University of Waterloo
Faculty of Engineering
Department of Electrical and Computer Engineering

Brew It Yourself

An Automated Single Vessel Home Brewery System

Group Number: 2016.019
Consultant: Douglas Harder

Kevin Nause (20413332)
Mathieu Tremblay (20420813)
Scott Wood (20379649)
Steve Jung (20411563)

Date: February 12, 2016

Abstract

Automated Single Vessel Home Brewery System

The art of home brewing has been steadily gaining popularity over the past 35 years alongside the rise of craft breweries in North America, so much so that in 2010 there were over 2000 craft breweries in the United States, after starting with only 8 in 1980. The objective of this project is to combine home brewing experience with engineering design, and construct a single vessel brewing system. By maintaining a strict control of key parameters, the brewing process is regulated using a combination of fluid mechanics, heat transfer, digital controls, power systems, embedded robotics and mobile development. The Robot Operating System (ROS), allows for a design where sensors can be added to a modular setup and provide feedback. By receiving feedback from temperature readings, density measurements and capacitance monitoring, the brewing process can be accurately recorded, shared, and automated by the system. The single vessel design allows for reduced complexity compared to the traditional home brewing method which requires various components, constant monitoring and heavy maintenance.

Acknowledgements

The Brew It Yourself team would like to acknowledge a number of people for providing guidance along the way. Project consultant Douglas Harder, for helping during the brainstorming phase and with interim financial backing; Steve Innocente of Innocente Brewing Company, for providing technical assistance with key problems during the design phase; Andrew Svoboda, for assisting with pre-build trials of the all-grain brewing process; and Rick Forgett and the rest of the Engineering Machine Shop staff, for assisting with the fabrication of our vessel and providing feedback on design decision

Contents

1 High-Level Description of Project **5**
- 1.1 Motivation . . . 5
- 1.2 Project Objective . . . 5
- 1.3 Block Diagram . . . 5
  - 1.3.1 Description of System . . . 6
  - 1.3.2 Designing and Not Designing Components . . . 7

2 Project Specifications **8**
- 2.1 Functional Specifications . . . 8
- 2.2 Non-Functional Specifications . . . 10

3 Detailed Design **11**
- 3.1 Mechanical Design . . . 11
  - 3.1.1 Material . . . 11
  - 3.1.2 Tank Structure . . . 12
  - 3.1.3 Integration of Waste Removal System . . . 13
- 3.2 Electrical System . . . 15
  - 3.2.1 Controller Designs . . . 15
  - 3.2.2 Power Electronics Design . . . 21
  - 3.2.3 Prototype Circuitry . . . 23
- 3.3 Computer System . . . 24
  - 3.3.1 Embedded Computer . . . 24
  - 3.3.2 Sensors and Microcontrollers . . . 24
  - 3.3.3 Web Server and Database . . . 27
  - 3.3.4 Sparging Algorithms . . . 28
  - 3.3.5 REST Web Server . . . 29
- 3.4 Mobile Application . . . 31
  - 3.4.1 Real Time Data . . . 33
  - 3.4.2 Input Controls and Interaction . . . 34

4 Discussion and Conclusions **35**
- 4.1 Evaluation of Final Design . . . 35
- 4.2 Use of Advanced Knowledge . . . 36
- 4.3 Creativity, Novelty, Elegance . . . 36
- 4.4 Quality of Risk Assessment . . . 37
- 4.5 Student Workload . . . 37

Acronyms **38**

Beer Terminology **38**

Technical Terminology **39**

Appendix A Prototype Hazard Disclosure **41**

Appendix B Symposium Floor Plan Request **42**

Appendix C Mechanical Materials **43**

List of Figures

1 Block Diagram outlining the interactions between the computer, electrical, and mechanical systems . . . 6
2 Illustration of the double wall design for the fermenter . . . 11
3 Illustration of the double wall design for the fermenter . . . 12
4 A render of the full vessel . . . 13
5 A render of the filter basket . . . 14
6 A render of the chugger pump mounting position . . . 14
7 A front view of the mechanical design of the conical fermenter . . . 15
8 A heat system diagram for a closed volume of liquid [1] . . . 16
9 Hysterisis loop temperature controller and plant model . . . 17
10 PID temperature controller and plant model . . . 17
11 PID controller parameters . . . 18
12 Lossless hysteresis loop control step response with control signal . . . 18
13 Lossless PID control step response with control signal . . . 19
14 Lossy hysteresis loop step response with control signal . . . 19
15 PID controller step response for various thermal conductivities . . . 20
16 Critically lossy hysteresis loop step response with control signal . . . 20
17 Large Servo Motor Power Circuit . . . 21
18 Small Servo Motor Power Circuit . . . 21
19 Hysteresis Loop Controller Power Circuit . . . 22
20 PID Controller Power Circuit . . . 22
21 Thyristor Bridge Motor Power Circuit . . . 23
22 Electrical Prototype . . . 23
23 State diagram for the water controller component . . . 27
24 State diagram for the temperature controller component . . . 27
25 A UML diagram showing the relation between tables and their entries in the logging database 27
26 Flow Design for Notification Service . . . 33
27 Flow Design for Data Statistics . . . 34
28 Flow Design for User Input Activity . . . 35

List of Tables

1 An overview of each functional specification of the project . . . 8
2 An overview of each non-functional specification of the project . . . 10
3 Criterion Table . . . 32
4 Decision Matrix . . . 33
5 Project Contributions . . . 37
6 Alloying elements of AISI 304 Stainless Steel [2] . . . 43
7 AISI Type 304 Stainless Steel Mechanical Properties [3] . . . 43
8 Alloying elements of 5052 Aluminium [2] . . . 43
9 Type 5052 Aluminium Mechanical Properties [3] . . . 43

1 High-Level Description of Project

1.1 Motivation

The art of home brewing has been steadily gaining popularity over the past 35 years alongside the rise of craft breweries in North America, so much so that in 2010 there were over 2000 craft breweries in the United States, after starting with only 8 in 1980 [4]. The traditional method for homebrewing requires various components, constant monitoring and heavy maintenance. There should be a solution which reduces complexity, making it much more affordable and practical for home use. We hope to create a single vessel system that would make the home brewing process precise, automated and compact, all at a reasonable price.

1.2 Project Objective

The objective of this project is to combine homebrewing experience with engineering design, and construct a single vessel brewing system. By maintaining a strict control of key parameters, the brewing process is regulated using a combination of fluid mechanics, heat transfer, digital controls, power systems, embedded robotics and mobile development. Robot Operating System (ROS), allows for a design where sensors can be added to a modular setup and provide feedback. By receiving feedback from temperature readings, density measurements and pH monitoring, the brewing process can be accurately recorded, shared, and automated by the system.

1.3 Block Diagram

Figure 1 shows relevant links between the physical, mechanical design of the brewing system, as well as the computer systems and their underlying software.

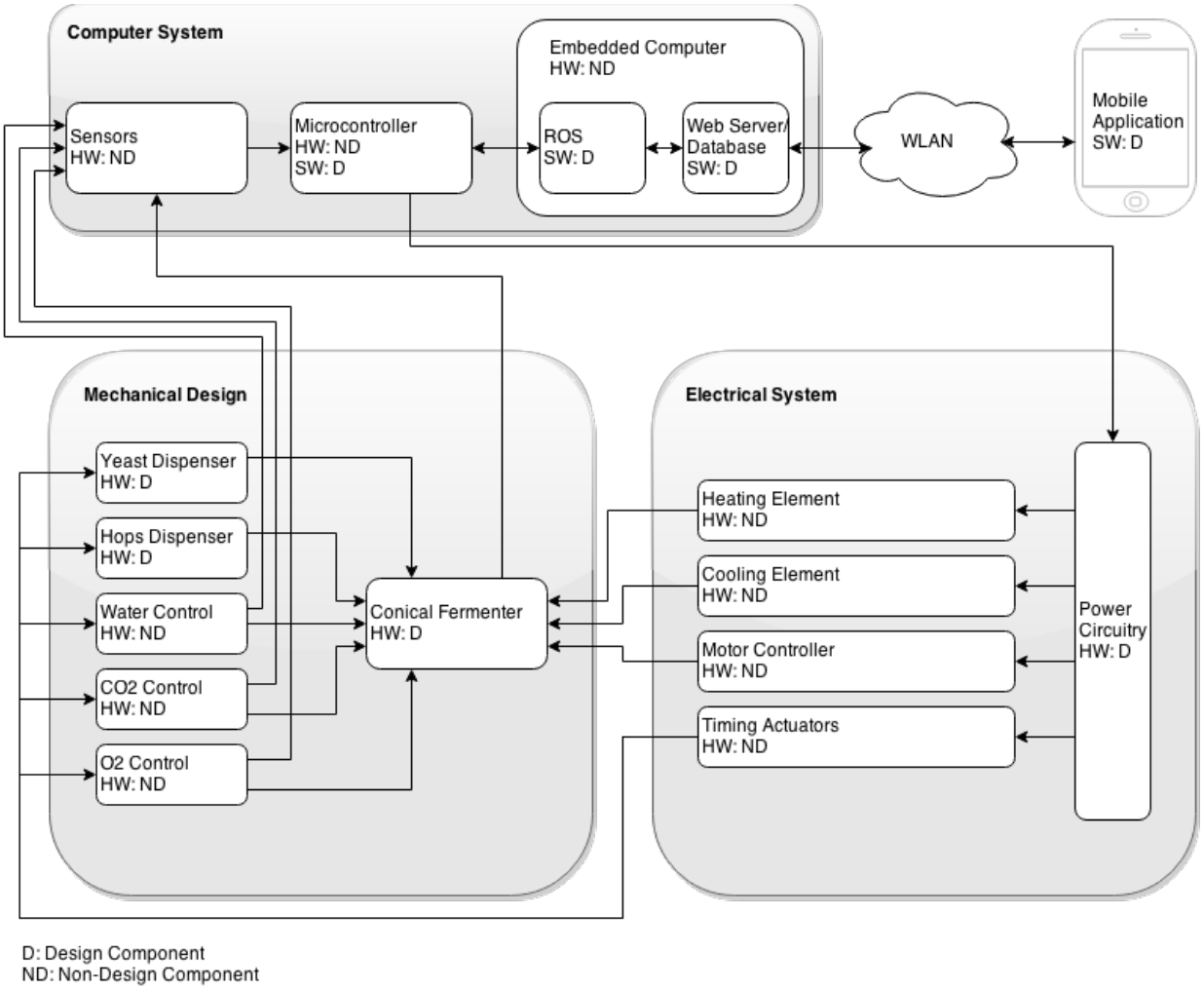

Figure 1: Block Diagram outlining the interactions between the computer, electrical, and mechanical systems

1.3.1 Description of System

The single vessel brewing system, as shown in Figure 1, contains various sensors which forward the environmental parameters to the microcontroller. The data from the various sensors accurately represents the state of the fermenter and feed into digital control loops running on the microcontroller. Data from sensor modules is obtained by a microcontroller via an Inter-Integrated Circuit (I2C) interface. This data is then sent to an embedded computer system over Universal Serial Bus (USB) where it is logged in a database and used to control relevant subsystems. Control commands are sent from the embedded computer to the microcontroller, where the appropriate component can be communicated with in order to regulate the brewing environment. The embedded computer system is then able to send diagnostic information and push notifications to a mobile device through a Wireless Local Area Network (WLAN) connection.

