Automated Brew Extractor

Group 10

Robert Bower – Electrical Engineer
David Rodriguez – Computer Engineer
Alonzo Ubilla – Electrical/Mechanical Engineer
Kleber Valencia – Electrical Engineer

December 4, 2014

Table of Contents

1.0 Executive Summary

The proposed Senior Design project at hand is to effectively build, format, test and engineer an automated home brewing system tailored to the specific needs of the home brewer and his/her recipe(s). With no doubt in mind this is something that has already been invented and can be mass produced for profit with parts and/or materials that are much cheaper. Nevertheless the main goal is not to present a for-profit project, instead it is to bring a *tailored* finished product that can be freely manipulated by the home brewer for a more precise and genuine home tasting brew finish. The automated home brewing system, or A.B.E as it is to be known, will be comprised of a portable stable aluminum frame, along with the various sensors and circuits that are needed for the internal system, a central unit for data gathering and processing, three to four microcontrollers for ease with the internal process communications, and finally a web server available for the user of the Internet along with a very simple Android application for on-the-go monitoring.

There are many moving parts to this specific design, discussed in much detail throughout this paper, but the outcome should still be roughly 99.0% equal to as if it had been done with human supervision (the remaining 1% can be accounted for the human/machine errors in the process). With an automated brewing system, the user can not only be more effective with the specific brewing recipe but they can also aim for a much higher percentage in overall quality. Being able to control specific sensors (whether they be level or temperature sensors), start and end times, along with gathering and displaying this invaluable information makes this mechanically driven logical system a home brewer's dream come true. A home brewer is just that, a home brewer, meaning most times rather than not the brewer is taking this as a hobby or as a personal/family tradition; either way the time factor is a huge overhead for this process if done manually. Typically taking anywhere between four to six hours (this time does not include the fermentation period nor the bottling period which occurs post-fermentation). Making this practically a half-day to an all-day event, for some home brewers this would not be a problem, for this case, the group is looking to simplify the lives of the other home brewers who don't want to manually spend this much time every couple of months/weeks for a home brewed beer.

Creating the solution to this time management versus home brewing quality problem is at the top of this project's objectives. Therefore this objective, along with demonstrating this project's engineering capabilities, will be seen all throughout this report and of course in the live presentation of this one of a kind automated brewing system. With such a project, spread out over several months there will be times when things won't go according to plan, either with the manufacturers' parts or just with the design/test phases of this project. the main goal as a group regarding this matter will be to stay very close to the proposed: design specifications, testing requirements, build schematics, circuit design

layouts, software schemas, etc. that are documented in the following pages of this document. The build time for this specific project is estimated to be around ten to twelve weeks, this time includes debugging and time needed for modifications and/or issues that typically arise during a project of this magnitude and scope. The finished product will in fact produce drinkable beer, but due to regulatory laws this part of the project will not be showcased in the Senior Design Showcase in April 2015, unless given special permission by the required University of Central Florida department.

2.0 Automated Brew Extractor Description

This section will cover the description of the Automated Brew Extractor as well as discuss the motivation, goals, objectives, requirements, and specifications. As students who enjoy making beer, the group identified several features that they would like to see included in the final design.

2.1 Motivation and Goals

Making beer is an enjoyable and rewarding experience. This is science people can eat. Whenever one sets out to make beer, the first thing they must do is acquire the equipment necessary to accomplish this task. The equipment can range in price from about 100 to 1000 thousand dollars depending on both the quantity one wishes to make, and the type of process one wants to go through. After one has acquired the equipment, the actual brewing process itself requires attention and can be very time consuming. The set up process alone, takes around 2 hours and makes a mess if they choose to brew inside their home. The actual time spent making the wort that will be fermented, takes significantly less time than the set up and cleanup process. One of the main motivating factors for this design project was the significant amount of preparation time required. The group plans to significantly reduce, if not eliminate the process of setting up the equipment. Aside from the set up and clean up time, the equipment required to transfer liquids into different containers is a bit of a hassle to maintain and store. The brewer is required to have an automatic syphon, and lots of rubber tubing to transfer the liquid into different containers. The brewer must also have tools to securely connect and disconnect hoses to the containers they plan to use. The complexity and hassle of liquid transfer was another main motivating factor for the design project. The group plans to completely eliminate the need for equipment to transfer liquid from container to container. This will reduce the time required to set up the transfer equipment as well as help prevent the introduction of contaminants to the brew system, which is always a major concern when preparing substances that will be ingested by humans. Now looking closely at the actual brewing process, most home brewers use their kitchen stove as their source of heat when making home brewed beer. Even though this is probably the easiest way to heat things at home safely, there is no accurate way to control the temperature of the stove. Having said that, one can simply measure the temperature and adjust the power

setting of the stove to either raise or lower the temperature during the process, but this requires constant attention and is very inaccurate. Home brewers can also invest in temperature controllers and separate heating apparatuses, but these can be costly and require lots of equipment. The lack of a reliable, controlled heating process, and need for equipment to accomplish this goal was another huge motivating factor for the design project. The group wants to completely automate the heating process using PID control as well as have this information made available to the user. Another key step in the brewing process that motivated the design project, was timing. Home brewers must accurately track time using egg timers or something similar. There are systems in place that track the process for you, however, most home brewers opt to monitor the system constantly. This motivated us to design the system with built in timing control. Another major goal for the design project was to have the system time each step in the process on its own, and prepare itself for the next step so that the home brewer can focus on more important aspects of the process. On the next page Table 2.1 shows a breakdown of all the project motivations and goals.

Table 2.1: Motivations and Related Goals

Motivation	Goal
Time Consuming Set up / Cleanup process	**Eliminate the need for equipment setup by having one piece of equipment to handle the brewing process. Incorporate a self-cleaning cycle to eliminate the need for lengthy and messy cleaning process.**
Need for equipment to transfer liquids between tanks during the process.	**Eliminate the home brewer's involvement in liquid transfers by automating the liquid transfer process.**
Inaccurate or Unreliable temperature control or the need to purchase equipment for temperature control.	**Incorporate accurate temperature measurement as well as temperature control through the use of PID and heating elements.**
Need for automated timing and control	**Program the microcontrollers to time each step in the process and respond accordingly.**

2.2 Project Objectives

In this section the group will outline the objectives for the Automated Brew Extractor. The main objective of the design project is to automate the process of extracting sugar from barley to allow the home brewer to focus on the data, rather than the process. This main objective can be broken down into several small objectives. The objectives and there sub categories are listed below.

1. Design a counter-top system that can be used for the sugar extraction process.
    a. Make a frame that can hold all of the compartments needed for the sugar extraction process.
    b. Design the frame so that it can be moved easily, and also allow for the transfer of liquids without additional tools.
    c. Design the frame to hold a display unit to allow the user to monitor all of the process data.
2. Automate liquid transfer
    a. Configure the setup of the system to permit gravity fed flow in order to reduce loss of product in the transfer lines.
    b. Use solenoid valves to automatically control water fed into the system as well as the path the product takes through the system.
    c. Use microcontrollers to monitor and control the liquid transfer process.
    d. Use microcontrollers to monitor the liquid levels of each compartment.
3. Automate the heating process
    a. Use microcontrollers and PID algorithms to monitor and control the heating process of the system.
    b. Use a built in heating element to achieve specific temperatures throughout designated by the user and eliminate the need for an outside heat source.
    c. Incorporate a system that will monitor temperatures and verify that they are within the safe limit. This will prevent personal injury and property damage.
4. Automate the Timing of the process
    a. Use microcontrollers to monitor the timing of each step and react as needed, depending on what stage of the process the system is currently in.
    b. Set triggers to activate steps in the process.
5. Incorporate a user interface for the system
    a. Design an easy to use integrated user interface that can send recipe information to the system.
    b. Design an interface that allows the user to monitor all process information in real time.
6. Make all process data available to the user
    a. Log all data
    b. Make the data available via a web server.
    c. Make the data easy to understand

2.3 Automated Brew Extractor Requirements

In order for the system to accomplish the goal of automating the process of extracting sugar from barley, the system must satisfy certain requirements. These are the requirements necessary for the design project to truly be an Automated Brew Extracting System. The list below illustrates the requirements that the group outlined for the project.

1. The user must be able to place the ingredients into the system, and let the controls handle the rest of the process.
2. The user must be able to input recipe specifications, and have the system respond appropriately to these specifications.
3. The system must produce a safe and consumable finished product.
4. The system must operate safely when not under supervision.
5. The system must control all heating, flow, and filling processes according to specifications set by the user.
6. The system must have an interface that allows the user to monitor all system related information.
7. The system must log all process information, so that the information can be reviewed on a later date.
8. The system must be portable and easy to use.

2.4 Automated Brew Extractor Specifications

2.4.1 The Brew Kettle

This section will cover the specifications for the brew kettle that the group decided to implement in the design. The brew kettle will be used in all heating processes for the system. The Brew kettle is used initially to heat the water, then again to boil the sugar water extracted from the barley. This sugar water must be boiled in order to make sugar that can be used by the yeast during fermentation. Below is list of the specifications and equipment connected to the brew kettle.

- 1 - 7.57 liter brew kettle
- 1 - 2000 watt heating element
- 1 - temperature probe
- 2 - ¼” NPT female threaded outlets
- 2 - ¼” NPT manual ball valves
- 2 - ¼” NPT 24V solenoid valves

- 2 - pressure sensors

2.4.2 The Mash Tun

This section will cover the specifications for the mash tun that the group decided to implement in the design. The mash tun holds all of the grain during the sugar extraction process. The grain is put into the mash tun by the user before the process begins. A pump connected to the brew kettle sends 70 degree Celsius water into the mash tun and sits for 1 hour. Below is a list of specifications and equipment connected to the mash tun.

- 1 – 7.57 liter insulated cooler
- 1 – temperature probe
- 2 – ¼" NPT 304 SS female threaded outlets
- 2 – ¼" NPT 304 SS manual ball valves
- 1 – Grain Bed
- 2 – pressure sensors

2.4.3 The Cooling System

The section will cover the specifications for the cooling system that the group chose to implement in the design. The cooling system extracts heat from the wort after the boiling process has taken place. This helps put the finished product at a safe temperature so that the yeast can be added to the wort. Below is a list of the specifications and equipment connected to the cooling system.

- 1 – 10 gallon insulated cooler
- 2 – ½" 304 SS female threaded outlets
- 1 – Plate Chiller
- 1 – 7.57 liter per minute pond pump
- 1 – temperature probe
- 2 – ½" 304 SS ball valves

2.4.4 The Chugger Pump

The section will cover the specifications for the wort pump that the group decided to implement in the design. The wort pump will be used to transfer all liquids throughout the system, excluding the liquid be transferred through the plate chiller on the cold side. The specifications of the wort pump are listed below.

- In line 304 SS pump head
- 2 – ½” NPT male threaded inlets
- Maximum head of 4.1 meters
- Maximum flow of 22.7 liters per minute
- 1/20 HP
- 115 V 50/60 Hz
- Non – Submersible
- 1.7 Full Load Amps
- Maximum Liquid Temperature through pump 121 degrees Celsius
- FDA Food Compliant
- UL Recognized

2.4.5 The Control Unit

This section will cover the specifications of the control unit that the group will be implementing in the design. The control unit monitors all system processes and controls the flow of the product through the system. Below is a list of the specifications for the control system.

- 2 – Atmel ATMEGA328 8-bit microcontrollers
- 12 – relay outputs
- 1 – solid state relay output
- 2 – programming ports
- 1 – 5V power input
- 1 – 5V power output
- 1 – 3.3V power output
- 1 – Two Wire Interface Port
- Communications via TWI (I^2C)
- On board logic level shifter

2.4.6 The Interface

This section will cover the specifications for the user interface that the group are implementing in the design. The interface will take in information from the user and send that information to the control unit. Below is a list of specifications for the interface.

- 1 – Embedded Computer with Linux Operating System
- 1 – 19” HP flat panel computer monitor

- 1 – keyboard
- 1 – mouse
- Wi-Fi connectivity
- Two Wire Interface to the control unit
- 12 - general purpose input/output ports
- Data logging capabilities

3.0 Research Related to Automation and Home Brewing

When Prohibition was passed in 1920, Americans began brewing their own alcoholic beverages at home. This may not have been the beginning of home brewing, but Prohibition certainly pushed Americans find other means to enjoy the beverages they love. When Prohibition ended in 1933, the beer market was tightly regulated forcing most small time brewers or start-ups, out of the market altogether. In 1979 Jimmy Carter deregulated the beer market and made it possible for small time breweries to gain a foothold in the American beer industry. Figure 3.1 illustrates the growth of microbreweries in the United States since 1979.

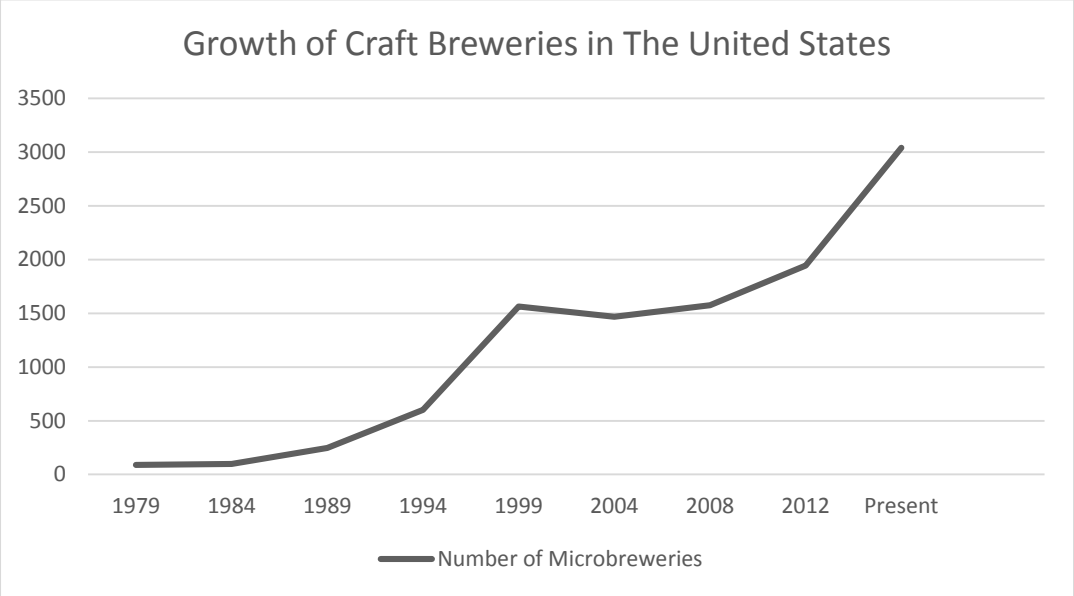


Figure 3.1: *Growth of Craft Breweries in the United States*

The growth of craft breweries in the United States brought along with it, a market for home brewing products. In the sections to follow there will be an analysis of the existing systems that inspired the automated brewing system that the group is designing.

3.1 Existing Similar Projects and Systems

There are several home brewing systems that one can purchase when looking to get started. They range in complexity and price. There are three systems in particular that the group reviewed during the research. These systems were chosen because they contained similar design specific features that the group is trying to capture in the project. These systems are the “Brew Boss”, “SYNEK”, and the “Pico Brew.”

3.1.1 “The Brew Boss”

One of the systems the group came across during the research was “The Brew Boss.” This system was one of the simpler designs that the group found. “The Brew Boss” uses a 7” Android tablet to communicate via Wi-Fi with the system. The tablet allows the user to control temperature and timing during the brewing process. Alongside control, the tablet allows the user to monitor temperature and timing variables while the brewing process is taking place. Aside from control, “The Brew Boss” allows the user to do all-grain batches as well as extract brewing. The system comes in a 120V or 240V version. The image on the next page shows “The Brew Boss” system. “The Brew Boss” contains many of the features that the group is looking to capture in the design. This system has connectivity and control which gives the user all of the necessary information that one would want during the brewing process. Below is an image and block diagram of the system. Figure 3.2 on the next page show the actual system and a block diagram breakdown.

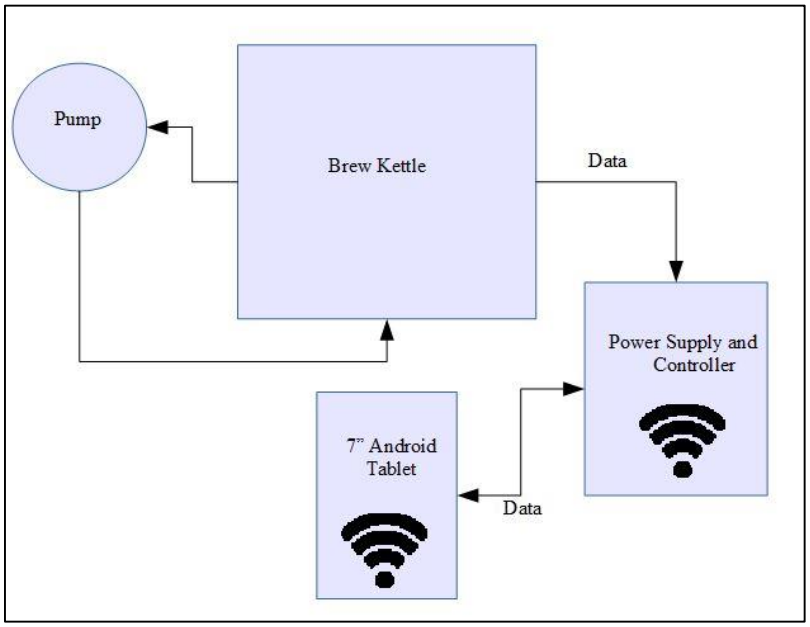


Figure 3.2: *Block Diagram of Brew Boss brewing system.*

3.1.2 “SYNEK”

The next system the group discovered during the research is called “SYNEK.” This is the first crowd funded draft beer dispensing system that has several backers on Kickstarter. This system is more of a preservation device than a brewing device. However, “SYNEK” utilizes temperature and pressure control to keep beer fresh. Although this strays away from the actual brewing process, the controls behind preserving the beer were of interest to us. “SYNEK” uses digital temperature control and a cartridge system to allow the user to obtain beer from any bar, pub, or restaurant, and bring it back to their very own home. The temperature controlled box squeezes the cartridge to keep the beer at pressure which greatly extends the shelf life of the beer. Figure 3.3 and 3.4 show the SYNEK system.

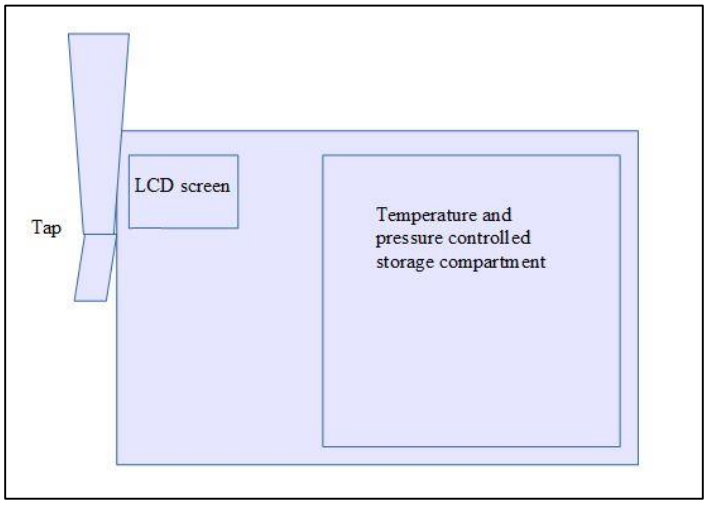


Figure 3.3: *Block diagram of SYNEK system*

Figure 3.4: *SYNEK beer preservation system.*
(Source: SYNEKsystem.com)

3.1.3 “Pico Brew”

The “Pico Brew” is the exact design that the group is trying to achieve. This system is an all in one brewing system and beer dispenser. The “Pico Brew” is a suit case sized stainless steel box with a polycarbonate drawer that pulls out from the front of the box. Inside of this polycarbonate drawer, the user can put grain, hops, and other flavor additives that they wish to add to the beer. Once all of the ingredients have been deposited inside of the drawer, the user sets all of the recipe parameters through an OLED screen that is attached to the system. Once all of the recipe parameters are set, the “Pico Brew” initiates the brewing cycle and controls all of the user specified inputs. This system closely monitors temperature, liquid level, and time. Aside from controlling the user specified parameters, the “Pico Brew” system is also fully automated in terms of liquid transfer. The beer starts in the main reservoir inside of the polycarbonate drawer, and ends up inside of the fermenter at the appropriate temperature for pitching yeast. All of the liquid transfers occur inside of the steel box. The “Pico Brew” perfectly captures the goal that the group has in mind for the system. Since the 3 systems that this section covered, vary greatly in design and purpose, Table 3.0 shows a break-down of the features of each unit. The tables will include the plans for the unit in red for comparison purposes. Figures 3.5 and 3.6 show the PicoBrew system and block diagram of the system configuration.

Figure 3.5: *Pico Brew craft brewing system*
(Courtesy of PicoBrew)

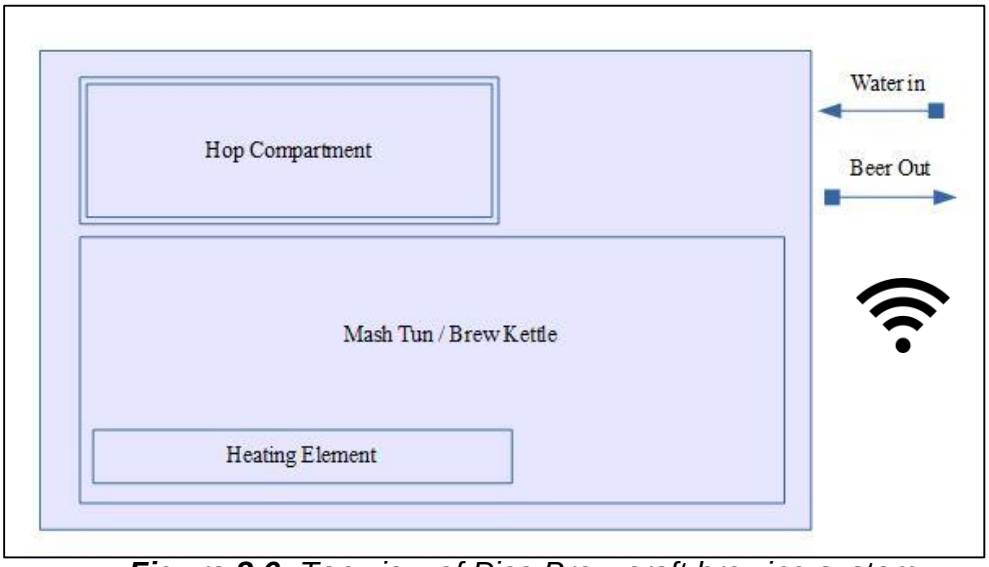


Figure 3.6: *Top view of Pico Brew craft brewing system.*

Table 3.0: Systems Comparison					
Name	**Temperature control**	**Liquid Level control**	**Flow Control**	**Automated Ingredient Addition**	**Connectivity (Wi-Fi)**
Brew Boss	Yes	No	No	No	Yes
SYNEK	Yes	No	Yes	No	No
Pico Brew	Yes	Yes	Yes	Yes	Yes
Automated Brew Extractor	Yes	Yes	Yes	Yes	Yes

3.2 Relevant Technologies

While examining the existing designs, there were several things that the group came across that would have been advantageous to include in the design. The main goal is automating the process of extracting sugar from barley so that it may be used to produce alcohol. In order to achieve this goal this project will be using many of the technologies found in the 3 systems examined in the previous section.

The most important aspect of extracting sugar from barley would be temperature control. The process must hit the perfect temperature in order to extract the maximum amount of sugar from the barely without burning off too many sugars. The “Brew Boss” as well the “Pico Brew” both use a heating element and computer control to maximize the accuracy of the temperature during the brewing process.

It was not specified in the design how exactly they achieved this goal, however, PID seems to be the most efficient means of controlling temperature throughout these processes. Since the group intends to use a heating element and PID control in the design, this will be discussed in detail throughout the design portion of this document. Aside from heating the liquid, removing heat from the liquid is also an important part of the brewing process. The "SYNEK" system uses some form of heat exchange, similar to that of a small refrigerator, to keep the temperature of the beer in a desirable range which increases the shelf life of the product substantially. For the design project purposes, the group will be using a counter flow plate chilling system to remove the heat from the finished product. The speed at which the heat is removed, dictates how well particulates, left over from the heating process, coagulate and precipitate from the finished product. This increases clarity and helps to remove unwanted flavors from the beer. Table 3.1 shown below compares the technologies related to heating discussed above. The group will include the technologies it intended to be used, in red, for purposes of comparison.

Table 3.1: Heating and cooling system comparison

System Name	Heating System	Cooling System
Brew Boss	Heating Element	None
SYNEK	None	Refrigeration Unit
Pico Brew	Heating Element	Refrigeration Unit
Automated Brew Extractor	Heating Element	Counter Flow Plate Chiller

Since the goal is to fully automate the sugar extraction and breakdown process, flow control is another "must have" for the system. The only system discussed previously in this section that utilizes flow control is the "Pico Brew." Transferring liquids from place to place is necessary to treat the liquid during and after sugar extraction. There are several ways to achieve the goal of transferring liquids. Some of the most popular methods are gravity fed systems. This involves transferring liquid by water pressure from the city, to a tank that sits high above the rest of the system. Then by opening and closing valves, gravity can be used to allow the liquid to move down the system to the various places that it needs to go during the process. Another method is pump control. Using pumps, the liquid can be transferred throughout the system depending on what stage of the process is taking place. The "Pico Brew" uses a reservoir that is filled from the tap to introduce liquid into the system. Since the design of the system is proprietary, it was hard to discern how exactly the "Pico Brew" transfers liquid from place to place. Through observation of the system, it would appear that the "Pico Brew"

uses a combination of automatic valves and pump control. The group intends to use pump control, gravity feed, and automatic valves to direct the flow of liquid through the system. This will be discussed in detail throughout the design portion of this document. Table 3.2 below illustrates a breakdown of the flow control system. The group will include the plans for the design project in red for purposes of comparison.

Table 3.2: Flow control comparison			
System	**Gravity Fed**	**Pump Fed**	**Automatic valves**
Brew Boss	No	No	No
SYNEK	No	No	No
Pico Brew	Yes (Speculated)	Yes (Speculated)	Yes (Speculated)
Automated Brew Extractor	Yes	Yes	Yes

Another important aspect of the designs that were reviewed was the interface. All three systems featured some method of data input and monitoring. The “Brew Boss” featured a 7” Android tablet which allows the user to monitor all of the heating processes taking place. The tablet could also be used to input data for the recipe settings, which is achieved via Wi-Fi communication. For the “SYNEK” system, the only interface available is an LCD screen that shows temperature readout and settings. This allows the user to monitor the temperature of system at all times and make adjustments accordingly. This is probably perfect for what the “SYNEK” system is because this is mainly a preservation device. The “Pico Brew” utilizes an OLED screen which gives the user much higher resolution when trying to view process variables and settings. The “Pico Brew” also allows the user to monitor data via their PC, tablet, or phone through a server that is supported by the “Pico Brew” website. For the system, it is intended to use a PC to input data and also have that data available through some kind of data logging mechanism. This will be discussed in more detail throughout the design portion of this document. Table 3.3 below illustrates the interface options for each of the systems discussed. The plans will be included for the design project in red, for comparison purposes.

Table 3.3: Interface comparison	
System	**Interface Type**
Brew Boss	7” Tablet with Wi-Fi Communication
SYNEK	LCD Screen
Pico Brew	PC/Tablet/Phone and OLED Screen

Automated Brew Extractor	PC (Raspberry Pi Computer)

3.3 Strategic Components

In this section there will be a discussion of some of the strategic components that will be used in the system. All of the components were chosen with the goal of implementing the requirements. The first component that will be examined is the Raspberry Pi computer. This computer was chosen to serve as the interface and I^2C master. Although there are several comparable systems that could accomplish the same goal, the online development community behind the Raspberry Pi is the best in its class. This made several websites available as resources throughout the design stage of this project. Since the system needs an interface that allows the user to input specifications and monitor data, the Raspberry pi seemed like a great starting point. This computer allows us to program in the Python language and generate a graphical user interface, to receive data from the user. The graphical user interface will be discussed in detail throughout the design portion of this document. The Raspberry Pi also contains general purpose input and output pins that are also directly accessible through the use of the Python language. These pins can be used as digital inputs and outputs or, drive an I^2C bus. The Raspberry Pi comes equipped with an I^2C module that allows the computer to drive any I^2C device connected to the bus. Since the ability to move data from the interface, to the microcontrollers is required, this was an attractive feature for this project. The I^2C bus configuration is discussed in detail throughout the design portion of this document. The Raspberry Pi is the most critical component in allowing all of the subsystems to work together. It is able to serve as a user interface, data transmitter, data receiver, and data logger.

The components chosen to help us accomplish of goal of automating the brewing process were the microcontrollers. The system uses two ATMEGA328P microcontrollers to manage all of the processes as well as relay information back to the raspberry pi. Specific details about this configuration will be discussed throughout the design portion of this document. The use of two microcontrollers allows us to divide the work required by the system into two separate jobs. This allows us to make the programming for the microcontrollers simpler as well. Instead of combining all of the code on one chip, the group will divide the code into two smaller, more manageable code sections. This also helps us reduce the time it takes to debug certain functionality should a problem arise during testing or operation.

3.4 Possible Architectures

While researching for this project the team came up with several system configurations. Diagrams and descriptions of the possible system configurations can be seen below. Each system description and diagram was considered as a potential configuration. The finalized system configuration is shown in the design portion of this document. The group will take a look at each of the possible configurations and discuss the advantages and disadvantages of each. The first configuration the team will examine is a side-by-side brewing system. In this configuration all of the equipment is located at the same level. Liquids can then be moved from one side of the set-up to the other side via pumps and valves. Figure 3.7 shows the side by side configuration that was initially considered for the design.

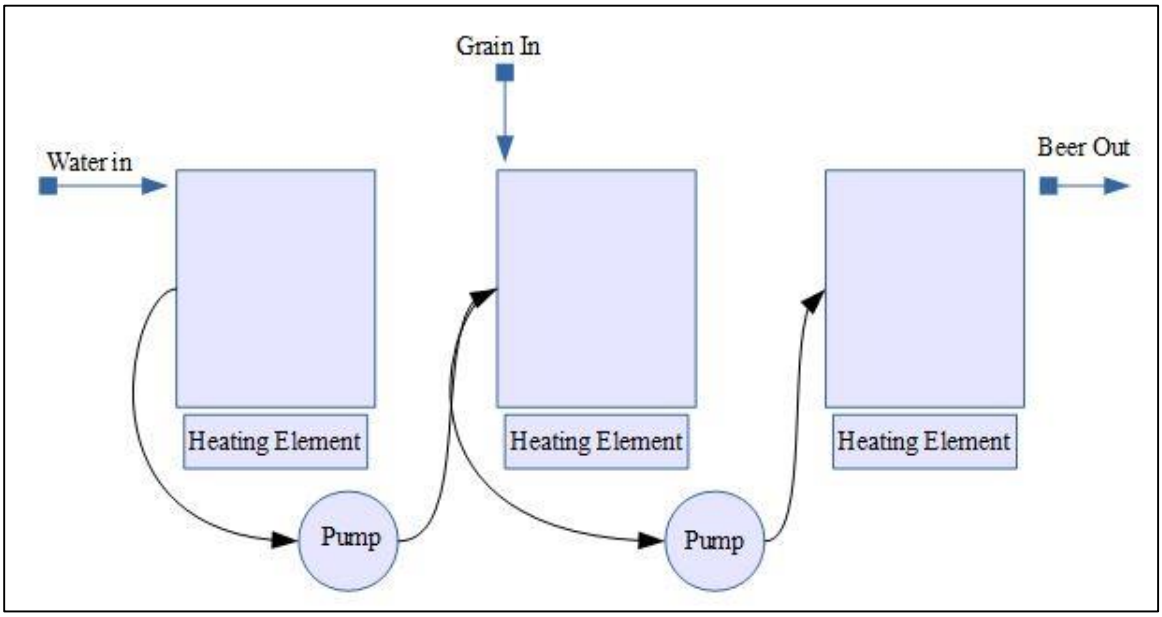


Figure 3.7: *Side by Side brewing system.*

Ultimately the team decided to not use this configuration for several reasons. Firstly, loss is a big issue in a side by side configuration. Without the use of self-priming pumps, the lines used to transfer liquid from one side of the system to the next, will be full of the product and will ultimately be lost by the end of the brewing cycle. This is because liquid pumps cannot pump liquid once air is in the line. Another major disadvantage of this system was the heating element arrangement. Some configurations use electric burners and others use propane heating systems. While propane is probably the most efficient, the equipment is expense, which is why the team opted for the heating element configuration. The side by side configuration also requires the use of two transfer pumps to move the liquid from one side of system to the other. The pumps needed to move high temperature liquid, and still keep that liquid safe for human consumption, are expensive as well. The side by side configuration also takes up a lot of space horizontally. This is not desirable unless the user has a lot of space free in his or

her home. Ideally the team wanted something compact that could easily find a place in the user's home. There are some advantages to using this configuration however. The fact that all of the brewing apparatuses are on the same level, make it easy for the user to see what is going on inside of the system. This also gives the user the ability to add ingredients easily if necessary. From an automation point of view, the side by side configuration offers no real advantage. The next configuration the team examined was the self-contained system. This system is much like the Pico Brew discussed previously in this document. All of the systems and processes required to make the beer would occur inside of one "box." Although this would be the most ideal configuration, the manufacturing of these device would be too expensive for the budget. Figure 3.8 shows the self-contained brewing configuration.

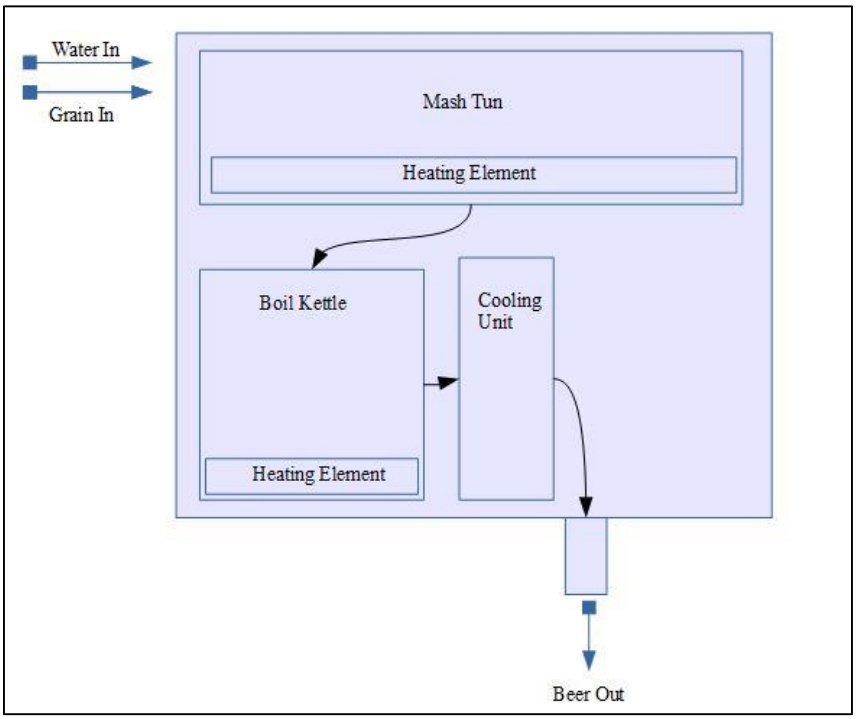


Figure 3.8: *Self Contained Brewing Configuration*

4.0 Brew Extractor Hardware and Software Design Detail

This specific section will cover the integration details of the hardware and software aspects of the automated system. It will go into the specifics of each individual subsystem, such as the structure design, heating and cooling systems, communications, analog data acquisitions, system power supply, user interface and functionalities.

