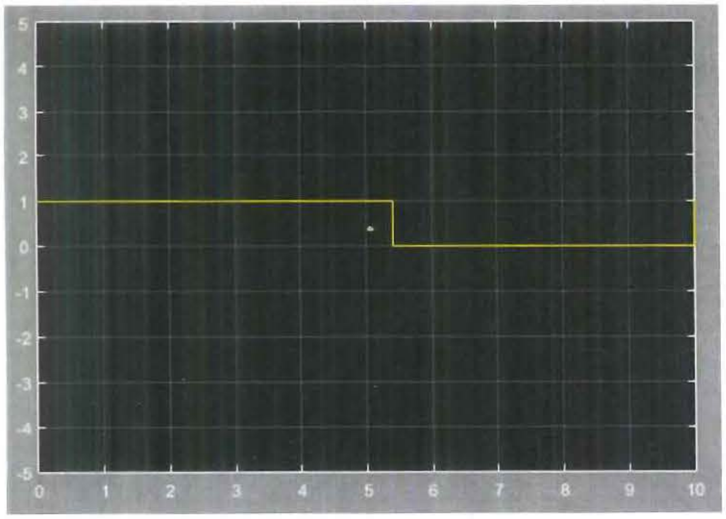


Figure 13. The output of the fuzzy logic control system as measured in Simulink.

This represents the output of the control system. The AC system receives an 'ON' signal for a little over half the period and then an 'OFF' signal for the remaining time.

Outside	62
Inside	65.5
CO_2	525
Output	0.353

Figure 14. A set of inputs given to the fuzzy logic system and the resulting output.

These inputs simulate an environment with cold indoor and outdoor temperatures and a person present. The output is a duty cycle of the HVAC system. Because the output is positive it will be sent to the heating system.

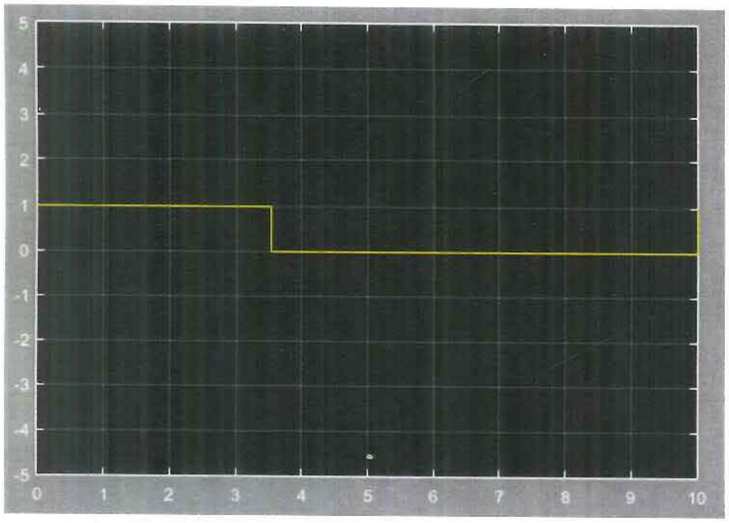


Figure 15. The output of the fuzzy logic control system as measured in Simulink.

This represents the output of the control system. The heating system receives an 'ON' signal for a little over a third of the period and then an 'OFF' signal for the remaining time.

8.0 Evaluation of Design

Having completed our project, we now look back to evaluate the final product against the goals we set for ourselves. There were many constraints that governed the work we did and many problems that forced us to use creative solutions. There were also aspects of the project that did not meet our expectations. The following is a discussion of these creative solutions and concerning aspects.

8.1 Creative Solutions

First and foremost, we were able to create a thermostat that successfully controls the temperature of an environment to a comfortable level. This in and of itself was a more difficult task than we expected as it requires a deep understanding of the underlying systems and interconnections. We overcame several obstacles on the way to success, creating solutions that elegantly solved the problems we faced.