1.3.2 Designing and Not Designing Components

The following is an outline of the components in the block diagram and the requirements associated for each Designing (D) and Non-Designing (ND) component.

- **Sensors**
  ND: Sensors being used in the vessel are not being designed. Instead, off-the-shelf sensors are being purchased and integrated into the system. Relevant sensor types for this system include, but aren't limited to, pH sensors, volume sensors, flow meters, temperature probes, etc.

- **Microcontroller(s)**
  ND: The hardware of the microcontroller is not being designed since that's outside the scope of this project. The system instead uses an off-the-shelf microcontroller. D: The software running on the microcontroller is being designed. This software mainly consists of digital control loops for interfacing with the sensors and power electronic circuitry.

- **Conical Fermentation Vessel**
  D: The mechanical features and dimensions of the conical fermentation vessel are being designed to hold various sensors and other electrical and mechanical components.

- **Power Electronic Circuitry**
  D: The power electronic circuitry is being designed to power the pumps, motors, actuators, and heating/cooling mechanisms. The circuitry takes input from the digital controller running on the microcontroller and outputs the appropriate power to the end devices.

- **Heating and Cooling Elements**
  ND: The heating coils and refrigeration unit used to regulate temperature of the vessel are not being designed. Instead, off-the-shelf or salvaged and adapted components from existing appliances (e.g. hot water tank coils, home air conditioning heat pump) are being used and the power electronic circuitry is being designed to power these devices.

- **Motor Controller**
  ND: The motors, pumps, and mixing mechanisms used in our system are not being designed. Instead, off-the-shelf or salvaged and adapted components from existing appliances (e.g. blender mixing prop, aquarium pumps) are being used and the power electronic circuitry is being designed to power these devices.

- **Timing Actuators**
  ND: The actuators used in our system are not being designed. Instead, off-the-shelf solenoids and servo motors are being used and the mechanical subsystems that they will actuate are being designed. The power electronic circuitry is also being deigned to be able to power the actuators.

- **Embedded Computer System**
  ND: The hardware and operating system of the embedded computer system is not a design objective as it is outside the scope of this project. Instead an off-the-shelf embedded computer is purchased and a Debian based Linux operating system is installed to satisfy the requirements for the development environment.

D: The embedded computer system is configured with a Web Server, Database, and ROS. Microcontrollers can be modularly added to the system via USB and automatically configured as ROS nodes through negotiations.

- **Mobile Application**
  D: The mobile application is designed such that the user can receive push notifications and see various sensor data during the brew process.

2 Project Specifications

This section outlines the functional and non-functional requirements of the project design.

2.1 Functional Specifications

Table 2.1 describes each functional requirement and highlights whether it is essential or not to the completion of the project.

Table 1: An overview of each functional specification of the project

Specification	**Classification**	**Description**
Completion of the Brewing Process	Essential	The device automatically completes the brewing process, consisting of these steps: • Mash • Sparge • Boil Wort • Dispense Hops • Aerate the Wort • Pitch the Yeast • Kegging and Dispensing In its entirety and in the correct order. For more information on the brewing process and the terms used, please see the Glossary.
Heating Unit	Essential	Mashing the grains, sparging the grist, and boiling the wort all require high water temperatures. The system is able to accurately heat the contents to a minimum of 110°C within 3°C of error.

Cooling Unit	Essential	Yeast pitching and fermentation happen immediately after boiling the wort, but require temperatures a specific temperature (typically 20°C). The system is able to rapidly cool the wort to a temperature within 3°C of the target temperature so that the yeast may be pitched safely.
Temperature Regulation	Essential	The system is able to control temperature for each step in the brewing process. An error of 3°C is the target specification.
Trub Removal	Non-Essential	Remove the undesired sediment and other byproducts at the bottom of the fermenter so that ageing may take place within the vessel. The user does not have to maintain the trub accumulated by the brewing process.
Application Notifications	Non-Essential	The system provides notifications to the user updating them on the current state or action being taken. Additionally errors or warnings can be sent, in cases of emergency such as clogs, low oxygen or carbon dioxide, or an improper environment.
Database	Essential	The system is able query the specific steps it needs to automate for a specific brew. Additionally, it can store various data involved in brewing including temperature, pH levels and density measurements to generate logs and reports.
Reproducibility	Non-Essential	The system is able to record recipes, ingredients and steps for various brews. The user can recreate the same type of brew if they select one of the recipes.

2.2 Non-Functional Specifications

Table 2 describes each non-functional requirement and highlights whether it is essential or not to the completion of the project.

Table 2: An overview of each non-functional specification of the project

Specification	**Classification**	**Description**
Temperature Accuracy	Essential	The system accurately regulates the temperature required for the brewing process within 3°C of the targeted temperature.
Volume Control	Essential	The end result must be greater than or equal to the target yield volume. Target yields typically consist of 15.5 gallons, 7.75 gallons, and 5.16 gallons. These are the standard sizes of a half barrel keg, quarter barrel keg, and a sixth barrel keg respectively.
Size	Essential	The dimensions are limited such that the vessel can fit within a standard 36 inch residential door frame.
Mobility	Essential	The system is able to be relocated by a single person using the aid of caster wheels and or a standard 18 inch wide utility dolly.
Sanitation	Essential	The system employs SUS 304 stainless steel to maintain food-grade sanitation conditions. The vessel is self cleaning since a sanitary environment is crucial for the brewing process.

3 Detailed Design

3.1 Mechanical Design

The mechanical design component of this project is comprised of two aspects: the structural design of the fermenter and the thermodynamics of the heating and cooling systems. There are many sources for pre-fabricated fermenters for home brewing, but the custom nature of this device required the construction of a purpose-built tank, shown in Figure 2. The tank features a double walled construction to enclose the heating and cooling systems of the inner tank as well as provide thermal insulation.

Figure 2: Illustration of the double wall design for the fermenter

3.1.1 Material

To meet the requirements of food grade equipment, the components of the fermenter that will come into contact with the brewing ingredients are to be constructed from stainless steel. While the Canadian Food Inspection Agency (CFIA) (CFIA) does not outline specific details for the materials to be used in food processing, it states that, "Food contact surfaces of equipment and utensils are smooth, non-corrosive, non-absorbent, non-toxic, free from pitting, cracks or crevices, and able to withstand repeated cleaning and sanitation" [5]. To meet the non-corrosive guideline set by the CFIA, American Iron and Steel Institute (AISI) 304 stainless steel was the specific alloy of stainless steel chosen for the construction of the food-contacting surfaces of the fermenter. 304 is a member of the Austenitic Stainless Steel family (see Austenite), which have the best general corrosion resistance of any family of stainless steel. They also contain low levels of carbon, aiding in weld-ability. The low levels of carbon also act to reduce the generation of Inter-granular Corrosion caused by welding, over the life of the component [6]. Tables of the composition of 304 and pertinent mechanical properties are included in C. The outer tank of the fermenter will not come into contact with the brewing ingredients, and as such, other materials separate from stainless steel may be used. Aluminium was chosen to be the material for the outer tank due to its low mass and financial cost.

3.1.2 Tank Structure

When contemplating the mechanical design for a double-walled system such as this fermenter, the wall thickness for the inner tank should be determined early as this will dictate the structural support required between the inner and outer tanks, as well as the frame to support the entire vessel. Forced carbonation of different brews can reach up to 30psi. The American Society of Mechanical Engineers (ASME) outlines a function for determining the wall thickness of a pressure vessel as

$$t = \frac{P_{work} \times r \times FS}{\sigma_{uts} \times E} \tag{1}$$

Where:
P is the working pressure of the vessel
r is the radius of the vessel
FS is the factor of safety
σ_{uts} is the Ultimate Tensile Strength (UTS) of the material
E is the non-dimensional efficiency of a welded seam

Therefore:

$$t = \frac{206.84kpa \times 254mm \times 9}{505MPa \times 0.6} \tag{2}$$

$$t = 1.560mm \tag{3}$$

The recommended thickness derived from this equation is approximately 1/16 of an inch, or 16 gauge when used to describe sheets of stainless steel. The outer tank thickness can now be specified using the mass of the inner tank and the brewing ingredients. The design of the system features collars at the top and bottom of both tanks to distribute the load evenly around the perimeter of the outer tank. The deflection and shear formulas used to determine the forces acting on the tank collars were first introduced in ME 220, Mechanics of Deformable Solids, and ME 322 Machine Design.

Figure 3: Illustration of the double wall design for the fermenter

Formula 4 was found using the Mechanical Engineer's Handbook [7]. The maximum stress experienced by the collar of the outer tank occurs at the inner edge of the collar. The UTS of aluminium can be used as this stress to determine the minimum sheet thickness required to support the fermenter.

$$\sigma = \frac{3w}{mt^2(a^2 - b^2)}\Big(a^4(3m+1) + b^4(m-1) - 4ma^2b^2 - 4(m+1)a^2b^2\ln(\frac{a}{b})\Big) \tag{4}$$

Where:
t is the sheet thickness
m is the reciprocal of Poisson's Ratio
a is the outer diameter of the collar
b is the inner diameter of the collar
w is the applied force on the collar

Therefore:

$$t^2 = \frac{3w}{m\sigma(a^2 - b^2)}\Big(a^4(3m+1) + b^4(m-1) - 4ma^2b^2 - 4(m+1)a^2b^2\ln(\frac{a}{b})\Big) \tag{5}$$

The resultant thickness is 2.05mm, or 0.08in. Accounting for a safety factor, a sheet thickness of 0.125in. was selected to meet the requirements for the outer tank.

3.1.3 Integration of Waste Removal System

The main challenge following the the design and construction of the structure of the fermenter is the separation of desired brewing products from spent ingredients. The full rendering below, Figure 4 shows the full fluid circuit.

Figure 4: A render of the full vessel

The first step in the process is the removal of the batch from the fermenting vessel. A large, two inch ball valve was selected to mitigate any potential blockage in the system which could be caused by the viscous and tacky mash. After the batch ingredients pass through the valve, they enter into a chamber featuring a wire mesh basket held within the lower container, shown in Figure 5.

Figure 5: A render of the filter basket

The secondary vessel features a removable, clamp-on door attached to the mesh basket. This aids with cleanup and preparation for the subsequent brew. Leftover ingredients and trub can be contained in this vessel during the brewing process until the user is ready to remove any waste. A "Chugger Pump" is then used to replace the remaining brew into the primary vessel to proceed with fermentation process. A rendering of the fitment of the pump is shown below in Figure 6.