4.1 Initial Architectures and Related Diagrams

When beginning the project the group had several ideas of how they wanted to achieve the goal of automating the sugar extraction process. Throughout the research stage the group came across several great designs that influenced how they would achieve that goal. This section will cover the initial designs for the project that group came up with after the research stage. Initially the group decided to configure a 22 Liter brewing system. Table 4.0 shows the components required to operate a 22 liter brewing system. The group eventually decided that the cost of implementing a 22 liter brewing system for the design project would be too costly mainly because of issues that occurred during the heat exchange process. This issue is discussed in detail in the upcoming sections. Not only were there challenges during the heat exchange design, this configuration took up a lot of space. This may not be convenient for someone who does not have a lot of space. This also makes the system incredibly time consuming to clean, which is something the group wanted to avoid in the end. Table 4.1 shows the configuration for the 22 Liter brewing system.

Table 4.1: Configuration of 22 Liter System

Equipment Needed	Quantity Needed
37 Liter Brew Kettle	1
37 Liter Mash Tun	1
22 Liter Glass Jug (Carboy)	1
Plate Chiller	1
Stainless Steel Head Pump	1
½" Stainless Steel Solenoid Valves	8
½" Copper Tubing	10 Meters
70 Liter Cooling Reservoir	1
20 Liter Sparge Tank	1

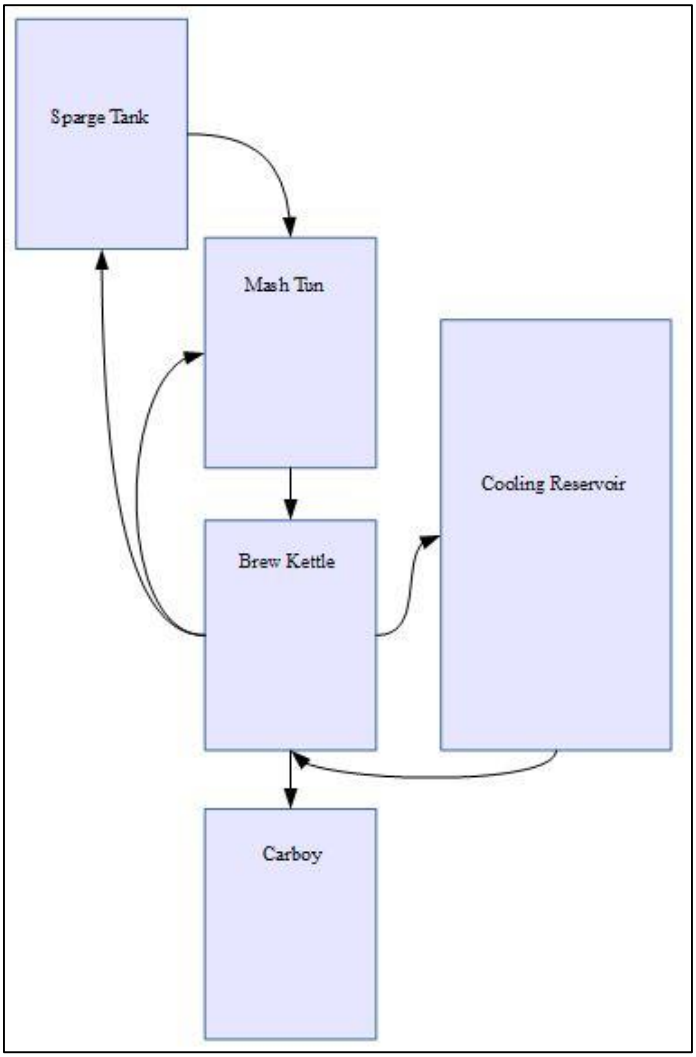


Figure 4.1: *The 22 Liter Brewing System Configuration*

4.2 Brew Extractor Support Structure

In this section, the design process for creating the Brew Extractor's support structure is reviewed in detail. Some of the key design requirements for the structure were a strong, lightweight, low profile, modular structure.

Much like a great suspension bridge, a masterfully designed sky scraper, or even a chair in a living room, a strong and properly designed support structure is necessary in order to ensure safe and reliable operation. The maximum loads of each component in this system are not entirely large, and most common building materials would suffice in terms of basic strength and vibration resistance; however, the temperatures achieved during device operation are an important factor in material choice. During the wort "boil" cycle, the kettle has the potential to reach temperatures of 100 ºC in the form of boiling water. The boiling water will not only conduct its high temperatures through the kettle and into the structure, but

the rising water vapors will also condense onto the structure, requiring the structure to be resistant to corrosion. In Table 4.2 below, different possible building materials are compared by considering some important properties. Each property is ranked on a scale between 1 and 5, 1 being undesirable and 5 being ideal.

Table 4.2: Support Structure Material Comparison

Material	Strength	Weight	Corrosion & Wear Resistance	Cost	Operating Temperature
Steel	5	2	1	5	5 (500 °C)
Stainless Steel	5	2	5	2	5 (760 °C)
Aluminum	4	4	4	4	5 (460 °C)
Wood	3	3	1	5	4 (300 °C)
PVC	2	4	5	5	1 (70 °C)

In order to choose a main structure material the material properties are weighted with a factor that dictates how important each is in relation to the others. For example, weight is not as large of a factor as corrosion\wear resistance, neither of which are as important as strength and operating temperature. The cost factor depends on how funds are acquired and allocated throughout the project. Table 4.3 shows the weights designated to each material property and Table 4.4 shows the weighted total scores of each material.

Table 4.3: Material Property Weights

Strength	Weight	Corrosion & Wear Resistance	Cost	Operating Temperature
5	3	4	3	5

Table 4.4: Material Weighted Scores

Steel	Stainless Steel	Aluminum	Wood	PVC
75	82	85	63	62

From the material comparisons done above, it is clear that Aluminum is the choice material for the main skeleton support structure. Aluminum is a largely versatile material and it is available in a huge range of sizes and shapes. Aluminum also has an advantage of being a softer metal, allowing it to be cut and drilled with ease.

The next structural hurdle to overcome would be the dimensional design of the stand including major part orientation. The process flow and full subsystem dimensions are the key factors in choosing the stand design. Ideally the entire structure will consume as little space as possible, as one goal of the design is for the system to be able to live on an average kitchen counter top. One major limiting factor for physical size constraints is the size of the cold water reservoir. The cold water reservoir will be too large to fit onto the countertop design; therefore, it will be omitted from the stand design and will simply be placed in a nearby location within range of the stand. The key sub-systems occupied by the stand are the sparge water tank, the mash tun, the kettle, and the carboy. Due to the gravity fed nature of most of the system, a 4-teir setup will be used. Other systems that will be incorporated into the stand but are not being incorporated in the stand design include pumps, solenoids, and tubing for flow. These are not being considered in the stand design because their sizes are currently unknown and it is assumed that their footprints will be small enough that adding them into the design will be an unchallenging task once the parts have been chosen.

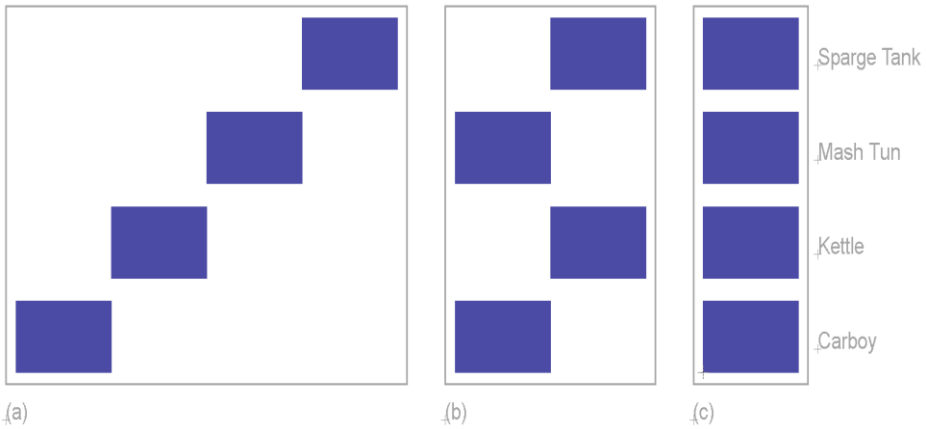
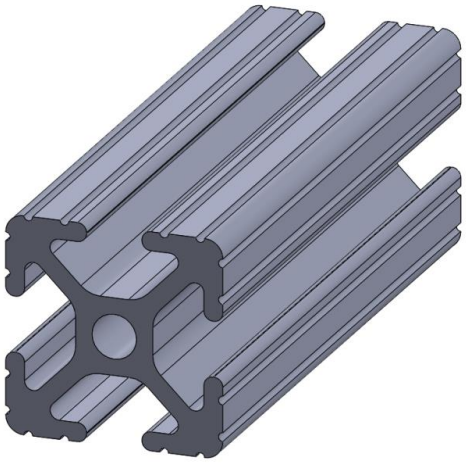


Figure 4.2: *Sub-system Configurations*

As stated earlier, total system footprint would ideally be as small as possible. Figure 4.2 above shows different generic 4-tier configurations. In configuration (a) the sparge tank. mash tun, kettle, and carboy are arranged in a ladder configuration where each stage of the brew process moves in one direction. This configuration would be the simplest to implement and allow for the most room for "extras" underneath the top 3 stages; however, this will also consume the largest area possible and is therefore not an entirely desirable configuration. Looking at configuration (c), a tower (completely vertical flow) would have an ideally small footprint and would also be the most aesthetically pleasing form that the system could take. Form (c) does come with a few negative aspects though, the form factor leads to a very top heavy design during the grain steeping cycle where the carboy and kettle are empty and the mash tun and sparge tank are filled water and grain. This is a dangerous scenario where the only solution is to increase the system footprint. Form (c) also leaves very little room to add the pumps and solenoids to the system as everything, in this stacked fashion, is within close proximity of each

other. Configuration (b) combines designs (a) and (c) into a hybrid configuration that can have both a small footprint, and a "stacked" design with open room for additional hardware.

Finally, assembly of the stand is to be considered. With aluminum, there are a huge number of options for assembly which include but are not limited to welding, fitted pieces, and hardware; each with its own appealing characteristics. Unfortunately not all of these options will fulfill the design requirements for the structure. Specifically, while welding a stand together would create a very strong structure, it will lose modularity due to the inherent permanence that is a weld. Hardware allows for the assembly and disassembly of the stand while maintaining a strong structure when designed properly. One option for a hardware assembled design would be to purchase raw aluminum tubing and drill holes for hardware. This is a viable option and allows for a lot of creativity in shaping of the structure. Another option is to use extruded slotted aluminum profiles known in the industry as 80/20. These extruded profiles allow for the construction of a fully modular stand with strong joints which is also modifiable without the need of a machine shop full of tools. The 80/20 1010 series aluminum profile is shown below in Figure 4.3(a).



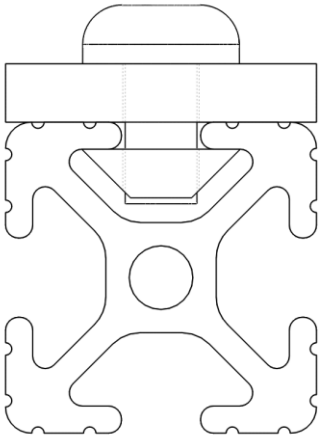


Figure 4.3: (a) *080/20-1010 Series Profile.* ***(b)*** *80/20 Fastener.*

Utilizing profiled aluminum bars like this allows ease of assembly, modification, and without major tooling required. In Figure 4.3(b) the standard assembly fastener is shown. The trapezoidal figure inside the slot of the aluminum profile is a block with a threaded hole in the center of it, to attach objects to the profile, the block is simply slid to the desired location and a bolt clamps the object being fastened to the outside of the profile. Figure 4.4 shows a render of one possible stand configuration using the 80/20 profiled aluminum bars. A bill of materials required to assemble this style of stand is shown in Table 4.5.

Figure 4.4: *Configuration (b) using 80/20 extruded aluminum profiles.*

Table 4.5: Stand Bill of Materials			
Item	**Quantity**	**Part Number**	**Unit Cost**
80/20 Aluminum - 10 series, 97" length	3	1010-97	$31.30
80/20 10 Series, Fasteners (25-pack)	1	3393-25pk	$17.12
80/20 10 Series, Corner Bracket	8	4119	$3.81

4.3 Brew Extractor Thermodynamics

Across the entirety of the brewing process, thermodynamics are at play. The brewing process requires a great deal of temperature changes, many of which are time sensitive. When the brew extractor user enters a recipe into the central control unit, it is necessary that the brew process adheres closely to the user set temperatures and times. A lack of accurate repeatability defeats the purpose of the automated brew extractor. This is where thermodynamics come into play. Using

some basic thermodynamic and fluid heat transfer concepts, it is possible to obtain estimates of timing, temperatures, and volumes required to achieve the user's recipe. For example, sourcing a part for the kettle heating element requires knowledge of fluid volume, target elapsed time, and fluid properties.

Similarly with the wort cooling system, the system needs to be capable of reaching the user defined target temperature with the cooling device and have cooling capacity to spare. Thermodynamics and fluid heat transfer concepts are used to calculate the necessary starting temperatures of the cooling fluid as well as estimate the total cooling capacity of the system given a worst case scenario. Being able to calculate some of these figures beforehand will give the user insight into what kind of changes to the recipe could affect brew time or even overload the system.

4.3.1 Kettle Heater

Throughout the brewing process water and wort must be heated to user specified temperatures for user specified amounts of time. It is important that the method of heating is able to be controlled in a manner that will allow for the user defined recipes to be repeatable and accurate. First the heater must heat water to a temperature for steeping the grain. This process was reviewed in higher detail in the *Process Control* section, it is important to note that this temperature will vary based on the recipe input by the user. The next stage of heating requires sparge water to be heated to another user defined temperature for extracting as much wort from the steeping process as possible, this process must be accurate in order to ensure the sparge water mixing with the wort does not cause any unwanted heating or cooling. Finally the heater is used to "boil" the wort. The temperature of the wort "boil" process is not necessarily at the point of boiling, but is actually another user defined temperature that will likely be near the boiling temperature of water. It is clear now that the heater will be used throughout the brewing process many times and it is therefore an incredibly vital component to consider.

Four heating methods were considered and are outlined thoroughly in the following sections. The four methods considered were gas burner, stove top style resistive heating element, water heater style resistive heating element, induction heating coil. The factors to be considered in each of these heating methods included size, efficiency, cost, and power output (effective heating time).

Figure 4.5: *Gas burner style heater.*
(Courtesy of WEBstaurantstore.com)

The first method considered was the gas burner. An example of a gas burner is shown in Figure 4.5. This method of heating is what most home brewers use when any sort of heat needs to be introduced to the system. It allows for the quickest heating due to the immense amount of energy released during carbon fuel combustion. Burning a carbon fuel may not seem like the most efficient method of heating liquid, but it is actually more efficient than utilizing electrical appliances when the source of the electrical power is considered. Power plants than are burning carbon fuels to convert chemical energy into electrical energy and subsequently deliver this energy to a home suffer large losses throughout the process. Burning the fuel for direct heating is a more efficient method of using the carbon fuel. That being said, the efficiency being considered for this project is viewed from a different perspective. The available energy is compared to the energy transferred into the fluid for heating, and in this respect, gas heating is fairly inefficient due to the inherent inability to direct heating directly to the liquid and not into the surrounding air and structural materials. The temperatures reached in the gas flame also are high enough to cause loses from radiation. There is also the inconvenience of possibly running out of fuel during brewing alongside the dangers of leaving a burning fuel unattended during brewing that reduce the favorability of this method.

Before reviewing the next three methods, it is necessary to evaluate a key limitation of the next three options. This limitation is that the power source for the entire brew extractor is limited to a common household outlet which are usually tied into a 20A breaker. Assuming household mains voltage to be 110V, this limits

the power consumption of the entire assembly to 2200 watts. A safe margin from this limit would be to cap the total power consumption to 2000 watts. Assuming the rest of the electronics used throughout the project consume no more than 200 watts, this leaves 1800 watts available for heating.

$$t = \frac{C_p \cdot \rho_m \cdot V_m \cdot (T_f - T_i)}{P}$$

According to thermodynamics, the time for a fluid to change in temperature given the system properties is shown in the equation above. In this equation, *t* is time, C_p is the specific heat of the fluid, ρ_m is the density of the fluid, V_m is the volume of the fluid, T_f is the final temperature of the fluid, T_i is the initial temperature of the fluid, and *P* is the thermal power being transferred into the fluid. Assuming the initial fluid temperature is 25°C, the fluid being heated is water, the volume of fluid being heated is 1 $^1/_2$ US gallons, and the heating element is able to transfer all power drawn (1800 watts) into thermal energy with no losses; then time to reach boiling temperature would the following calculation:

$$t = \frac{\left(4.18\ \frac{kJ}{kg \cdot K}\right) \cdot \left(999.9\ \frac{kg}{m^3}\right) \cdot (5.678(10)^{-3} m^3) \cdot (373\ K - 298\ K)}{1.8\ kW} \cdot \left(\frac{1}{60} \cdot \frac{min}{sec}\right)$$

$$t = 16.74\ min$$

This figure represents the amount of time it would take to boil 1 $^1/_2$ gallons of water given no energy loss during the heating process, assuming 1800 watts of energy being fed into the system. In the following three options, this is the theoretical limit to heating speed, but other inherent losses will cause the speed to drop.

***Figure 4.6**: Stovetop style resistive heating element.*
(Courtesy of WEBstaurantstore.com)

The second method considered was the stovetop style resistive heating element (STRE). An example of a stovetop style resistive heating element is shown in Figure 4.6. This device uses mains power to heat a coil of resistive material which the brew kettle would sit on top of. Resistive heating is a very effective technique because it is very efficient, in the sense that nearly all of the power drawn will be converted into thermal energy. It is also a fairly cost effective solution as resistive heating elements are very common and easy to manufacture. The device is fairly slim therefore its size can be easily accommodated into the design of the stand too. One drawback of this style of device is that it is used to heat the kettle primarily, which in turn heats the liquid inside through conduction, this causes a slight loss in efficiency. The device also could also obstruct the measurement of the kettle weight, which will be used to track the fluid level within the kettle, because both the stovetop style resistive heating element and the weight sensor need to be placed below the kettle.

Figure 4.7: *Water heater style resistive heating element. (Courtesy of WEBstaurantstore.com)*

The next method considered was water heater style resistive heating element (WHRE). An example of a water heater style resistive heating element is shown in Figure 4.7. This device functions similarly to the stovetop style resistive heating element in the sense that they both take in mains power and convert this power with high efficiency into thermal energy. The fact that this device does not function as a stand or come with extra power controls actually makes this even more cost effective however. The heater shown in the figure shows mounting holes for mounting the element directly into the side of the kettle, which allows the heating element to be in direct contact with the liquid being heated, this causes less losses as the primary item being heated is the liquid and the kettle is only heated through conduction. These types of heaters are also available with the mounting option of threading the element into a fitting on the side of the tank/kettle which allows for standard industrial sealing methods to be used to prevent leaks. Since the element is placed in the side of the kettle it takes up virtually no space outside of the kettle and is therefore the smallest option.

Figure 4.8: *Inductive heating coil (stovetop style).*
(Courtesy of WEBstaurantstore.com)

The final option considered was an inductive heater. An example of a stovetop style inductive heater is shown in Figure 4.8. The fundamental principal of an induction heater is that it utilizes a high current carrying coil to induce eddy currents into magnetic materials placed within the magnetic field of the coil. The eddy currents induced into the magnetic metal object cause the object to heat. In the case of this project, a stovetop style heater is not necessary as the coil could be wrapped around the brew kettle and the current induced directly into the walls of the kettle. Again this style of heating will be susceptible to greater losses due to the fact that the kettle is being heated primarily and the liquid through conduction. Designing this style of inductive heater will also be much more costly and time consuming than a resistive heating element. The material used in the brew kettle could also affect efficiency negatively, because most brew equipment is stainless steel, and stainless steel in only mildly magnetic. Induction heating efficiency is directly proportional to the magnetic permeability of the material within the coils magnetic field.

Now that all four options of have been reviewed, a comparison of the options is done with weighted parameters; size, efficiency, cost, and power output (effective heating time) is shown in Tables 4.6(a), (b), and (c).

Table 4.6(a): Comparison of Heating Methods

Method	Size	Efficiency	Power Output	Cost
Gas	2	5	10	3
STRE	5	7	3	8
WHRE	10	9	3	10
Induction	8	7	3	3

Table 4.6(b): Heating Parameters Individual Weights			
Size	**Efficiency**	**Power Output**	**Cost**
5	3	4	3

Table 4.6(c): Comparison of Heating Methods Weighted			
Gas	STRE	WHRE	Induction
74	82	119	82

It is clear that the most desirable method to use given these weights is option number three, the water heater style resistive heating element. These elements are available in range of output wattages for different applications. In the case of this project, a 1500 watt resistive heating element will be used as these elements are typically found in increments of 500 watts. Knowing that the element is nearly 100% efficient, the lossless time required to bring 1 $^1/_2$ gallons of water to a boil from room temperature is recalculated using this lower power element. Plugging 1.5 kW into the formula used above yields an ideal time of roughly 20 minutes. A bill of materials for this heating element and its installation hardware is shown in Table 4.7

Table 4.7: Heater Bill of Materials		
Item	**Part Number**	**Unit Cost**
Stainless Steel Heating Element - 1500W, 1" NPT	Camco 02142	$9.42
Stainless Steel Weld On Bulkhead Fitting, 1" NPT	(MCM) 1698T32	$10.12
Viton O-ring, AS568A, FDA Approved (5-pack)	(MCM) 5577K219	$8.28

4.3.2 Wort Cooling

Depending on the user defined recipe, the target wort "boil" temperature could range anywhere from 93 °C (199 °F) to 100 °C (212 °F). When brewing the target maximum of 1 $^1/_2$ US gallons of unfermented beer, it can be quite a huge undertaking to cool that volume of fluid to below room temperature in a short period of time. It is necessary to use a heat exchange system capable of rapid cooling with plenty of headroom to spare. One key piece of knowledge to take from the basic principles of thermo dynamics is that when using one fluid to cool another the ΔT (difference in fluid temperatures) is proportional to cooling time assuming all other system parameters are held constant. That is, if the fluid being cooled is 50 °C and the target temperature is 30 °C, a coolant at 0 °C will reach this target temperature more quickly than a coolant running at 10 °C. This is because heat flux is proportional to the difference in temperature given a static temperature

exchange medium, much like current is proportional to a difference in electrical potential (voltage) given a static electrically conductive medium (such as a resistor).

Given the maximum capabilities of the automated brew extractor to be a volume of 1 $^1/_2$ US gallons at a temperature of 100 °C, and the assumed lowest likely target temperature for fermentation is 15 °C, it is possible to calculate the "cooling load" using the following formula:

$$Q = C_p \cdot \rho_m \cdot V_m \cdot (T_f - T_i)$$
$$= \left(4.18\ \frac{kJ}{kg \cdot K}\right) \cdot \left(999.9\ \frac{kg}{m^3}\right) \cdot (5.678(10)^{-3} m^3) \cdot (288\ K - 373\ K)$$
$$= 2.017\ Megajoules\ OR\ 1911.7\ BTU$$

In order to achieve this magnitude of cooling capacity, a much larger reservoir of cold fluid must be used to absorb the thermal energy in the heated wort. A ten-gallon cooler will be used as a cold fluid reservoir due to space constraints. This reservoir will have a connection to city water and will be filled to capacity minutes prior to utilization to minimize losses. A 10 gallon reservoir of city reservoir at room temperature will not have the capacity to reach an equilibrium temperature of 15 °C or lower because room temperature is already above the target low temp. In order for the wort and the cooling fluid to exchange thermal energy and have an equilibrium temperature of 15 °C or lower, the cold reservoir must be lower than 15 °C. In order to achieve this, the user will deposit a predetermined amount of ice into the cold water reservoir prior to brewing. The amount of ice necessary can be calculated using basic thermodynamic principles, as well as recipe parameters such as wort volume and boil temperature. By adding ice to the reservoir the cooler is taking advantage of waters high latent heat of fusion. Latent heat of fusion is defined as the amount of energy required to be removed from a substance to change from a liquid to a solid per unit mass. This value is equivalent in the reverse direction, that is, it can also be viewed as the amount of energy added to a substance per unit mass to cause a state change from solid to liquid. In the case of this cooler, the ice is absorbing the thermal energy from the hot wort but not changing temperature until all of the ice has changed state to liquid water.

It was calculated that a 1 $^1/_2$ gallon batch of 100 °C wort exchanging thermal energy with a 10 gallon cooler of room temperature (25 °C) water mixed with 9 kg (20 lbs.) of ice will in a lossless closed system will reach an equilibrium temperature of 12.7 °C. The losses within the system will most likely work in favor of the user because the temperature differential with the surroundings is greater for the wort. That is to say that since the wort will not only be exchanging heat with the cold water reservoir, but also with the air via natural convection as well as solids it comes in contact with via conduction. The large temperature differential will work to help cool the wort. Even so, if a recipe calls for a 1 $^1/_2$ gallon batch of 100 °C wort with a

target temperature of 15 °C, the user will most likely be prompted to add additional ice to lower the equilibrium set point further. Adding an additional 1 kg of ice will lower the equilibrium point of the heat exchange process to 10.4 °C which is a safer margin. In the end, however, the user will have the ability to add as much ice as seen fit, the program will simply suggest a bare minimum amount of ice to ensure the target temperature is reachable.

In order for this heat exchange to take place, a medium between the two fluids is necessary. A counter-flow heat exchanger is used to cause this interaction between the two fluids. A counter-flow heat exchanger is any device that allows the flow of two substances in opposite directions separated by a thin thermally conductive layer of material. The flow allows for a high rate of heat transfer through the thin wall between the two substances without mixing the two substances together. In this project, the cold water from the reservoir will be pumped into one side of the counter flow heat exchanger while the hot wort is pumped into the opposite side. The coolant water will then flow out of the counter-flow heat exchanger at a much higher temperature, and into a water to air finned heat exchanger which utilizes forced convection to dissipate a great deal of the thermal energy in the water into the air. The water then returns to the cold water reservoir for recirculation. Meanwhile the wort is circulated back into the kettle until the mean wort temperature in the kettle has reached the target temperature. A block diagram of the system is shown in Figure 4.9.

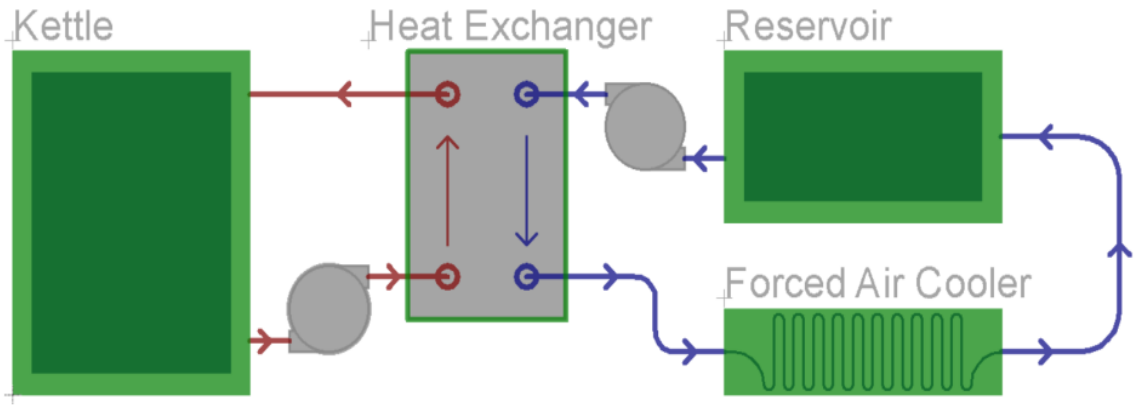


Figure 4.9: *Block diagram of heat exchanger system.*

There are many different types of counter flow heat exchangers available, the main difference between most of them being size and geometry, both which affect cooling speed. For this project, budget is a large concern therefore an economical counter-flow plate chiller was chosen. A plate chiller uses the concept of a counter-flow heat exchanger, but the conducting path between the two fluids are thick copper or stainless steel plates. The fact that the conduction medium is thin allows for larger thermal flux through the plates and the geometry of the plate maximizes surface area available for heat transfer, all the while being an inexpensive device to manufacture. Most commonly available counter-flow plate chillers are also stainless steel, which is ideal for this project due to stainless steels corrosion

resistance and inert nature. The plate chiller chosen is shown below in Figure 4.10. This plate chiller, from dudadiesel.com, is capable of a maximum of 17,000 $^{BTU}/_{hr}$ of cooling potential, assuming this cooling capacity is attainable, the a 1 $^{1}/_{2}$ gallon batch of 100 °C wort example would be cooled in roughly 7 minutes. Table 4.8 below shows the bill of materials for installation of this heat exchanger.

Figure 4.10: *B3-12A counter-flow plate heat exchanger. (Courtesy of dudediesel.com)*

Table 4.8: Wort Cooler Bill of Materials		
Item	**Part Number**	**Unit Cost**
Stainless Steel 10 Plate Heat Exchanger, 1/2" hose barb	B3-12A	$48.87
Stainless Steel Hose Clamps, SAE 6, 10-pack	(MCM) 54155K11	$11.13
24V Submersible pump, 30L/min	(eBay) 1E96208F54	$6.99

4.4 Brew Extractor Automation and Control

There are several subsystems involved in achieving the goal of automating the process of extracting sugar from barley to produce beer. Each of the automation and control subsystems was designed to satisfy certain criteria needed for each step of the process. In order for this process to be truly automated, the group needed to incorporate, communications, recipe control, temperature control, fluid level control, and fluid flow control. All of these automated subsystems work together to produce the finished product. This section will cover all of the automation and control subsystems designed for this project.

4.4.1 Communications

The communications for the system was a crucial instrument that the group needed, to achieve the goal of automating the process of sugar extraction. The group needed to be able to send and receive information from the user interface,

into the system, and have the system interpret that data as usable and reliable information. In order to accomplish this goal the group decided to use the Phillips I^2C communications protocol. Since the group decided to interface three control devices, I^2C was desirable because it required less space to implement, making the printed circuit board cheaper to manufacture. The programming associated with implementing this protocol also seemed more compact and intuitive, when compared to SPI. Implementing the SPI protocol would have occupied twice as much space as the I^2C protocol, when looking at the additional hardware needed to successfully implement. The group was also interested in learning to use the I^2C protocol because it seems to be the most commonly used form of communication between integrated circuits that are on the same circuit board. Figure 4.11 shows a block diagram of how the group connected each of the control devices to the I^2C bus.

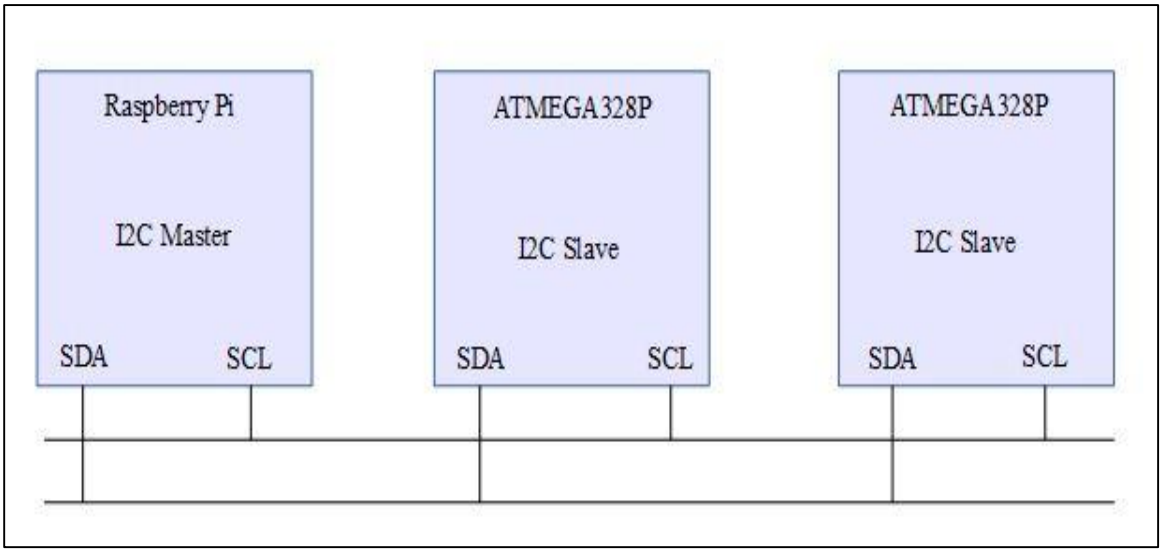


Figure 4.11: *Block diagram of* I^2C *bus*

Had the group decided to use SPI as the protocol for transferring and receiving data, an additional two connections would be needed per chip, and additional two connections would be need on the Raspberry Pi. This configuration seemed less desirable because not only would this occupy more space on the circuit board, it would also take pins away from the microcontrollers that could be used for something else. Aside from the additional connections that would be required, the protocol itself seemed less intuitive for the design project. SPI uses two shift registers to exchange data between devices. Since the system only needs to send data in one direction a majority of the time, it seemed more intuitive to use the I^2C protocol. Figure 4.12 shows a block diagram of the SPI implementation would have looked like, had it been chosen for the design project.

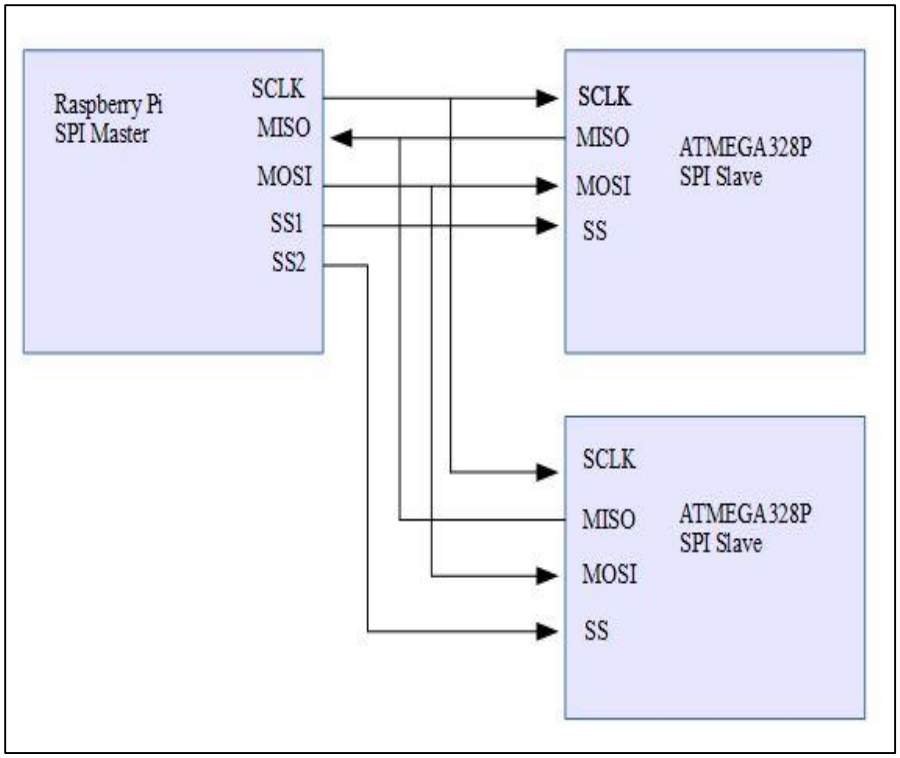


Figure 4.12: *Possible SPI configuration*

Although all of the devices connected to the I^2C bus support I^2C, the group could not simply connect them together because of their operating voltages. The Raspberry Pi operates a 3.3V I^2C bus and because the chosen voltage for the ATMEGA328P microcontrollers is 5V, the group could not connect them directly to one another. The data sheet for the microcontrollers indicates that the I^2C bus must operate at 0.7*Vcc. The Raspberry Pi is not 5V tolerant. Although this may not destroy the Raspberry pi, it can significantly reduce its operating lifetime. It would have been possible to operate the microcontrollers at 3.3V, however, this would significantly reduce the resolution of the sensors. In order to overcome this challenge the group implemented a level shifting circuit using two 2N7000 n-channel MOSFETS. The serial data lines, and serial clock lines, were interfaced to this circuit. The raspberry pi supplies the 3.3V needed to drive the gates of each MOSFET. In addition to the level shifting MOSFETS, the group also needed pull up resistors to connect the power to the low voltage side as well as the high voltage side. Each I^2C module pulls the bus low to send and receive data. Since each device provides some level capacitance to the bus, the group calculated the optimal pull up resistor value to be 1.5k Ohms. The group needed to calculate this value to ensure that the devices were getting the appropriate edges required by the I^2C protocol. Figure 4.1 shows the level shifting circuit the group implemented in the design including the pull up resistors and terminal blocks used to connect the Raspberry Pi to the printed circuit board. If SPI had been chosen to be the main protocol for communications, an additional four pull up resistors would have been needed along with four additional transistors. This would have occupied a

significant amount of space on the printed circuit board that the group is planning to have manufactured when the design is finalized.