The main problem that we dealt with is the separation in style between the fuzzy logic system output and the input required for the HVAC system. The fuzzy logic system produces a value on a spectrum, known as an analog signal. Conversely, the HVAC system requires a signal that is either 'ON' or 'OFF,' known as a digital signal. In order to bridge this gap in style, we decided to use the analog output of the fuzzy control system as a duty cycle for the input of the HVAC system. Put simply, this means that the fuzzy control system controls what percent of the time in a given period the HVAC system is 'ON' and what percent of time it is 'OFF.' In this way we were able to retain the subtlety afforded us by the fuzzy control system while still operating within the framework required by the HVAC system.

This use of the fuzzy logic system output as a duty cycle also allows us to preserve the HVAC system that the thermostat is controlling by ensuring that it does not turn on and off frequently. This quick switching results in decreased length of life for the HVAC system. By using cutoff points for the duty cycle, we reject values close to 0 or 1 and replace them with exactly 0 and 1. So, for example, a duty cycle of 0.05 would not result in the heater being on for 5% of the time, instead it would be on for 0% of the time for that period.

The benefits of using fuzzy logic extend well beyond the above solutions. One of the ideas for future work on this product is to make it more customizable for each user. It is a fact that comfort is not static. Certain users may prefer warmer environments while others may prefer cold ones. Incorporating a simple way to change what the thermostat considers 'warm' and 'cold' would be vital to creating a more perfect thermostat. Fortunately, fuzzy logic allows just this thing. The membership functions discussed above are very fluid and can be changed within the flow of normal use, if programmed to do so. This means that a thermostat can learn a user's preferences and cater to them.

8.2 Concerning Aspects

While working on this project we ran into a variety of challenges, obstacles, and concerns. These concerns range from hardware issues to design flaws. Addressing these concerns will aid groups who work on this project in the future by maintaining continuity.

The first concern is that the CO_2 sensor has several undesirable properties including volatility, heat, size, and efficacy. The testing of the CO_2 sensor involves exaggerated situations: having multiple people enter and exit a small room in a short period of time. We are worried about the sensor's resolution for picking up on subtle occupancy changes in a larger home. Thus the CO_2 sensor may limit the effectiveness of the entire project if it does not adequately register important occupancy data. The fuzzy logic control is only as accurate as the data it receives.

There are several obstacles derived from using Simulink on the Raspberry Pi. Although there is a plethora of resources available for integrating the temperature sensor with an Arduino board, there is nothing available for integrating it with Simulink. The task of interfacing the sensor with Simulink is made especially difficult by the sensor's 1-wire protocol.

Another difficulty is quantifying comfort for the user. Energy savings is a straightforward metric of comparison, but thermostats exist first to provide comfort for the user.

9.0 Future Works

There are several potential improvements that can be made to our project in the future. The first is a continuous process of tuning the fuzzy logic system. By adjusting the peaks or making them user-adjustable, the product can have increased user comfort and personalization. Beyond tuning the fuzzy logic system, the algorithm could be ported onto a lower energy and lower cost microcontroller. There are several options for achieving this, including creating a custom PCB that includes the microcontroller and the various sensors on one chip.

Another improvement that we considered but ultimately did not implement is a budget mode. The budget mode allows the user to set a budget guideline for how much they wish to spend on HVAC utilities per month. This is achievable by increasing the acceptable range of temperature that is considered 'comfortable.' This process would require measuring the amount of energy used by the HVAC system as well as using the updated price of energy in each region. This may require some hardware or software additions to measure energy use or to maintain an accurate price for electricity, but we believe a budget mode is a valuable addition to the Smart Thermostat.

While there are advantages to a smart thermostat that does not require an internet connection for basic function, adding this functionality should be considered as integrating control of the thermostat into a mobile phone application has value. This control could also be integrated into a Wifi or Bluetooth connection that is more local to the environment the thermostat controls.