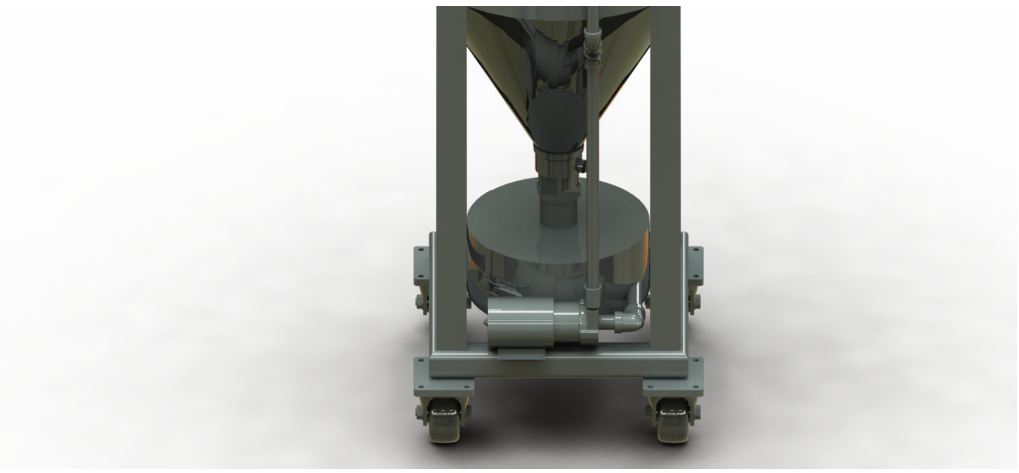

Figure 6: A render of the chugger pump mounting position

The chugger pump [8] was chosen for its durability and resilience, specifically when moving large quantities of sugar rich wort. The integration of these components allows for the completion of an all-grain brew, an increasingly popular method used by contemporary home brewers.

In summary, the mechanical design and renderings contributed to the final mechanical prototype as shown in Figure 7.

Figure 7: A front view of the mechanical design of the conical fermenter

3.2 Electrical System

The electrical system consists of two main parts, the controller and the power electronic circuitry. These two subsystems are coupled closely together in that the control signals produced by the controller are amplified by the power electronics and sent to each plant. The main plants that are controlled and powered by the electrical systems are the heating element, cooling system, mixing motor, and various small similar actuators.

3.2.1 Controller Designs

Of the four controllable subsystems, the heating and cooling for fine temperature regulation is the highest priority for robust design. The pump and actuators are necessary for the final realised system, but can easily be constructed with off-the-shelf components and minimal design effort. The actuators used in the system to open/close various valves and chutes are implemented using servomotors as the electrical component. The controllers used to control these actuations outputs a simple on-off signal and does not require feedback in the form of sensors. The continuous-time transfer function model of a DC motor with motor voltage as input and gear angle as output is shown Equation 6

$$P(s) = \frac{K}{s(\tau s + 1)} \tag{6}$$

Which is open-loop unstable when tracking a step input. However, when tracking a ramp (i.e. a constant velocity) the system is stable, though with large steady-state error. This makes intuitive sense because when a constant voltage is applied to a DC motor, the speed of the motor stabilizes to a constant value but the position of the motor goes unstable and increases indefinitely. Because of the fact that we don't require any sort of motor position tracking, nor do we require precise speed control, the control system for the pump motor is an open-loop system without any feedback. Instead, the gear ratio of the gearbox that is powered by this motor is constructed to allow us crude speed control by simply changing the voltage output to the motor.

Since the temperature model of the main volume of water is the same regardless of increasing or decreasing temperature, with only a sign-change to account for, the cooling system is assumed to function identically to the heating system in terms of power delivered to or taken from the water. Figure 8 shows a sketch of the physical system, and the equation that relates heat input to heat storage and heat loss is described in Figure 7.

$$Qh = Qs + Ql \tag{7}$$

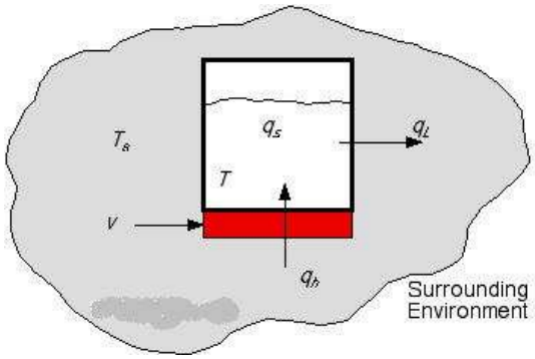

Figure 8: A heat system diagram for a closed volume of liquid [1]

If we differentiate with respect to time, we get Equation 8 [1].

$$P = C \times \frac{dT}{dt} + k(T - T_a) \tag{8}$$

Where P is the power input into the system, C is the thermal capacity of the water, k is the thermal conductance of the tank, T_a is the ambient temperature of the room, T is the temperature of the water, and t is time. Taking the Laplace transform of both side, we get Equation 9.

$$P(s) = C \times s \times T(s) + kT(s) \tag{9}$$

Resulting in Equation 10

$$\frac{T(s)}{P(s)} = \frac{1}{Cs + k} \tag{10}$$

Our transfer function model of the heating of a volume of liquid. Finding the value of the parameter C

is relatively straightforward, it is equal to the mass of water in the system multiplied by the specific heat capacity of water. Since our system is designed to hold 50L of water, we get the result shown in 11

$$C = 50L \times 1\frac{kg}{L} \times 4200\frac{J}{kgK} = 210,000\frac{J}{K} \tag{11}$$

Since the inner tank of our conical vessel is well insulated, we know that the value of k (in units of Watts per Kelvin) is small. However, calculating k mathematically is difficult and it is easier to instead perform some system identification on the completed system. The controller used to control the temperature of the volume of water can be implemented in many different ways, however we are deciding to look at 2 main options: a Proportional-Integral-Derivative (PID) controller, and a hysteresis-thermostat controller. Pictured below in Figure 9 and Figure 10 are Simulink models created in MATLAB to simulate these two possible controller schemes. Additionally, the tuned PID parameters used in the latter diagram, as calculated using the built-in Simulink PID Tuner tool-kit, are listed in Figure 11.

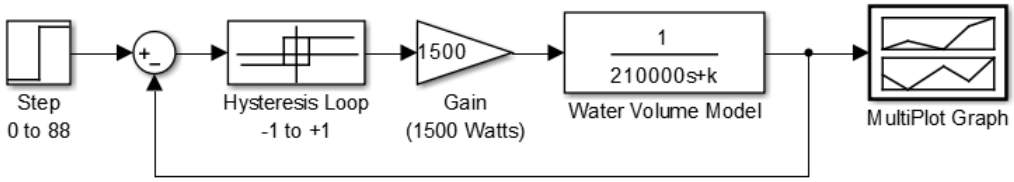

Figure 9: Hysterisis loop temperature controller and plant model

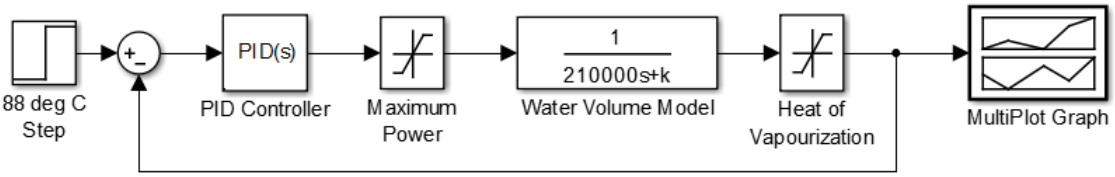

Figure 10: PID temperature controller and plant model

Controller parameters

Proportional (P):	46.1237643092313
Integral (I):	8.08607295128549e-05
Derivative (D):	-108891.122439572
Filter coefficient (N):	0.000423576902100788

Compensator formula

$$P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}}$$

Figure 11: PID controller parameters

In these models, the step input has an amplitude of 88 degrees Celsius, simulating the heating of water from 10°C to 98°C; and the maximum allowable control signal is 1500W, which equates to 12.5A on standard 120V household power. We are looking to optimise a number of parameters with the selection of one of these control systems. Specifically, we want the best combination of settling time and steady-state error versus cost and complexity of implementation. Pictured below in Figure 12 and Figure 13 are the step responses of each of these systems, simulated assuming a k-value of zero (well insulated, lossless vessel).

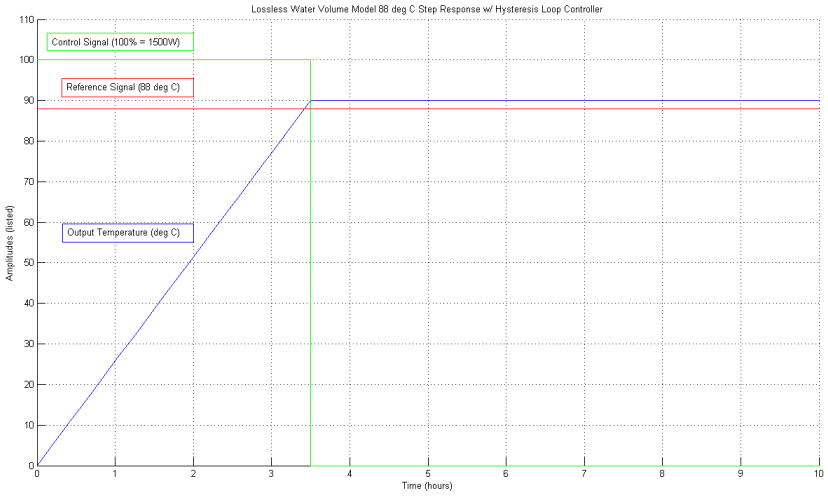

Figure 12: Lossless hysteresis loop control step response with control signal

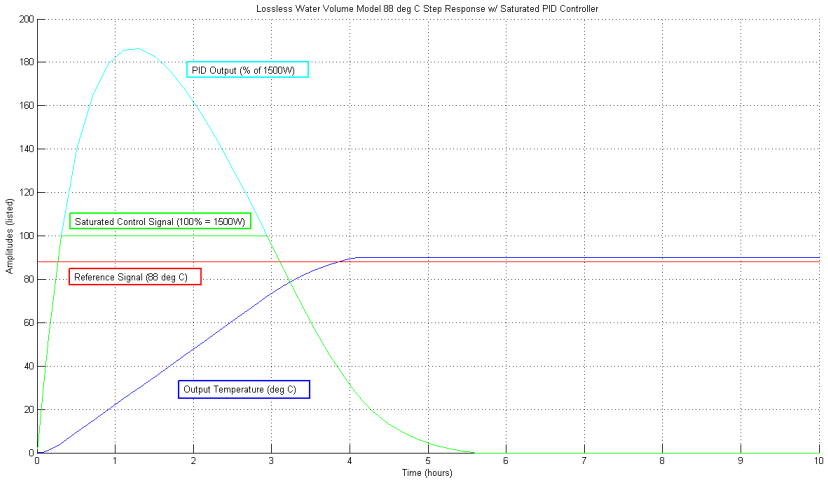

Figure 13: Lossless PID control step response with control signal

It can be seen from these plots that the hysteresis controller reaches the full-value thirty minutes earlier than the saturated PID controller. Since this model is perfect and lossless, the operation of the hysteresis loop isn't very apparent. If we increase the thermal conduction value to a modest level (k = 10), we get the response plotted in Figure 14. In this figure, we can clearly see that the switching period is around 20-minutes, which is long enough to ensure that we aren't over-switching our relays to run this controller. The new settling time is close to five hours in this plot.