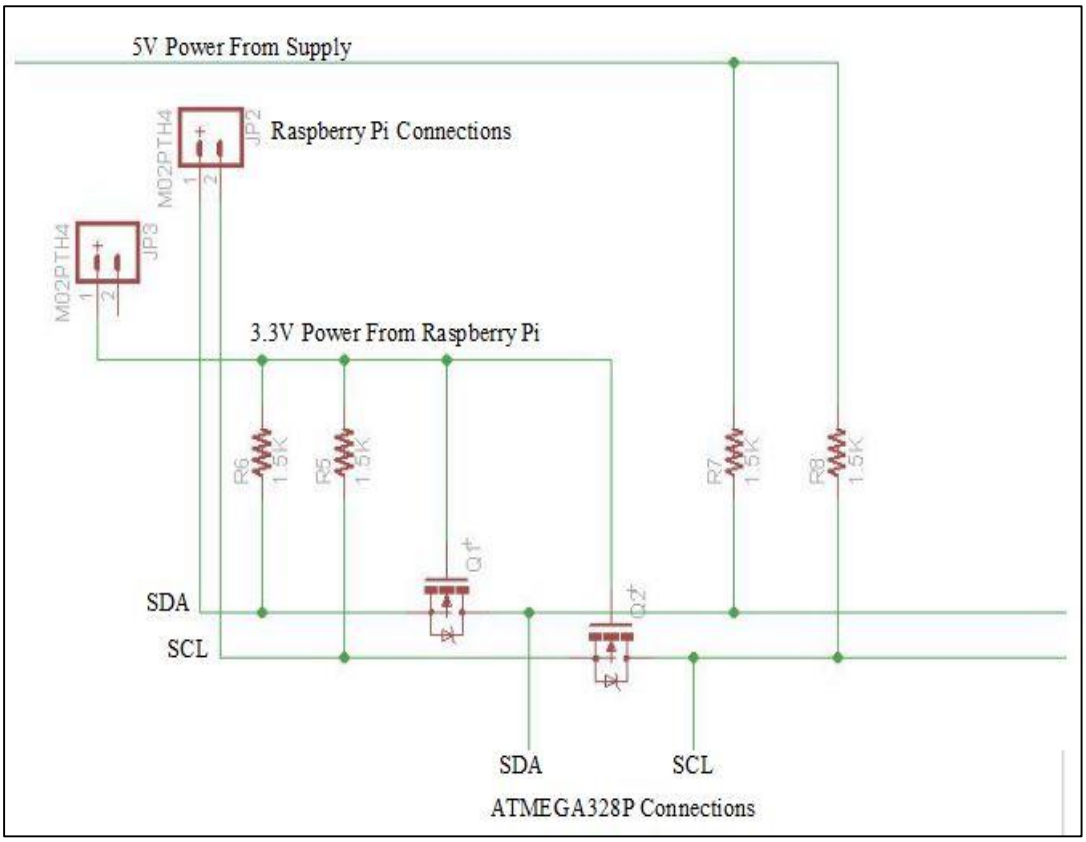


Figure 4.13: *Level shifting circuit*

Aside from the level shifting circuit, the group also wanted a way to utilize the remaining inputs and outputs on the microcontrollers that were not used to operate the system. In order to achieve this goal the group included male header pins that would allow them to connect the programmer and utilize all of the remaining input and output ports. This would allow the group to program the chips in circuit. This functionality could be useful if the user ever decided they wanted to make changes to the existing system configuration. This will also allow for flexibility with the current system configuration, giving the user a creative grasp of their equipment and freedom to change that equipment as necessary. Figure 4.14 shows the connections made in order to implement the group's plan for in circuit programming. Figure 4.15 shows the finalized communications circuit for the project.

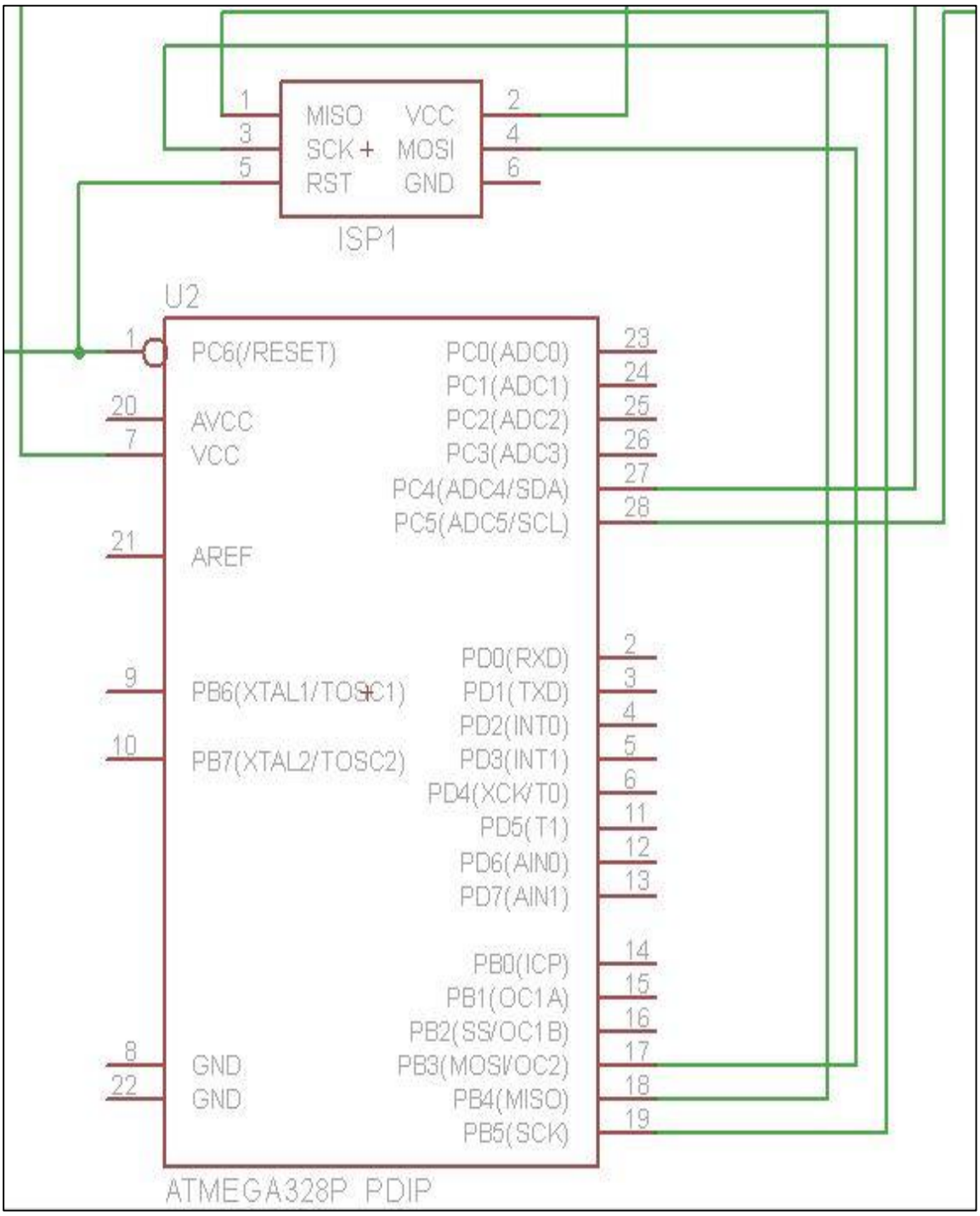


Figure 4.14: *Male header connections for in circuit programming*

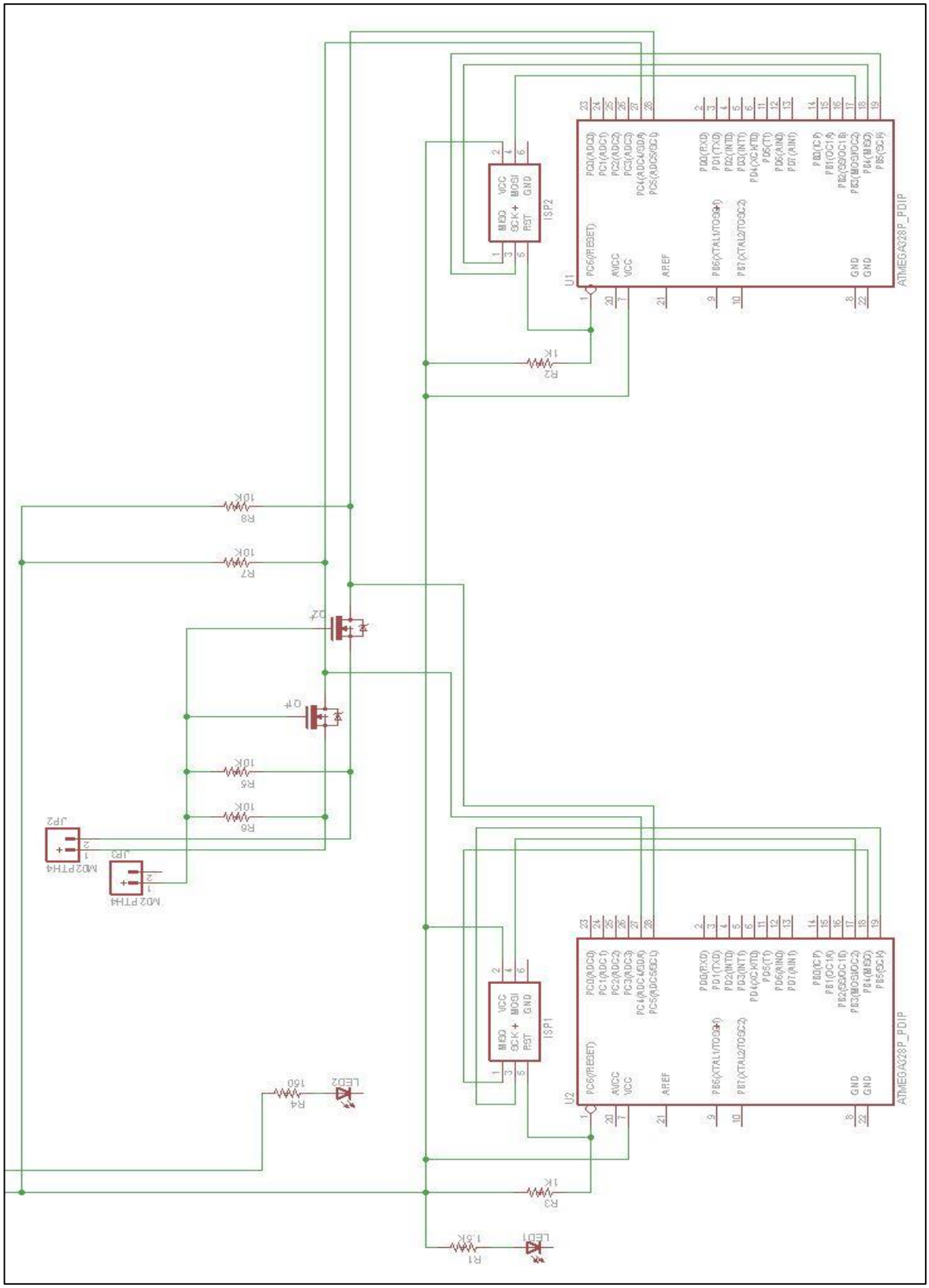


Figure 4.15: *Complete communications circuit*

The communications subsystem will be expected to transmit all of the necessary data between the chips and the user interface. It will also provide the user with the ability to change any of the functionality of the input and output pins as needed. Using the I^2C protocol, the level shifting circuit, and in circuit programming capabilities, the communications system will provide all of the functionality that the design project needs. Aside from the hardware used to implement the I^2C protocol, the group also had to prepare the software side. In order to have a functioning I^2C bus, the group obtained the I^2C slave drivers for the microcontrollers from Atmel. Although this driver contained all of the necessary code to control the registers that operate the I^2C module on the microcontrollers, significant modification was needed to make the driver work for the design project. The I^2C driver is interrupt driven, therefore, delays are needed to allow certain steps of the control program to execute before transmitting data. Without the delays the I^2C module would remain in an active state preventing the microcontrollers from performing other tasks. This was also controlled by only sampling the data at specific intervals. For example, instead of continuously reading the temperature of one step of the process, the microcontroller only reads the temperature every three seconds. This is sufficient in helping the group obtain meaningful data, and also helps stabilize the control process by preventing the rapid fluctuation of the data. All of the delays were implemented using the Python code that was written for the Raspberry Pi I^2C master. Since the ATMEGA328P microcontrollers operate as slaves, only the Raspberry Pi can initiate a transfer of data. Adding the delays on the microcontroller side would have been ineffective considering the interrupt would essentially override the delay operation. Figure 4.16 and Figure 4.17 Show how the program uses delays to smooth out the operation of the system.

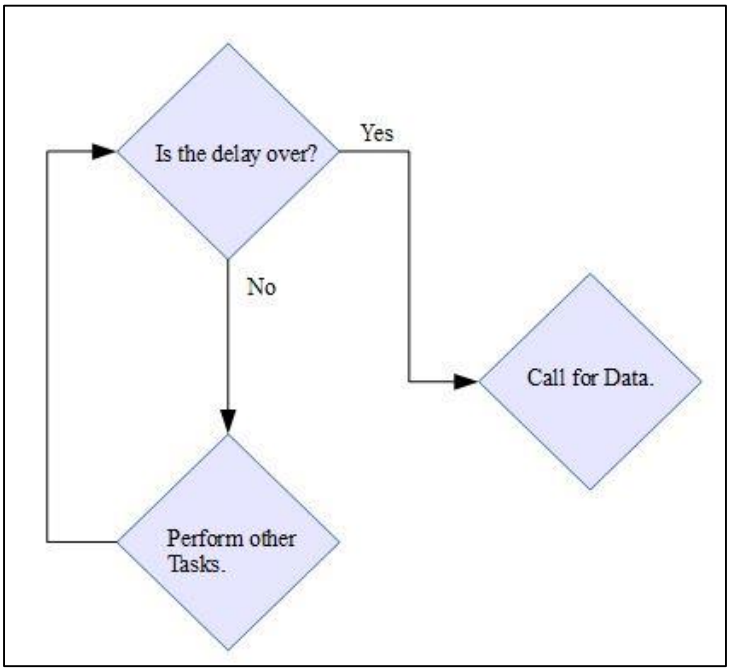


Figure 4.16: *Raspberry Pi I^2C Master Delay Structure*

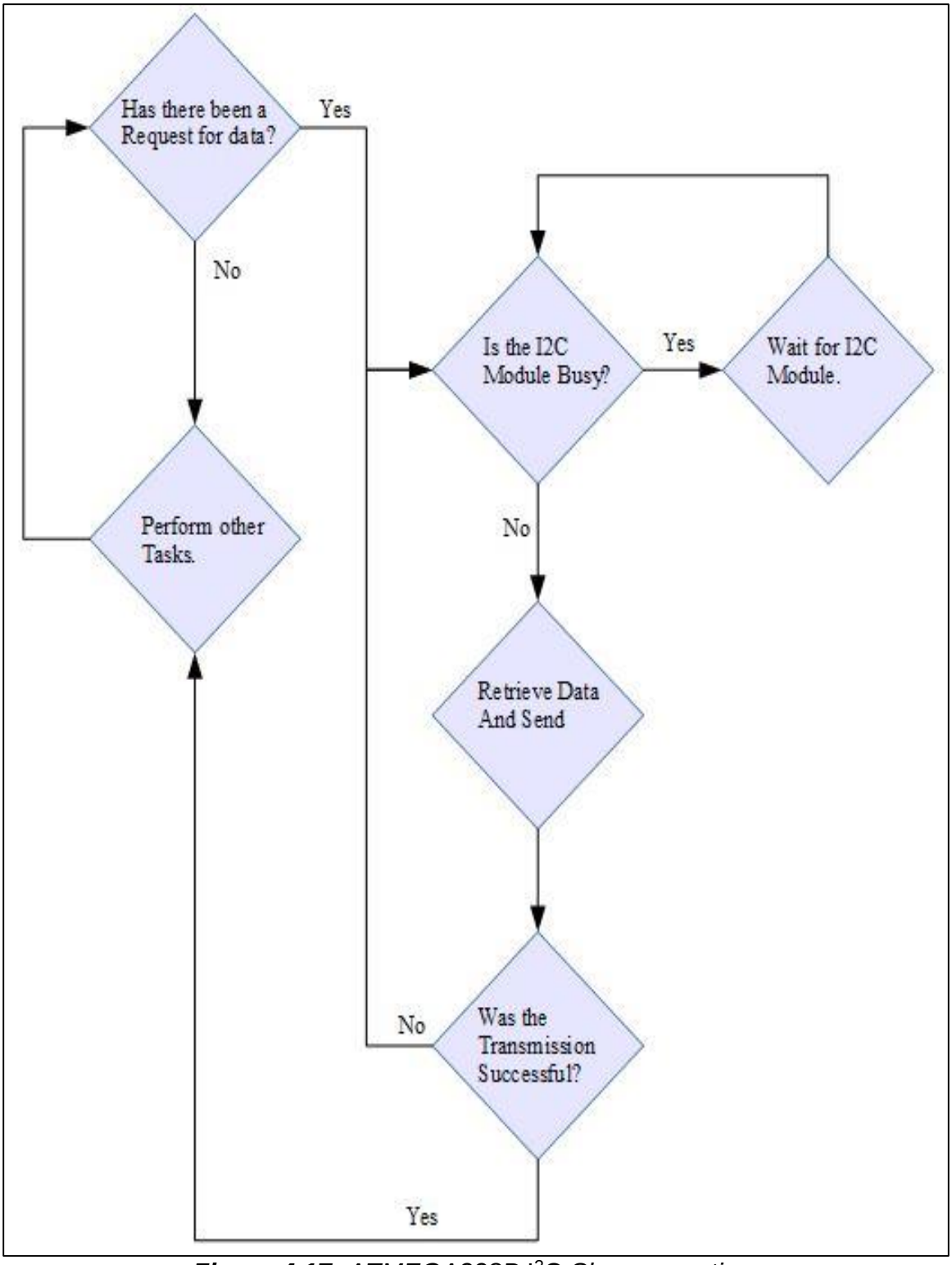


Figure 4.17: *ATMEGA328P* I^2C *Slave operation*
(Visibility decreased with smaller size)

4.4.2 Recipe Inputs

With the communications setup implemented, the group could now look at how they were going to control the system using recipe information. Each brewing cycle will be driven by specific temperatures, boil times, and liquid levels. In order for the system to be truly automated, the user must be able to specify this information, send it out to the system, and have this system respond appropriately. When the user enters the data into the interface, and sends the data to the microcontrollers, the Raspberry Pi will work with the microcontroller to make sure that the recipe information is assigned to the appropriate variables in the microcontroller's code. In order to accomplish this goal, the Raspberry Pi will first send a hexadecimal value that corresponds to the type of data that will follow. The microcontrollers will call a function that returns the hexadecimal value sent by the Raspberry Pi, then decide which variable needs to be assigned the next byte of data that will be sent. Once the second byte of data is received, the microcontroller places that value according to the previously received hexadecimal value and waits for the next variable to be sent. This allows the microcontroller to differentiate between a temperature specification, time specification, and liquid level specification. The hexadecimal codes are shown in table 4.9.

Table 4.9: Data codes for recipe information

Hexadecimal Value	Variable To Assign
0x01	Mash Temperature
0x02	Boil Temperature
0x03	Pitching Temperature
0x04	Mash Time
0x05	Boil Time
0x06	Mash Liquid Level
0x07	Boil Liquid Level

Included in the recipe codes there are also some other commands that the microcontrollers can interpret. Since the group want to use the Raspberry Pi as an interface, it will also need other functions such as starting a recipe cycle, stopping a recipe cycle, and a cleaning cycle. These are other things that they found to be a necessity when trying to implement an automated home brewing system. The command codes will function very much in the same way as the recipe input codes. Table 4.10 shows the list of command codes.

Table 4.10: Command codes for system operation	
Hexadecimal Value	**Operation to Perform**
0x08	Start Recipe
0x09	Master Stop Command
0x10	Clean Cycle Start

4.4.3 Process Control

Once all of the necessary data has been received by the appropriate variables, and a start command has been issued, the brewing cycle will begin. As previously mentioned the group divided the work of process control, between two microcontrollers. The first microcontroller will have received all of the temperature specifications while the second microcontroller will have received liquid level specifications and timer specifications. These two will work together to monitor the process and ensure proper operation. Table 4.11 shows how the work load is divided among the microcontrollers.

Table 4.11: Division of labor amongst microcontrollers	
Microcontroller 1	**Microcontroller 2**
Maintain Proper Temperatures	Control Fluid Levels
Disable Heating if an above necessary temperature occur	Control Fluid direction (flow paths)
Disable Heaters When Steps are Complete	Monitor the Timing of Each step in the Process
Store temperature information so that it may be retrieved by the Raspberry Pi	Store Liquid level information so that it may be retrieved by the Raspberry Pi

The process will start by filling the boil kettle to the specified level as input by the user. Once that level has been achieved, the heating cycle will begin. When the Mash Temperature specified by the user has been achieved, the first transfer of liquid from the boil kettle to the mash tun will take place. Once the level in the mash tun has been achieved, the mash timer will begin. When the mash timer is done, the liquid in the mash tun will return to the boil kettle during the sparge process. This rinses the sugar water from the grain that is in the mash tun. Now that the boil kettle has all of the liquid returned, the boil is ready to begin. The boil temperature will be achieved and maintained for the specified amount of time indicated by the user. Once the boil is complete, the liquid will be transferred through the cooling

unit until the pitching temperature is achieved. After this step is complete, the liquid is transferred to the carboy. This completes the automated portion of the process. The user will add the yeast and attach the airlock for fermentation. Throughout all of these steps the microcontrollers will send information back to the Raspberry pi to indicate when it has completed a step. This helps coordinate all of the work between the two microcontrollers. The Raspberry Pi, upon receiving notification of a step completion, will then send new instructions back to the microcontrollers in the form of the hexadecimal commands shown previously. Table 4.12 shows the hexadecimal notifications for step completion that will be sent back to the Raspberry Pi. Figure 4.18 shows how the notifications effect the system throughout the process.

Table 4.12: Process Notifications	
Hexadecimal value	**Notification**
0x01	Mash Step Complete
0x02	Boil Step Complete
0x03	Cooling Step Complete
0x04	Cycle Complete

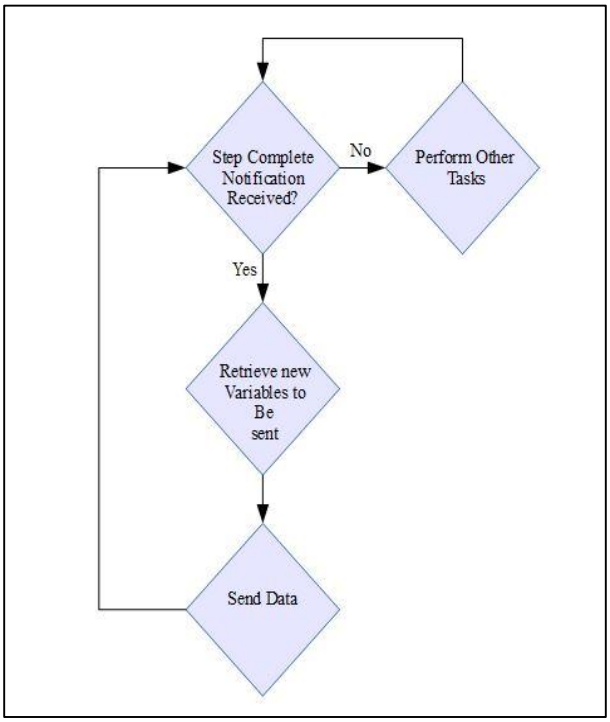


Figure 4.18: *Flow chart for process notification handling*

4.4.4 Temperature Control

Throughout the process there will be several stages where temperature control is a necessity. The entire sugar extraction process can be very sensitive to temperature when dealing with how efficient the system is at removing sugar from barley. There are very specific temperatures that are used in the brewing community. For the purposes of this project, the group decided to use PID temperature control to handle all of the heating processes used in this system. This was the most desirable means of temperature control because it will allow the system to hit accurate temperatures given changes in the surrounding area. For example, if the ambient air around the brewing system is warmer or cooler, the PID temperature control can compensate for the changes in the surrounding. It should also be mentioned that all of the heating control will be taking place on one microcontroller. It was determined to be less confusing to have one microcontroller dedicated to temperature control. This was stated previously in the design section of this document.

4.4.5 Fluid Level and Flow Control

While the heating control is taking place, the second microcontroller will handle all of the fluid filling and transfer. For each stage of the process valves will work together with the pump to transfer liquids to their appropriate destinations. Figure 4.19, 4.20, 4.21, 4.22, 4.23 and 4.24 show each of the stages and valve configuration for those steps of the process. The Filling and transfer stages are also listed below.

1. Valve to main water supply opens and the filling of the brew kettle Begins.

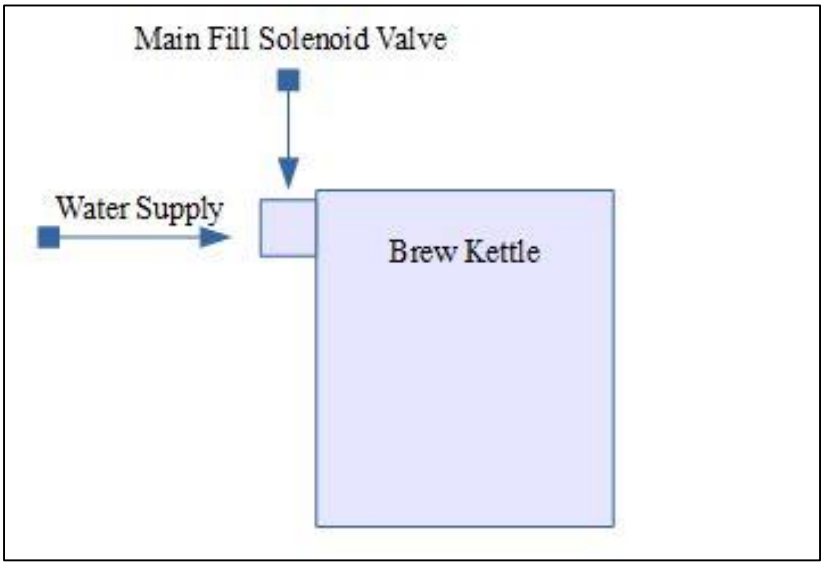


Figure 4.19: *Main Fill Stage*

2. Once the brew kettle is full and mash temperature is reached, water is transferred from the brew kettle into the mash tun.

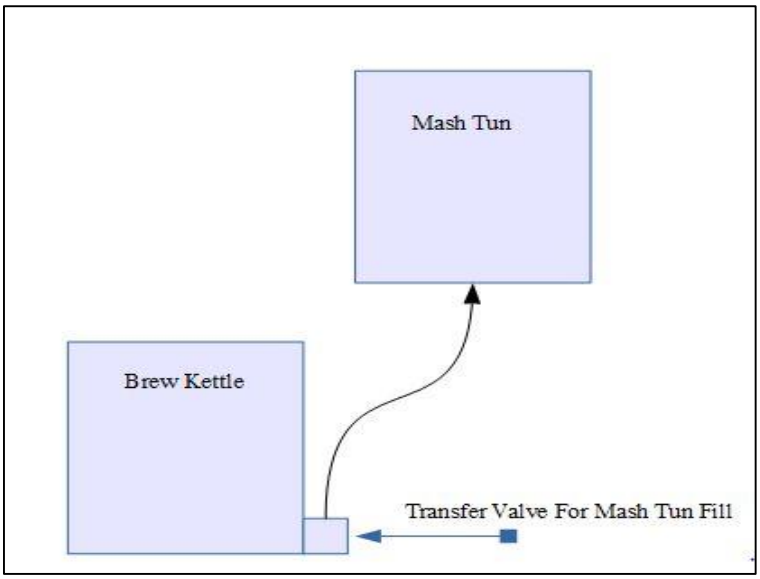


Figure 4.20: *Mash Fill Stage*

3. Once the mash tun is full, the sparge tank is filled with heated water from the brew kettle.

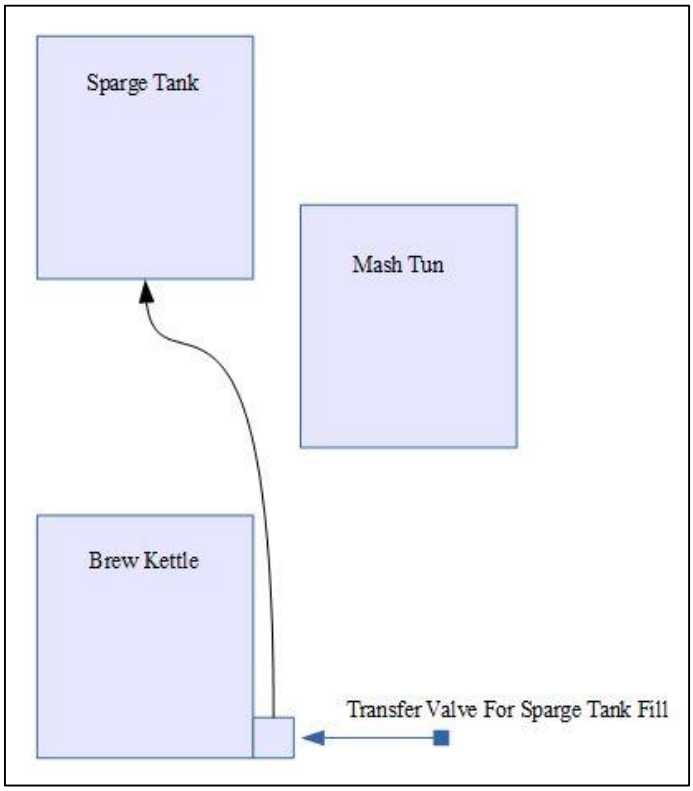


Figure 4.21: *Sparge Fill Stage*

4. After this step, the remaining processes except for the cooling will be fed by gravity. Once the mash time is completed, the liquid will drain from the sparge tank into the mash tun and the liquid in the mash tun will drain back into the brew kettle.

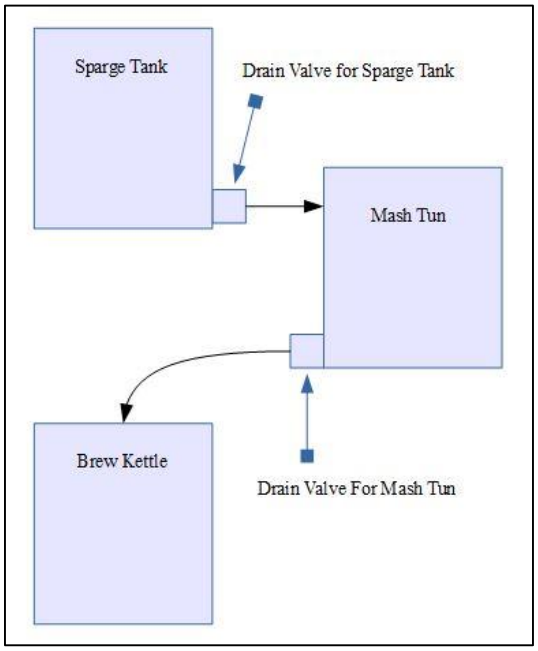


Figure 4.22: *Sparge Stage*

5. Once the brew kettle is filled, the boil will take place. Once the boil is complete, the liquid will be cycled through the cooling unit until the specified temperature is achieved.

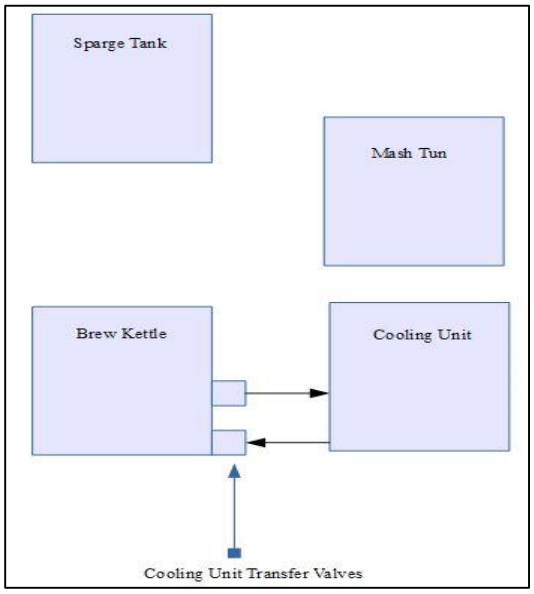


Figure 4.23: *Cooling Stage*

6. Once the cooling stage is completed, the transfer valve for the carboy opens, and the liquid is drained from the system.

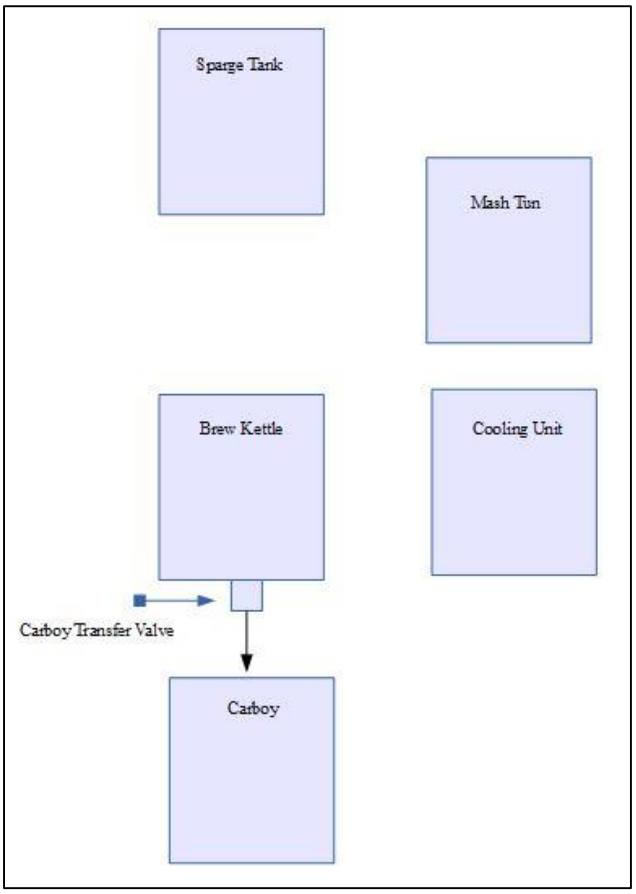


Figure 4.24: *Carboy Stage*

7. This completes the fluid transfer control process.

4.5 Brew Extractor Analog Data Acquisition

The group decided to keep track of the temperature, fluid levels, and pH of the fluids in the brew extractor. The data will be collected using an array of sensors that constantly is tracked. The fluid levels and temperature are important for the automation process. The pH information is for the user to keep track of the wort and sparge water pH and adjust as necessary. It is important that the worth maintain a pH of 5.3-5.5. To collect and store the data a combination of microcontroller and a web server is going to be implemented.