10.0 Ethical Analysis

The fundamental moral reason for our project is to reduce consumption of natural resources. It is widely known that Americans use a disproportionate amount of the world's resources. If everyone on Earth consumed as many resources as Americans, we would need 4.1 Earths to sustain ourselves (Elert). As engineers, it is our ethical responsibility to preserve natural resources so that the global population can equally benefit from them. We must use them in a sustainable way so that future generations do not suffer from our wasteful behavior. The purpose of our thermostat is to efficiently operate the heating and cooling units in homes in order to use less energy. Lower demands for electricity will allow a higher percentage of California's electricity to be generated sustainably, with solar and hydroelectric power plants. As it stands, 45% of California's electricity comes from natural gas, and around 8% comes from coal. (California Energy Commission) We have a finite amount of these resources, and are using them at an alarming rate. In addition, burning natural gas and coal pollutes the environment and contributes to climate change. Our project is thus tied into important environmental ethical issues.

While the IEEE engineering code of ethics provides a useful framework for ethical analysis of our project, it has some limitations. As the article "The Good Engineer: Giving Virtue its Due in Engineering Ethics" by Charles Harris points out, most of the rules are negative, meaning they list things that engineers should not do. Negative rules are useful for preventing engineers from harming the stakeholders, but do not provide a helpful guide for decision making and motivation. When we analyze our project from the standpoint of virtue ethics, we find that there are several key habits which compensate for the purely negative nature of the code of ethics. Two important non-technical excellences mentioned in the Harris paper are analyzed below.

Within the field of philosophy of technology, Harris highlights the theme of technology fragmenting and impersonalizing human experience. He gives an example of how computers have transformed communication between people, saying that it is increasingly abstract and impersonal. We must consider how our device affects the human experience and whether the benefits of the device compensate for the drawbacks. Our device falls into the category of smart appliances, which attempt to simplify and automate processes in the home. This simplification often comes at the cost of the user's understanding of the process and maintenance of the process. We must be careful to specify the limits of our thermostat's "intelligence" so that users do not expect it to do more than it should.

Respect for nature and the environment can be viewed as a virtue and therefore falls into the realm of virtue ethics. As was explained earlier, our device will decrease consumption of natural resources, which is a way to show respect for nature. We must be careful how we market our product, however, since it could be improperly used and give the false impression that it is still eco-friendly.

The nature of our device and the systems it is connected to make safety a paramount concern in our design. First of all, the thermostat controls the furnace and AC systems in a house. Although both of those systems have built-in safety mechanisms, we want the total system to have multiple levels of redundancy and we will therefore have shutoff mechanisms in the thermostat. There are also proper ways the AC unit should be used to maximize its lifetime. For example, the thermostat should not rapidly turn the AC unit on and off in a short period of time.

Thermostats which are wifi connected must use a secure connection to the router or access point. The thermostat occupancy information is in fact sensitive information which burglars could exploit. If a thermostat detects an empty house for a few days, this is a good sign to a burglar that the homeowner will be away from the house for the inactive duration. Although the internet-of-things provides many conveniences, having so many things connected to the internet raises significant privacy issues.

11.0 Bibliography

Elert, Emily. "Daily Infographic: If Everyone Lived Like An American, How Many Earths Would We Need?" *Popular Science*. N.p., n.d. Web. 05 June 2015.

Meier, Alan K.(2008). RESIDENTIAL THERMOSTATS: COMFORT CONTROLS IN CALIFORNIA HOMES. *Lawrence Berkeley National Laboratory*. Lawrence Berkeley National Laboratory: Lawrence Berkeley National Laboratory. Retrieved from: http://escholarship.org/uc/item/2vc7h48t

Sherwin, Elton B. *Addicted to Energy: A Venture Capitalist's Perspective on How to save Our Economy and Our Climate*. United States: Energy House, 2010. Print.