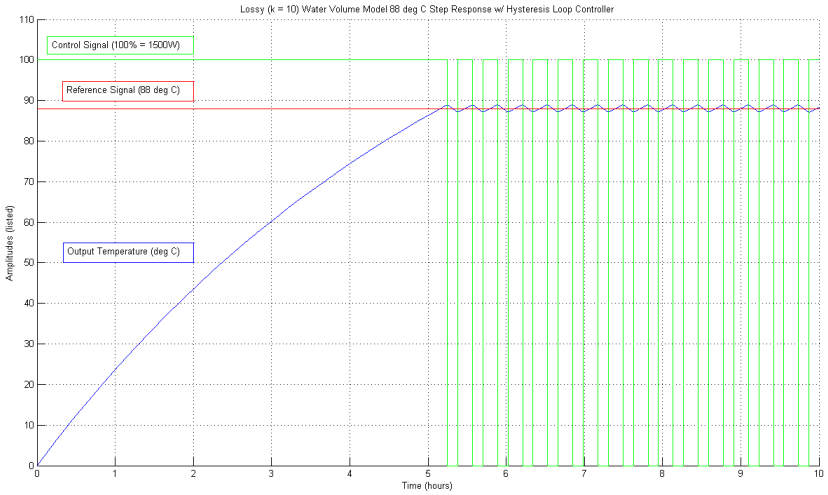

Figure 14: Lossy hysteresis loop step response with control signal

We can compare this to Figure 15 which plots the response of the PID system with varying k-values. In this plot, we see that for many values of k, the response time is also close to 5 hours, but as k increases the steady-state error quickly worsens to an unacceptable level.

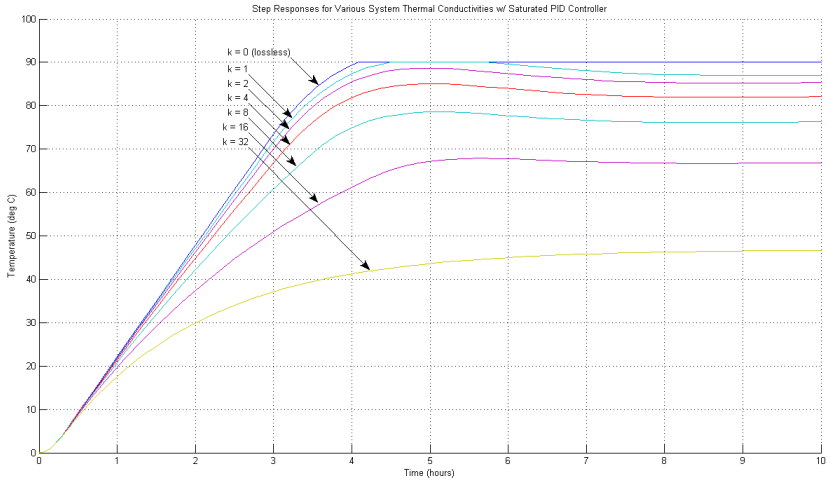

Figure 15: PID controller step response for various thermal conductivities

For the PID system, the "critical k-value", defined as the k-value for which the steady-state error is beyond an acceptable level, is below $k = 4$. If we compare this to Figure 16 where the critical k-value of 16.67 for the hysteresis controller is plotted, we can see that the hysteresis controller is much more robust when faced with uncertain values of k since the critical value is over four times larger.

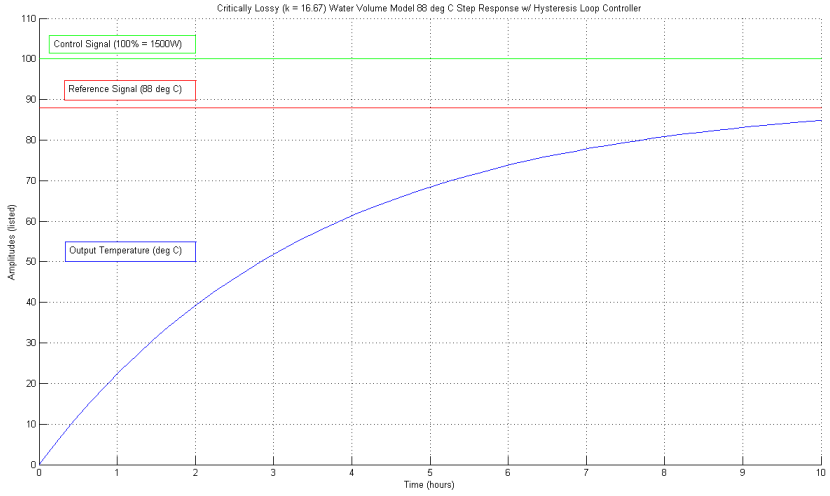

Figure 16: Critically lossy hysteresis loop step response with control signal

As such, the hysteresis-loop thermostat controller is being chosen to implement the heating and cooling of the vessel. Additional to the superior performance, the hysteresis controller has the added benefit of reduced implementation complexity in terms of the required power electronic circuitry, as discussed in the next section.

3.2.2 Power Electronics Design

The purpose behind the design of the power electronic circuitry is to be able to effectively implement the control strategies required to meet our design criteria. With the chosen control systems discussed in the previous section, the basic design of the power electronic circuits is straightforward. Depending on the size and power requirements of the servomotors needed to actuate the mechanical chutes and valves of the system, one of two circuits can be implemented. For large servos, the circuit pictured in Figure 17 can be implemented to power the motor.

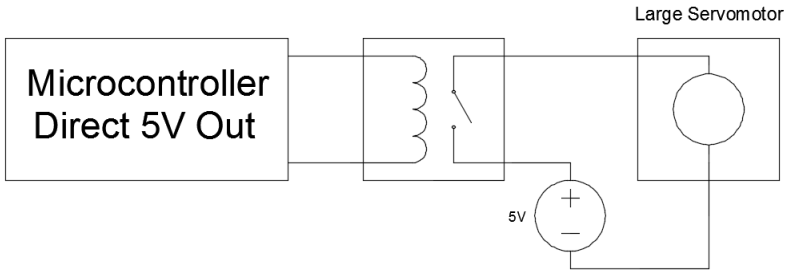

Figure 17: Large Servo Motor Power Circuit

However, for smaller servos that require less current to function, the relay and external DC source can be removed from the circuit completely, and the motor can be powered directly from the microcontroller's 5V output pins, as depicted in Figure 18.

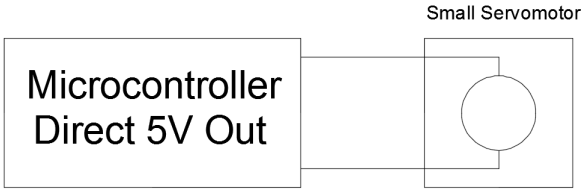

Figure 18: Small Servo Motor Power Circuit

The heating element alluded to in the previous section is realised through the use of immersion hot water tank heaters. The original design called for a length of insulated resistive wire wrapped around the vessel, but that became too difficult to source parts for and implement. Additionally, the original design for the cooling system revolved around a compressor-driven refrigeration cycle with coolant coils wrapped around the vessel walls, but once again the sourcing of a refrigeration system proved too difficult to implement. Instead, the cooling is realised by an array of 20 solid-state Peltier cooling tiles scattered evenly around the vessel walls. Heat from the hot side of the tiles will be sunk away via aluminum blocks with water channels, where water passes through them to an outer reservoir with a larger forced-air heat sink. The Peltier tiles

are powered through a purchased 12V, 1500W supply with a relay on the 120V input such that the control circuitry becomes identical to the heater.
Figure 19 and Figure 20 illustrate the circuits for the hysteresis control and the unselected PID control schemes respectively. The Hysteresis control is realised with a simple circuit consisting of a relay between the 120V power supplied by the house and the immersion heating element. The PID controller could have been realised with a triac whose firing angle is varied to modulate the power output to the heating coil.

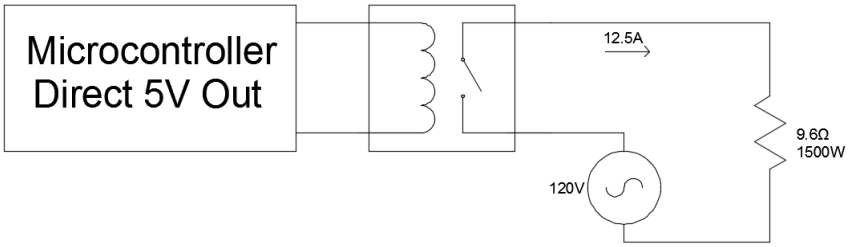

Figure 19: Hysteresis Loop Controller Power Circuit

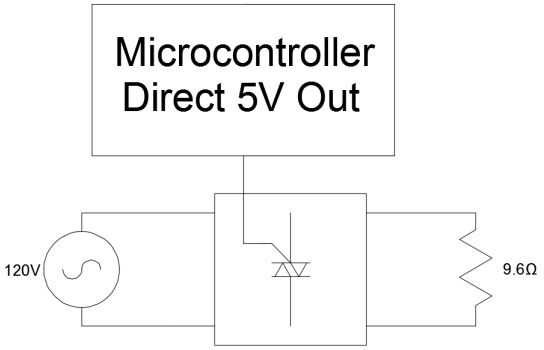

Figure 20: PID Controller Power Circuit

The pumping system is driven by a DC motor, and the circuit used to power it is pictured in Figure 21. It consists of a step-down transformer to reduce the mains voltage down to 12 volts to power the motors, then a full-bridge thyristor rectifier is used to simultaneously rectify the AC input voltage to a DC voltage and control the magnitude of the output voltage by varying the thyristor firing angle . The DC motor can be modeled as an inductance in series with a resistance, followed by a reverse-emf voltage.
The original design called for an additional large DC motor to power a mechanical mixer that would be submersed in the vessel, however it was decided that the turbulence generated from boiling would suffice to aid in the extraction of sugars from the grain.

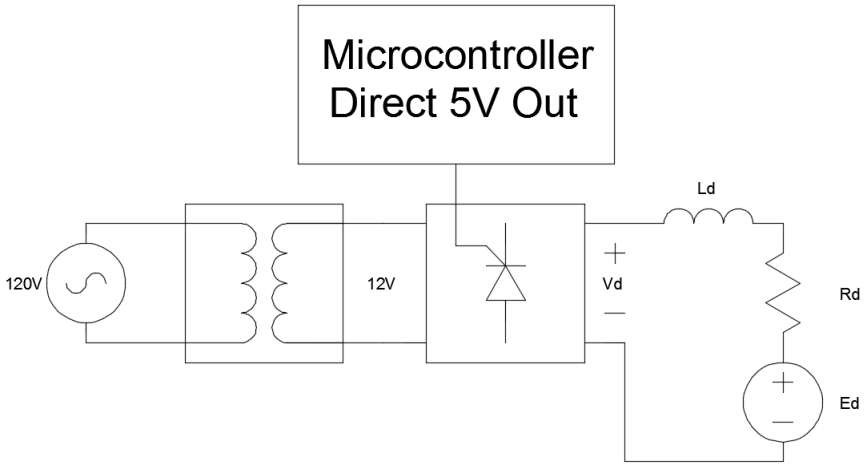

Figure 21: Thyristor Bridge Motor Power Circuit

3.2.3 Prototype Circuitry

Since we only recently received our vessel back from the machine shop from welding, none of the test electronics have been installed. However, Figure 22 shows a test board where the relays have been tested by powering loads, controlled by the microcontroller on the far right. The next step is to install these components and their duplicates into a NEMA-rated electrical box, affix the actuatable components to the vessel, and run the wiring between everything.