The plan initially was to collect the analog data and send it through an ADC so both the web server can collect the data and the microcontroller will be able to use the data. As the team progressed a microcontroller was picked due to its performance, price, and amount of analog to digital pins. The microcontroller

needs to come with analog to digital pins to remove the need for an ADC. Having one would take up board space. Once the data has been collected, it can be sent from the microcontroller to the web server.

4.5.1 Sensors

The temperature sensors will be used to keep track of the kettle water, wort, and sparge water. The sensors will be used to maintain target temperatures depending on the step in the brewing process. For instance the user will want the water to be heated to 95˚C before being pumped into the mash tun. Also the sparge water used to wash the sides of the mash tun need to maintain 76˚C. The wort also should be around this temperature when it returns to the kettle. The most important point being when the wort needs to be chilled for the distillation. The wort needs to go from 95˚C to 20˚C in an expedient manner and the process shouldn't stop until it is chilled. The temperature sensors will keep track of the changing temperature and send a signal to the solenoid switch to close. This will let the fluid sensor to control the pumps returning the wort form the plate chiller.

The next sensor needed are fluid level sensors. The importance is obvious and necessary to the system. The fluid level sensors ensures that there are no leaks in the A.B.E. Pumps will move fluids between the different containers. If an unequal amount of fluids is found the user is altered to a leak in the tubes. The fluid sensors will also alert the user to wort being left in the lines if there are no leaks. This sometimes happens, in a system such as this, product tends to get stuck in the lines if the pumps are not strong enough to move it through the lines. The other purpose of the fluid level sensors will be to send the signals to the solenoid switches to open and close for the pumps connecting the kettle to the mash tun and the plate chiller. The fluids sensors become integral in running the pumps controlling the flow of fluids throughout the system, especially while the wort flows through the plate chiller. The temperature sensors and the fluid sensors are both important for this process. The group needs the solenoid switches to remain open and the pump running until the target temperature is met and the fluid level reached.

The pH sensors are only important when the grains are added to the heated water. The pH needs to be maintained for the yeast to create an excellent batch of beer. The target pH is 5.4-5.6. Depending on the location the user is in the water's starting pH is important; however, that becomes part of the recipe inputted by the user. Sending the data of the pH levels will allow for A.B.E to adjust the levels while the sparge water is added to wash down the sugars as the wort water is brought back to the kettle. The A.B.E will add Calcium and Magnesium Salts to help lower the pH in the mash tun during the process. The group doesn't need to be concerned with increasing the pH because the sparge water being added increases the pH. Small adjustments in a brews pH affects the taste of the batch made. The information provided by the pH probe will assist the user to creating a

better tasting beer. There are negatives to using a pH probe to automate the process but that will be discussed later with the selection of the probe itself.

4.5.1.1 Temperature Sensors

The operating range of the Brew Extractor 20˚C-95˚C. There are three ways the group looked into for obtaining temperature readings: Thermocouples, resistance temperature detector (RTD), and thermistors. Thermocouples, while having the largest range of temperature values, didn't have the accuracy needed for the group's application. They also lose accuracy over time much quicker than an RTD or thermistor. Because high temperature measurements are not planned and accuracy is fairly important to the system, the group decided to look further into RTDs and thermistors.

Thermistors and RTD perform in the same functionality at lower temperatures. As temperature increases so does the resistance in the RTD. For thermistors it depends of the doping type thus creating two types of thermistors: Positive temperature coefficient (PTC) for p-type and negative temperature coefficient (NTC) for n-type. Three ways to use these devices are to create a current source and measure the changing voltage due to the change in resistance in the device. Another is to use a simple voltage divider circuit with a reference resistor. The third is to use a Wheatstone bridge circuit and use either the RTD or thermistor as one of resistors and measure the voltage across the bridge. Perhaps the biggest difference between the two are the temperature ranges and sensitivity of the two devices. RTDs have a large temperature range and a linear change in voltage. This works well with the application but the device doesn't react quickly to changing temperatures and is more expensive then thermistors. Thermistors react quickly and, although do not maintain a linear temperature to voltage, will help monitor the wort more accurately especially during the cooling process. One thing to note, the thermistor can maintain a mostly linear curve at lower temperatures usually around 100˚C or below. Because of the average cost of an RTD and the operating range being relatively small, the group has decided to use thermistors for temperature sensing.

There are three immersion thermistors the grouped looked into using for A.B.E.: Vishay NTCAIMME3C90080, Vishay NTCAIMME3C90373, and the Honeywell 590-59AD02-104. The first thing the group wanted out of the temperature sensor was the ability to be immersed in both the water and the wort. The temperature sensors are going to be exposed to products meant for human consumption. This leads to the second most important part, the housing of the thermistor. The housing needs to be food safe; thus, the housing needs to be stainless steel. The previously mentioned thermistors fulfills the criteria. The two Vishay thermistors found have an advantage in being that can be screw mounted. This make it easier to have multiple points of temperature sensing to see how the wort/water is boiling and the wort is cooling off. It also makes it easier to avoid the heating elements at the

bottom of the kettle. Another feature these two sensors have are the can be soldered directly onto a PCB or used in a breadboard for testing.

The two Vishay sensors do have their differences. The NTCAIMME3C90373 is a bullet shaped sensor with PVC cable. This sensor is the easiest to mount. Collecting accurate data easily because it maintains the most linear temperature to resistance ratio. The thermistor unfortunately has only a temperature range of -25˚C to 105˚C. While within the temperature range there is a discrepancy at the high end. The group runs the risk of obtaining inaccurate data while the wort and water are boiling. The NTCAIMME3C90080 is a rod shaped thermistor with similar mounting. The rod length is 6.5 inches. Female fasten is needed to collect the data off the thermistor. The best feature is the temperature is the temperature range of -25˚C to 125˚C. The housing is also 100% water tight. The NTCAIMME3C90373 is only water proof up unto the brass or ring. When the group first designed the brew extractor the brew kettle was expected to be much larger. This rod type shape was perfect and could fit along with the heating elements. With the change in the amount worth the A.B.E will make, the rod no longer fits in the kettle.

Finally the Honeywell 590-59AD02-104. Similar to NTCAIMME3C90373 in its bullet shape form this sensor is immersed in the fluid from the top or any other form of mounting. The temperature range is -60˚C to 300˚C. The leads are covered in nylon can be implemented with a Wheatstone bridge, using a current source, or a voltage divider circuit. This thermistor though is not completely watertight. The group would have to construct a thermal well for the thermistor. This particular thermistor also is more expensive than the other two. Performance wise this is the best thermistor except for not having the perfect housing. The price of this thermistor is more than the others and the group plans only using several for A.B.E.

The thermistor the group decided to use is the NTCAIMME3C90373. The temperature range maybe smaller than the others but still performs well within the operating range. The price of this particular thermistor is fantastic at only $1.31. This makes it easier to get several temperature measurements different parts of the kettle. The housing is already made for it and can easily be mounted with a sealing O-ring and screw. Table 4.13 lists the main operating constraints the group was researching into.

Table 4.13: Thermistor Comparisons

Thermistor	Temperature Range	Casing
NTCAIMME3C90373	-25˚C-105˚C	Stainless steel bullet with brass O seal
NTCAIMME3C90080	-25˚C-125˚C	Stainless steel threaded rod
590-59AD02-104	-60˚C-300˚C	Stainless steel bullet with exposed leads

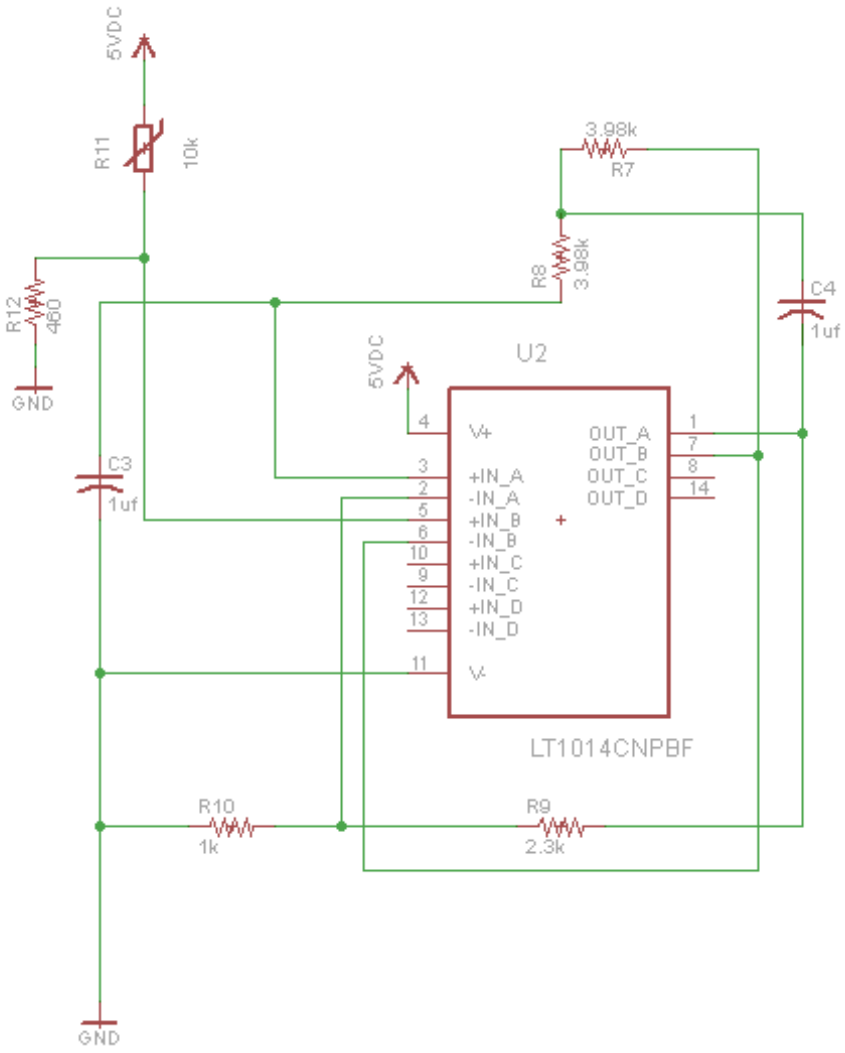


Figure 4.25: *Thermistor Circuit Design*

Figure 4.25 above illustrates how the group plans to measure the changing voltage according to temperature. The NTCAIMME3C90373 is a negative temperature coefficient. As the temperature increases the resistance decreases. In the voltage divider that means as temperature increases the measured voltage decreases. The reason behind the unity buffer is to make it easier to implement a low pass filter and amplifier circuit if necessary. V1 and V2 are both 5V. The team needed to ensure the voltage entering the operational amplifier is less than 2.5V so accurate readings can be taken by the microcontroller.

4.5.1.2 pH Sensors

The pH sensors or probes work like a glorified multimeter. Measure the change in voltage between and electrode and the solution the probe is in. No matter which probe the group decides to use a basic unity gain circuit will be used to read the change in volts depending on the pH. Because of the small range in pH, the group must also amplify the signal going to the microcontroller. The change in voltage will be in smaller then 0 to 60mV. The amount of amplification has not been decided yet. Figure 4.26 illustrates the initial planned circuit is below without amplification. Amplification will be needed because mV measurements will be taken. A low pass filter will be needed to deal with the noise coming from the 115 VAC powering the system.

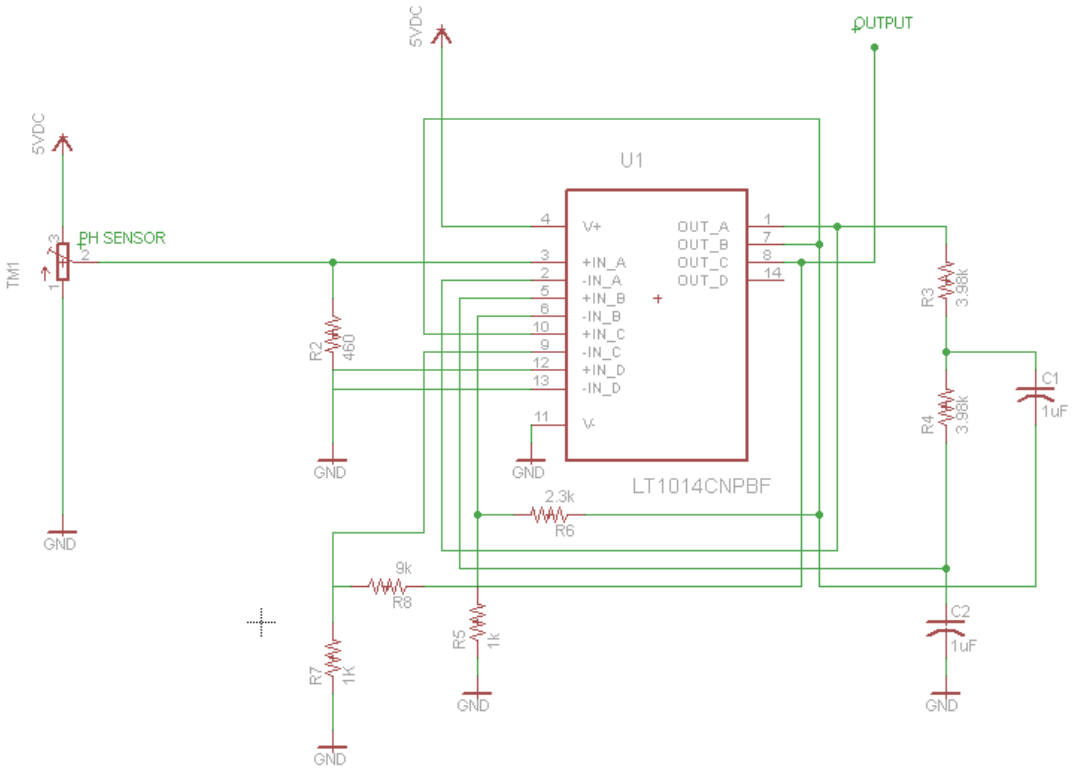


Figure 4.26: *pH Probe Circuit Design*

The wort made while brewing needs to maintain a pH of 5.2-5.6. This wort will also be at higher temperatures then what a probe normally checks which changes the type of probe the system needs. The probe selected needs to be able to operate at temperatures at and above 95˚C. The probe used must also be able to accurately check the pH at those temperatures not just measure it. Most pH probes are made for room temperature and pH of fluids change along with temperature. If one looks only at water with a pH of 7 and increase the temperature then there doesn't appear to be any error. Take the pH of a base or acid and that

is no longer true. Some probes come with an automatic temperature compensation (ATC) while others are manual. In other similar applications most people ignore the change in pH due to temperature. Because the system's target range is small, the group cannot ignore temperature. The last thing the probe needs to be is low maintenance. Probes, no matter how good, eventually will need to be recalibrated or replaced. Also while A.B.E is not in use the probe must be submerged in liquid. The final part of the criteria is cost. Most probes are expensive and making one had proved to be problematic. The probes the group looked into are Milwaukee MA917B/1, the Fluka 53162, and ALDRICH Z113077.

The Milwaukee MA917B/1 offers ATC and uses a BNC connector to collect data. The Milwaukee uses a glass shaft with makes it food safe. This probe operates at 0˚C to 100˚C. This is barely within the upper operating range. This probe does require less maintenance once calibrated. The approximate change in voltage to pH is about ±60mV depending on the change. At more acidic pH's this change is greater which is where the operating range is taking place. This is important to note when amplifying the voltage to see the changes more accurately. This shows that this probe might not be as accurate as the system needs with a ± .25pH variance. Storage of this electrode is important in order for the user to increase longevity of the probe. While not in use it is important to store in 7.0 buffer solution versus just storing in distilled water. The electrode solution may migrate from the probe if this is done. This electrode can last from six months to a year when cared for properly. What makes this probe viable is the cost being only $82 versus others that easily go to $120 or more.

The Fluka 53162 is very similar to the Milwaukee in its tolerances, operating range, and housing. It has a glass shaft, operates at 0˚C to 100˚C, and an accuracy of ± .25pH. The probe also offers ATC. The data sheet lacks the approximate change in voltage to pH so experimentation would be involved in obtaining the data and how much amplification it would need to see the difference, especially because of acidic nature the wort is in. The probe uses an S7 connector for data collection. The probe uses cartridge electrodes so it will last longer without maintenance and makes it easier for the user to replace. The Milwaukee uses a gel electrode. For the user, this probe has less hassle for storage because tap water can be used. This probe is more expensive at $110.

The ALDRICH Z113077 uses a pin connection instead of a BNC connection. While not really making a difference it does somewhat simplify the overall design for the probe. The operating temperature for the probe is -5˚C to 110˚C. This outperforms the other two prospective probes. It has a glass shaft and gel electrode like the Milwaukee. It shares an accuracy of ± .25pH with the other two probes. This probe would also require experimentation to see the voltage to pH change. Storage of this probe also will require a 4.0 pH buffer or a 7.0 pH buffer. This probe also has manual temperature correction. That is counterproductive to the automotive process. This probe is the most expensive out of the three having a price tag of $117.

The pH probe the group decided to use is the Milwaukee MA917B/1. This model overs the pH to voltage ratio and notes the change as the solution becomes more acidic. The probe is the cheapest of the three and the BNC connector is easy to implement. Finding an extremely accurate probe was troublesome considering the budget. Looking at the Milwaukee probe it is the most useful. It performs the best considering the conditions. Price played a big part in this decision. The only place this probe falls short is storage and maintenance. This probe requires more user interaction then was initially planned on. Table 4.15 below does a comparison of the various pH probes researched.

Table 4.15: pH Probe Comparisons

Probe	Temperature Range	Accuracy	Storage	Voltage/pH
Z113077	-5˚C-110˚C	±.25pH	Buffer Solution	unknown
53162	0˚C-100˚C	±.25pH	Tap Water	unknown
MA917B/1	0˚C-100˚C	±.25pH	Buffer Solution	±60mV/pH

4.5.1.3 Fluid levels

Fluid sensors play an important part in the system. They will control the solenoid switches controlling the flow between kettle, mash tun, plate chiller, and finally the keg. There are several different ways to measure fluid levels such as: load cells/ strain gauges, floats, phototransistors with LEDs, and for simple fluids a solid state electronic tape.

Strain gauges are an easily implemented way for measuring the fluid levels in the different tanks the systems has. It works the same way an electronic scale works at home. It measures the change of weight, in this case amount of fluid, buy measure the voltage change as the resistance in the strain gauge changes. The circuit would implement a Wheatstone bridge and would act as one of the four resistors on the circuit. A.B.E would then measure the voltage across the bridge to calculate the fluid levels. Another implementation circuit would is to use all four resistors in the bridge as strain gauges and have to be place in opposite directions of one another and create a balance for the load. This sort of configuration would make the measurements extremely accurate and sensitive to change is fluid level. Because the strain gauges will be in opposite directions it also counters the effect heat will have on the system. Figure 4.27 below helps to illustrate the strain gauge placement.

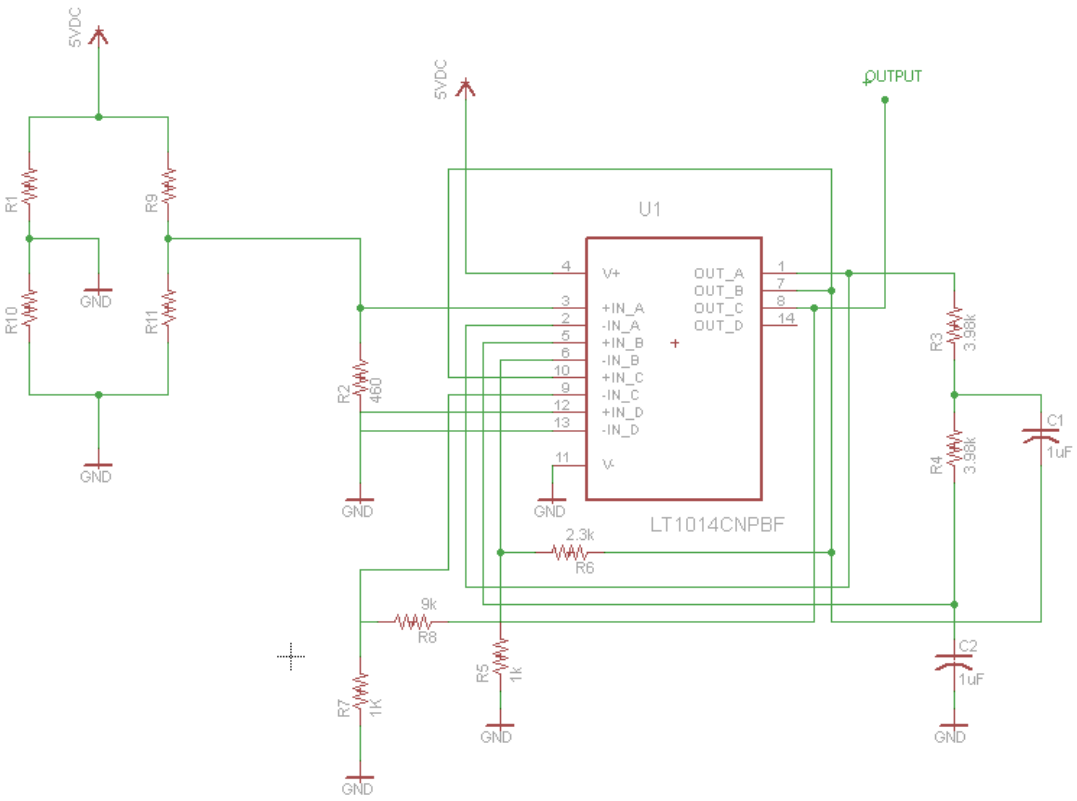


Figure 4.27: *Strain gauge Circuit*

Floats or float switches are a basic solution to measure fluid level. It's a mechanical solution. While not something the group will use for the kettle or mash tun environments provides an easy solution to preventing the overflow of the water tank. The fluid level in the water tank is not as important to the system like the sparge tank level or kettle level.

Phototransistors with LEDs provide an interesting new way of measuring the fluid levels in tanks. The phototransistor acts like a trigger waiting for light to be present of not. An LED is housed with a phototransistor in a dome and waits for liquid to be present. When no liquid is present the LED will be reflected onto the transistor. With fluid present the LED light escapes the dome indicating there is liquid at the sensors position. This however makes the system discrete in measuring fluid level. While very fast, resolution will depending on the numbers of LED-phototransistor pairs are in the system. A device that works like this is the Honeywell LLE Series. This device does integrate directly into a microcontroller freeing up an analog to digital pins on the microcontroller.

Finally a solid state "eTape" for measuring fluid levels in a tank. It is a solid state device that acts like a resistor. The functionality is similar to a thermistor or RTD. As the water level rises in the tank the resistance of the eTape decreases. This drop in resistance is linear in fashion. Figure 4.28 below is shown to display the effects depth has on the resistance. The drawback is that this only work in water

or other low acidic fluid environments. The temperature range on the tape though is -9˚C to 65˚C. This limits the tape to the sparge tank and water tank; however, this covers an electrical solution over the float switch.

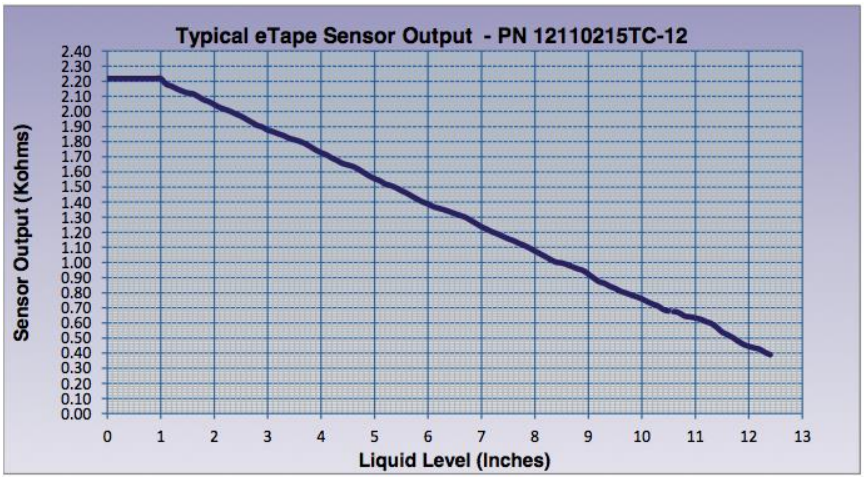


Figure 4.28: *eTape Depth vs. Resistance*

4.5.2 Components

Two of the most important components to the brew extractor are the operational amplifiers and microcontroller(s). The number of microcontroller is limited to the number of pins it has and how the team will unitize them. The microcontroller will be tasked with storage and sending of the data collected by the sensors; however, if the data is not seen clearly or over a large enough scale the user could lose data. Thus the groups need for operational amplifiers.

The operational amplifier will be used in two way. The first is obviously the amplification of the analog signals before it reaches the microcontroller. Many of the signals are two low to see any significant change it voltage for instance of the pH sensors only changes ±460mV per pH level. That does not provide the necessary resolution to track properly considering slight changes only should about 46 mV. The second is the role it will play in the extractor's active filter circuits.

The main purpose of the operational amplifier is to filter out noise for the brew extractors more sensitive measurements. The noise the team is most concerned with is coming from the AC to DC converter driving the system's power. This leads to the operating conditions the operational amplifier is required to have.

The operational amplifier needs to be low power in the milliwatt range. It must also be low noise as low noise as possible. Amplifying the noise will be detrimental to the data acquisition. The final criteria is the operating voltage range. Operational amplifier needs to be able to operate from a single power supply because the team does not plan on using any negative power supplies. The next is that the

operational amplifier can operate with only 5VDC. With the operating range being small, the circuit designs have to take in consideration the outputs coming from the sensors may hit the rails after amplification.

4.5.2.1 Microcontrollers

The microcontroller is an amazingly versatile piece of hardware that is found in nearly any electronic component that requires some form of control, from refrigerators to RC cars, to light switches. The function of the microcontroller in this project is to implement cause and effect situations as well as communicate data to a more powerful central computer. The range of microcontrollers available in today's market is truly vast, with huge scope of options like inputs, outputs, communication protocols, size, and analog to digital converters, led controllers, PWM controllers, and many others. The number of microcontroller options available is so great, that many of the available options are capable of performing the same tasks, at similar price points, footprints, and availability. It is because of this that only a few microcontrollers were compared when venturing to choose the right one. When comparing these microcontrollers, the parameters considered were size, number of analog inputs, number of digital inputs and outputs, communication protocols available, size of memory, and ease of use. The microcontrollers that chosen for analysis were the Texas Instruments MSP4302553, Freescale Semiconductor MC9S08SE8CRL, Microchip Technologies PIC16C73B, and the Atmel ATMEGA328P-PU. Table 4.16 compares some of the hardware features of these microcontrollers. All of the microcontrollers chosen use PDIP packaging so that the group could gain experience soldering printed circuit board through-hole components.

Table 4.16: Hardware Comparison of Select Microcontrollers

	Texas Instruments MSP4302553	**Freescale Semiconductor MC9S08SE8CRL**	**Microchip Technologies PIC16C73B**	**Atmel ATMEGA 328P-PU**
Package	PDIP-20	PDIP-28	PDIP-28	PDIP-28
ADC Channels	8	10	5	6
ADC Bit Size	10	10	8	10
I/O	16	24	22	23
Communication	I^2C, UART, SPI, IrDA	SCI	I^2C, USART, SPI	I^2C, USART, SPI
Supply Voltage	1.8 - 3.6 V	2.7 - 5.5 V	4.0 - 5.5 V	1.8 - 5.5 V
Timers	2	2	3	3
RAM size	512 Byte	512 Byte	192 Byte	2 kByte
Program Memory	16 kByte	8 kByte	4 kByte	32 kByte
Max Clock Freq.	16 MHz	20 MHz	4 MHz	20 MHz

It is clear that a few any of these options could be made to work for the application of controlling the automated brew extractors I/O interface, analog input, and communication. The task of choosing the right microcontroller to use ends up being based more on user experience rather than hardware capabilities, however, the elimination process can place weight on certain hardware parameters to make one microcontroller more desirable than the others. The communication protocol the group decided to use to communicate between microcontroller and the main computer was the I^2C protocol, this decision eliminated the Freescale Semiconductor MC9S08SE8CRL from further consideration as it does not have I^2C communication built into the hardware. Next the decision to use a 5VDC supply was made. Having a higher supply voltage would increase the accuracy of the analog to digital converters onboard as well as allow for the I/O pins to function with more reliability. The Texas Instruments MSP4302553 microcontroller has a maximum supply voltage of 3.6VDC and was therefore eliminated from further consideration. This leaves the final choice between the Atmel ATMEGA328P-PU and the Microchip Technologies PIC16C73B. Comparing some of the key features of these two chips reveals that the Atmel ATMEGA328P-PU is superior in nearly all fields. The Atmel chip has more ADC channels, a higher ADC bit size, more I/O pins, and substantially larger RAM size and program memory size at no extra cost. These factors led to the decision to use the Atmel ATMEGA328P-PU.

The Automated Brew Extractor requires constant reading of 6 different analog signals as well as control of 12 outputs for operating solenoids, pumps, actuators, and heaters. This is achievable with a single Atmel ATMEGA328P-PU; however, in order to make programming simpler for the group, the decision to split the work load onto two separate chips was made. The analog signals that need to be sampled are measuring the weight of subsystems that require certain fluid levels as well as the fluid temperature at different points of the brew process. The temperature and weight measurements are then logically split between the two controllers so that different people are able to work on these programs simultaneously. The I^2C communication protocol is interrupt based and is initiated by the host computer; therefore, timing does not need to be done by the microcontroller and communication between each microcontroller and the host computer will never clash. All unused I/O pins on each microcontroller will be routed to empty output trigger circuits to allow for later device upgrades.

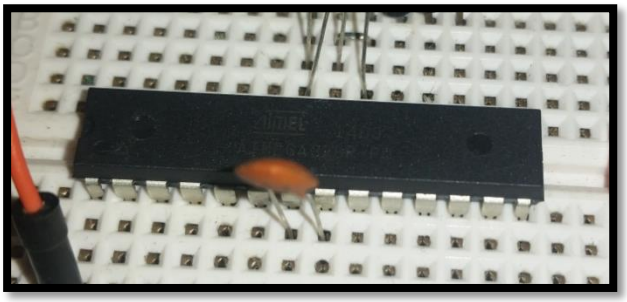


Figure 4.29: *Atmel ATMEGA328P-PU microcontroller.*

The Atmel ATMEGA328P-PU microcontroller, shown in Figure 4.29, is in the AVR family of microcontrollers, most notably used in the popular development board "Arduino". Development for this family of microcontrollers has become so popular due to its use within Arduino development boards, that a large host of programming and development tools are available for these chips. Atmel has released its own proprietary development studio, free of charge, called "AVR Studio", which includes everything necessary to begin development on any AVR chip including line by line debugging. Also due to the popularity of the AVR chips, low cost programmers are available to flash programs onto the chips. All of these factors, as well as Atmel's comprehensive and easy to navigate datasheets, make the Atmel ATMEGA328P-PU microcontrollers an ideal platform for the Automated Brew Extractor.

4.5.2.2 Operational Amplifiers

As with any application using analog sensors is needed to amplify the signal output for it to be of any use to the system. The operational amplifiers will act as unity buffers between operating stages. The signals also need to be filtered to reduce the voltage noise coming from the other components in the system. For the most part low pass active filters will be used 2nd order filters will be use. The planned type of filter is a Butterworth filter. Figure 4.30 below is an example of a 40Hz low pass filter. The team has shown other examples of how the system will implement the operational amplifiers for the sensors.

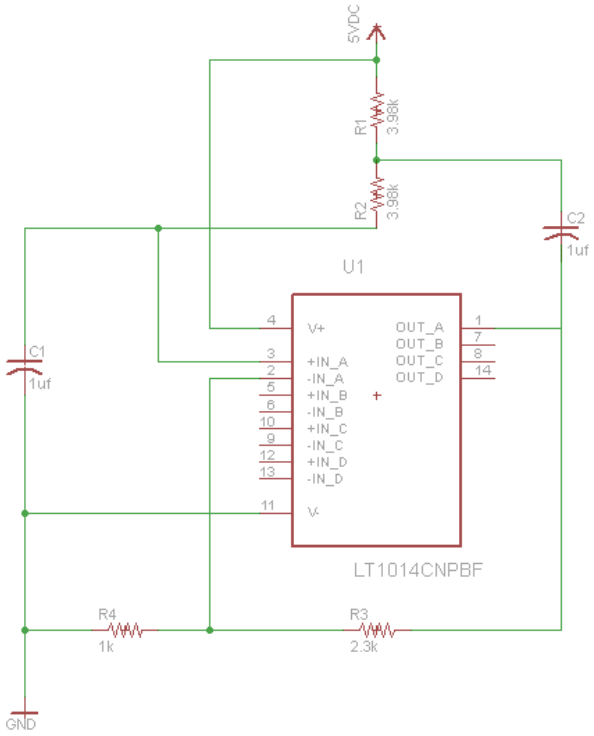


Figure 4.30: *Second Order Butterworth Filter*

Several operational amplifiers (operational amplifiers) were considered for the brew extractor. The operational amplifiers needed to be unipolar because the application lacks a negative voltage source. The circuits are also used as if normal amplification is needed on that powered side. It needed to be low power and be able to run on 5 V_{cc}. The operational amplifiers that met the criteria: TI LM158, LT1006, LT1013DN, and the MCP60x family of op amps.

The TI LM158 is the first operational amplifier the group considers for the amplification stage and filter designs. The real draw of this operational amplifiers the low power consumption has compared to other operational amplifiers. It comes packaged only in a dual chip. It requires the highest turn on voltage but well within the voltage supply range of 5V. It is also the most expensive of the operational amplifiers considered. The group plans to order in bulk and that price does not agree with the budget.

The Linear Technology LT1006 is the next operational amplifier. Very similar in performance to the LM158J. It has a voltage supply range 2.7V to 22V. This is well within the operational range. At max operational voltage the most power it will draw is 3.4mW. This is the highest draw out of the three operational amplifiers. Another down side is it only comes with one operational amplifier per chip. Another chip is the same operational ranges is the LT1013DN; however, this chip is available in dual/quad chips. An added bonus is that it draws less power than its single operational amplifier brother. The downside to this operational amplifier is that it requires 5V to operate. For the system that give no leeway in the voltage supply and presents the chance the operational amplifier won't turn on.

The Microchip MCP60x family of operational amplifiers. The reason the group looked at the family of operational amplifiers because the only difference between the chips was the amount of operational amplifiers on one chip. The operating range on this operational amplifiers is the smallest at 2.7V to 6V. It has a medium power draw on the system compared to the other operational amplifiers. This chip is also ideal for the driver amplifiers for the ADC pins on the microcontroller. The real draw this operational amplifier has is the price tag to the other features it has. At only 44¢ it is the cheapest in the bunch. The negative to this chip is the lack of dual/quad chips. Meaning that these chips will take up more space on the teams PCB. Because of the small operating range though the group will have to be careful with the tolerance of this operational amplifier. Table 4.17 below shows a comparison of the operational amplifiers researched for filters and amplification. Parameters such as power dissipation, price, and supply voltages are considered.