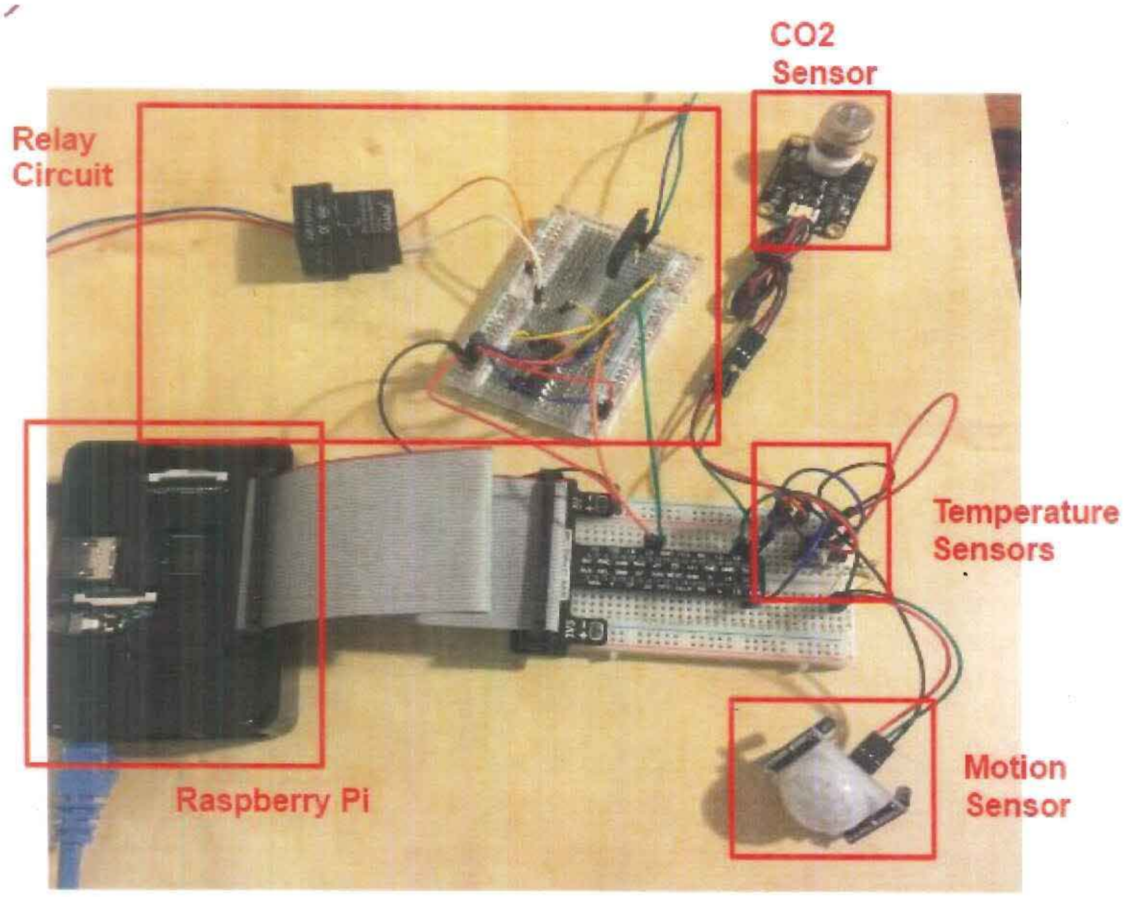
Therese Peffer, Marco Pritoni, Alan Meier, Cecilia Aragon, Daniel Perry, How people use thermostats in homes: A review, Building and Environment, Volume 46, Issue 12, December 2011, Pages 2529-2541, ISSN 0360-1323, http://dx.doi.org/10.1016/j.buildenv.2011.06.002. (http://www.sciencedirect.com/science/article/pii/S0360132311001739)

"Total Electricity System Power." *Total Electricity System Power*. California Energy Commission, n.d. Web. 05 June 2015. http://energyalmanac.ca.gov/electricity/total_system_power.html