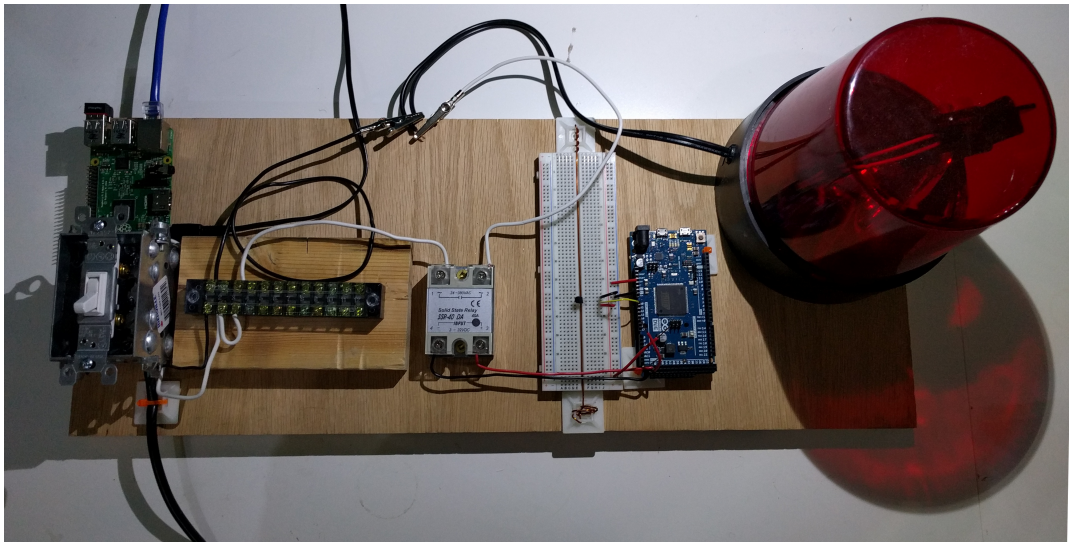

Figure 22: Electrical Prototype

3.3 Computer System

The computer systems consist of three main components, the embedded central computer, the microcontroller and sensor pairs, as well as a Web Server and Database pair to allow for storage and communications. The detailed description of each subcomponent and the methods of interaction between them is discussed in this section.

3.3.1 Embedded Computer

The primary function of the embedded computer system is to aggregate data in close to real-time from the multiple sensors used in the design. Analysis of this data is crucial for determining the current progress in the mash, sparge, and boiling of the wort. For specific analysis of sensor data please see subsection 3.3.2.

For ease of repeatability, availability, and cost the Raspberry Pi was selected. At the time of design and creation of this report, this is the most recent hardware revision of the Raspberry Pi. The relevant hardware specifications for the Raspberry Pi is a quad-core Acorn Reduced Instruction Set Computing (RISC) Machine (ARM) processor, 1GB of Random Access Memory (RAM), and a Bluetooth dongle for local wireless communications (rated to 50m). Since the outer shell of the fermenter is aluminum, the casing for the Raspberry Pi can act as a heat sink, allowing it to be safely overclocked to 1GHz, 500MHz, and 500MHz for the ARM processor frequency, sdram frequency, and L2 cache (core) frequency respectively. The configuration file below allows for a substantial performance boost with negligible thermal impact.

```
arm_freq=1000
sdram_freq=500
core_freq=500
over_voltage=2
arm_freq_min=400
sdram_freq_min=250
core_freq_min=250
```

The operating system chosen was the ARM build of Ubuntu 14.04 Long Term Support (LTS), codename Trusty Tahr, as well as ROS Jade Turtle for sensor coordination and communication. Ubuntu 14.04 was chosen over the default Raspbian operating system for increased security due to the use of SELinux policies. Since the Raspberry Pi will be acting as a Lamp Apache MySQL PHP (LAMP) server as well for mobile interaction and remote notifications, enforcing read only access over the configured port of choice is a necessity. For an in depth overview of the mobile application please see subsection 3.4. Additionally, the LTS version of Ubuntu was chosen for a maximum of five years support, as well as increased compatibility and reliability for repository packages specific to the trusty distribution.

3.3.2 Sensors and Microcontrollers

Every sensor in the system is paired with a microcontroller to enable a modular design and a plug and play like support. By adding more sensors to the system, more information can be provided as feedback to enable better logging and regulation of the brewing environment. The only mandatory sensors for the system are a main temperature sensor for the wort and flow meters on the liquid inputs and outputs of the system. By introducing additional sensors, such as pH and density sensors, certain yield thresholds of interest in the sparge and fermentation procedure can be met rather than approximated. Additionally, this will allow for

a tier based configuration for the automated vessel to reduce cost and complexity if desired. As the end goal for this project is to have a community influenced input, having a customizable sensor configuration with open hardware and software pairs is encouraged. For maximum affordability the Arduino compatible Teensy-LC microcontroller is used to poll the sensors and report data, however any Arduino compatible microcontroller may be used. The bridge between the ROS instance on the Raspberry Pi and the Arduino compatible microcontroller is achieved through the use of the ROSserial package. Example code for obtaining data from a TMP102 temperature sensor over I2C at address 0x91 [9] can be seen below [10].

```
#include <Wire.h>
#include <ros.h>
#include <std_msgs/Float32.h>

//Set up the ros node and publisher
std_msgs::Float32 temp_msg;
ros::Publisher pub_temp("temperature", &temp_msg);
ros::NodeHandle nh;

int sensorAddress = 0x91 >> 1; // From datasheet
long publisher_timer;

void setup()
{
  Wire.begin();      // join i2c bus (address optional for master)
  nh.initNode();
  nh.advertise(pub_temp);
}

void loop()
{
  if (millis() > publisher_timer) {
  // step 1: request reading from sensor
    Wire.requestFrom(sensorAddress,2);
    delay(10);
    if (2 <= Wire.available()) // if two bytes were received
    {
      byte msb;
      byte lsb;
      int temperature;

      msb = Wire.read(); // receive high byte (full degrees)
      lsb = Wire.read(); // receive low byte (fraction degrees)
      temperature = ((msb) << 4); // MSB
      temperature |= (lsb >> 4); // LSB

      temp_msg.data = temperature*0.0625;
      pub_temp.publish(&temp_msg);
    }
```

```
  publisher_timer = millis() + 1000; //publish once a second
  }
  nh.spinOnce();
}
```

Additionally, controllers can be linked to the system via Arduino compatible microcontrollers and receive inputs based on sensor data gathered via ROS publications. To improve the resolution or accuracy of data and controls, the polling interval of each of the microcontrollers can be calibrated appropriately. Since most sensors will remain idle for a majority of the brewing process, their sampling rates can be scaled using a simple algorithm as suggested in the code sample below.

```
#include <std_msgs/Float32.h>

int sample_rate = MIN_SAMPLE_RATE;
int sample_count = 0;

void scale_rate(std_msgs::Float32 prev, std_msgs::Float32 curr)
{
    if(prev == curr)
    {
        sample_count++;
    }
    else
   {
        sample_count--;
   }

   if(sample_count == SCALE_THRESHOLD)
   {
         sample_rate = increase_rate();
      sample_count = SCALE_THRESHOLD >> 1;
   }
   else if(sample_count == 0)
   {
      sample_rate = decrease_rate();
      sample_count = SCALE_THRESHOLD >> 1;
   }
}
```

The two fundamental controllers that require communications for the operation of the system are the water controller and the heating and cooling controllers. Figure 23 demonstrates a state diagram for the water controller and Figure 24 demonstrate the state diagrams for toggling the heating and cooling states of the temperature controller. The variables volume, temp, target and offset represent the current volume, current temperature, desired resultant, and amount of allowed deviation from target respectively.

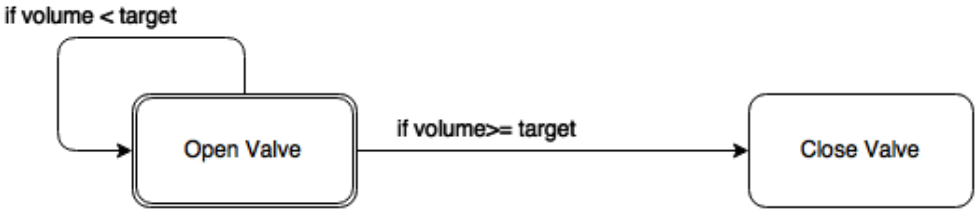

Figure 23: State diagram for the water controller component

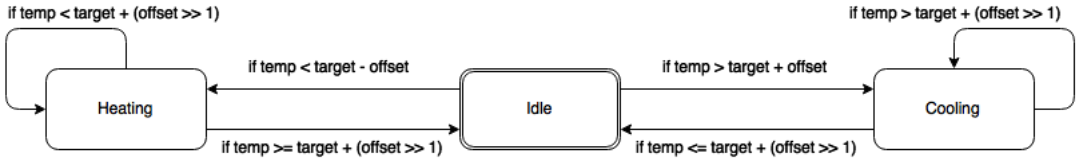

Figure 24: State diagram for the temperature controller component

3.3.3 Web Server and Database

A LAMP server is the current solution for delivering information from the embedded computer to the mobile application. All communications and parameters are to be communicated using a JavaScript Object Notation (JSON) format. The database will consist of relational connections between recipes, instructions, ingredients, sensors, and logged data as demonstrated in Figure 25.

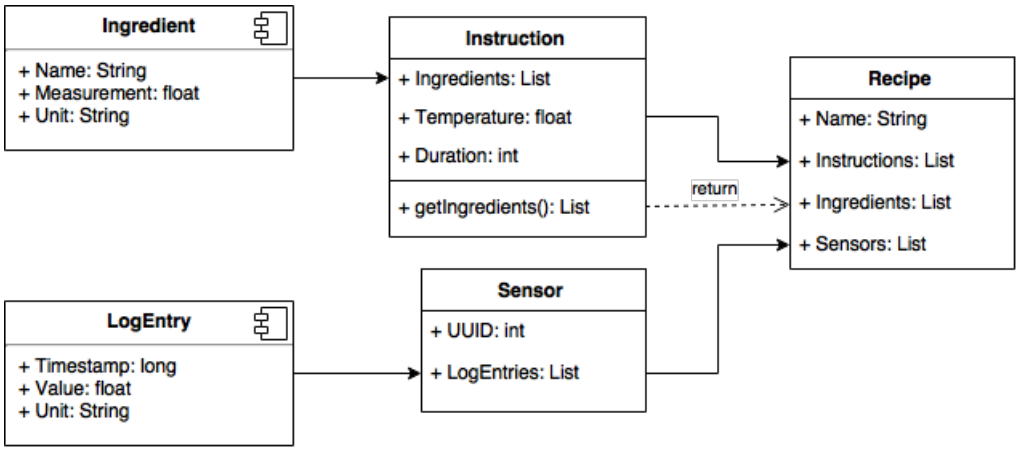

Figure 25: A UML diagram showing the relation between tables and their entries in the logging database

An important component of the web server is the server to client communication. The communication will be done through using the Google Cloud Messaging (GCM). This service allows a server to send messages to clients on various platforms [11]. There are two types of messages that can be sent using GCM, notification and data messages. The differences between the notification and data messages is that GCM automatically displays the notification messages on the client's behalf. Where as the data messages has to be parsed by the client in order to be used. Another difference is while the notification type messages has fixed key value

pairs, data type messages can have custom ones. Both message types are in JSON format, which is shown below.

```
{
    "to" : "XYZ",
    "data" : {
    "degrees" : "85",
    "component" : "Water"
    }
}
```

The "to" value holds the unique application id and the data block holds the key value pairs that are encapsulated in the message. For message passing, this application uses the data message type. This is because notification messages does not offer custom key-value pairs that is required for transferring information from server to client. Although additional processing is done in the client side, data messages can be flexible to handle the various state changes that the system may undergo. In conclusion, the GCM component of the web server uses the Extensible Messaging and Presence Protocol (XMPP) protocol that sends data type messages to the client.