Table 4.17: Operational Amplifier Comparison				
Operational Amplifier	**Supply Voltages(V)**	**Max Power Dissipation (mW)**	**Price ($)**	**Number of Amps/chip**
MCP60x	2.7-6	1.65	0.44	1
TI LM158J	3-32	7.50E-04	7.74	4
LT1006	2.7-22	3.4	1.92	1
LT1013DN	5-22	2.75	2.67	2
LT1014DN	5-22	2.75	4.17	4

4.5.3 Output Circuit Designs

The I/O circuit designs represent the integration of components being controlled by the microcontroller as well as the components that trigger cause and affect situations within the microcontroller programming. A thermocouple cannot simply be connected to the microcontroller input and be expected to get accurate temperature readings. The analog inputs must be amplified and filtered to obtain as consistent and accurate data as possible. Similarly, microcontrollers are incapable of operating devices which requite > 5VDC and/or >20 mA. The outputs must have circuitry that will successfully trigger a high voltage and/or high current device while isolating the microcontroller from the damaging effects of the high voltage or high current. The following subsections review how these goals are achieved as well as outline a plan for the I/O ports and ADC port on each of the microcontrollers being used.

In this section the design process leading to a finalized "standard" microcontroller controlled output circuit will be reviewed. The function of these standard circuits is to have a default circuit that will take a very low current low voltage signal from an output pin of the microcontroller and will be able to activate some of the higher power electrical components used throughout the system, components such as: resistive heaters, solenoid valves, relays, and actuators.

Microcontrollers are, in general, very low power devices. In fact, most microcontrollers are only able to output signals in the 5ma-25ma range due to their small size and low cost. This creates a problem for someone looking to switch on a solenoid valve or control a heater using a microcontrollers output ports. Microcontrollers also run on relatively low input voltages, in the range of 1.8V-5.5V, which limits the types of electronics they can directly power or control with the output ports. Often the devices that need be switched on or off are being powered by higher voltages. It is for this reason that it is important to implement some form of isolation between the microcontroller output circuit and the higher voltage circuit that is being controlled.

Two main devices will be utilized in the following designs that provide high degrees of circuit isolation as well as control. First is the electromechanical relay. A relay utilizes the magnetic field created by a properly designed coil to physically pull or push contacts in a different circuit. The amount of electromagnetic force required to move the contacts in the other circuit will tend to be greater when the contacts can handle a greater current. In a roundabout fashion, the current required on the coil side of the relay is related to the load capability of the output side of the relay because of the change in electromagnetic force required to move these contacts.

The other circuit isolation technique used is the opto-isolator. An opto-isolator can take many forms but only two will be discussed in this text, those are the optocoupler and the optotriac. An optocoupler is a 4-pin device with a 2 pin input side and a 2 pin output side. This device functions by placing a light emitting diode on the input side and a BJT on the output side, where the base of the BJT is biased by the input side LED. This creates a device that allows current to pass through the output side only when the LED on the input side is lit, all while being completely separate circuits. An optotriac functions in a very similar manner. The main difference between the optocoupler and the optotriac is that the optotriac has a triac on the output side rather than a BJT. A triac can be modeled as a pnp transistor connected to an npn transistor by joining the base of the pnp transistor with the collector of the npn transistor and the base of the npn transistor to the emitter of the pnp transistor. This creates a 4 layer device that is able to pass an alternating current signal through its output when the gate is biased. The gate of the triac is nearly synonymous with the base of the BJT in the optocoupler, in the sense that when the LED on the input side is turned on, the gate/base is biased to allow current to flow through the output of the device.

Given the previous circuit isolation devices, a total of four output "turn-on" circuits were considered. It is important to note that each of the four following example circuits assume the microcontroller used has the current output capability to light an LED or bias a BJT directly. In each of the 4 following options VCC is assumed to be the operating voltage of the microcontroller and therefore equivalent to the output voltage at the microcontroller output pin. VDD is considered to be a DC voltage greater than VCC, and 115VAC is mains voltage connected directly to a household wall socket or the power supplies mains throughput.

The first option utilizes a low power optotriac to enable the gate of a larger power triac. The power triac then allows current to pass through it to the load, which in this and the following three cases, has been modeled as a solenoid valve because that will be the most common load in the system, the circuit is shown in Figure 4.31. Inductive loads such as solenoid valves tend to cause large spikes in current when initially triggered, for this reason an RC snubber is connected in parallel to the power triacs output terminals to protect it from damage. Since triacs are typically used for switching of AC sources, the solenoid in this example circuit is shown to run directly from 115 VAC mains voltage, but it could easily be any other device that is powered by mains voltage.

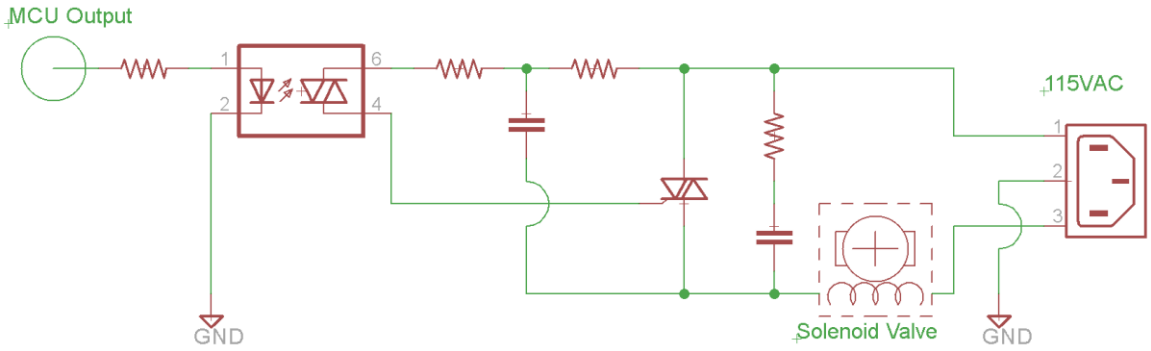


***Figure 4.31**: Output trigger circuit #1, used for switching 115 VAC.*

The second output circuit once again utilizes an optotriac for high voltage isolation from the microcontroller circuit. This circuit; however, assumes that the designer is able to find an optotriac with an output current rating high enough to run the solenoid valve directly. The RC snubber circuit is still used to protect the optotriac in this circuit. An optotriac that has the capacity to run the solenoid valve directly may require a strong LED to bias the gate on the triac internally; because of this, it may be wise to power the LED with a direct connection to a 5V power source and use a low power BJT to switch the LED on and off according to the microcontroller output. In Figure 4.32 it is assumed that the microcontroller is able to put out enough current to activate the optotriac fully.

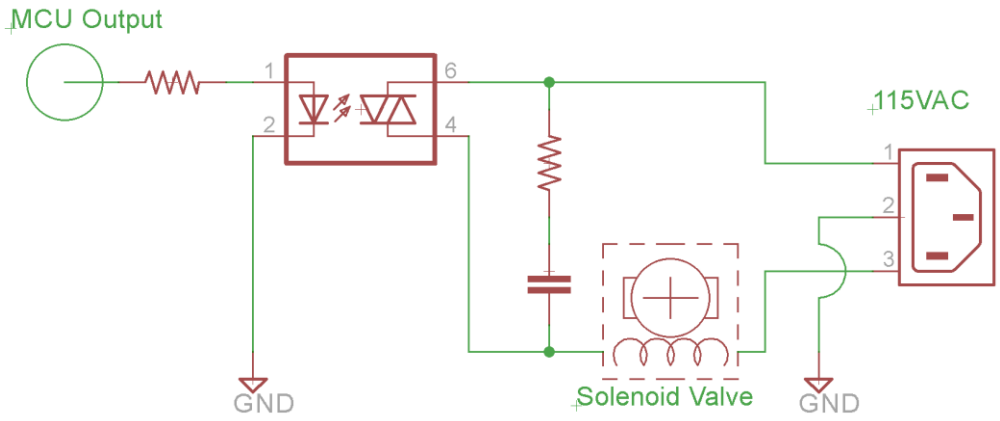


***Figure 4.32:** Output trigger circuit #2, used for switching 115 VAC.*

The third output circuit considered utilizes a relay for voltage isolation between the microcontroller circuit and the higher voltage circuit. This circuit is the most versatile circuit for switching higher power loads because by choosing a properly sized relay and BJT, nearly any sized load at any reasonable voltage, AC or DC, can be switched on; the circuit diagram is shown in Figure 4.33. In this circuit, the microcontroller output pin is used to bias a BJT into conducting which then grounds the relay. The relay coil acts an inductor and is capable of storing a charge, this can be damaging to the BJT. The diode placed in parallel with the relay coil is placed there to protect the BJT from charge stored by the relay coil. When the magnetic field around the coil is building or collapsing, creating a spike in

voltage/current, the diode causes the charge to dissipate as heat through the coil itself as well as the diode. It is also important to note that in this case, V_{cc} is equivalent to the microcontroller power input voltage, and is therefore equivalent to the microcontroller's output port voltage. V_{dd} on the other hand, represents any voltage, AC or DC that is used to power virtually any load.

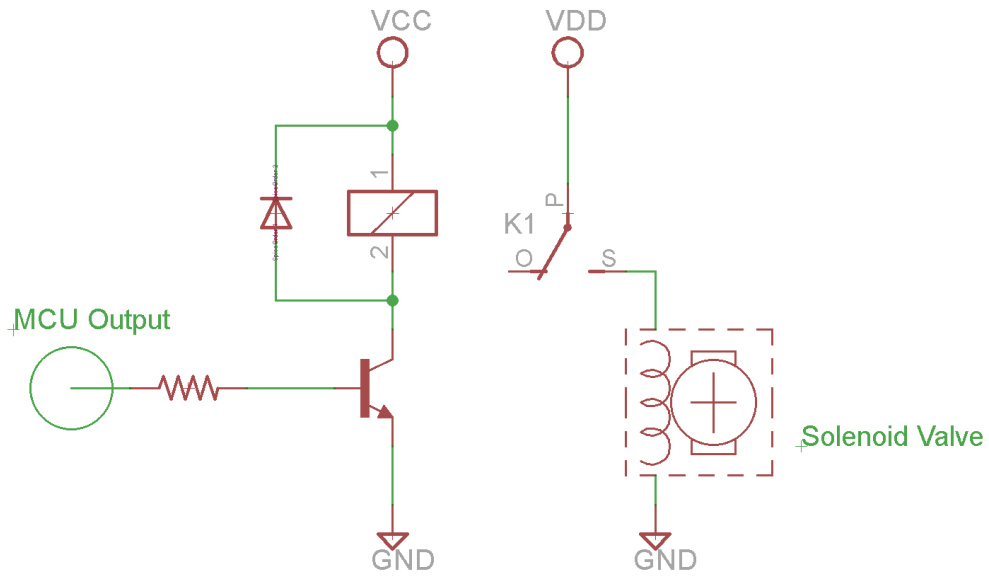


Figure 4.33: *Output trigger circuit #3, used for switching any reasonable AC or DC voltage.*

The fourth and final output circuit considered utilizes an optocoupler for voltage isolation between the microcontroller circuit and the higher voltage circuit. This circuit is just as versatile as the previous circuit for switching higher power loads because it is able to trigger solenoids that require an AC voltage source, however, the circuit is more robust in the sense that it separates high voltage and high current from the microcontroller circuit using both an optocoupler and a relay; the circuit diagram is shown in Figure 4.34. In this circuit, the microcontroller output pin is used to bias a BJT into conducting which then grounds the optocoupler input. The optocoupler input side is then triggered which grounds the relay coil which is operating at a higher voltage, therefore requiring a lower current to pass through the output side of the optocoupler. Finally this completed conduction path for the relay coil triggers the output of the relay to enable the solenoid. The relay coil, once again, acts an inductor, this can be damaging to the optocoupler. The diode placed in parallel with the relay coil is placed to protect the optocoupler from charge stored by the relay coil when the magnetic field around the coil is building or collapsing, creating a spike in voltage/current, the diode causes the charge to dissipate as heat through the coil itself as well as the diode.

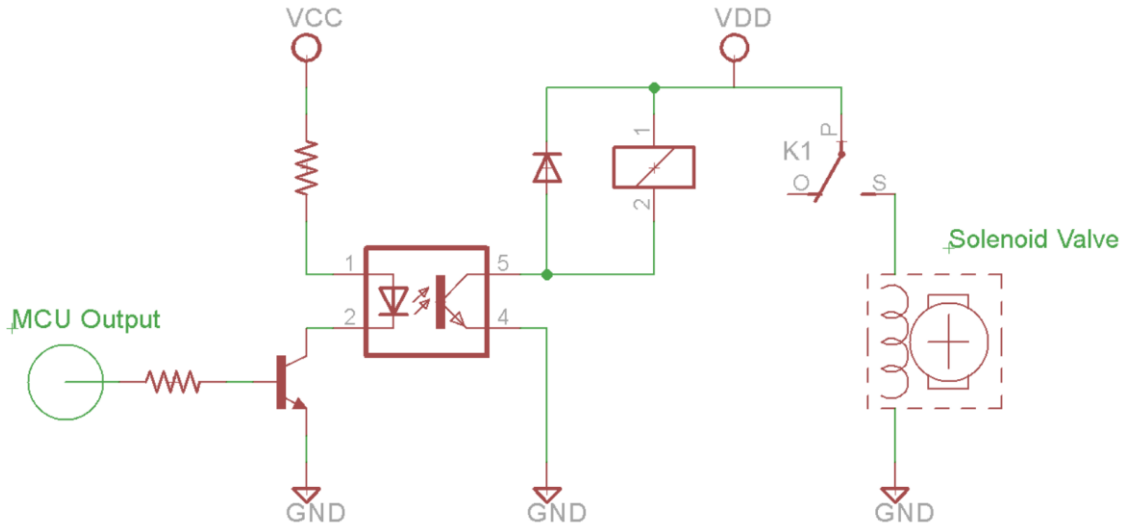


Figure 4.34: *Output trigger circuit #4, used for switching DC voltage equal to the solenoid trigger voltage (special case).*

The four output circuits designed are summarized and compared below in Table 4.18. Each output trigger circuit configuration is analyzed to determine which one will be most suitable for the application of triggering solenoids, actuators, and heating elements by considering elements including but not limited to: bill of material count, bill of material cost, versatility, and durability.

Table 4.18: Trigger Circuit Comparison

	Pros	Cons
Trigger Circuit #1	• Durable for switching AC loads.	• High BOM count. • High BOM cost. • Only AC loads.
Trigger Circuit #2	• Low element count. • Simple circuit design.	• Operating near device limits. • Only AC loads.
Trigger Circuit #3	• Useful for switching any AC or DC loads.	• Operating near device limits. • High BOM cost.
Trigger Circuit #4	• Durable for switching AC or DC loads.	• High BOM count.

The decision was made to eliminate as many AC loads as possible in order to reduce issues with noise, shielding, and switching wear. Since trigger circuit #4 can be easily adapted to switch any kind of load and has the versatility as far as design goes to allow operation in ranges far from maximum ratings, it is chosen to do the switching for the bulk of the output operations. Figure 4.34 shows the design when triggering a load that operates at the same DC voltage as the relay coil, but this design can easily be adapted to trigger high current AC loads by removing

VDD from the common pin on the relay and applying the AC source to that pin instead. This design then isolates the low DC voltage microcontroller from the higher DC voltage relay trigger circuit, which is isolated from the mains voltage via said relay. This is the type of adaptation that is required to operate the heating element in the boil kettle as it will most likely be a resistive AC load. The adapted circuit design is shown in Figure 4.35.

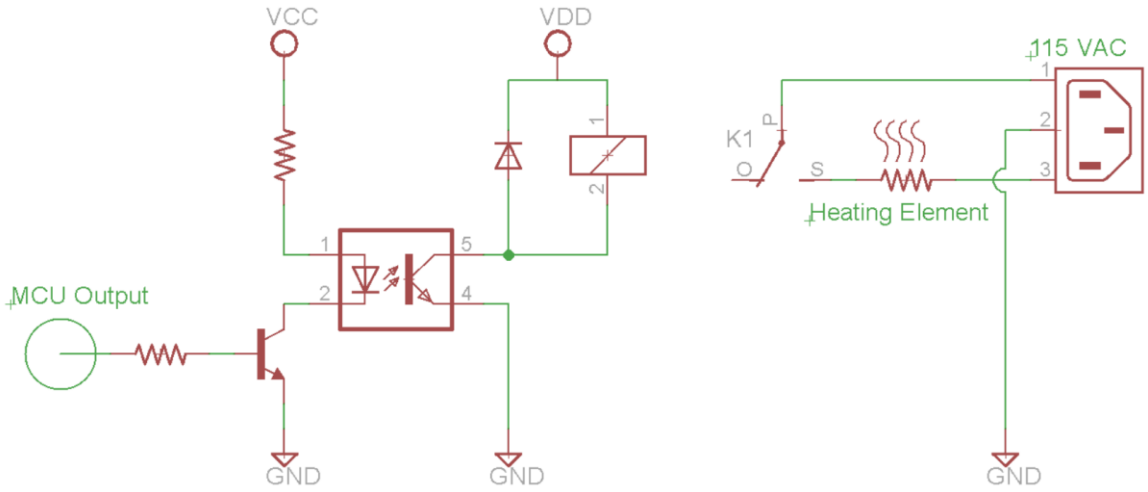


Figure 4.35: *Output trigger circuit #4, used for switching an AC resistive heating element (special case).*

This circuit does have a larger bill of materials/element count than most of the other options, but it was determined that the versatility and durability of the circuit outweigh the minor price that will be paid in board space and element cost.

When choosing elements to turn this conceptual circuit design into a functional bill of materials, certain assumptions must be made. Current through the IC's, electro mechanical, and electro optical devices is the main concern when choosing components. In order to keep current as low as possible, devices to be switched will operate at a reasonably high DC voltage. The highest common DC voltage used in the industry is 24VDC, which devices are commonly made to operate at. It is assumed that the common solenoid used throughout this project will be one with coil operating parameters that are shown in Table 4.19 below.

Table 4.19: Solenoid Operating Parameters

Voltage	Current	Power	Coil Resistance
24 VDC	333.3 mA	8 W	72 Ω

Knowing these operating parameters, it is possible to begin finding components for the design. A relay that can handle over 333mA of current on its output side and consumes as little current as possible on its input side is ideal. Next after finding the proper relay, an optocoupler with a high output current capability and a low input current draw is necessary. The transistor chosen to operate the

optocoupler input must be capable of being saturated by the microcontroller output voltage, VCC, which will be assumed as 5VDC. Finally the resistor values must be chosen to ensure proper operation. The final bill of materials per solenoid is shown in Table 4.20 along with key features considered in the design.

Table 4.20: Trigger Circuit Bill of Materials

Part	P/N	Design Features	Cost
Relay (low contact current)	OJ-SS-124LMH2	• I_{coil} = 8.3 mA (nominal) • Rated Current = 3 A (max)	$1.12
Relay (high contact current)	FTR-K3JB024W	• I_{coil} = 31.2 mA (nominal)	$2.21
Optocoupler	FOD817A	• $I_{collector}$ = 50 mA (max) • $I_{forward}$ = 20 mA (nominal) • $V_{forward}$ = 1.4 V (max)	$0.43
Transistor	2N3904BU	• $I_{collector}$ = 200 mA (max) • V_{ce} = 0.3 V (max)	$0.19
Resistor 1	MFR-12FTF52-1K	• R = 1000 Ω (1% tolerance)	$0.12
Resistor 2	RN55D1650FB14	• R = 165 Ω (1% tolerance)	$0.10
Diode	1N4149TR	-	$0.09

The bill of materials for the default output trigger circuit shows two different relays, a low contact current relay, and a high contact current relay. The purpose of having two different relays rather than solely using the high contact current relay is to save on utilized board area. As will be discussed later in this report, manufacturing of custom printed circuit boards is typically charged by the square inch. This fact alone is reason enough to attempt to create a board layout that functions properly with as small of a footprint as possible. The low current relay will not be able to switch devices such as heaters and larger pumps, but the low current relay has a much smaller footprint and will suffice for actuators and solenoid valves.

4.5.4 I/O Pin Layout

In this section the input and output pins of the microcontrollers will be discussed in a specific manner. The pin inputs and outputs will be split between two microcontrollers as discussed previously in section 4.5.2.1. The temperature and fluid level measurements will be split between the two microcontrollers which will be referred to as MCU1 and MCU2, respectively, from this point on.

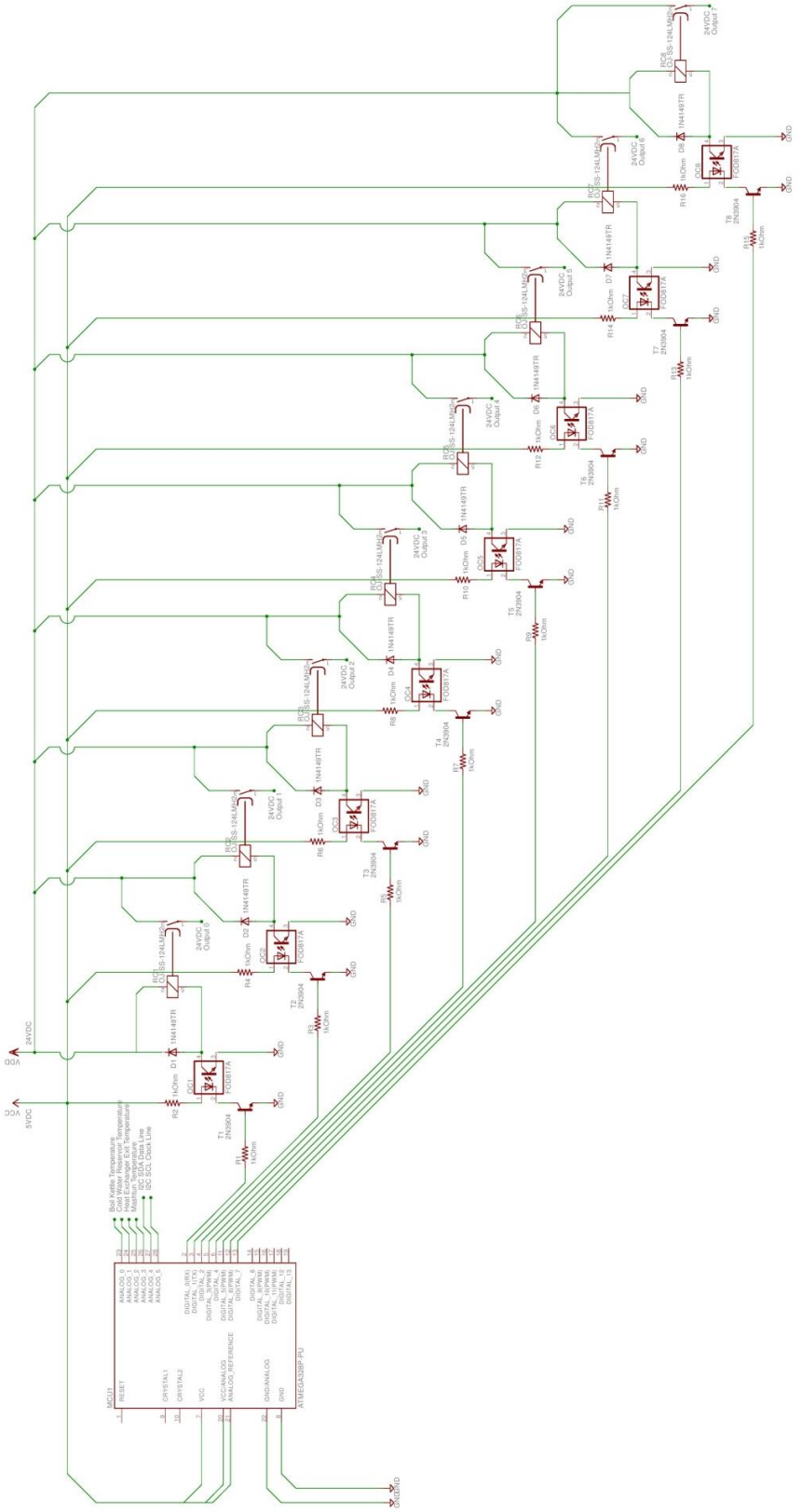


Figure 4.36: *MCU1 output port circuit.*

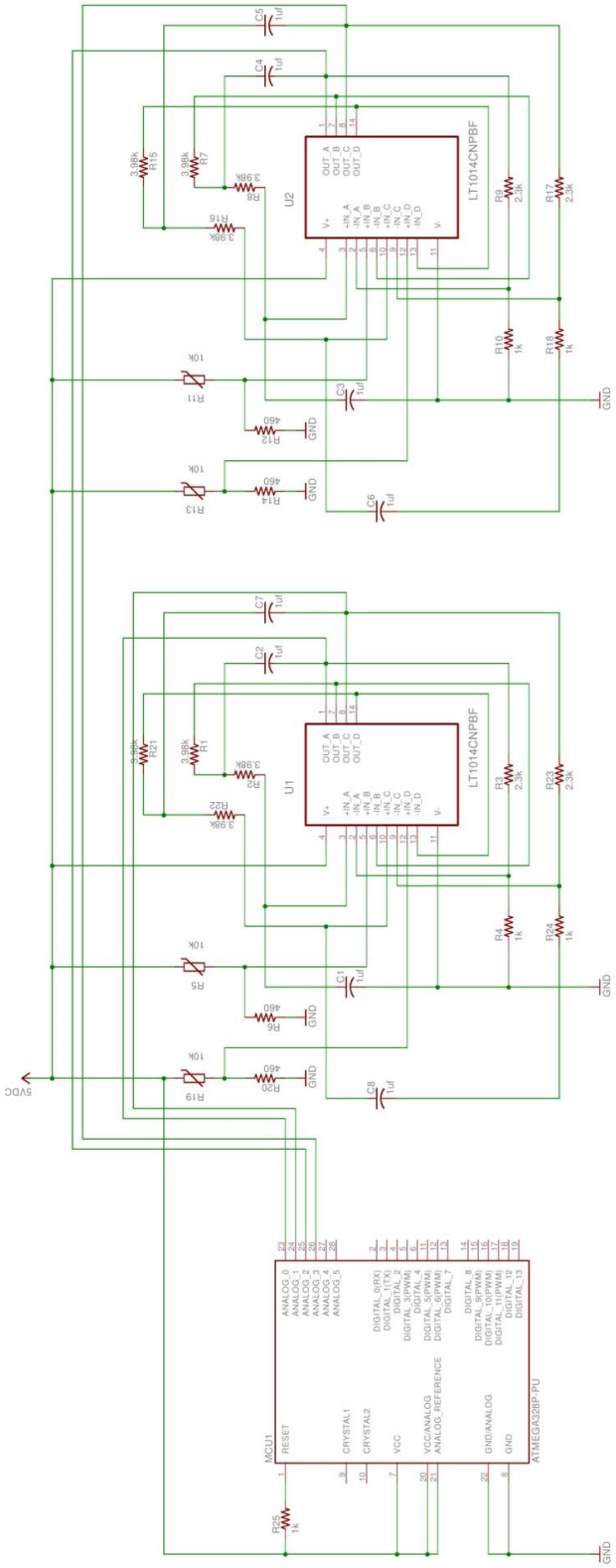


Figure 4.37: *MCU1 analog input port circuit.*

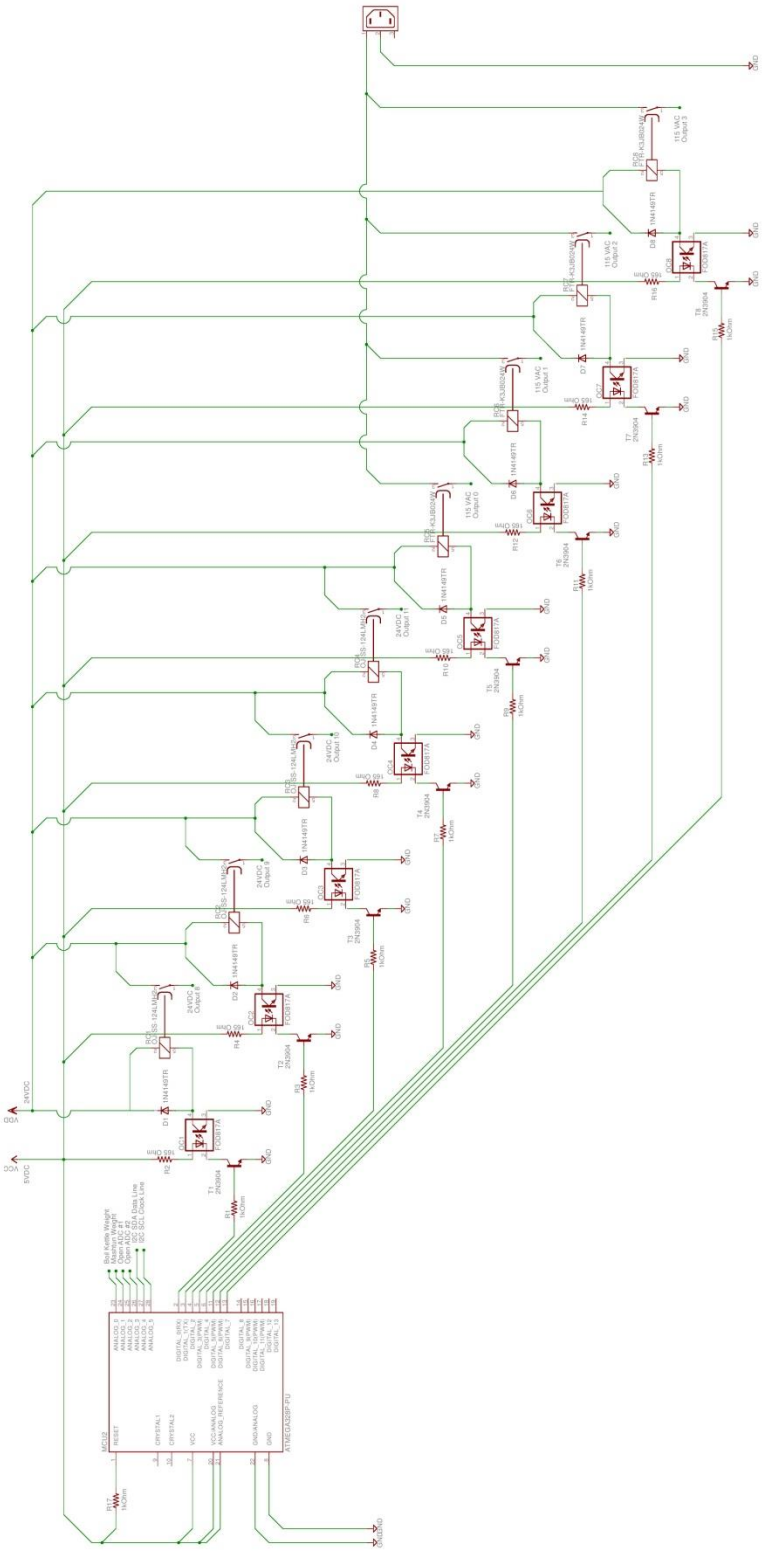


Figure 4.38: *MCU2 output circuit.*

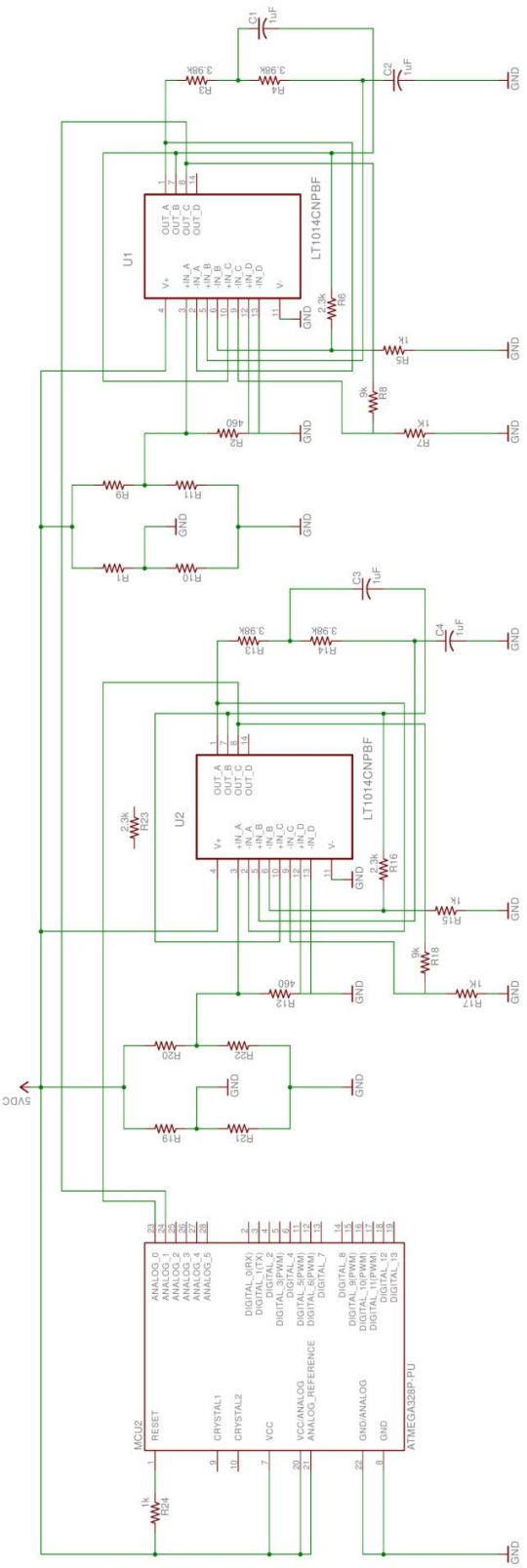


Figure 4.39: *MCU2 analog input port circuit.*

Figure 4.36 shows the MCU1 output circuit diagram, all of the transistors connected to the microcontrollers I/O port will be connected via a cable, all of the circuitry beyond the transistors will be placed on an entirely separate board in order to keep the high voltage away from the microcontroller and operational amplifier circuitry. Figure 4.37 is a continuation of the MCU1 I/O design showing the analog temperature measurement circuits with amplification and filters. Figure 4.38 displays the output circuit for MCU2, four of the outputs are connected to low current relays and the remaining four outputs on the same port are connected to high current relays which trigger 115VAC mains voltage. This totals up to 12 low current relays on 24VDC and 4 high current relays switching 120VAC. Figure 4.39 shows the analog inputs for MCU2 which are the strain gauges that are used as fluid level sensors. Finally Table 4.21 explains which pins/ports are dedicated to which measurements.

Table 4.21: Microcontroller Pin Assignments

Pin #	Port	Function
MCU1		
2	D.0	Low current output trigger. (Solenoid #1)
3	D.1	Low current output trigger. (Solenoid #2)
4	D.2	Low current output trigger. (Solenoid #3)
5	D.3	Low current output trigger. (Solenoid #4)
6	D.4	Low current output trigger. (Solenoid #5)
11	D.5	Low current output trigger. (Solenoid #6)
12	D.6	Low current output trigger. (Solenoid #7)
13	D.7	Low current output trigger. (Solenoid #8)
23	C.0	Analog temperature. (Boil Kettle #1)
24	C.1	Analog temperature. (Boil Kettle #2)
25	C.2	Analog temperature. (Heat Exchanger Exit)
26	C.3	Analog temperature. (Mashtun)
MCU2		
2	D.0	Low current output trigger. (Coolant Pump)
3	D.1	Low current output trigger. (Open)
4	D.2	Low current output trigger. (Open)
5	D.3	Low current output trigger. (Open)
6	D.4	High current output trigger. (Wort Pump)
11	D.5	High current output trigger. (Heater)
12	D.6	High current output trigger. (Open)
13	D.7	High current output trigger. (Open)
23	C.0	Analog weight. (Boil Kettle Liquid Level)
24	C.1	Analog weight. (Mashtun Liquid Level)

4.6 Brew Extractor Power Supply

In this section, the design process for creating the Brew Extractor's main power supply is reviewed in detail. The role of the power supply in this project is to supply power to every component within the system in a safe, stable, and reliable manner. The challenge in the power supply design for this system is the different voltage potentials employed throughout the system as well as their varying loads. Table 4.22 below shows the different voltages being utilized and some of the components in the system that will function on that specific voltage.