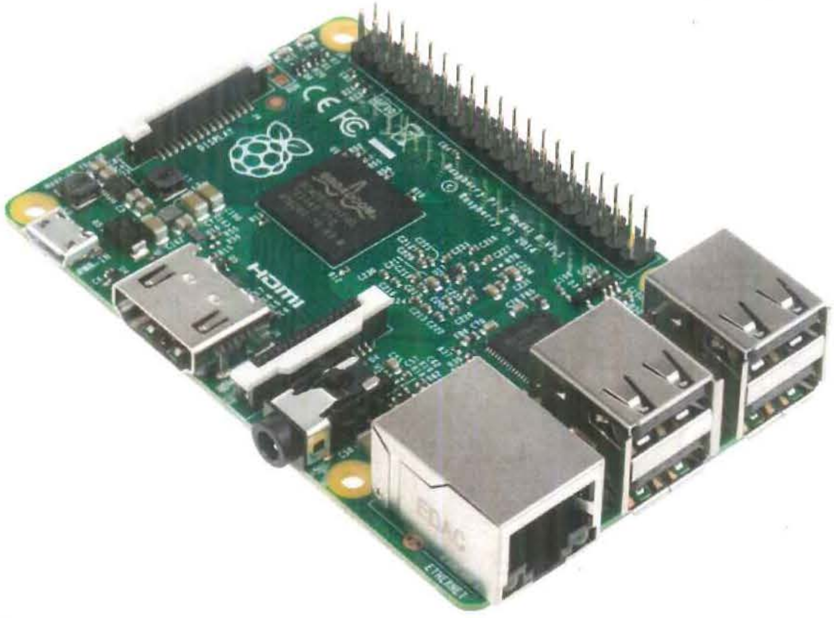
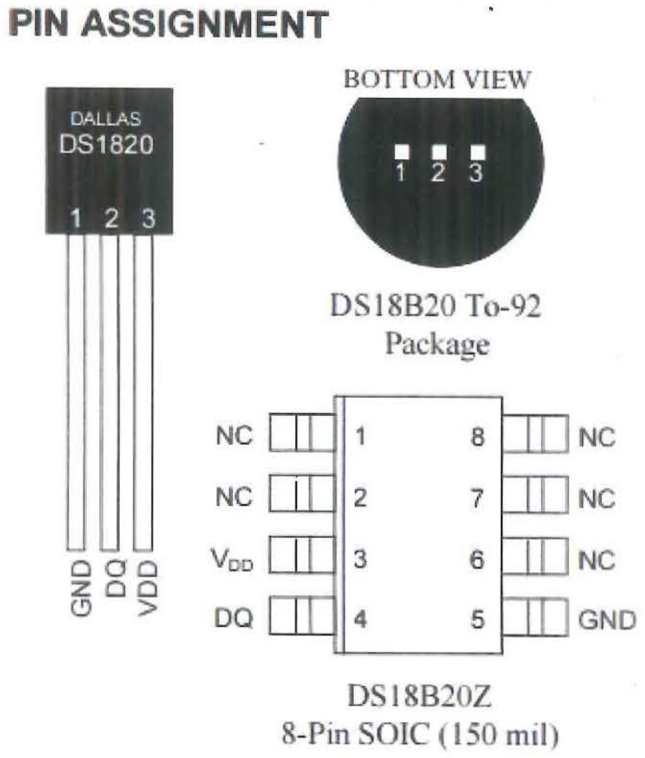
12.0 Appendices