3.3.4 Sparging Algorithms

The no-sparge technique uses approximately 25% more grain than a standard recipe and allows for the sparging process to be bypassed with no ill effects to the brew. As a result, this method produces a richer, smoother-tasting wort with the same gravity as a standard recipe, but with a mash and optionally a lauter process that makes the wort more robust and pH stable. Not only is the end result more desirable to the consumer, but having a pH stable wort allows for a reduced risk of souring due to bacteria and diminishes the chances of abnormal deviations. Ultimately, this process greatly reduces the complexity of the all grain brewing process as well as the overall complexity of the system.

The following calculations combine the scaling-up of the grain bill with a three-step infusion-mash that makes the whole process more manageable [12]. Below is a list of the relevant symbols and terms used for input parameters, constants, and output parameters respectively.

The input parameters to the no sparge:
OG is the standard recipe original gravity
G_r is the standard recipe grain bill
V_r is the standard recipe batch size in gallons
V_b is the standard recipe boil volume in gallons

Relevant constants to be considered:
k is the water retention coefficient (0.125 gal/lb)
R_r is the standard recipe conversion rest mash ratio (typically 1 qt/lb)

The output parameters:
S is the scale-up factor for the grain bill
G_n is the no-sparge grain bill in pounds

BG is the no-sparge boil gravity
R_n is the no-sparge final mash ratio (qt/lb)
W_n is the no-sparge total water volume in quarts
W_{mo} is the mash-out water volume in quarts
V_t is the no-sparge total mash volume in quarts

The scale up factor is calculated by using Equation 12 and is used in determining the no-sparge grain bill.

$$S = \frac{V_b}{(V_b - kG_r)} \tag{12}$$

The no-sparge grain bill is calculated using Equation 13.

$$G_n = S \times G_r \tag{13}$$

The no-sparge boil gravity is adjusted by using Equation 14.

$$BG = OG \times \frac{V_r}{V_b} \tag{14}$$

The total no-sparge water volume in quarts is determined by Equation 15.

$$W_n = 4(V_b + kG_n) \tag{15}$$

By using Equation 16 the no-sparge mash ratio is calculated.

$$R_n = \frac{W_n}{G_n} \tag{16}$$

The volume of water used for the mash-out in quarts is determined by Equation 17.

$$W_{mo} = G_n(R_n - R_r) \tag{17}$$

Finally, the total no-sparge mash volume in quarts is calculated using Equation 18. The volume of 1 pound of dry grain, mashed at 1 quart per pound, has a volume of 42 fluid ounces or 1.3125 quarts. Higher ratios only add the additional water volume, but can be used for different styles of recipes.

$$V_t = G_n(1.3125 + (R_n - 1) \tag{18}$$

By using the methods listed above, the no-sparge process is successfully implemented while maintaining compatibility with standard recipes and the all grain brewing process.

3.3.5 RESTful (REST) Web Server

The RESTful Web Server is implemented using the Jersey Framework. The Jersey Framework provides various Application Program Interfaces (APIs) and services to streamline web server development. Services such as JSON parsing and formatting, error logging and Uniform Resource Identifier (URL) mappings are all handled by this framework. There are two main functionalities that is implemented in the web server; the *GCMNotificationResource* and the *BrewLogResource*. Each functionality is mapped to a URL endpoint

called a Jersey Resource. A sample of the *GCMNotificationResource* is shown in the code sample below. The following code samples are all written in Java.

```
@Path("gcm_notification")
public class GCMNotificationResource {

    GCMSender gcmSender;

    @GET
    @Path("/send")
    public Response sendSimpleMessage(String message) {
        GCMSender.gcmSendMessage(message);

        return Response.ok().build();
    }
}
```

The above resource is called whenever the system needs to notify the client. The resource calls the function *gcmSendMessage* which sends the message to the GCM server. The GCM server is an external service that is used to push notifications to client applications. A sample code of the *gcmSendMessage* function is shown below where the message being sent to the client is put into a JSON object to be passed to the GCM server.

```
public static void gcmSendMessage(String message) {
        // Prepare JSON containing the GCM message content. What to send and where to send.
        JSONObject jGcmData = new JSONObject();
        JSONObject jData = new JSONObject();
        jData.put("message", message);
        // Where to send GCM message.
        jGcmData.put("to", "/topics/global");

        // What to send in GCM message.
        jGcmData.put("data", jData);

        // Create connection to send GCM Message request.
        URL url = new URL(GCM_URL);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestProperty("Authorization", "key=" + API_KEY);
        conn.setRequestProperty("Content-Type", "application/json");
        conn.setRequestMethod("POST");
        conn.setDoOutput(true);

        // Send GCM message content.
        OutputStream outputStream = conn.getOutputStream();
        outputStream.write(jGcmData.toString().getBytes());
}
```

[13] Code above is based on GCMSender.java provided by Google.

This satisfies the functional requirement "Application Notification" seen in the Functional Specifications Section 2.1, where the system can provide notifications to the user simply by calling the URL with the message that it wants the user to receive.
The *BrewLogResource* is implemented to fulfil the "Database" requirement as seen in Functional Specifications Section 2.1. The resource provides a access point from the server to the database. A sample of the resource code is shown below:

```
@Path("brewlog")
public class BrewLogResource {

    private BrewLogService brewLogService;

    @GET
    @Path("/{brewid}")
    @Produces("application/json")
    public Response getBrewLogs(@PathParam("brewid") int brewid) {
        List<BrewLog> brewLogs = brewLogService.getBrewLogs(brewid);

        if (brewLogs.size() > 0) {
            return Response.ok().type(MediaType.APPLICATION_JSON_TYPE).entity(brewLogs).build();
        } else {
            return Response.noContent().build();
        }
    }

    @PUT
    @Path("brew_data/")
    @Consumes("application/json")
    public Response putBrewData(BrewLog brewLog) {
        brewLogService.putBrewLog(brewLog);
        return Response.accepted().type(MediaType.APPLICATION_JSON_TYPE).build();
    }
}
```

This resource allows both reads and writes to the database. This gives the functionality of storing various data involved in brewing through write operations and querying the database through read operations.

3.4 Mobile Application

The mobile application subsystem provides an interface to the user and consists of three main components. The first is a service that provides real time data of the current brew to the user, the second component is an interface that provides data analytics recorded by the sensor, as well as providing an interface that allows the user to input controls.
Platforms considered to develop the mobile application includes the Native Android platform, the iPhone Operating System (iOS) platform and the HyperText Markup Language 5 (HTML5) web platform. Table 3 below shows the criterion used to choose the appropriate platform. GCM support ranks the platforms

based on the amount of support available for client development. This is the most important criteria since the Web Server mainly communicates with the clients using the GCM service, making it essential that the client is able to process the message. The implementation criteria ranks the platforms based on the amount of time required to develop, members previous knowledge about developing for the platform and amount of developer support that is available. This criteria is important because there is a limited amount of time for development, and the application must be done on time. Accessibility ranks the platforms based on how many smartphones can use the mobile application. The more smartphones that the platform can support, the higher its rank. The last criteria, ranks the platforms on the equipment and software tools required to develop the application on the platform. The platforms will be ranked based on the amount of costs to develop the application.

Table 3: Criterion Table

Criteria:	**GCM Support**	**Implementation**	**Accessibility**	**Resource**
Weight:	40%	25%	10%	15%

Each development platform is compared based on previous knowledge. A score from 1 to 10 is given to each criteria for the 3 platforms. A weighted sum is calculated to determine the appropriate platform to develop the application.

GCM Support: Both Android and iOS platforms have GCM client APIs developed by Google, and there are detailed steps in how to create a GCM client [11]. HTML5 based apps do not have GCM client APIs, but there are 3rd party support available for development, giving the platform a score of 5.

Implementation: The members have the most experience with Android platform, and have little to no experience with the others. Also, both Google and Apple have released various Software Development Kit (SDK)s and documentation for developing Android and iOS applications respectively. The members do not have any experience in developing for HTML5 web applications, nor is there are any SDKs or documentation. However, all 3 platforms have various support online, from forums, blogs and repositories all having sample code and logic that can be very helpful in development. Considering this, a final score of 8 for Android, a score of 6 for iOS and a score of 4 for HTML5 is given.

Accessibility: The most accessible application to develop would be the HTML5 application. This is because all mobile devices can use HTML5 applications through a web browser. However, both Android and iOS have their own application distribution (Google PlayStore and Apple app store). 2014 market share of mobile devices shows that 80.7% of smartphones run the Android OS while 15.7% of smartphones run iOS [14]. Therefore a score of 10 for HTML5, a score of 8 for android and a score of 1.6 for iOS is given based on how many percentage of smartphone users can be reached.

Resources: Developing the application for Android can be done using the Android Studio Integrated Development Environment (IDE) and can be done on various computers [15]. There are no specific IDEs to develop for the HTML5 platform, but web applications can be developed using any text editor. However, developing iOS applications involve getting additional resources since they can only be developed on Apple computers. Therefore a score of 8, 6 and 4 is given to Android, iOS and HTML5 respectively.

Table 4: Decision Matrix

Criteria:	GCM Support	Implementation	Accessibility	Resource	TOTAL
Native Android	10	8	8	8	8
Native iOS	10	6	1.5	4	6.25
HTML Web App	5	4	10	6	4.9

The decision matrix shows that the best option to develop the mobile application is through the Android platform. The application implements the GCM client side of the system to communicate with the GCM Application Server running on the embedded system. The following subsections will focus on the component design for the Android application.

3.4.1 Real Time Data

This component of the application provides data from the brew system to the user's device. There are main two methods that the application provides data to the user. The first method is to display the current state of the brew in system through push notifications. Whenever a change is detected in the vessel such as temperature change, process change or any other state changes there might be, there would be a notification on the client's device to let the user know. The second method of the component is to provide interface that provides data statistics based on sensor data recorded by previous brews. This section will focus on the design for both methods. Figure 26 demonstrates the design flow for the push notification service.

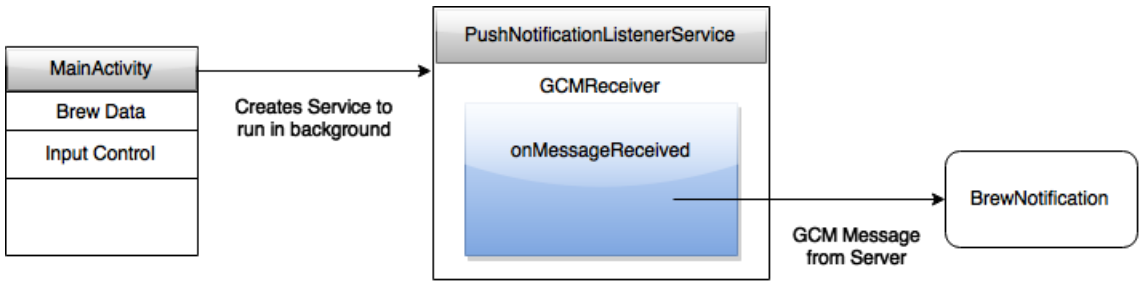

Figure 26: Flow Design for Notification Service

MainActivity: This activity starts on application launch (when user taps on the app icon). The activity will start the PushNotificationListenerService to run in the background.
PushNotificationListenerService: The Listener uses the GCMReceiver class to listen for and to receive the data messages from the server. Whenever a JSON message arrives to the client side, the onMessageReceived function is called to parse the JSON message.
onMessageReceived: This function parses the JSON message received by the Listener and creates a type of BrewNotification to display to the user.
BrewNotification: This object will encapsulate notification created for the end user. Types of BrewNotification are based on priority: max, medium and low. Max priority notification is given to messages that relay critical information about the brew system (overheat, pressure build up, etc). Medium priority notifications are created for important messages (brew changing state, brew completed). Low priority notifications are

given to messages that contain messages which the user does not necessarily have to know (boil temperature reached).
To summarize, once the server sends a message to the client application, the Listener service will process the message through the onMessageReceived function. Once the function parses which type of message it is, it displays the appropriate BrewNotification to the user.
The next part of the component displays previous data statistics recorded by the sensors on previous brews. The data displayed includes temperature readings, volume, sugar concentration and other sensor measurement. This gives the user the ability to look at past brews and possibly make adjustments. Figure 27 illustrates the application flow for the data statistics.