Table 4.22: List of Required Power Supply Outputs

Voltage	Use
115 VAC	Mains system input, resistive heating element, solenoids
24 VDC	Solenoid valves
5 VDC	Microcontroller power, analog reference voltage, sensor amplifier power
3.3 VDC	I^2C communication, raspberry pi

The design of the power supply begins with the current requirements at each specified voltage. The current outputs for each voltage is estimated by roughly summing the maximum current ratings for the components running on that respective voltage. For example, at 24 VDC only solenoid valves will be drawing power from the power supply; therefore, the estimated current requirement from the 24 VDC rail of the power supply design is the sum of the total current drawn by every solenoid valve. It is always good to have a factor of safety as well in the event that some component fails, the system will have the headroom to push out as much current as necessary in order to safely activate a fuse or breaker which will shut that specific rail down.

If the maximum current capable of being drawn from a certain rail is in the range of 1-1.5A, it is then possible to use voltage regulating integrated circuits, like the LM78xx series chips, for a reliable and stable voltage output device. These integrated circuits are very economical and easy to use, which is one reason they are very common in many circuits that require voltage regulation. Another good reason to use this integrated circuits, is that they have a multitude of different protection circuits built in to the chip that will shut it down when certain ratings are exceeded. To use the LM78xx series voltage regulators as an example again, they employ a thermal overload protection circuit as well as a short circuit protection. The amount of power that the IC needs to dissipate is shown in equation below.

$$P_{diss} = (V_{in} - V_{out}) * I_{load}$$

These regulators, essentially dissipate the excess voltage beyond their set output voltage as heat, while the total heat is a function of the current draw from the load. In the power supply being designed, three of these regulators could be used in a

series formation with "outputs" in between each stage, but power dissipation becomes an issue. Take, for example, the largest voltage difference between any two DC rails. In this power supply, that would be the 24 VDC and 5 VDC rails. Assuming the 5 VDC and 3.3VDC cascaded sources are able to draw 1.5A of current that leaves the 5 VDC linear voltage regulator to dissipate:

$$P_{diss} = (24 - 5) * 1.5 = 28.5\, watts$$

That is a huge amount of power to dissipate, even with a very large heat sink. This type of configuration would also limit the maximum current draw of the parent regulator when the lower tier regulator begins to draw more current. The combination of these problems will lead to a severely handicapped power supply and an overall poor design. The circuit in bypasses the tiered relationship between the regulators allowing each to draw 1.5A on its own, but the power dissipation issue is worsened. These regulators have a minimum dropout voltage of around 2V, meaning, the input voltage to the regulator is required to be at least 2V higher than the output voltage in order to operate. Using these numbers, if the 3.3V regulator were to draw 1.5A it would need to dissipate:

$$P_{diss} = [(24 + 2) - 3.3] * 1.5 = 34\, watts$$

This form of power supply design is also very poor and would not perform adequately. These also assume that 1-1.5A is adequate current output which simply is not the case for each voltage.

Another method of regulating a voltage is with the use of a "zener diode". A zener diode utilizes the breakdown voltage characteristics of a reverse biased p-n junction. The breakdown voltage can be manufactured to almost any reasonable voltage. In the case of a power supply, the zener diode would simply be used as a reference voltage while a bipolar junction transistor would be used to amplify current. This type of supply is still unable to supply enough current to outperform the voltage regulator ICs discussed previously. It is for this reason that a second transistor is added to the design. Now, instead of the transistors DC current gain being restricted to "ß" of one BJT, the cascaded configuration of transistors increases the DC current gain capabilities to "$ß^2$". This transistor configuration is known as a "Darlington Pair", and they are available in a single package.

The output voltage of this supply will be regulated to a value which is calculated by subtracting the voltage drop across the base of the darling pair from the zener voltage of the diode. The power consumed by the Darlington pair transistor package is then exactly the same as the power dissipation formula for the regulator ICs discussed above. The advantage to using this circuit instead of the regulator integrated circuits is that the maximum current draw is now determined by the transistor specifications, which are typically able to handle a much higher current than the integrated circuits.

Both of the voltage regulation methods that have been discussed thus far assume an unregulated DC input as V_{in}, but how does the system get that input? What is readily accessible in all households is 115 VAC mains power. This is going to be

the source for the power supply being designed. In order to convert the alternating voltage source into a direct current source with a lower voltage, a power transformer, rectifier diodes, and filtering capacitors must be used.

The transformer, rectifier, and filter combination can be viewed as the "unregulated" DC power supply, because the output signal after the filter will be a jumpy DC signal. Next the design of the unregulated supply will be reviewed.

With a transformer, the designer is able to dictate the output voltage based solely on the ratio between the number of windings on the primary side of the transformer (where the 115 VAC is connected) to the number of windings on the secondary side of the transformer. The relationship between the output voltage and number of windings is proportional.

The fact that it is so simple to increase or decrease the voltage on the secondary side of the transformer allows for transformer manufacturers to mass produce transformers with nearly any reasonable voltage output. This fact allows the design to leave this parameter as a variable which can be manipulated after the design is near finalization without much consequence. The rectifier to be used will be a simple 4 diode configuration which will effectively rid the power supply of negative voltages.

The addition of a filter will clean the output of the rectifier such that voltage will reach the peak of the sinusoid, then, rather than drop immediately, it will stay relatively close to the peak voltage as long as the current draw from the power supply stays low enough. In order to achieve this, a smoothing capacitor is used. The capacitor can only store so much charge, therefore the size of the capacitor used will decide the rate of voltage decline between sinusoidal peaks. This rate is also a function of current draw, because the larger the current draw, the faster the smoothing capacitor will discharge. The calculation for the capacitor value to use can be shown as a function of these elements:

$$C = \frac{I_{load}}{f * V_{ripple}} \quad \text{or} \quad V_{ripple} = \frac{I_{load}}{f * C}$$

Where I_{load} is the current draw from the unregulated supply, *f* is the frequency of the rectified signal, and V_{ripple} is the peak-to-peak differential voltage between the peaks of the rectified sinusoid and the lowest voltage reached before the next sinusoidal cycle recharges the capacitor. An important note is that now that the signal being smoothed is a rectified signal, it is no longer the same frequency as the mains voltage. The frequency of the rectified signal is now double that of the input signal, so in the case of a standard 60 Hz mains line that has been rectified with a full wave rectifier, the frequency used in this formula is 120 Hz.

Increasing the size of the smoothing capacitor can help to eliminate the ripple, but with an increasing load, the capacitor required to reduce the ripple becomes far too large. In this respect the design must strike a balance between smoothing capacitor size and acceptable voltage ripple. This unregulated supply ripple could have an effect on the regulator circuit if the voltage dips low enough.

In order to obtain a highly regulated 24VDC output a tool called WEBENCH® created by Texas Instruments was utilized to automatically create regulated 24VDC output. The tool is able to create multiple designs with varying efficiencies, footprints, bill of materials count, and total cost to suit nearly any application given the users input constraints.

The output of the unregulated DC power supply is assumed to be a target of 25.25VDC with a maximum ripple of ± 0.75V making the range of possible regulator input voltages to 24.5 - 26V. The maximum current output is assumed to be 3A, much greater than what will most likely be necessary, but will provide good headroom. Putting these figures into the WEBENCH® power supply design tool yielded a total of 16 different designs. Some of these designs had the possibility of becoming unstable so they were eliminated from consideration. Three designs were chosen to be compared. These reference designs are shown below in Figure 4.40(a), (b), and (c) respectively.

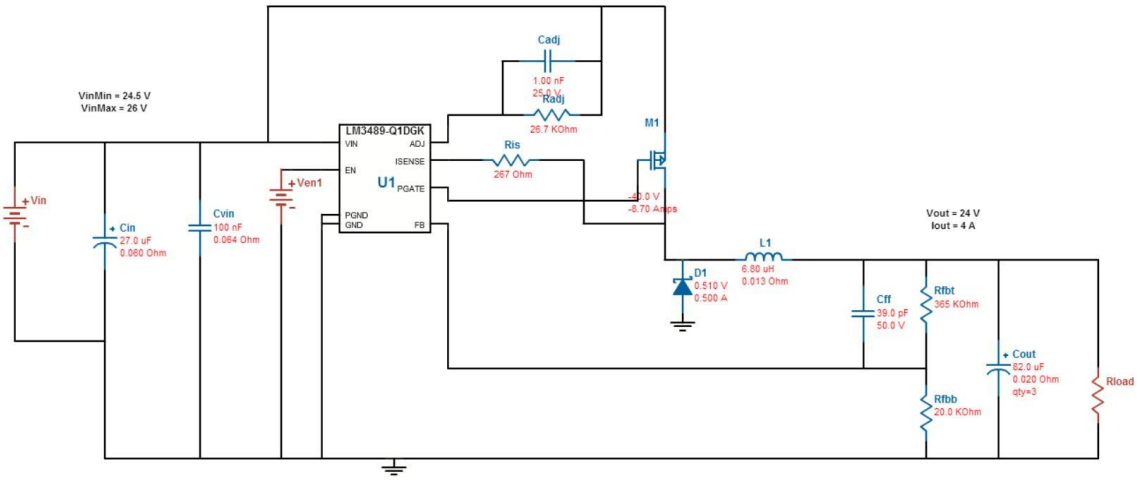


Figure 4.40(a): *Reference design 1 for 24VDC power supply.*

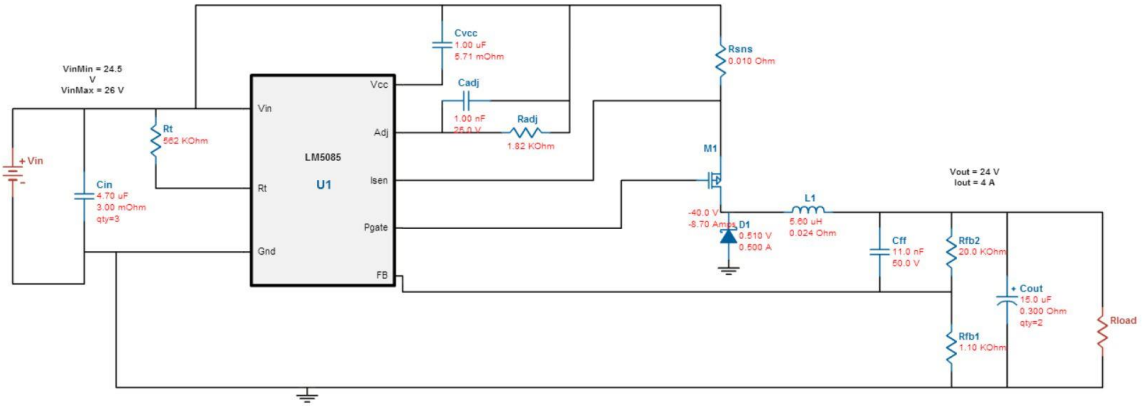


Figure 4.40(b): *Reference design 2 for 24VDC power supply.*

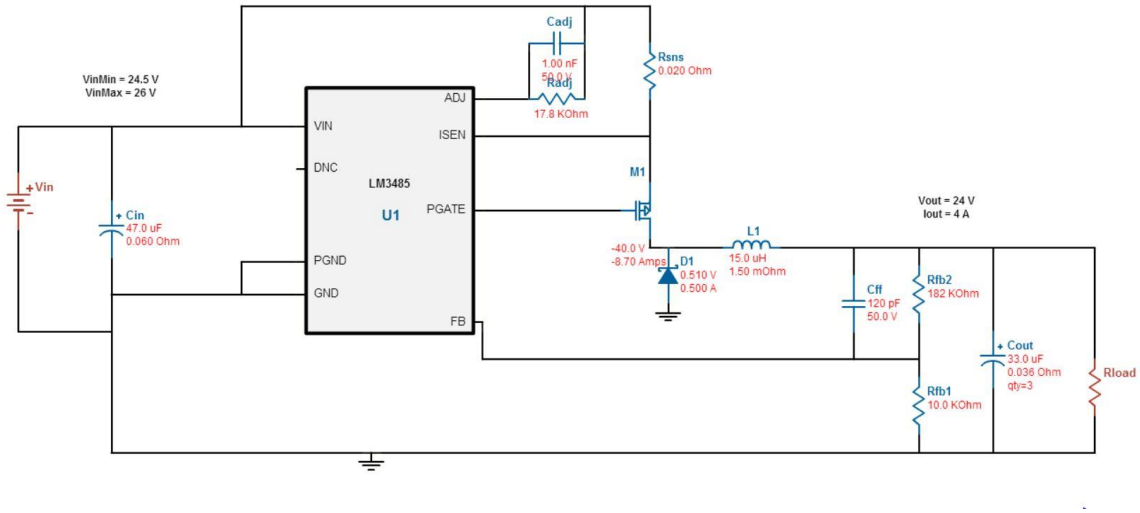


Figure 4.40(c): *Reference design 3 for 24VDC power supply.*

All of the designs chosen are buck converter topologies. These designs were chosen because they represent the strengths and weaknesses associated with power supply design. While these designs look very similar, efficiencies, footprints, bill of materials count, and total cost of each design is different. These parameters are compared for each of the designs below in Table 4.23.

Table 4.23: Buck Converter Power

	Efficiency	Footprint	BoM Count	Cost
Design 1	99%	617 mm^2	15	$4.76
Design 2	98%	337 mm^2	17	$7.59
Design 3	99%	1329 mm^2	14	$4.39

Any of these designs would work well for the application. Design 3 was chosen due to its larger footprint making it easier to make changes to the board if something were to go wrong, as well as the low cost and low element count. Since this supply is a 24VDC, the microcontroller and I^2C bus still require 5VDC and 3.3VDC. These voltages will not be used for high current purposes, so simple linear regulators discussed earlier will suffice for these applications. In the case of the 5VDC supply, two separate regulators will be used, one powering the microcontroller board, the other powering the output trigger board. The combination of the unregulated supply ~25VDC supply, regulated 24VDC supply, 5VDC linear regulators, and 3.3VDC linear regulator creates the full power supply for the Automated Brew Extractor. Figure 4.41 demonstrates the efficiency of the power supply 24VDC rail against current output. Figure 4.42 shows the power dissipated by all of the components in the 24VDC power supply rail. The schematic for the full design is shown in Figure 4.43. A full bill of materials for the components used in the power supply design is shown in Table 4-6.

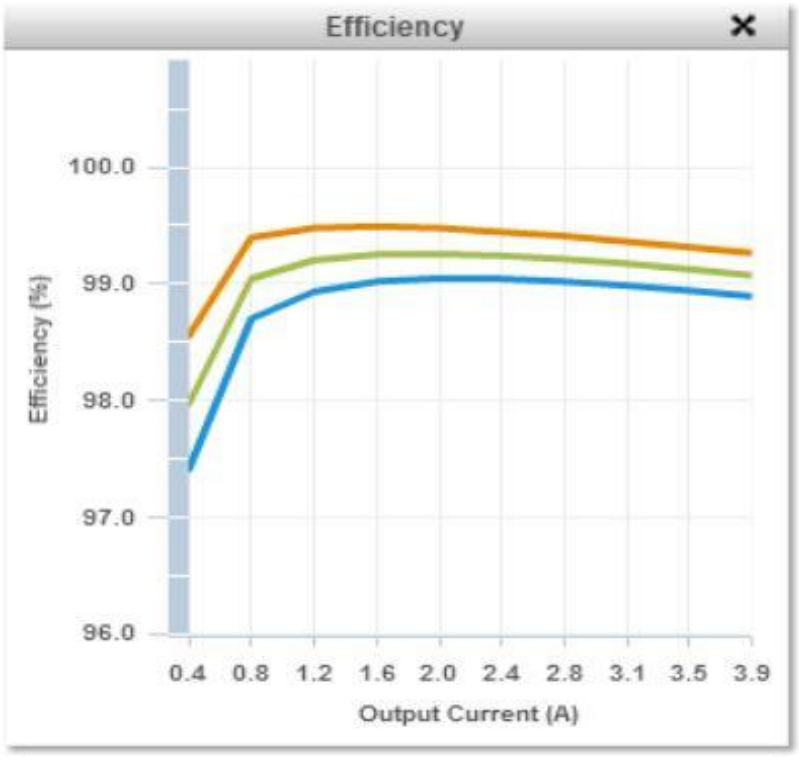


Figure 4.41: *Efficiency vs output current of Design #3*

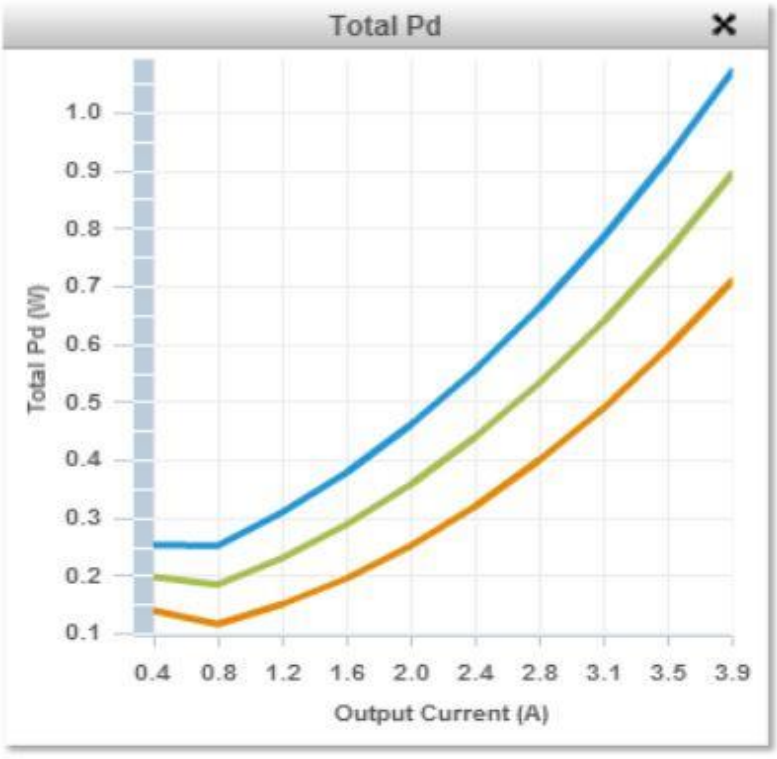


Figure 4.42: *Power dissipated vs output current of Design #3*

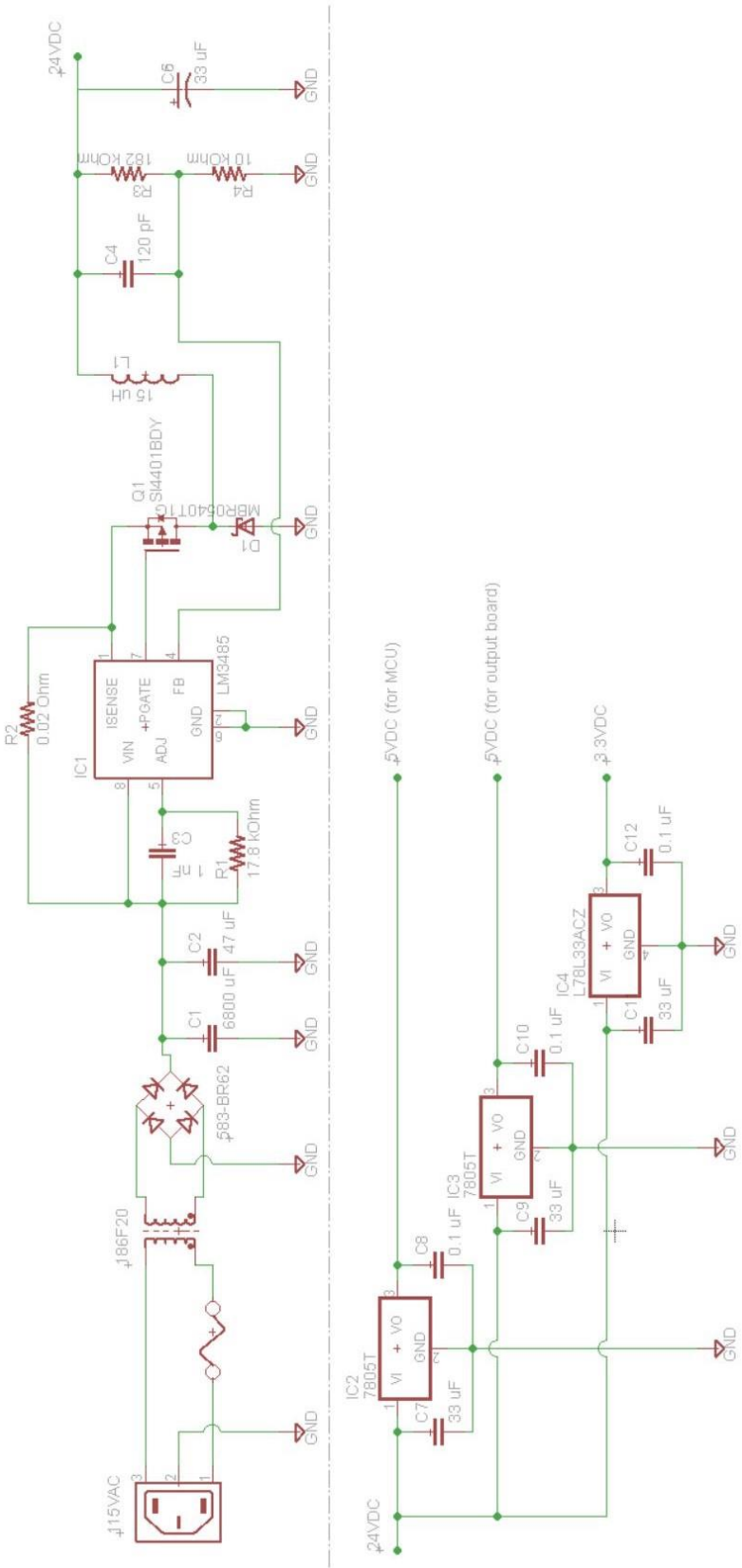


Figure 4.43: *Completed Power Supply Design*

Table 4.24: Power Supply Bill of Materials			
Item	**Part Number**	**Quantity**	**Unit Cost**
Transformer	186F20	1	$22.36
Bridge Rectifier	583-BR62	1	$0.96
C1 - 6800 uF	-	1	$1.68
C2 - 47 uF	-	1	$0.42
C3 - 1 nF	-	1	$0.15
C4 - 120 pF	-	1	$0.15
C5 - 33 uF	-	4	$0.20
C6 - 0.1 uF	-	3	$0.06
R1 - 17.8 kΩ	-	1	$0.10
R2 - 20 mΩ	-	1	$0.10
R4 - 182 kΩ	-	1	$0.10
R5 - 10 kΩ	-	1	$0.10
L1 - 15 uH	-	1	$0.12
Zener Diode	MBR0540T1G	1	$0.36
MosFET	SI4401BDY	1	$1.26
Buck Controller	LM3485	1	$1.68
5V Linear Reg	LM7805T	2	$0.69
3.3V Linear Reg	L78L33ACZ	1	$0.42

4.7 Raspberry Pi B+, Data Logging & In-System Web Server

One of the previously discussed purposes of having an automated wort extractor was that a home user could have control of the entirety of the automated process from beginning to end while being able to obtain system data for post-production analysis. In order to obtain this goal, the group has decided that the system, since it will already be connected to the Raspberry Pi, will have its own internal

Apache/MySQL server running on the Pi itself. Controlling this service will be a Python 2.7 driven GUI (graphical user interface) that will be connected to a laptop or desktop in order to transmit the user driven interface data. With the Raspberry Pi being the central unit, it will also need to have a couple Python 2.7 scripts that will interface between the GUI being displayed, the internal Apache/MySQL server and the ATMEGA328P control chips. This functionality in specific, was the team's purpose all along: to have the Raspberry Pi Model B+ handle all of the routing communications that weren't internal to the actual automated system (i.e. the sensors, control valves, power control signals, etc.).

All in all the Raspberry Pi will act as a traffic light situated in a very busy intersection. Directing outgoing traffic, that is the data gathered from the internal sensors and various control valves, and the incoming traffic into the system itself, mainly being the user inputted data for the specified brew recipe and all of its parameters (or taking already stored data for a recipe stored on the server). The overall idea here is that the system will generate and control about 99% of the data that is navigating throughout the Raspberry Pi at any given time and the remaining, fractional 1% will be derived from the initial recipe initiation done by the user; which is important as it is needed to initiate the process.

Throughout the whole process the system will be reading and relaying data that will be transferred from the control chips to the Raspberry Pi and then to the tables in the server accordingly. As mentioned above several Python 2.7 scripts will take care of translating the incoming system data to its appropriate destination; reading the incoming data written in C language and interacting with the Python 2.7 language. In doing so, accessing and manipulating this information will be much cleaner, simpler and more manageable for later analysis if the home brewer sees it necessary to access this valuable information. The server installed on the master unit, the Raspberry Pi, will be a key part in the system because of its centralized work flow layout. The server, alongside the interface, will undertake fully the task to store previous inputted 'favorite' recipes, along with giving the user the option of opening previously saved recipes which will be stored on the server at all times. These values and their relevant data will be stored in their own separate tables in order to avoid any data compromise/mismanagement.

Having this option undoubtedly gives the home brewer quite the versatility, not to mention this software option inside the GUI allows the system to be very user-friendly and flexible in its interaction with the user. The server helps to organize the massive amount of small time-driven data that will be generated in cycles throughout the process, keeping in line with the initial concept that this system will be as autonomous as possible. The concept of this data logging, from the sensor readings to the recipe initiations, is that in the end the home brewer will be able to access all of this rich-data after process termination. This can be hours after or days, as long as the system keeps generating the system data the user will be able to obtain this information by being able to access the server on the Raspberry Pi through the internet or the Android application (under consideration).

The group will have a website running off of the Raspberry Pi's internal IP address that will be available for access through any internet network as long as the user has the correct IP address. In this modern age of smartphones and data mobility having this system offer this option was not only crucial for the group but also somewhat necessary. They wanted to appeal to the modern home brewers whom in some cases have become quite tech-savvy fairly quickly. If the system is allowed to go off on its own after the home brewer initiates the process, while also allowing him/her to access 'real-time' data (there might be a few seconds delay between batches of information depending on the network connectivity and also because the system might have some short periods of lag) is something that is highly beneficial to all parties. Just this functionality alone gives the main goal of freeing the home brewer from the strenuous task of being present to take measurements, record them and keep track of specific recipe restricted given times, a huge win. The group wants to reduce the amount of physical human overhead and having this system do data logging and web server acquisitions to this data, make this an undeniably needed modern day functionality.

4.7.1 Raspberry Pi Model B+ Introduction & Overview

In this section the advantages as to why the group decided to select the Raspberry Pi Model B+ as the driving machine behind this automated system will be discussed. As seen below in Figure 4.44, this small CPU can easily be comparable in size to a modern credit card with the dimensions shows above. The size of this small, yet powerful CPU was a top decisive factor in choosing it, along with its many incorporated high-tech parts which set it apart from its predecessors and other competitors out in the market currently. The group from the beginning knew that they would need a powerful and cost efficient central unit which could handle the highly involved software aspects of this automated system, with little to no addition of hardware components to the central unit. In the end it was the Raspberry Pi Model B+ which landed the long term job. Not to mention the cherry on top: its operating system is a Linux based operating system called *Raspbian* (*Raspbian* is a registered trademark of the Raspberry Pi Foundation); this makes the world of a difference when integrating this part with the rest of the circuit since Linux is one of the less complex operating systems out there. This is also true not only from a programmer's point of view but also from a designer's point of view, making the design aspects of the project and the ATMEGA328P chips much simpler.

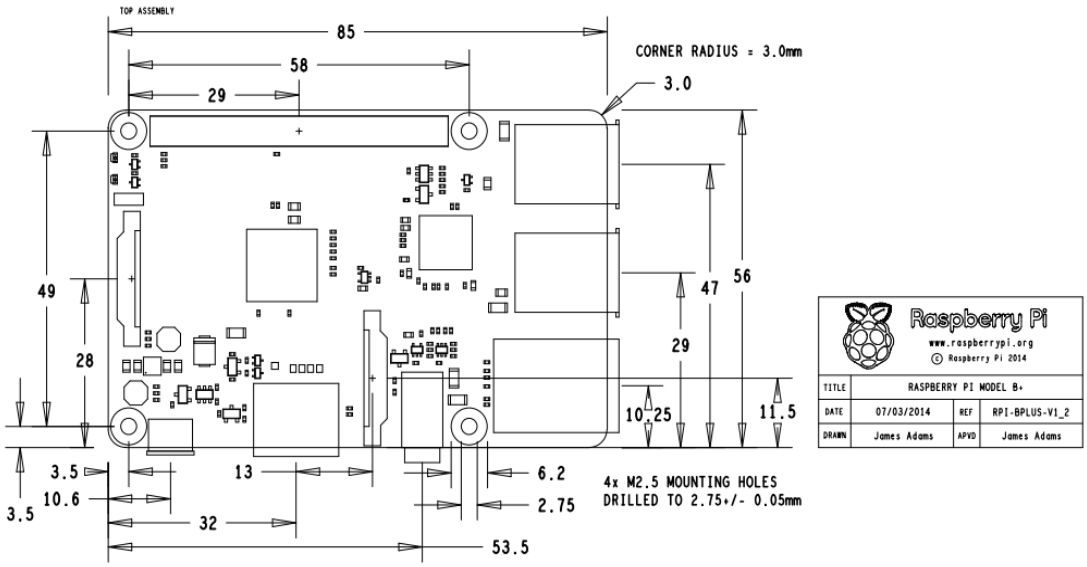


Figure 4.44: *Mechanical Specifications for Raspberry Pi Model B+ (Courtesy of the Raspberry Pi Foundation)*

This small, powerful unit has seen its popularity rise since its incorporation in the technology market for not only its simplicity in operation but also its cost along with the versatility in handling various tasks. The Model B+ comes at a cost of just $35, before tax and shipping, making it a very competitive modern day "all in-one" computer for small time developers, programming enthusiasts or beginners. Having just the Model B+ alone looks boring, but it can deliver a lot more than meets the eye at a first glance. Of course if the job requires a little more components, the Raspberry Pi Model B+ can be adjusted to hold various external components. As seen in Figure 4.45 below, the Model B+ has four USB ports, along with an HDMI port, a 4-pole 3.5mm jack, an Ethernet socket, 40 GPIOs (seven of these being general GPIOs) and a DSI display connector (for a ribbon cable connection). This credit card-sized computer can handle high capacity operations since it comes with a 512MB RAM capacity along with its ARM 1176JFZ-S 700 MHz processor.

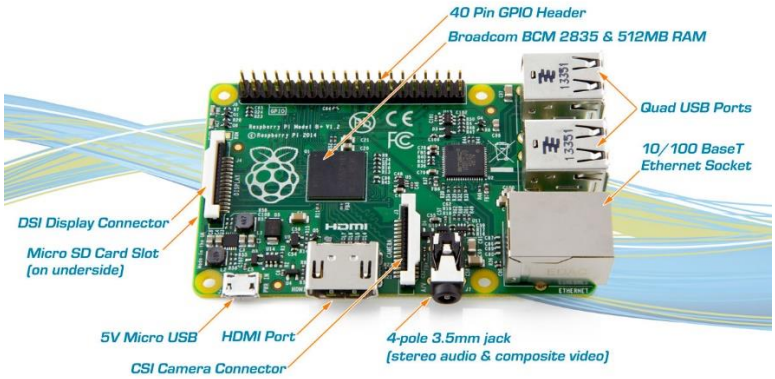


Figure 4.45: *Layout of Components for Raspberry Pi Model B+ (Courtesy of the Element14 Community)*

The specifics for the pin layouts can also be seen in Figure 4.46 below. The Model B+ also offers one 3.3V source along with two 5V sources, allowing for the group to utilize these three pins to their advantage when sending control signals to the ATMEGA328P chips and all other connected devices. Pin #27 and #28 will also be very useful since the group will be utilizing the Raspberry Pi Model B+'s I^2C (pronounced 'I squared C') input/output protocol. This great, already built in tool, serves as the platform for the internal communication protocols that the Raspberry Pi will handle. It is an extremely useful tool that allows the group to facilitate communications and transfers of data between the ATMEGA328P chips and the central unit. Another fascinating fact about this model according to the *Element 14 Community* is the fact that it is very low power in consumption; in specific just operating with the basic Wi-Fi adapter, a USB connected keyboard and a wireless mouse it is able to keep under 700mA or less than 3.5 Watts (this is using the suggested 5V micro USB power supply for the Model B+). With more than enough, this affordable pocket sized computer weeded out other models that were in consideration for the job. Seeing as to how this project also required, as mentioned in the executive summary, a user interface, the Model B+ running its small Raspbian operating system (3.2 GB large) on an installed 8GB Micro SD card allowed for the group to create a great interface and the many different functionalities which could not have been achieved so easily with another small based CPU in its place.