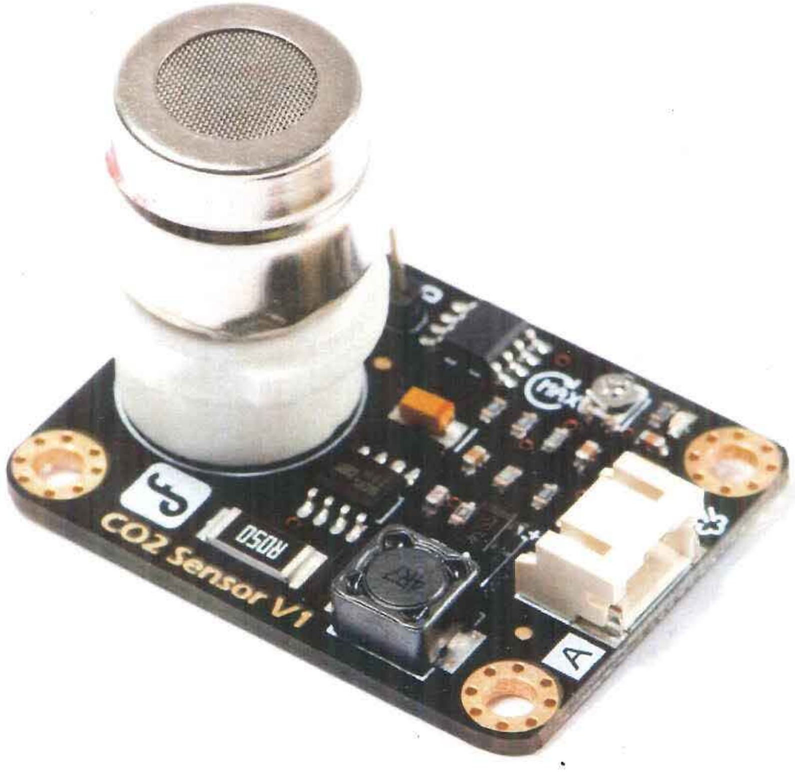
Appendix A: Parts

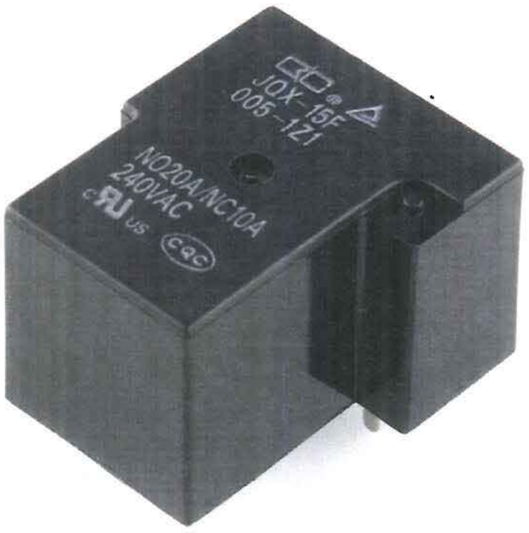
A.1 Complete Setup



A.2 Individual Components

Part	Picture
Raspberry Pi	
Temperature Sensor	 PIN ASSIGNMENT DALLAS DS1820 1 2 3 GND DQ VDD BOTTOM VIEW 1 2 3 DS18B20 To-92 Package NC 1 NC 2 VDD 3 DQ 4 8 NC 7 NC 6 NC 5 GND DS18B20Z 8-Pin SOIC (150 mil)

CO_2 Sensor	
Motion Sensor	

Solid State Relay	
Magnetic Relay	

Appendix B: Bill of Materials

Sensor	Part Tag	Price	Purchase Link
Computer/MCU	Raspberry Pi	$46.99	https://www.raspberrypi.org
Temperature Sensor	DS18B20	$4.25	https://www.sparkfun.com/products/245
CO2 Sensor	MG811	$56.05	http://www.dfrobot.com/
Solid State Relay	S208T02	$4.95	https://www.sparkfun.com/products/10636
Magnetic Relay	JQX-15F	$2.95	https://www.sparkfun.com/products/10924
Motion Sensor	HC-SR501	$0.80	http://www.amazon.com/Pyroelectric-Infrared-Motion-Sensor-Detector/dp/B008AESDSY/ref=sr_1_2?ie=UTF8&qid=1433747777&sr=8-2&keywords=PIR+motion+sensor