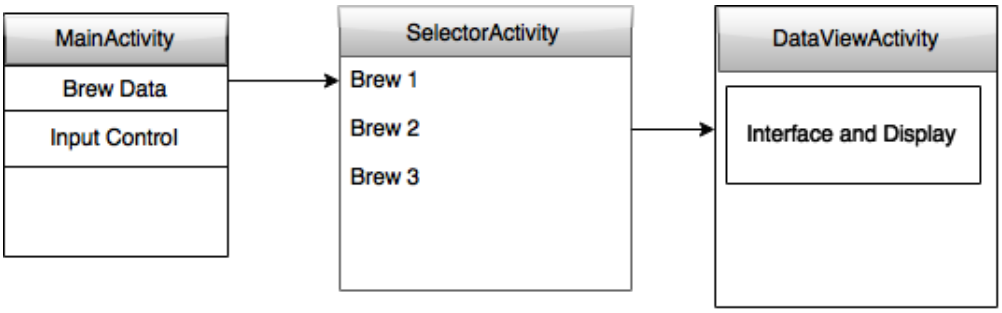

Figure 27: Flow Design for Data Statistics

MainActivity: The main activity (same from the previous component) allows the user to launch the activity to select previous brews.
SelectorActivity: This activity contains a selectable list of all previously done brews. By selecting one item launches the DataViewActivity based on the brew.
DataViewActivity: Displays various data (temperature, pressure, yield percentage) of the brew selected by the user. Providing this information allows the user to view and experiment with previous brews to enhance the quality of future brews.
In conclusion, the real time data component of the application is designed to keep the user notified about the brew system, whether that would be about the current state of the brew process or about previous brew data. This enhances user experience such that the user does not have to constantly check the brew process to receive information since the system informs the user.

3.4.2 Input Controls and Interaction

The second component of the application is to allow the user to control the brew system with the mobile application. The goals of this component is to allow the user to start the brew process through the application and to provide control over various conditions within the brew process such as the maximum temperature, the target sugar concentration and others that would affect the brew's quality. Figure 28 below shows the design flow for this component.

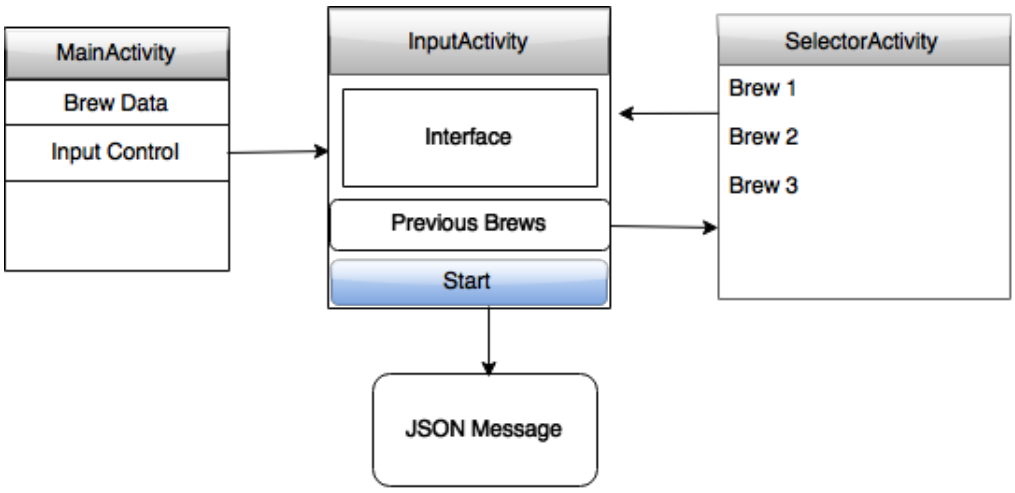

Figure 28: Flow Design for User Input Activity

MainActivity: Option to select the input control activity, the main activity first checks if the mobile device is on the same local network as the vessel. If the device is not on the same network, the MainActivity will not allow the user to select the "Input Controls" option. This is because the mobile application is designed to allow users to perform write applications only when connected on the same network as the web server, otherwise the application allows read-only operations. Although this will limit the application's usability, it ensures a high level of security on the brew system, where it does not allow others to exploit the application over the internet to cause the main vessel to malfunction.
InputActivity: The InputActivity allows the user to select a previous brew process to modify or create a brand new brew process. The user can change various states such as max temperature for boils, number of hours and other conditions that might affect the brew. The user is also able to start the brew from the interface, by tapping the START button snapped to the bottom of the interface.
SelectorActivity: If the user selects a previous brew, the activity retrieves the relevant data from the server, and populate the interface of the InputActivity.
If the user taps the START button in the InputActivity interface, the application creates a JSON file with all the data mapped to a key value. The client will send the JSON file to the server and wait for a response. An error response from the server will have another key explaining the details of the error. The errors are based on lack of ingredients in the vessel or a malfunction detected by the system. A success response from the server means that the system started the brew.

4 Discussion and Conclusions

4.1 Evaluation of Final Design

The final design meets the objective and specifications listed in the project specification document. The combination of the subsystems creates an automated brew system that can control the climate within the vessel. These controls are brought about by the inclusion of a temperature regulation which requires real-time temperature feedback to maintain precision. The functional requirements "Database" and "Application Notification" is implemented through a RESTFul web server with each functionality mapped to a Jersey

Resource by a URL. There is also an additional specification to create a user interface so that the brewing device may be operated via a mobile device.

In summary, the mash component of the brewing process and the trub removal component can be achieved by using the filter basket outlined in Section 3.1.3, Integration of Waste Removal System. The sparge requirement of the brewing process is accomplished by utilizing the no-sparge method outlined in Section 3.3.4. Temperature regulation and the creation of the wort is controlled by both the heating and cooling systems, their respective designs, and implementations which is shown in Section 3.2.1. The final requirements of the brewing process such as dispensing the grains, dispensing yeast, dispensing hops, aeration of the wort, and the kegging process is all achieved by actuating the appropriate valves. Since the implementation of these components is rather trivial, a general overview of the system is shown in the Block Diagram in Section 1.3. The Application Notifications and Logging Database are handled by the Computer System in Section 3.3.3 and allows for user interaction with the prototype device. Finally, the physical design of the conical fermenter and the feedback from control systems and sensors contribute to the success of the entire system; temperature accuracy, volume control, volume size, mobility, and sanitation is achieved.

Overall, each essential Functional Specification and Non-Functional Specification outlined in Sections 2.1 and 2.2 respectively have been achieved and the all grain brewing process can be successfully automated.

4.2 Use of Advanced Knowledge

The subsystem designs all require upper year knowledge to complete. For developing the communications portion of the embedded computer system SELinux policies enforcing read only access is a crucial security precaution. Proper security enforcement can minimize the potential for malicious attacks on the system as outlined in ECE 458, Computer Security. Developing the Web Server and the Mobile Application includes knowledge from such courses as ECE 356 (Database Systems), ECE 454 (Distributed Systems) and ECE 452 (Software Design and Architectures). Designing databases requires knowledge of various schemas, relations, normalization and indexing that are involved in the ECE 356 contents. Designing a mobile application involves the knowledge of software design processes, methods and implementing those designs that is taught in ECE 452. Finally, the brew system requires various sensor networks and communication between a centralized server to multiple clients which involves knowledge from ECE 454. The design and analysis of the control systems used in the electrical subsystems require knowledge of control systems that are taught in both ECE 380 (Analog Control Systems) and ECE 481 (Digital Control Systems), and the design and analysis of the power electronic components uses concepts taken directly from ECE 463 (Design & Applications of Power Electronic Converters).

4.3 Creativity, Novelty, Elegance

The novelty of our design is that it involves so many components from various Engineering backgrounds to complete. The system allows the user to have precise controls of the brew but at the same time offer flexibility in various environmental conditions during the process. One of the most elegant part of the project is that it takes the entire brew process and compacts it into one single vessel. The use of a double walled tank enables the heating and cooling apparatus to be mounted inside the device itself, adding to the singular design aspect of the fermenter.

4.4 Quality of Risk Assessment

The power electronic systems pose many hazards to the building of this system. The custom 120V circuitry is inherently dangerous to test, and proper precautions must be taken to avoid electrocution. The heating components of the system are capable of getting very hot, so proper care must be taken to avoid burns and potentially starting a fire. The motors can be very dangerous if anyone puts their hand near the spinning parts while the system is live, so it is important to make sure that everything is powered down and disconnected before attempting to adjust any of the motor actuated assemblies.

The risks were mitigated by taking precautions before using the system. The application of insulators, heat sinks and fans were required to avoid injury. One of the problems that was faced during the implementation was getting rid of the by-products created during the brewing process, known as the trub. This problem was solved by revising the mechanical design of the prototype and integrating a waste removal system as seen in Section 3.1.3.

4.5 Student Workload

Table 5 below shows the amount of work contributed by each group member.

Table 5: Project Contributions

Group Member	**Percentage of Workload**
Kevin Nause (Embedded Systems)	25%
Mathieu Tremblay (Electrical)	25%
Scott Wood (Mechanical)	25%
Steve Jung (Mobile Application)	25%

The workload was divided on the various skills of each group member. Each group member was assigned a major component of the project based on their area of interest and expertise.

Acronyms

AISI American Iron and Steel Institute.

API Application Program Interface.

ARM Acorn RISC Machine.

CFIA Canadian Food Inspection Agency (CFIA).

GCM Google Cloud Messaging.

HTML5 HyperText Markup Language 5.

I2C Inter-Integrated Circuit.

IDE Integrated Development Environment.

iOS iPhone Operating System.

JSON JavaScript Object Notation.

LAMP Lamp Apache MySQL PHP.

LTS Long Term Support.

PID Proportional-Integral-Derivative.

RAM Random Access Memory.

REST RESTful.

RISC Reduced Instruction Set Computing.

ROS Robot Operating System.

SDK Software Development Kit.

URL Uniform Resource Identifier.

USB Universal Serial Bus.

UTS Ultimate Tensile Strength.

WLAN Wireless Local Area Network.

XMPP Extensible Messaging and Presence Protocol.

Beer Terminology

Grain Bill The required amount of grains for the mash in order to reach the desired gravity necessary for the wort..