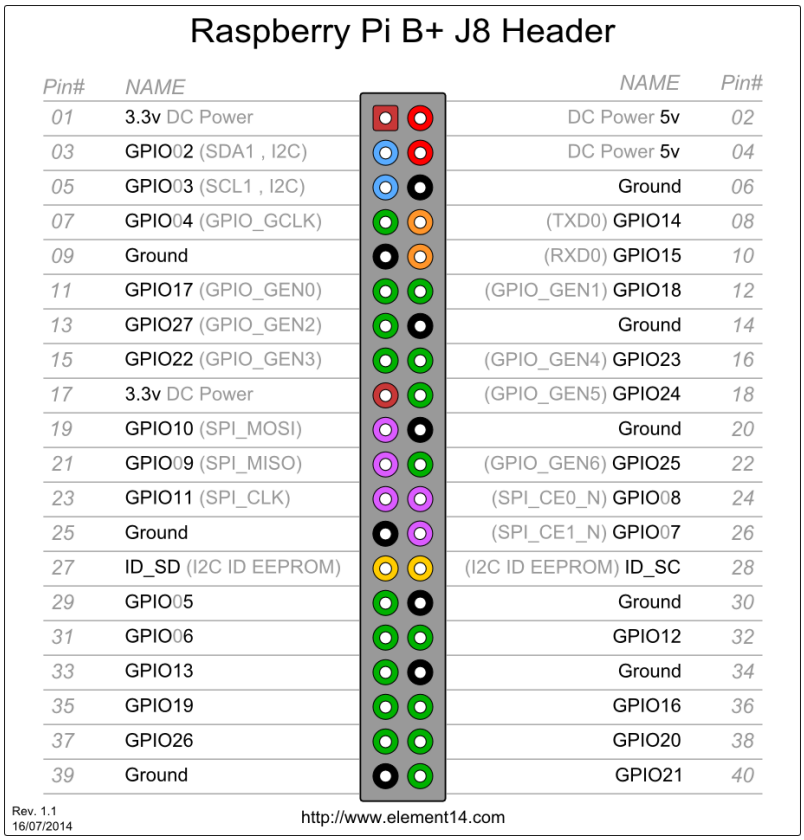


Figure 4.46: *GPIO Pin Layout for Raspberry Pi Model B+ (Courtesy of the Element14 Community)*

4.7.2 Raspberry Pi B+ vs. A13-OLinuXino, Cubieboard2 & Banana Pro

Table 4.25: Raspberry Pi B+ VS. Other Modern Single Board Computer Models (Information for Raspberry Pi courtesy of the Raspberry Pi Foundation, A13-OLinuXino Wi-Fi courtesy of Olimex, Cubieboard2 courtesy of the Element 14 Community, Banana Pi courtesy of Banana Pi)				
Single Board Computer Name:	**Raspberry Pi Model B+**	**A13-OLinuXino Wi-Fi Enabled**	**Cubieboard2**	**Banana Pi**
Developer:	Raspberry Pi Foundation	Olimex	Cubieboard	LeMaker
Release Date:	Summer 2014	April 2012	November 2012	2014
Cost:	$35.00	$68.70	$49.00	$54.99
Processor:	ARM11	ARM Cortex-A8	ARM Cortex A7-Dual Core	ARM Cortex A7-Dual Core
SoC (Software on Chip):	Broadcom BCM2835	Allwinner A13	Allwinner A20	Allwinner A20
GPU:	Dual Core VideoCore IV	ARM Mali-400	ARM Mali-400	ARM Mali-400
Clock Speed:	700 MHz	1.0 GHz	2 x 1.0 GHz	2 x 1.0 GHz
RAM/Memory:	512MB SDRAM / None	512MB / 4GB NAND Flash	1GB DDR3 / 3.4GB NAND Flash	1GB DDR3 / None
OS Image (Linux/Android):	Linux	Android	Linux OR Android	Linux OR Android
Power Supply:	5V, 2A	6-16V (Battery supported)	5V, 1-2A	5V, 2A
GPIO Count:	27	8	-	7
I^2C *Support:*	Yes	Yes	Yes	Yes
HDMI Port:	Yes	No	Yes	Yes
Ethernet Port:	Yes	No	Yes	Yes
USB Port(s):	4 hosts	4 hosts (3 for users)	2 hosts	2 hosts
Video/Audio Out:	Yes	Yes	Yes	Yes
Dimensions:	85mm x 56mm	120mm x 120 mm	100mm x 60mm	92mm x 60mm
Weight:	45g	*n/a*	*n/a*	48g

In the following section and in Table 4.25 above, the detailed specs and comparisons between the Raspberry Pi B+ and its other three competitors will be made. Although there are quite a few other “small single board computer” models out there, these four are the ones that more closely met the needs of the group and fulfilled most of the objectives list. The three runner-ups are as follow: the A13-OLinuXino by Olimex, the Cubieboard2 by Cubieboard and the Banana Pi by LeMaker. The group did not have a specific budget for the central unit that would be controlling most of the communications internally and externally, nevertheless prices for all four SBC’s (single board computer) have been listed.

At the bottom of the price range, $35.00 to be precise, is the chosen CPU, the Raspberry Pi Model B+. As mentioned in the above sections, this model just outperformed the others when it came to the needed requirements for this project. The group knew that they would need a small computer with a good size RAM memory along with the ability to extend system memory if needed. It most of the SCB’s the operating system image is mounted through an SD or Micro SD card, and most offer the option to open up all of the card’s memory for system memory (as *Raspbian* does). The group also needed a central unit with the capability of running in a low-power mode at times when it is idle, along with the support for HDMI to be connected to any monitor. Having a high number of general GPIO’s was a very must have feature, along with also having a high number of GPIO pins. Compared to the other three models the Raspberry Pi B+ had the highest number of GPIO’s in total out of its 40 pin header, and 7 general purpose GPIO’s. The other models had high pin counts too, but most of the pins had been designed for other special purposes and functionalities.

Supporting I^2C in one of those pin(s) configurations was also important, since the group had already decided that was the communications protocol that they would be using all throughout. If the Raspberry Pi can be seen as the central unit, I^2C is the central artery that leads to the “heart” of this digital communication highway. Without I^2C support the chosen SCB would have not been of service to the group, thankfully all four models had a special pin supporting I^2C communications. Another factor was the ability to quickly give internet connection to the central unit, whether it be through an Ethernet port connection or through the ability to do it wirelessly through Wi-Fi. This option is important since the user will be accessing the central unit and its internal server constantly throughout the automated process. The Raspberry Pi beats the three competitors by giving the group4 general purpose USB ports, with no strings attached as the A13-OLinuXino does.

Having the correct RAM capacity along with the right amount of available system memory was also a huge concern for the group since most of the time these small based single board computers don’t focus too much on high capacity memory systems. Luckily the four models selected had a good high amount of RAM and on-board memory. The highest being both the Cubieboard2 and the Banana Pi with both having 1GB DDR3 RAM and the Cubieboard2 having an extra 3.4GB

NAND Flash memory. On paper so far it would seem that the A13-OLinuXino by Olimex would be winning the race, but the Raspberry Pi B+ is still leading.

This is mainly because of two to three reasons: one being that the Raspberry Pi B+ is still cheaper, two the power supply for the A13-OLinuXino is much higher (6-16V) making the power supply for the system a little higher than the group wanted, and third the Raspberry Pi B+ is the newest/latest model out of the two. With its 512MB RAM and its 700MHz clock speed along with its option to extend the SD card for extra system memory beyond the 3.2GB needed for the operating system, the Raspberry Pi B+ just seems to fit the necessities of the project. Not to mention in size compared to the other models the Raspberry Pi is the smallest in size.

Worth mentioning is that all of the models compared in Table 4.6-1 above operate with the "System on Chip" (SoC) integrated circuit design, which works to the users advantage since the system will be a much smaller, faster, more compact integrated computer chip generating most often much lower power consumption than other designs. In the end the winner of the group's vote was the Raspberry Pi Model B+ for its outstanding performance in theory and on paper compared to the A13-OLinuXino Wi-Fi, the Cubieboard2 and the Banana Pi single board computer models.

4.7.3 External/Internal Communications With Raspberry Pi B+ Overview

At the moment, this portion of the project is still under careful planning and construction while the group waits to have all of the parts together so they could start to build a concrete model of chip-to-chip communications. What is known so far is that the ATMEGA328P chips will be the managers of the incoming sensor data taken straight out of the system by the various, carefully placed sensors (i.e. temperature sensors, PH sensors and level sensors). In turn it will then turn on its communication portal with the Raspberry Pi through the I^2C protocol, who will then delegate the appropriate procedures to get the incoming data to its correct recipient location. There will be a hardline connecting the Raspberry Pi to the ATMEGA328P chips, this communication has been decided that it will not be done wirelessly. At this time the group knows that it will be using a ribbon cable to connect the correct GPIO pins to the correct ATMEGA328P chips.

On the other hand, the communication output from the Raspberry Pi will be of the same manner as that out of the input protocol. Once the Raspberry Pi has data it needs to get out, at this time this is only done at the initiation of the process, it will alert the chips connected to its I/O pins and pass the data along accordingly. The group decided it would be much simpler in design and debugging purposes if the data sent from the Raspberry Pi to the ATMEGA328P chips would be done in the order they were received through the interface application the user used to input

the initialization data. Since each ATMEGA328P chip will take care of at least one type of sensor, no more than two sensors on a single chip, it will need a set amount of time to process the new given data and pass it along to the sensors/other connections it may control. By allowing the communication to be sent and received in a particular order it makes the delay on the chip's side to be lower since there will be order in the data flow from the beginning and hopefully no confusions or "NACKS" (no acknowledgment) sent from the chips.

Since the system will be using solenoid valves to direct the liquid flow in several parts of the process, these valves would need to receive electrical signals that would indicate what position they should be in (open/closed) all throughout this precisely timed automated process. These solenoid valves will be controlled by the ATMEGA328P chips themselves, no interface with the Raspberry Pi should be needed for this point. But relaying this opening and closing actions will be relayed to the Raspberry Pi, so if at a certain point the user wants to know why or what valve did or didn't close they can easily see that recorded information.

Another important aspect that needs to be detailed, which was touched upon earlier, is the communication protocol that will be used to implement all of these communications within the integrated system. For the purpose of not making this part any more complicated, the group will be using the I^2C protocol that already comes packaged (not installed) on the Raspberry Pi. According to definition provided by *www.WhatIs.com* "The I^2C (Inter-IC) bus is a bi-directional two-wire serial bus that provides a communication link between integrated circuits (ICs)". This is more than enough for what the system requires, for the majority of the time the I^2C will just be receiving incoming system data after the system begins. Unless something goes wrong it shouldn't have sending and receiving signals at the same time, nevertheless the group has implemented a try and catch system to prevent the system from crashing, stopping abruptly or producing unwanted results. The specifics for this solution though will be discussed in later drafts when the system is being wired, written and tested in its environment.

4.7.4 Data Logging Software Details

The specifics of the data logging is still being written and it is in its infancy stage; for the moment what will be discussed are the steps that will be taken to get the software aspects of the data logging functionality working properly. Along with what will be needed to get this step from the design stages to the fully automated and functional stage. Since the build time for the system is estimated to be at roughly ten to twelve weeks, at each phase of the build time frame portions of this very important highly dependent software component must also be in the pipeline for integration with the system.

Data logging can also be thought of as the data mining portion of this project, since in that sense the system is also giving the user the capability of being able to go back after the termination of the process and look at the data that was fed into the

server from the system. This form of data logging is small in scale but will be very precise and time drive, which is something that even at a smaller scale takes precision and perfectly calibrated hardware and software components. It all starts from the time the user starts up the '*Automated Wort Extractor Recipe Application'* interface and decides to enter the required variable inputs for the various system start/end points. By start and end points this means the required brewing instruction(s) needed to make the wort extraction process correctly. This refers specifically to temperatures before/after the wort enters boil kettle, how long should the wort be circulating through the wort chiller and what temperature it should exit, etc.

The specific functionalities along with their equivalent recipe implications will be discussed in much greater detail in section *4.7.3* of the document. Following this inputting of data the data will be saved, as a specific recipe with a name given by at the user's discretion. All of the collected data will be stamped with a date and time when saved to the server entries to avoid confusion and mislabeling information. This will be helpful not only during the test stage but also for when the user is combing through the data post-process termination. After relieving the interface of its data the next part takes us to the actual start of the process, in which it is kicked into high gear and from this moment forward it will start to collect its information. Up until the wort makes its way into the glass carboy at the bottom of the frame at which point it is ready for fermentation and the system can alert the user and safely go into standalone mode or shutdown.

4.7.5 Data Logging: Microcontroller Communications

This section goes more in depth about the specifics of the numerous data transfers and communications between the Python scripts in the Raspberry Pi and the C language written drivers for the ATMEGA328P chips. Every time a sensor is supposed to read some data, for example the temperature inside the boil kettle, it will translate this signal from the ATMEGA328P chip to the Raspberry Pi to the Python script then lastly to the correct table on the server. Now of course the specifics of this may vary from sensor to sensor because each one will have different frequencies in which it will need to be reading data at, but the overall idea stays the same. Each step after the data is taken requires precise calculation and manipulation of the communication flow using the I^2C protocol and any internal commands that the Raspberry Pi may offer to get the data to the right place.

The reviewer of all this information can be said to be the Python script that will take this fresh raw data from the microcontroller organize it and translate it so that it can be shipped off to the server for logging. This script will not be seen by the user, like the GUI application, it will act in the background running at all times as long as the Raspberry Pi has power and listening for incoming data. Variable assignments will be made at this point in time so that it will correlate with the variables already

defined inside the tables of the server, this makes logging this information much more organized. Since the data being sent and received will be at a synchronized order the system should not have a problem during the variable assignment process. The Python script will also organize the data coming into it, it will make the decision to go into a specific set of code (or function for simplicity) based on what microcontroller sent the information. This allows the script to maintain total control over what data is where and what data is being written where and into what variables.

Having the information being sent and received be in that certain order set by the programmer from the beginning slows this recognition script to be more accurate and precise. Using data logging in the end, is for the purpose of having readable and understandable data at the end of a certain time period. This is the whole point of having this 'middle man' Python script embedded in the background receiving these bands of information. The most favorable thing about this piece of code is that it can be easily manipulated and changed if one of the microcontroller's changes or its function becomes to control a different sensor than it can adapt easily. At this point all the programmer would have to do is to adapt the code and verify that the order characteristics of the data have not been breached and are still intact. Having a script that can do this makes a big difference for any programmer, and techy-savvy home brewer, if they want to change specific layouts of the system at any given point.

4.7.6 Web Server Specifications and Usage

The web server will help to facilitate the access of the gathered information for the user; the server will serve as the human "eyes" overlooking the automated system. Since there will be no human interaction after the user begins the process from the comfort of the interface application, the time-driven data collected will give the user a very accurate insight into each of the system's subsystem. At any given point in time the user will have easy access to the data that was gathered by connecting to the Raspberry Pi's home network and use a standard web browser to navigate to the web site being hosted off of the Raspberry Pi's IP address. Issues with security are still being discussed and researched, since the user is leaving specific ports open on the network router they can cause unwanted breaches. The purpose of this functionality is not to put the user's internet service and its components in jeopardy but to allow for a modern mobile environment for data access. The other option of course is just to limit the external access to the Pi's internal data and just allow the user to access this data on the server if they are connected to their home network.

4.8 Brew Extractor User Interface Overview

The group decided that, having already a Linux environment on the Raspberry Pi itself, it would be more efficient and a lot more manageable to integrate the graphical user interface (GUI) using the Python 2.7 language. Since Linux and Python already get along so easily and there exists so many external resources on these two topics, it was a simple choice. At the moment the group is still discussing whether or not to incorporate a mobile application (most likely Android based), to allow a home brewer to connect remotely to the automated system to keep track of how the process is coming along. It will basically show the web server information in an organized matter, this data is the same data that the application will have access from the Raspberry Pi itself. It is still under discussion and an executive decision will be made at a later time in Senior Design II.

For the time being the majority of the discussion will be focused on the interface application that will give the user the creativity and accuracy that they will need to get their batch started. When coming up with how to get the user to feel in control of such an automated system, the group decided that it would be most appealing if the user could in turn see what he/she were doing actually turned into what they had hoped for. The user has a sense of complete control from the very beginning, even though this is a highly autonomous systems, the user commands it to begin and do what they tell it to do. In order to get this portrayal across to the user, the interface has to be well organized along with having an appealing touch. The group wanted to make sure the user knew exactly what the system needed to know right from the get go, but not so much information that it would drown and confuse the user.

Establishing a GUI to head this “first impression” friendliness was designed with non-technical users in mind. Seeing as it is the non-technical users whom this automated home brewing system should benefit the most. Examples of the interface and their specific characteristics and functionalities will be discussed in detailed in a later section. The group also needed a reliable way to communicate within the Raspberry Pi’s I^2C protocol and writing the Python scripts was going to be the easiest way to get internal communication across to the microcontrollers. Having the Raspberry Pi running Raspbian with Python being so light and already installed made the design process for this interface to be rather simple.

4.8.1 Python GUI & Android App Development

The automated process being created has to be driven by at least an initialization phase done by an actual human. This is where the group decided to establish a graphical user interface, and in this case it will be written in Python 2.7. Wanting to use technology to its most advantage, the group established that the user will bring up this interface “application” on the laptop/desktop connected to the automated system, enter all of the required information and send the information

collected on its way. From the point that the user verifies and send the information, the GUI will start its I^2C communications (done via the Raspberry Pi) with the microcontrollers on the other side of the data line feed.

After this is completed, the overall job of the GUI is finished and the GUI itself will be rendered idle for the continuation of the process. Closing the application could be an option that is still being investigated, at the moment the group has decided it will be left open in the background without any conflicts to the other tasks being handled. It is important to see the effect(s) of closing the initial window of communication, seeing as it would free up more memory for the Raspberry Pi to do its other tasks on the 512MB RAM that it comes equipped with. The reason this is being discussed now is because the GUI will also be communicating with the server on the Raspberry Pi, dealing with the functionalities mentioned in section *4.6.6*, and further testing with these scenarios is still needed. The entirety of this discussion will be further determined, analyzed and resolved during the test phases of this project in the coming semester; a layout of the testing phases will also be detailed in sections *7.3 Software Test Environment* and *7.4 Software Specific Testing* later in this document.

For the Android application portion of this project the group is still discussing as to whether to fully integrate it in the final portion of the project or not. It has been added to this document because it was in the initial draft for the scope of the project, and if it does become an integrated part of the final design it will be detailed in the final draft of this document next semester. Having the Android app integrated with the data communication levels is an excellent idea on paper and in thought, seeing as this could facilitate the user experience and information management and viewing on the go. The conclusion the group discussed was that it would be better to wait until all other major parts of the automated system were designed, drawn out, and build specifications were in place in order to see if in fact the group would be able to complete the project with incorporating the Android app or leaving it out. For this document the overview of the “functionality” of this Android app is discussed below.

The idea behind the Android app is to give the user the modern technological mobility and flexibility to the fullest, since one of the main objectives is to have the human user physically interact with the actual brewing system as least as possible. The app will have a very simplified yet appealing interface with a couple options to view the data that the system has been gathering. There could be an option to look at a graph representation of the different temperature sensors and their activity over a certain time period, and there could be an option to look up certain recipes that have been saved to the server on the Raspberry Pi. One thing this Android app will not be designed to do is to send data to the system, the only thing it may be designed to do is to send a “STOP” command to the Raspberry Pi that will in return tell the system to halt all of its operations. As mentioned above the full scope and design of this portion on the project will be assessed and put into the final draft after the build stages have begun next semester.

4.8.2 Discussion of User Interface Software Layout

The software layout of the graphical user interface is very much simplified as can be seen in the UML diagram in Figure 4.47. The bulk of dependencies of classes in this code are mainly centered on the Tkinter Python interface package. The interface itself sits on this “Tkinter Frame” that allows the programmer to insert objects and functionalities into the frame at any point in time. The bulk of the code sits on the window initialization function of the interface that basically brings to life all of the text boxes and buttons that are needed to gather the information from the user. From the UML diagram not much of this can be seen because as stated above this interface is “Tkinter driven”, the actual buttons, textboxes, info boxes, help messages, etc. are all derived from this package in Python 2.7.

The Tkinter frame is the overseer of all items that are located within its boundaries, and most of what’s inside of it is considered a “widget”. In order to have an appealing user interface the team needed to modify and rearrange these widgets using the four of the classes shown in the diagram below: Tkinter.BaseWidget, Tkinter.Grid, Tkinter.Pack and Tkinter.Place. The interface needs to have flexibility which is why the main window can be adjusted to various sizes and the widgets inside the Tkinter frame within this window will move accordingly. Of course when the user opens up the interface application it will have a set size, they may see the need to maximize or minimize the window. Currently there is not a scrolling bar option that will allow the user navigate smoothly through the application if it is resized too small, but this is a layout functionality that the group is working on to incorporate.

The group discussed that the interface should flow freely from top to bottom and left to right, as most users tend to feel more comfortable with that configuration. Another layout feature is that the interface application will have a top bar menu which will be displayed at all times for the convenience of the user(s). It will mainly feature important information about the application and an open/save option for data control. It is a bit of a drawback that the software layout is completely dependent on the Tkinter package and its hierarchical classes, but it was the best option the group found when it came to Python graphical user interface designs. All throughout the interface there will be “user error” detection, in case the user makes a mistake while using the interface. One example would be if the user accidently hit the “EXIT” button instead of the “SEND” button, the system will give them a chance to rethink their decision and continue. Further design specifications and interface layouts will be discussed in the next sections.

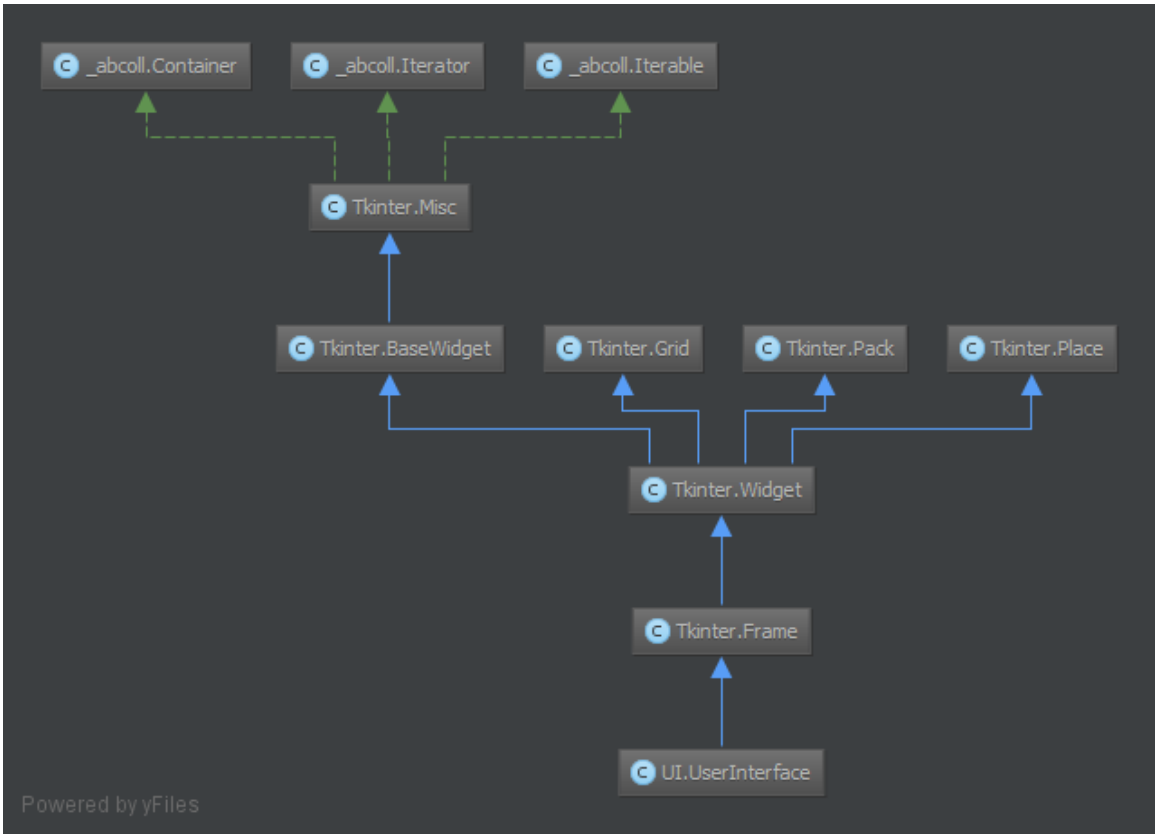


Figure 4.47: *UML Representation/Dependencies of Classes, Python 2.7 (Designed by David Rodriguez)*

4.8.3 Detailed Discussion of all UI Functionalities

The GUI itself will consist of a window, or “container”, that will pop up when the user runs the Python script on the laptop/desktop connected to the system. It will then show the home brewer the elaborate layout in which he/she will be able to edit the various input variables that he/she will have to enter to initiate the automated process. Of course as mentioned in earlier sections, the user has the choice to: 1. Enter a new recipe 2. Open an old recipe 3. Save a recipe (this option is unique, because it is available automatically from the system); further discussion in detail of each “recipe” option will be discussed later. Those options can be seen if the user navigates to the ‘File’ menu on the top left hand side as shown in the Figure 4.48 below. This preview of what the interface is going to look is at its best a rough draft of the final version, some of the text boxes will need to be modified for each sensor that needs to be initialized and for each control signal that will need to be set in the system by the user.

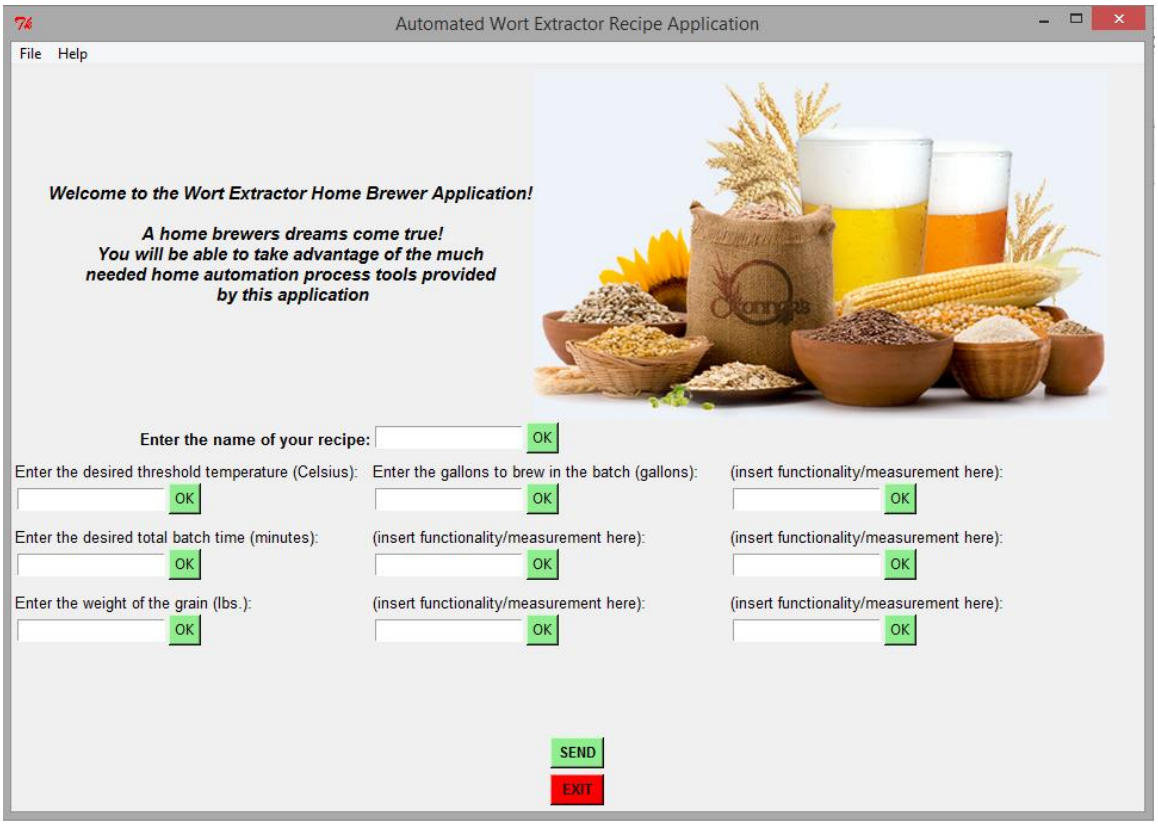


Figure 4.48: *Overview of Automated Home Brewing System Interface, Python 2.7 (designed by David Rodriguez)*

The window shown above in Figure 4.48 is very plain and straight to the point, the group did not want this to be a distracting interface by bombarding it with many different functionalities. The less the user is distracted the more focused he/she will be when filling out the form, because without the initialization of these variables the system won't start. As mentioned in earlier sections the group did want to give the sense of a "friendly" interface though, so that the user would not become uninterested and not feel connected to this process. Although the main objective is to take out the human interaction as much as possible for brewing beer at home, giving the sense that the user is still in control of parts of this process can be beneficial. The group did not want to create an automated system that was rugged and not look appealing to the home brewer's eyes. Having the interface have a sense of flow with eye-catching colors, along with giving the user that initial responsibility of initializing the system, makes for a very functional UI.

The main functionalities of this interface that can be seen are: the main menu options in the top left hand side, the save and exit buttons kindly located at the bottom of the window ('EXIT' colored red and 'SEND' green to catch the eye), the welcome message briefly stating the purpose of this application and lastly the text boxes that will be used to input the required system information. The text boxes in

particular will hold variable assignments that are of StringVar () type and will later be casted to the specific int or float data types as needed. Doing so makes the much formatting simpler and not constrained by the data type of each field. While at the same time giving the code and the programmer that flexibility to adapt the interface at later times. Each text box has its own individual green 'OK' button in order to let the user know that they have to check if what they've typed into the box is accepted by the program and it formatted correctly. This "checking" call will be done with a simple command function called each time the user presses the 'OK' button. Because this is an interface, for every action there's an action driven response that will occur, whether it is an information box like the one seen in Figure 4.49 below, or an error sound/error message that will communicate to the user the issue(s). For the "Information Message" seen below, it is found under the 'Help' and 'About' menu options in the top left hand side. It gives a brief overview of the purpose and options the user may want to utilize.

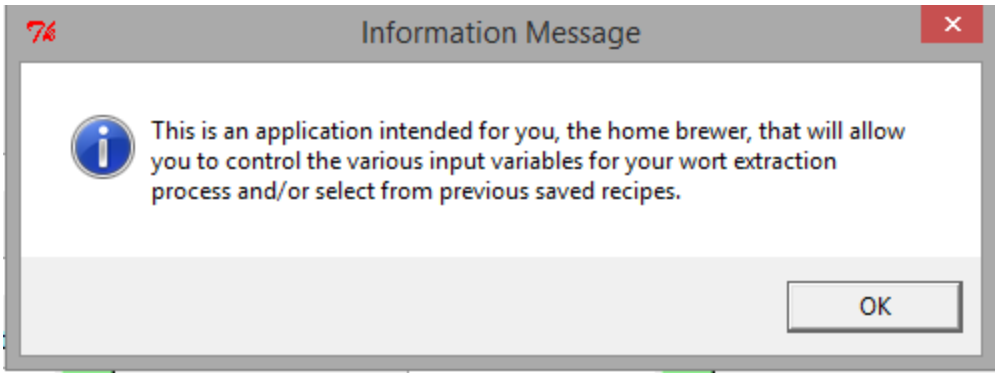


Figure 4.49: *Overview of Functionality: 'About' Option found in the 'Help' Menu Python 2.7 (Designed by David Rodriguez)*

The information message box seen above is an example of an "event-driven action". It was important to have a message display like this in order to provide the user with a more in depth explanation of the application. In the figure shown below, Figure 4.50, it can be seen in more detail the layout for the various labels, text boxes, and buttons. That exact formatting will be used for the other text entry boxes that are not shown in Figure 4.50; these "widgets" are set on the Tkinter frame itself and placed in specified x and y coordinates carefully calibrated so that even if the window expands the widgets are still in place properly. Many of the widgets added to this interface are derived from the Tkinter Python GUI library, using this tool is very simple and very common since it is Python's standard interface tool alongside the Tk GUI toolkit and there are many open source help tools for it. In the two images below it can be seen how the text entry box interface will look and feel to the user; it is straight forward, clear in instruction and it even reminds the user to enter the values in specific units (the units shown below are just for test purposes, the specific units for each input variable have not yet been finalized 100% and may or may not follow the metric system, these specifics will be decided when in the build stage).

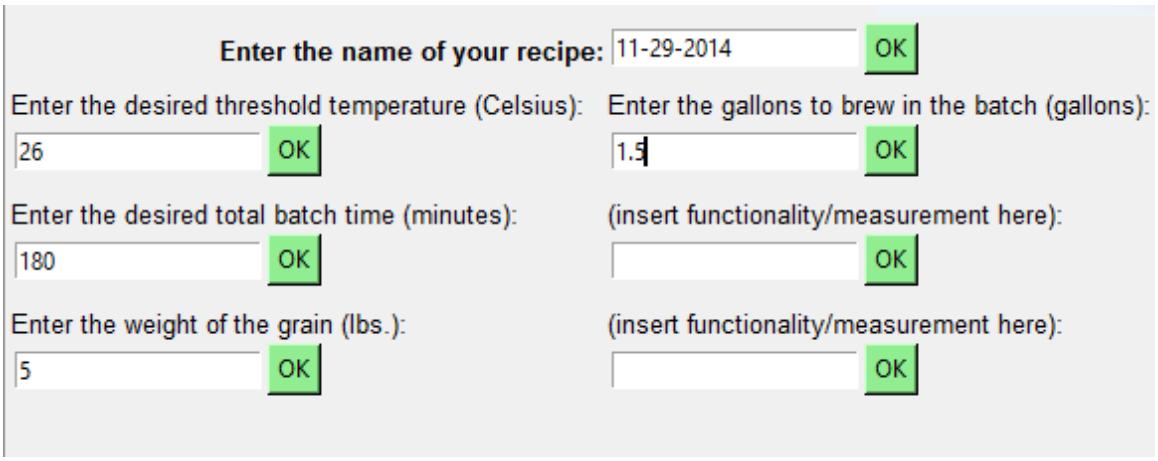


Figure 4.50: *Overview of Different Input Variable Text Boxes Python 2.7 (Designed by David Rodriguez)*

In the image below, Figure 4.51, a closer look is taken at two of the most important features of this interface: the send and exit functions. Of course the user has the option of just exiting from the interface window like all other window modules, but the group wanted to centralize these two options to give that "flow" characteristic to the design. The functionality of the red 'EXIT' button is simple: to exit the interface. Nevertheless the group also wants to have the user tend to exit the program, if that is their true intention, through the red button located near the bottom. It is ideal for the user to utilize this functional button because it will alert him/her in the case that they did not wish to truly exit. The functionality of the green 'SEND' button is to correctly open up the I^2C communication portal between the interface on the Raspberry Pi and the ATMEGA328P chips connected to the GPIO's, and send the data on its correct path. Once the user selects to send all of the inputted information, the system will collect it in the specific order (not specified at the moment, the group will decide at a later time) and pass it along to the corresponding data line to the correct ATMEGA328P chip.

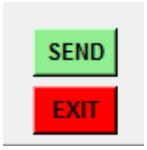


Figure 4.51: *'Send' or 'Exit' option for GUI, Python 2.7 (Designed by David Rodriguez)*

Figures 4.52 and 4.53 below show the event-driven response that the interface will have when either one of the functions discussed above are pressed by the user. The first figure explicitly shows the user that once he/she clicks 'Yes' all of their information will be sent to the system and saved appropriately in the server. As planned these actions will be done simultaneously, unless otherwise interrupted by the Raspberry Pi itself. There will be a check to make sure all of the fields have been properly formatted, so if the user selects 'Yes' and there is an error the system will communicate that to them. The 'No' option from this pop-up will simply

take the user back to the main interface application. For Figure 4.3 the system will clearly inform the user of the hazards of the exiting the main application at the time. In this case no data is saved, the Raspberry Pi will not be programmed to have some sort of “backup” for unsaved progress. Once the user opts in to exit the interface application will shut down and all communications will be turned off.

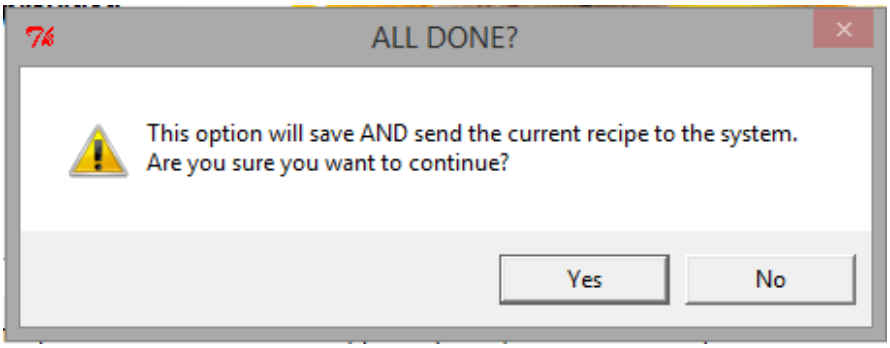


Figure 4.52: *‘Send’ Option Selection Prompt, Python 2.7*
(Designed by David Rodriguez)

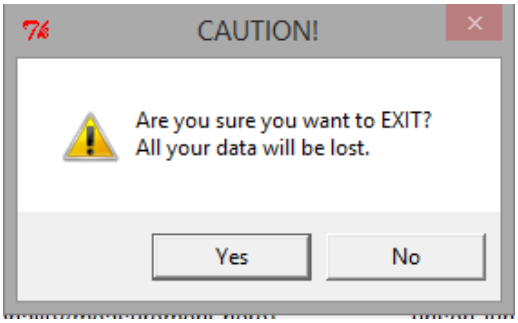


Figure 4.53: *‘Exit’ Option Selection Prompt, Python 2.7*
(Designed by David Rodriguez)

The whole purpose of the different functionalities of this interface are to also cut down the time needed to figure out what the user should type into each box and where. Since most brewing kits come with specifically laid out instructions and list of measurements that need to be taken at each of the crucial steps throughout the wort extraction processes. What the group decided was to get as much of that relevant information into text entry boxes that the user can fill out straight from the recipe the specific brewing kit brought, in turn having the system follow most of the same guidelines as that specific recipe. Transferring the data from paper to digital seems cumbersome, and in most cases it is, but for this automated system the last thing the group wanted was to add more overhead into a process that’s already time consuming on its own. The simple design and great flow layout put that drawback on the back of the user’s mind while interacting with interface application. The group would like to have the process be started with as little human interaction as possible and more technology driven interactions, leaving the cumbersome tasks to be more easily managed.