Appendix C: Code

C.1 Code for CO_2 Sensor

```
/******************MG-811 Gas Sensor Module V1.1*****************************
Author:  Tiequan Shao: tiequan.shao@sandboxelectronics.com<script cf-hash="f9e31"

License: Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0)

Note:    More sophisticated calibration is required for industrial field application.

                                        Sandbox Electronics    2012-05-31
************************************************************************************/

/***********************Hardware Related Macros************************************/
#define         MG_PIN                  (0)     //define which analog input channel you are going to
use
#define         BOOL_PIN                 (2)
#define         DC_GAIN                  (8.5)   //define the DC gain of amplifier


/**********************Software Related Macros***********************************/
#define         READ_SAMPLE_INTERVAL         (50)    //define how many samples you are going
to take in normal operation
#define         READ_SAMPLE_TIMES          (5)      //define the time interval(in milisecond)
between each samples in
                                        //normal operation

/*********************Application Related Macros*********************************/
//These two values differ from sensor to sensor. user should derermine this value.
#define         ZERO_POINT_VOLTAGE           (0.324) //define the output of the sensor in volts
when the concentration of CO2 is 400PPM
#define         REACTION_VOLTGAE           (0.020) //define the voltage drop of the sensor
when move the sensor from air into 1000ppm CO2
```

```
/****************************Globals**********************************************/
float        CO2Curve[3]  =  {2.602,ZERO_POINT_VOLTAGE,(REACTION_VOLTGAE/(2.602-
3))};
                                                     //two points are taken from the curve.
                                                     //with these two points, a line is formed which is
                                                     //"approximately equivalent" to the original curve.
                                                     //data format:{ x, y, slope}; point1: (lg400, 0.324), point2:
(lg4000, 0.280)
                                                     //slope = ( reaction voltage ) / (log400 –log1000)

void setup()
{
   Serial.begin(9600);                          //UART setup, baudrate = 9600bps
   pinMode(BOOL_PIN, INPUT);                          //set pin to input
   digitalWrite(BOOL_PIN, HIGH);                     //turn on pullup resistors
}

void loop()
{
   int percentage;
   float volts;
   volts = MGRead(MG_PIN);
   percentage = MGGetPercentage(volts,CO2Curve);
   if (percentage == -1) {
      Serial.println( "400" );
   } else {
      Serial.println(percentage);
   }

/*   Serial.print( "ppm" );
   Serial.print( "       Time point:" );
   Serial.print(millis());
   Serial.print("\n");
   if (digitalRead(BOOL_PIN) ){
      Serial.print( "=====BOOL is HIGH======" );
   } else {
      Serial.print( "=====BOOL is LOW======" );
   }

   Serial.print("\n");
*/
   delay(1000);
}



/**************************** MGRead ********************************************
Input:   mg_pin - analog channel
Output:  output of SEN-000007
Remarks: This function reads the output of SEN-000007
************************************************************************************/
float MGRead(int mg_pin)
{
   int i;
   float v=0;

   for (i=0;i<READ_SAMPLE_TIMES;i++) {
      v += analogRead(mg_pin);
      delay(READ_SAMPLE_INTERVAL);
   }
   v = (v/READ_SAMPLE_TIMES) *5/1024 ;
   return v;
}

/**************************** MQGetPercentage ********************************
Input:   volts   - SEN-000007 output measured in volts
```

```
      pcurve  - pointer to the curve of the target gas
Output:  ppm of the target gas
Remarks: By using the slope and a point of the line. The x(logarithmic value of ppm)
      of the line could be derived if y(MG-811 output) is provided. As it is a
      logarithmic coordinate, power of 10 is used to convert the result to non-logarithmic
      value.
********************************************************************************************/
int  MGGetPercentage(float volts, float *pcurve)
{
  if ((volts/DC_GAIN )>=ZERO_POINT_VOLTAGE) {
    return -1;
  } else {
    return pow(10, ((volts/DC_GAIN)-pcurve[1])/pcurve[2]+pcurve[0]);
  }
}
```