Gravity The density of the wort. This is often assumed to be the sugar density of the wort. The term OG refers to the original gravity of the wort. SG refers to the current specific gravity of the wort. FG refers to the final gravity upon the completion of fermentation. The difference between the FG and OG is often used to calculate the percentage of alcohol in the final product..

Grist The combination of milled grains to be used in a particular brew. Also sometimes applied to hops [16]..

Lauter To run the wort from the mash tun. From the German word to clarify. It uses a system of sharp rakes to achieve a very intensive extraction of malt sugars. [16]..

Mash (Verb) To release malt sugars by soaking the grains in water. (Noun) The resultant mixture [16]..

Pitch To add yeast..

Sparge To spray grist with hot water in order to remove soluble sugars (maltose). This takes place at the end of the mash [16]..

Trub The layer of sediment that appears at the bottom of the fermentation vessel upon the completion of fermentation..

Wort The solution of grain sugars strained from the mash tun [16]..

Technical Terminology

Austenite A non magnetic crystal structure of iron. May exist in solid solution with the aid of alloying elements..

Austenitic Stainless Steel An alloy family of stainless steels high in Austenite phase grain structure..

Inter-granular Corrosion Corrosion caused by varying composition between the grains and grain boundaries within a metal. Welding 304 stainless steel may lead to intergranular corrosion if not cooled properly..

Jersey Framework A REST API Framework written in the Java coding language..

Jersey Resource An object with a type associated data or a set of methods that operate on it..

Raspberry Pi 2 Model B The Raspberry Pi 2 Model B is the second generation Raspberry Pi. It replaced the original Raspberry Pi 1 Model B+ in February 2015..

Ultimate Tensile Strength The maximum tensile stress a material can experience before breaking or fracturing.

Yield Stress The stress at which a material begins to plastically deform..

References

[1] H. Suthar and J. Gadit, *Modelling and Analysis of the Simple Water Heater System.* Vadodara, India: International Journal of Electrical and Computer Engineering (IJECE), 2011.

[2] E. Oberg, *Machinery's Handbook 28th Edition.* South Norwalk, CT: Industrial Press Inc, 2008.

[3] MatWeb LLC, "Aisi type 304 stainless steel," 2000. http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=MQ304A. [Accessed 2015-07-01].

[4] A. Tepedelen, "History of homebrewing," 2013. http://allaboutbeer.com/article/power-to-the-people/. [Accessed 2015-05-19].

[5] Canadian Food Inspection Agency, "General principles of food hygiene, composition and labelling," 2014. http://www.inspection.gc.ca/food/non-federally-registered/safe-food-production/general-principles/eng/1352919343654/1352920880237?chap=3. [Accessed 2015-07-01].

[6] C. Matthews, *ASME Engineer's Data Book, 2nd Ed.* East Lansing, MI: ASME Press, 2001.

[7] M. Kutz, *Mechanical Engineer's Handbook, 3rd Ed. Book 1.* Mississauga, ON: Wiley, 2005.

[8] Chugger Pump, "Cpss-ci-1," 2015. http://www.chuggerpumps.com/home-brewing-pumps-chugger/chugger-pump-products/item/cpss-ci-1/. [Accessed 2016-02-10].

[9] Texas Instruments Inc., "Tmp102," 2015. http://www.ti.com/product/tmp102. [Accessed 2015-07-01].

[10] Open Source Robotics Foundation, "Measuring temperature," 2014. http://wiki.ros.org/rosserial_arduino/Tutorials/Measuring%20Temperature. [Accessed 2015-07-01].

[11] Google Inc., "Use google cloud messaging (gcm)," 2015. https://support.google.com/googleplay/android-developer/answer/2663268?hl=en. [Accessed 2015-07-01].

[12] J. Palmer, "Skip the sparge!," 2002. http://byo.com/malt/item/1375-skip-the-sparge. [Accessed 2015-12-03].

[13] Google Inc., "Gcmsender," 2014. https://github.com/googlesamples/google-services/blob/master/android/gcm/gcmsender/src/main/java/gcm/play/android/samples/com/gcmsender/GcmSender.java. [Accessed 2015-12-01].

[14] Gartner Inc, "Gartner says smartphone sales surpassed one billion units in 2014," 2015. http://www.gartner.com/newsroom/id/2996817. [Accessed 2015-07-01].

[15] Android Open Source Project, "Android studio," 2015. http://developer.android.com/sdk/index.html. [Accessed 2015-07-01].

[16] BeerAdvocate, "Beer & brewing terminology," 2015. http://www.beeradvocate.com/beer/101/terms/. [Accessed 2015-05-30].

Appendix A Prototype Hazard Disclosure

ECE498B: Prototype Hazard Disclosure Form Group number: 2016.019

Instructions: Answer all the following questions by putting an *X* in either the yes or no column. If unsure, answer yes. If you answer yes to any question, set up an appointment with the Lab Instructor to ensure your prototype is safe for the symposium. Include this completed form in your Final Report (Appendix A) even if you answer no to all questions.

Question: Does your prototype…	**yes**	**no**
1. include any circuitry that you designed or built by yourself?	X	
2. †include any circuitry that is not enclosed in an approved plastic or metal box?		X
3. †have a 120V AC power supply/adapter that is not approved by CSA, UL, ULC, or ESA?		X
4. †involve any other 120V AC circuitry/device that is not approved by CSA, UL, ULC, or ESA?	X	
5. †involve circuitry that is not connected to its power supply with a fuse and a switch?		X
6. †use high-capacity or high-density (e.g., lithium-ion or lead-acid) batteries?		X
7. †have exposed moving parts that may pinch, hit, or crush a person?		X
8. †have parts that are top-heavy and risk falling over?		X
9. have exposed sharp edges or points?		X
10. use strobe lights or unprotected lasers of any class?		X
11. involve projectiles or any part that can fly?		X
12. use high-pressure gases or liquids?	X	
13. emit dangerously loud sounds?		X
14. involve accessible components that reach temperatures above 40°C or below 0°C?	X	
15. emit non-trivial amounts of RF radiation?		X
16. involve irritating/dangerous chemicals, any biological materials, or any food/drink?	X	
17. involve x-rays or radioactive materials?		X
18. eject gas, particles, or fluids into the environment?	X	
19. involve any other hazard? Describe:		X

† *If you checked "yes" to any of these questions, you must contact the Lab Instructor by Feb 22 2016.*

Lab Instructor Inspection Report (needed only if "yes" is checked to any question above)

☐ Prototype is deemed to be unsafe in its current state and requires another inspection after changes are made.

☐ Prototype is deemed to be conditionally acceptable† for the symposium, and another inspection is not required.

☐ Prototype is deemed to be safe for the symposium in its current state.

†*State on the back of this form what minor changes are required for the prototype to be considered safe for the symposium.*

Lab Instructor Signature _____ Date _____

Project Consultant Inspection Report at Final Prototype Demonstration (needed for all projects)

yes	**no**	
☐	☐	The group appears to have accurately answered the above 19 questions.
☐	☐	The prototype presently appears to be safe for the public symposium.

Consultant Signature _____ Date _____

Disregard these sections for your Final Report.

Appendix B Symposium Floor Plan Request

ECE498B: Symposium Floor Plan Request Form Group number: 2016.019 .

Instructions: Answer all the following questions by putting an *X* in either the yes or no column. Provide additional information in the far right column as requested. Include the completed form in Appendix B of your Final Report, even if you answer no to all questions. Requests for equipment (e.g., an oscilloscope, a power supply, a large monitor, a computer, or a lab stool) should not be included on this form; instead, for such equipment requests use the on-line reservation system as described in the *Symposium Checklist and Schedule* document.

Question:	yes	no	if you answered "yes"…
1. Does your project require one or more hardwire internet connections at the symposium? **Note:** We provide Ethernet connections only, via a male RJ-45 connector. The connection comprises a static IP address in the uwaterloo.ca domain on a shared 100Mbps link. Do not count on the DC building wireless system being available for your project.	X		If yes, state how many connections you require: Two connections
2. Do you desire to use your own wireless router at the symposium? If yes, you will need to work with the university Information Systems & Technology (IST) people to get your setup approved well before the symposium date. Unapproved routers are strictly prohibited.	X		If yes, the course coordinator will contact you with information on what you need to do to get approval.
3. Each booth has a 7.5-amp 6-outlet power bar. Does your project require more than 7.5 amps of mains (120 V AC) power?	X		If yes, state how many amps you require in total: 15A
4. Each booth has a 7.5-amp 6-outlet power bar. Does your project require more than 6 outlets?		X	If yes, we will supply your booth with two power bars.
5. Do you intend to put any electronics/computers on the floor under your booth?		X	If yes, we will ensure you can do this safely.
6. Does your project involve projectiles or flying parts? (We have a large "cage" at the symposium for projects that involve projectiles or flying parts.)		X	If yes, state the nature of the projectile or flying part:
7. Does your project require special lighting conditions (e.g., dim light, bright light)? We will do our best to accommodate lighting requests, but no guarantees are made.		X	If yes, state the nature of the desired lighting conditions:
8. The default booth consists of a 5' wide by 2.5' deep table. The table is 2.54' tall. Does your project require extra table space (giving you a total table area of 7.5' wide by 2.5' deep)? We will do our best to accommodate space requests, subject to the urgency of the request and overall space and safety constraints.		X	If yes, explain specifically why you are requesting the extra table space:
9. The default floor space, including the table and area for you/visitors to stand, is around 6' by 6'. Does your project require extra floor space? We will do our best to accommodate space requests, subject to the urgency of the request and overall space and safety constraints.	X		If yes, explain why you need the extra floor space: The dimensions of the conical are 2' by 2' by 5' plus the space needed for connected accessories. State how much extra space you are requesting: Extra floor space to accommodate the conical, ideally an extra 3 feet in width.
10. Do you have any other special floor plan requests? Note: If you want an oscilloscope, a power supply, a large monitor, a computer, or a lab stool, read the instructions at the top of this form.	X		State your request: If possible a hose to supply water. Ideally, we would like to demonstrate temperature regulation on liquid within the tank.

Appendix C Mechanical Materials

This section contains important lookup values for mechanical and materials calculations.

Table 6: Alloying elements of AISI 304 Stainless Steel [2]

Alloying Element	Cr	Ni	C	Mn	Si	P	S	N
Wt. %	18-20	8-10.5	0.08*	2.0*	0.75*	0.045*	0.030*	0.10*

* Indicates maximum

Table 7: AISI Type 304 Stainless Steel Mechanical Properties [3]

Yield Stress	**UTS**	**Elasticity**	**Poisson Ratio**	**Hardness**	**Conductivity**	**Heat Capacity**
215 MPa	505 MPa	193 MPa	0.29	123	16.2 W/m-K	0.5 J/g°C

Table 8: Alloying elements of 5052 Aluminium [2]

Alloying Element	Cr	Cu	Fe	Mn	Mg	Zn	Si
Wt. %	0.15-0.35	0.1*	0.04*	0.1*	2.2 2.8	0.1*	0.25*

* Indicates maximum

Table 9: Type 5052 Aluminium Mechanical Properties [3]

Yield Stress	**UTS**	**Elasticity**	**Poisson Ratio**	**Hardness**	**Conductivity**	**Heat Capacity**
193 MPa	228 MPa	70.3 MPa	0.33	60	138 W/m-K	0.88 J/g°C