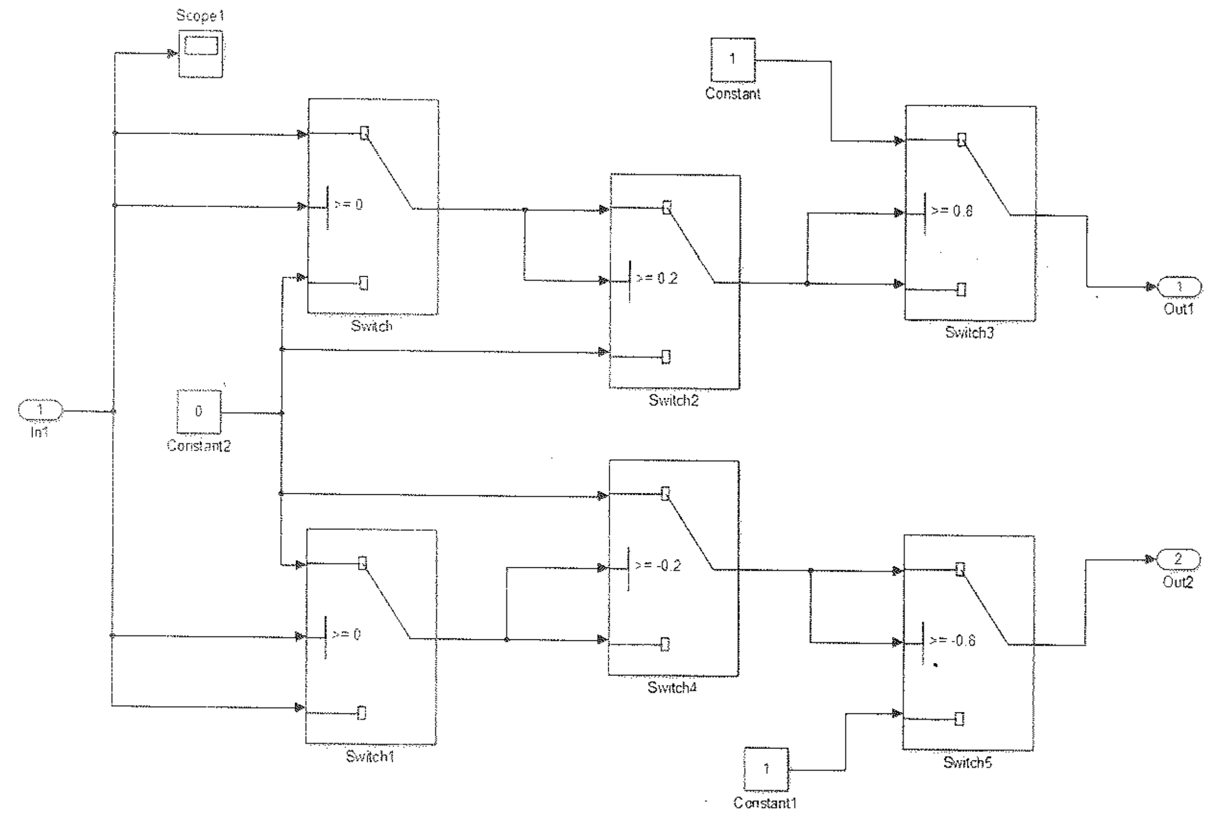
C.2 Code for Temperature Sensor

(parts supplied by BILDR.org)

```
#include <OneWire.h>
int DS18S20_Pin = 2; //DS18S20 Signal pin on digital 2

//Temperature chip i/o
OneWire ds(DS18S20_Pin);  // on digital pin 2

void setup(void) {
Serial.begin(9600);
}

void loop(void) {
 float temperature = getTemp();
 Serial.println(temperature);
 delay(100);
}

float getTemp(){
 //returns the temperature from one DS18S20 in DEG Celsius

 byte data[12];
 byte addr[8];

 if ( !ds.search(addr)) {
    //no more sensors on chain, reset search
    ds.reset_search();
    return -1000;
 }

 if ( OneWire::crc8( addr, 7) != addr[7]) {
    Serial.println("CRC is not valid!");
    return -1000;
}

if ( addr[0] != 0x10 && addr[0] != 0x28) {
   Serial.print("Device is not recognized");
   return -1000;
}

ds.reset();
ds.select(addr);
ds.write(0x44,1); // start conversion, with parasite power on at the end

byte present = ds.reset();
ds.select(addr);
ds.write(0xBE); // Read Scratchpad
for (int i = 0; i < 9; i++) { // we need 9 bytes
  data[i] = ds.read();
}

ds.reset_search();
byte MSB = data[1];
byte LSB = data[0];
float tempRead = ((MSB << 8) | LSB); //using two's compliment
float TemperatureSum = (tempRead / 16)*1.8 + 32;

return TemperatureSum;
}
```

Appendix D: Simulink Models

D.1 Duty cycle bucketization


Scope1
Constant
Switch
Switch2
Switch3
Out1
In1
Constant2
Switch4
Switch1
Switch5
Out2
Constant1
>= 0
>= 0.2
>= 0.8
>= -0.2
>= -0.8