5.0 Design Summary of Software & Hardware Integration

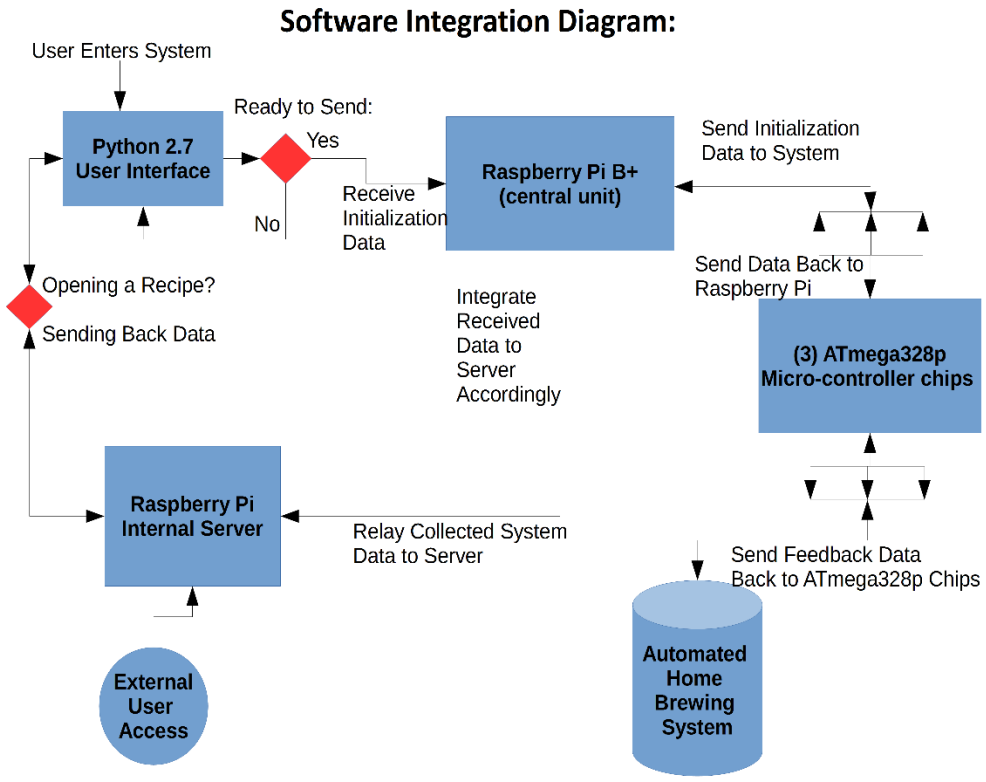


Figure 5.1: *Software Integration Diagram for Automated Home Brew System (Designed by David Rodriguez)*

In Figure 5.1 seen above, the group demonstrated the high level software integration of the automated system. For the hardware integration discussion it will be assumed that the object labeled “Automated Home Brewing System” in the diagram below will be the connection between the two diagrams. These two diagrams discussed in detailed are drawn to be a high level representation of the data flow and integration parameters that need to take place in order for this system to function as stated in the executive summary. Software and hardware integration are crucial discussion points in such a diverse and highly automated system. There are internally so many components that need to fit together and talk to each other, with the help of the software written for this system, that no one single part in this automated system can be seen as truly independent of its neighboring components.

From a purely software perspective the diagram below doesn’t make much sense, it is really more of a software blended with hardware interpretation which is what is needed. It is clearly depicted starting from the upper left hand side that the user

initiates this automated process at the beginning. They enter the Python 2.7 GUI application that appears to them on a monitor, they then will be prompted with various commands and visual cues by the interface as to what they are able to do next. When the user is ready the requested information will be transferred onto the first hardware component the data comes in contact with, the Raspberry Pi (also referred to as the central unit). In technical terms the computer on which the interface in running on is the first actual hardware component, but that is a part that can be interchanged whereas the Raspberry Pi is a static built component for the system.

If the user decides they want to go back for an older recipe the interface gives them that option. It is when this data comes at the intersection of the red diamond and the user is requesting access to specific data that is when a communication must be opened between the server and the interface. Hence the representation with a bidirectional arrow, information will be moved from server to interface and vice versa for the initialization period. Once the data is in the system, it is read into the Raspberry Pi taken in the order it came in and relayed to one of the three data feed lines that connect each of the three ATMEGA328P chips to the GPIO's. It is at this crucial point that software must maintain the correct order while navigating through the hardware components. The execution of the high priority of keeping the data order will either make or break the automated system from this point forward. The data that was received from the user will also be stored accordingly in its correct place in the server table frame. At this intersection only data will be flowing to the server from the Raspberry Pi, there will not be bidirectional data flow here.

From the diagram it can also be seen the access point the user has once the automated system has begun. It is also one directional because it is only accessing the collected data being fed from the Raspberry Pi at the appropriate time intervals all throughout the system's up time. Once the correct orders have been established and the data gathered from the interface is set up with its synchronization times, it will be sent to the system accordingly. It is clearly labeled as "Send Initialization Data to System", in which case the one data line feed will split into three different line feeds for each of the three microcontrollers. Once the microcontrollers have received their appropriate information and have been formatted, this takes place in the box labeled 'ATMEGA328P Micro-controller Chips', they then relay the information to the actual system and the correct parts/sensors accordingly.

Once this information is inside the automated system, the internal system communications will take over and being the process. The software job at this point is to remain attentive with open communication ports for when the flow of data begins. As can be seen in the figure, there is a bidirectional arrow connecting the ATMEGA328P chips and the automated system. The flow of data needs to be both directions since that it what the objective called for in the initial scope of the design. The communication from the sensors inside the system are not detailed in the diagram but their gathered information at the specified time intervals will be relayed through that bidirectional line. The flow of data from the automated system will

pass through the ATMEGA328P chips which will then integrate the data to the Raspberry Pi, it then goes to the server to be stored for later analysis by the user. The software integration portion of this project is a bit complex, next the specifics for the hardware components will be discussed.

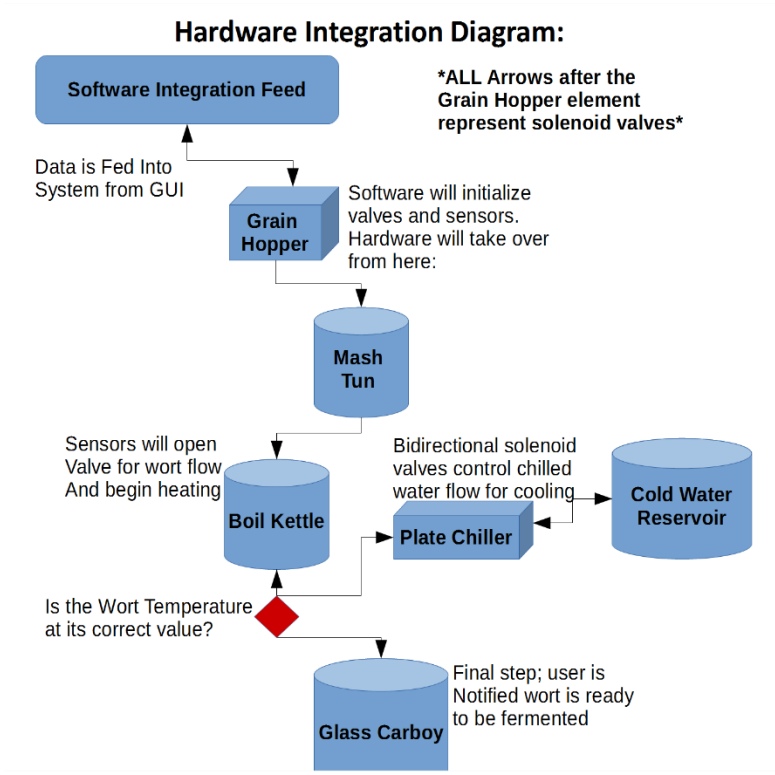


Figure 5.2: *Hardware Integration Diagram for Automated Home Brew System (Designed by David Rodriguez)*

Figure 5.2 shown above demonstrates the hardware integration into the automated home brewing system along with its high level individual components. In the previous section the software integration was discussed and certain aspects of the hardware were mentioned; this will cover the detail extensions of that. Once again as stated above with the software integration portion of this section, this diagram only covers the high level scope of the hardware integration aspect of this project. It will only show the necessary components needed to facilitate the integration process specifically. In turn the group looks to open the current design layouts to display this automated process' unique hardware integrations.

Beginning with the user entering their data into automated system, the hardware components will begin to play a role. The grain hopper, which is the container that will hold approximately 1.5 lbs. of grain, will be the shining star at the top of the frame. It will receive its signal to drop the grain into the mash tun for mixing from the interface application once the user hit the 'SEND' button. Once again at this point in this process all data has been checked for anomalies and corruption so that automated system is ready to go. Following the release of the grains the

solenoid valve on the mash tun will open up, with the signal sent specifically to it, to let the flow of water begin and the mixing process to start. The diagram above shows that the city water connection will be left on, leaving the solenoid valve to control the flow.

Once it is properly mixed, it must sit in the mash tun anywhere from 20-30 minutes in order for the water to absorb all of the flavors from the grain. Inside this mash tun there will be a small filtering hose that will sit at the bottom of the tank in order to drain only the extracted wort, leaving the grains behind. Another single, initiated by the time the user entered at the beginning of the process, will let the solenoid valve at the bottom of the mash tun be drained into the boil kettle. Once inside the boil kettle, the temperature for the heating elements have already been set and turned on. A temperature sensor along with a timer will let the system know when the wort is ready to be transferred to the cooling stage(s). Sometime before the wort begins to exit, the system will make a call as to when to add the needed hops and yeast to the wort. It is shown in the diagram next to the boil kettle but there is no specific design feature yet on how this will be done. The group will analyze different methods to make this injection into the wort as smooth as possible, while maintaining the integrity of the signals within the system. All throughout this intricate process the data being gathered from each internal subsystem is being analyzed by the Raspberry Pi's Python scripts. It is at these stages where the user can tap into the system to also make sure things are running smoothly.

As the wort reaches its threshold temperature, a signal will let the solenoid valve at the bottom of the boil kettle open up and begin to gravity feed the wort to the plate chiller located directly below it. The plate chiller has to inputs, one coming straight from the boil kettle and the other feeding in from the cold reservoir the group decided to include in order to cut down the cooling process. Another temperature sensor and a timer is needed to make sure the wort cooled properly and it is at the needed minimum temperature specified by the user's input data. At this point in the process the majority of the data taken from the various sensors has been received and stored. Once the system drains into the glass carboy sitting at the very bottom of the frame, the system will alert the user by pushing a notification to the: web server and web page, and the interface application. The group is still trying to discuss how to properly handle this last step, since it is crucial

6.0 Brew Extractor Prototype Testing

The prototype has many areas that need to be tested both mechanically and electrically. The first step is to test the extractor electrically though dry runs and see if the signals and data are being sent and received by the different components. Once the group knows that the electrical components are working, focus on testing the mechanics of the system can begin. These systems include the solenoid switches controlling the flow between kettles and the cooling

processes trough pumps. Also the method of cooling through the plate chiller. These test; however, are secondary to the teams design process.

Testing the automatic brew extractors electrical components can be broken down between two parts: the hardware and software. Hardware that will be tested are: the power supply, sensor outputs, filters, amplifiers, and the connection between the ATMEGA328P and raspberry pi. The software that needs to be tested are: communications between the ATMEGA328P and raspberry pi, the user interface, and reliability of the I^2C protocol.

The brew extractor has several hardware and software stages. Each of the hardware stages needs to be tested individually before applying to the system. For instance each of the sensors need to be tested before sending through the amplification and filter circuits. One place both hardware and software need to be tested together is the transmission of data between the ATMEGA328P and the pi. This is absolutely vital to system's web server functionality. This will be discussed in the hardware specific testing and software specific testing. Once individual stages have been tested the entire systems itself needs to be tested to see if the hardware and software have been integrated correctly.

As stated before a dry run will be used to test the overall electrical system. Brewing takes time and would be a mess to clean if the system fails to operate as expected. Certain aspects will be simulated such as the weight on the kettles and the temperature of the liquids to activate the pumps and solenoid switches. Other simulated aspects will be the pH in the sparge tank and the time it takes to cool the wort down. The exact methods will be specified later in the paper.

Once all the electronics are tested and operating as expected a full test of the project will occur. The test will see if the components will still behave under a live test. The secondary purpose of the test is to see if the structure for the brew extractor is built properly. For instance the strain gauges help to see if any leaks have sprung in the system. This project is supposed to be automated even though the group can see if all the components were built correctly during this run.

Testing the mechanical portion, while secondary to the overall project, is still important to the final project. This group's main focus was the electrical system, but what is the point of just having the electrical subsystem and no mechanical system to make the wort. The overall process is simple; however, there are several process that need to be address during the final test phases of the automatic brew extractor. The testing of the individual stages is covered later in the paper. For now the group will cover the final test stages assume each individual stage pass its test. The first step in testing the structure is seeing of it supports the brew kettle once filled and seeing if the strain gauges are taking the correct measurements. Once taken care of the water within the brew kettle is heated to the target temperature of 95˚C. Once the microcontroller has received that the target

temperature has been reached the solenoid switches will open and the pumps should turn on.

If all is operating correctly the strain gauges should take over the control of the pump and solenoid switches. As the mash tun is filled with hot water the grain from the grain hopper will fill the mash tun as well. The strain gauge in the heating kettle determines how long the pump runs and the grain hopper remains open. Once the brew kettle is emptied that will turn the solenoid switch off and gravity feed the wort back to the brew kettle. This will also turn on the sparge water system to wash the sides to capture all the sugars for the wort. The strain gauges in the boil kettle needs to be calibrated for the additional weight coming into kettle after the heating process. Once the target weight has been met the sparge water will be turned off and close the solenoid controlling the gravity feed.

The kettle strain gauges will now control the chilling process. The pumps control the flow of wort and chilled water through a plate chiller to lower the kettle. The final target temperature before fermenting is 30˚C. The temperature sensors will tell the pump and solenoid switches when to stopping sending the wort through the plate chiller and return it to the boil kettle. Once the wort is returned the user will bottle the wort for fermentation or chill further depending on type of beverage looking for. After checking the systems to see if any damage was found in any of the tanks, leaks in the tubes, or if the levers controlled by the solenoids are performing correcting then the system should be ready for general use.

6.1 Hardware Testing Environment

The hardware as mention before needs to be tested in stages to see if all the components are working properly. The typical environment most of the testing will occur will be primary through a bread broad. Using an oscilloscope to observe the correct outputs are being produced within the tolerance of the device. This is safer and will help with troubleshooting should it not work correctly on the PCB. The exception to this will be the power supply. The test will simply be an observing of the outputs through a voltmeter of oscilloscope.

Two separate power supplies are needed. A 5V for the electrical components on the PCB and a 24V for the solenoids. The group intends to use low resistances to observe the voltage dip from the two sources. Attention needs to be paid to the 5V power source and that two much dip does not occur with the test load. This 5V power supply needs to power the systems LT1014 operational amplifiers and the ATMEGA328P microcontrollers. Too much dip and the components may not turn on.

Now testing the different sensors. A 5Vcc will be used to power all of the sensors. The first one that will be tested is the NTCAIMME3C90373 thermistor. 5V will be sent through the thermistor will be wired in series of a reference resistor. The thermistor will be placed in several drink bodies of water of known temperatures to

check for accuracy of the thermistor. This thermistor temperature range reaches 105˚C and controls the turn on of the solenoid and turnoff of the heating element. To control misreading's in temperature due to thermistor variance, two thermistor will be used to measure the temperature in the heating kettle.

The next sensor to design an environment for are the strain gauges. They play an important role and thus must maintain accuracy throughout the process. They will need to be tested with and without the heating element in use to understand how heat effects the reading on the strain gauges. The stand will be used with the actual kettle fill with water to measure the accuracy and effectiveness of the strain gauges. Using varying temperature to see if it affects the measurements or not. The specifics are outlined later on.

Next the filter design and amplification steps are to be tested. This circuit is tested on a breadboard and simulated to see if the design works. The reason for both is to see if the theoretical calculations will filter out the 60Hz power supply noise affecting the sensor measurements. The breadboard is also used to test to see if the LT1014DN is turning on from the power supply.

The final test environment will be for the I^2C connections between the ATMEGA328P and the raspberry pi. An oscilloscope will be used to see if the appropriate signals are being sent back and forth between the two. This is initially tested through the software; however, should the task not be handled as expected then the previous method of testing shall occur. Many times the signal communication using I^2C is the hardware's and not the programming's error. The signal simply does not travel across the connections.

Now that each individual stage has been tested integrating the separate stages is the next task. The breadboard is used to test each of the next components. A 5Vcc is set up thanks to the power supply and powers both the thermistor and operational amplifier. Due to the amplification from the filter the voltage being measured needs to be attenuated. The thermistor measurements could easily reach the rails at low temperatures while observed through simulation. Testing is done to see the temperature to voltage range to help write the program to measure the voltage due to the temperature change not being linear.

The strain gauge implantation is similar to the thermistor. The 5Vcc is sent through the strain gauge whetstone bridge and the change in voltage is sent through a unity buffer. That measurement is then sent through the filter/amp stage and measured. The pH probe will be tested in a similar manner.

The next test stage is making sure the ATMEGA328P is receiving all the data and is recording them appropriately. The breadboard continues to be used as the test environment. The outputs from the previous test stages are used as the test signals. An oscilloscope is used to confirm that the output from the filter is correct and that is the input for the ATMEGA328P. Through the software it was confirmed

the data was input correctly in the ATMEGA328P. The final step is to see if the ATMEGA328P will power on the solenoid power supply once the appropriate inputs from the sensors are received.

The test environment for the solenoids is simple. It takes the signal for the turn on from the ATMEGA328P to turn on and remain on until the sensors signal to turn off. This will be left for the dry run of the complete system and the signals will be simulated first by using high and low voltage signals from the ATMEGA328P. Then once the solenoids are shown to be functioning correctly the signals from the sensors will be used to see if control is maintained through the test.

6.2 Hardware Specific Testing

The actual testing expectations and necessary results will be discussed here along with a summary table for each of the individual hardware testing stages discussed in the environment of testing.

Testing the 5VDC power source is first is the first step. The 24VDC power source for used for the solenoid will also be tested. As previously stated a low resistance will be used to test the effects of voltage lag on the system. The drops will be measured through a voltmeter. Voltage lag should not be less than 4.8VDC or a third power source will have to be created for the LT1014DN rated at 6VDC will be need. The LT1014DN will be tested using the two resistors wired in parallel as an input load and the operational amplifier as a part of a unity gain buffer. If an input is produced then the LT1014DN can operate with the 5VDC power supply. Table 6.1 and Table 6.2 show what results should be produced based on the different load resistances and amperage rating for the power supply, respectively.

Table 6.1: Test Loads for 24V/3A Power Supply

Loads(Ω)	Voltage Measurements(Volts)
3	23.5
5	23.8
8	24
10	24
12	24

Table 6.2: Test Loads for 5V/1A Power Supply	
Loads(Ω)	**Voltage Measurements(Volts)**
3	4.8
5	5
8	5
10	5
12	5

Each of the NTCAIMME3C90373 thermistors needs to be tested individually to test for accuracy and variance between them. The plan is to use five different water temperatures measuring from 25˚C - 95˚C. The test will be produced for both the thermistor before being amplified through the filter and after to make sure all calculations are correct. Table 6.3 records the test values and expected results for the thermistor before the filter stage. Table 6.4 will show the voltage values after filter stage. Using the table provided in the NTCAIMME3C90373 the group will convert voltage to temperature in the ATMEGA328P. For purposes of testing the output only the voltages are measured.

Table 6.3: Thermistor Temperature Test Values Before Filter					
Water Test Temp.	Nominal Value(V)	Thermistor 1(V)	Thermistor 2(V)	Thermistor 3(V)	Thermistor 4(V)
25˚C	0.220	0.209	0.2134	0.2244	0.2112
40˚C	0.398	0.3781	0.386	0.406	0.382
60˚C	0.781	0.74195	0.758	0.797	0.750
75˚C	1.188	1.1286	1.152	1.211	1.148
95˚C	1.851	1.758	1.796	1.889	1.778

The information gathered while testing the outputs of the entering the filter and amplification stage will determine whether or not an attenuation stage is needed. The maximum output from that stage cannot exceed a 5V. Another solution to consider is changing the power supply of the LT1014DN as previously discussed. The expected values are in Table 6.4.

Table 6.4: Thermistor Temperature Test Values After Filter					
Water Test Temp.	Nominal Value	Thermistor 1(V)	Thermistor 2(V)	Thermistor 3(V)	Thermistor 4(V)
25˚C	0.506	0.4807	0.491	0.516	0.486
40˚C	0.9154	0.870	0.888	0.516	0.486
60˚C	1.7963	1.707	1.742	1.832	1.724
75˚C	2.7324	2.595	2.650	2.787	2.623
95˚C	4.2573	4.044	4.130	4.342	4.087

The next sensor to be tested is the strain gauges. The maximum load the strain gauges can hold is 50kg. The method for testing the strain gauges will be to repeatedly test the same weight and see if the same output is produced each time. The weight used will be a 20kg weight and the output voltage will be measured. The output of the strain gauge circuit will be compared based on the voltage measured each time the test is run when only one strain gauge is used in the Wheatstone bridge. The measurements that are expected are outlined in Table 6.5 before the amplifier stage. Table 6.6 shows the strain gauge values after amplification through the filter. Then after observing the values obtained through the Butterworth filter a second amplification stage with a gain of 10 is needed to increase resolution of the measured weight. The total gain is the Butterworth filter gain multiplied with the cascaded amplifier, which comes out to be 23. These values are subject to change depending on the strain gauge chosen for the final design, other strain gauges may require higher amplification.

Table 6.5: 20 kg Load Applied to Strain Gauge	
Test 1	3.74 mV
Test 2	3.76 mV
Test 3	3.79 mV
Test 4	3.76 mV
Test 5	3.80 mV

Table 6.6: 20 kg Load applied to Strain Gauge Amplified	
Test 1	86.02 mV
Test 2	86.48 mV
Test 3	87.17 mV
Test 4	86.48 mV
Test 5	87.40 mV

The next stage of testing is communication of the ATMEGA328P and the raspberry pi. This stage is easily tested and mentioned in the testing environment portion. The oscilloscope will be used to see if data is transmitting through the I^2C connections. The software in the pi will confirm whether data was received or not by sending a message back to the microcontroller and vice versa.

Testing the solenoids will occur after the power supplies are tested with their load resistances. The solenoids are observed to see if they turn on or not when powered. The second stage of testing will occur while testing the strain gauge values and the software of the ATMEGA328P. If the solenoids do not react with the signal input from the ATMEGA328P then either the solenoid is not functioning or the software is not working. The other possibility is that the output pin is not functioning.

6.3 Software Test Environment

The testing environment for this very unique project would have to be in the exact environment that it is being designed to function in specifically. The group will have to have most of the system up and connected, preferably at least mounted components on the control board in place; this would be the best thing to do simply because it is hard to test software if one does not have the exact environment in which it is going to run it. The testing environment nevertheless could be simulated, in the case of the temperature, level or PH sensors to see if the microcontrollers are relaying the correct data and timing to the Raspberry Pi. The bulk of this software test will be to analyze the information that the microcontrollers are gathering at each step of the actual dummy test process. By dummy test, the group suggests running the automated system with half a gallon of grain, a very small amount so that precision and perfect timing can be analyzed through the information collection process.

The bulk of this software testing will most likely be done towards the end of the ten to twelve week build time set aside in the scope stated in the executive summary. Throughout this build time, certain components will be tested for accuracy and

timing as is the case of the three sensors mentioned above. But for other electrically driven parts a connection to the microcontroller and the Raspberry Pi should suffice. Parts like the solenoid valves will need to have been set into their containers and fastened correctly in order to test their reliability and efficiency in opening/closing the liquid flow at critical intersections of the brewing process. In order to test the integrity of parts like this, the group will have to have the main frame built to the specifications discussed in the sections above.

When running this test in this environment, the group will need to make sure that the pumps and hoses are aligned correctly since parts of this system will be gravity-fed. Positioning the equipment correctly is crucial to the correct outcome of the software test and its very valuable information. The group needs to know the specifics of the rate of flow of the liquid when it exits each container so that the timer for the next stage can be set correctly internally. Once calculations like that are figured out they can be adjusted for the up to one a half gallons that this automated system will be able to handle. Knowing the little details like that will facilitate the various testing phases that will need to be done in order to calibrate all of the internal systems.

Another big aspect that needs to be taken into consideration is the temperature of the environment in which the testing will take place. Although the majority of these parts are able to withstand over 100 degrees Fahrenheit temperatures, it is not a good idea to put this system in an enclosed area where temperatures can go several Fahrenheit degrees above normal room temperature (by room temperature the group will refer to it as being in between 76 degrees Fahrenheit and 78 degrees Fahrenheit). The reason being is that as all things, whether electronic, metal, wood, etc. they react differently to abrupt changes in their ambient temperatures. The group wants to make sure that this automated home brewing system will be specifically set up in an area that will hold those temperature boundaries, for testing and for operational purposes. This time driven, time sensitive information collecting machine needs to be at its utmost operation capacity near the 99.9% mark, which makes this testing environment restriction a must to follow.

Software components don’t really rely on the temperature on the surroundings, but the hardware that it talking to the software does. Therefore assuring a stable environment for the test to begin is crucial. Having the aluminum frame already built and the components in place is a very good way to insure this stable environment. Although software does become corrupt in occasion’s independent to the hardware and its surroundings that is something that will have to be handled through error detecting and error correcting scripts within the program(s). Another important aspect of the environment that needs to be controlled is that of the cooling process, seeing as there needs to be a “cold” reservoir in place for the wort chiller to utilize when minimizing the cooling time frame of the hot wort. This is another aspect of this system that relies heavily on the room temperature control in order to achieve the lowest possible cooling time.

These processes are mentioned because the software is dependent on their specific hardware components working properly, and the testing environment can clearly affect these processes. If the Raspberry Pi doesn't receive the correct signal from one of the microcontrollers it can't fully and reliable direct the traffic flow like it is supposed to. The reliability of the software in this test environment relies somewhat heavily on the ability of the hardware components, big and small, to function properly within their specific margin of errors. Data corruption can be taken during test phases, and if the testing environment is the culprit further analysis will be done to indicate how badly it affected the hardware and in turn the software communication corruption. The specifics for the handling of data corruption will be discussed in the following section geared towards the software testing.

6.4 Software Specific Testing

For this portion of the testing requirements the necessary software tools to achieve high quality, precise and lasting results will be discussed. In order to have a reliable software integrated with the hardware it is required that several high frequency test are done for the data communications portion of the system. The group has found that if the clock frequency and the period interval of receiving data are too high there will occur several fallouts of communications throughout the send/receive automated process. Finding the right balance of interval for data sending and the correct low clock frequency is key to making sure that this autonomous system doesn't break down with its internal communication protocols. Of course there is a need for an error handling code on the microcontroller side that will restore the communication layer if in fact it does break.

Testing to see how far the Raspberry Pi's RAM is able to go is another good idea. It does come with a standard 512MB RAM, which for a processor its size is quite excellent, but for the amount of back and forth communications it is best to make sure that the internal mechanism of the Raspberry Pi are up to the challenge. Conducting some sort of stress overload test would be of an excellent idea, although ideally the user would never want to reach the highest capacity for the automated brewing system since data logging is such dependent on the compact time intervals. This sort of stress test can be done by increasing the intensity in which the Raspberry Pi is reading the collected values; not necessarily does it have to be actual data, it could be coded into the chips to send high frequency, high priority data for an allotted time. Running out of RAM is never an option, especially if there is an automated process that will be running anywhere from two to three hours respectively. Making sure that all available extra memory is extended for the system to use will be necessary, it is at this point in the testing phase where the group will realize if they need a higher capacity Micro SD card or if the current 8GB one that is in scope will do. The group could also do away with unwanted operating system programs or cancel automatic updates in order to save memory. Since this is basically a closed system, except for the data access done externally by the

user, the Raspberry Pi will be doing the same job(s) over and over again. Having this repetition of tasks will put the Raspbian operating system at test to see how well it does with memory management and thread prioritization. It is not always good to give the system higher memory access but this portion of the software testing phase will conclude whether it is needed or not.

A crucial aspect of this communication highway is also how well the server will be able to update its tables efficiently and correctly while at any given time the user may want to access information within these tables. Testing the integrity of the server and its control over its information can prove to be beneficial if the user decides to collect data at a safer higher time interval. Reading and writing requests can be complex at times depending which portions of the data the user is looking to access. This makes for a very sticky situation if proper software testing is not done with this system functionality. Although theoretically the data is always, 100% available to the user, when an external request comes in to read certain data and the server is currently busy listening and/or talking to the Raspberry Pi then it can throw a warning error/flag. This can let the user know that some information requested may be delayed or not accurate. Doing this error control or error handling in the server side of the system is good practice, since the Raspberry Pi already has so many things to control at once.

The integrity of the ATMEGA328P chips needs to also be tested with scrutiny, since it is not known 100% whether they will work flawlessly or falter during a period of high requests from the system sensors. Since each chip will be in charge of a specific set of sensors, it is necessary to make sure that they are taking all of the readings coming in from the sensors at all times. The chips have to be multitaskers and jugglers all the same time since they will not be able to be idle while the system is functioning. The software dependencies dealing with these scenarios on the chips lie within their driver code(s) and the appropriate C header files. Therefore, the group must make sure to isolate code for handling errors occurring within these dependencies during the test phases. By ‘dependencies’ it is inferred to be the written software that will be placed on the chips themselves, this of course is the software that will work hand in hand with the hardware control units and sensors, so the reliability of that data needs to be at least 99.99% accurate for the system’s purposes. In order to test this specific software component it would be best if the group set up an environment where the sensors and control units would send/receive unordinary values of information. One example could be the putting a temperature sensor in constant hot, very hot, cold and very cold reading environments to make sure the reading capabilities of the ATMEGA328P chips are intact and that in fact the software is being able to keep up and read accurately what is going on.

The group’s main board will also have to be tested vigorously since it will integrate all of these data lines and power supplies from the hardware to the software. In specific the power supply will need to be tested as well, in case there were to occur a power outage or an abrupt spike in the current for the system the group needs

to make sure the software is able to respond quickly and correctly to these unknowns. A quick spike in current can affect some hardware components, in turn affecting the software. Although the group does have a power supply protection established, it is in the group's best interest to have the software listening in closely to the power supply status all throughout the process. In the case that there is a power failure or a power surge, the software must be able to perform adequately to maintain the system intact an operational. This can be tested with a small controlled lab experiment to see how well the board does with its power protection software.

In the end following these software specific testing guidelines will make sure that the automated system is running full speed and correctly. With so many integrated components, both electrically and in hardware, it can be easy to overlook certain software test specifics. Throughout the life cycle of the project, especially during the build time set for this system, the group will add more and more software specific tests. At the end of the day every software component needs to be tested for complete accuracy. The high level software components specified above will be followed step by step, and necessary steps will be taken, also recorded and incorporated, if unplanned behaviors occur during the testing phases.

7.0 Parts Acquisition and Bill of Materials

An overview of the total cost for the project systems is detailed in the tables below. Table 7.1 covers the communications materials, Table 7.2 covers the instrumentation materials, Table 7.3 covers the power supply materials, and Table 7.4 covers the apparatus materials. Table 7.5 sums the cost of each sub system.

Table 7.1: Communications Bill of Materials

Description	Part Number	Vendor	Quantity	Unit Price	Extended Price
ATEMGA328P	68T2944	Newark	2	$3.38	$6.76
MOSFET	58K9650	Newark	2	$0.38	$0.76
1kΩ Resistor	38K0327	Newark	10	0.02	$0.20
150Ω Resistor	38K0339	Newark	10	$0.02	$0.20
1.5kΩ Resistor	78R4869	Newark	10	0.32	$3.20
4.7kΩ Resistor	38K0376	Newark	5	$0.02	$0.10
Green LED	58K2496	Newark	10	$0.18	$1.80
Red LED	96K2564	Newark	10	$0.07	$0.70
				Subtotal	$13.72

Table 7.2: Instrumentation Bill of Materials					
Description	**Part Number**	**Vendor**	**Quantity**	**Unit Price**	**Extended Price**
Quad Op-Amp	LT1014DN	Linear Tech.	4	$4.17	$16.68
Thermistor	NTCAIMME3C 90373	Mouser	4	$1.88	$7.52
470Ω Resistor	02F1416	Newark	4	$0.10	$0.40
4kΩ Resistor	846FD35	Newark	12	$0.10	$1.20
1kΩ Resistor	5489451D	Newark	6	$0.10	$0.60
2.3kΩ Resistor	8416955D	Newark	6	$0.10	$0.60
9kΩ Resistor	2162FDB5	Newark	2	$0.10	$0.20
1uF Capacitor	1654484C	Newark	12	$0.19	$2.28
Strain Gauge	-	eBay	2	$6.85	$13.70
Relay	OJ-SS-124LMH2	Mouser	12	$1.12	$13.44
Relay	FTR-K3JB024W	Mouser	4	$2.21	$8.84
Optocoupler	FOD817A	Mouser	20	$0.43	$8.60
Transistor	2N3904BU	Mouser	20	$0.19	$3.80
1kΩ Resistor	MFR-12FTF52-1K	Mouser	20	$0.12	$2.40
165Ω Resistor	RN55D1650FB14	Mouser	20	$0.10	$2.00
Diode	1N4149TR	Mouser	20	$0.09	$1.80
				Subtotal	$84.06

Table 7.3: Power Supply Bill of Materials					
Description	**Part Number**	**Vendor**	**Quantity**	**Unit Cost**	**Extended Cost**
Transformer	186F20	Mouser	1	$22.36	$22.36
Bridge Rectifier	583-BR62	Mouser	1	$0.96	$0.96
C1 - 6800 uF	-	Mouser	1	$1.68	$1.68
C2 - 47 uF	-	Mouser	1	$0.42	$0.42
C3 - 1 nF	-	Mouser	1	$0.15	$0.15
C4 - 120 pF	-	Mouser	1	$0.15	$0.15
C5 - 33 uF	-	Mouser	4	$0.20	$0.80
C6 - 0.1 uF	-	Mouser	3	$0.06	$0.18
R1 - 17.8 kΩ	-	Mouser	1	$0.10	$0.10
R2 - 20 mΩ	-	Mouser	1	$0.10	$0.10
R4 - 182 kΩ	-	Mouser	1	$0.10	$0.10
R5 - 10 kΩ	-	Mouser	1	$0.10	$0.10
L1 - 15 uH	-	Mouser	1	$0.12	$0.12
Zener Diode	MBR0540T1G	Mouser	1	$0.36	$0.36
MosFET	SI4401BDY	Mouser	1	$1.26	$1.26
Buck Controller	LM3485	Mouser	1	$1.68	$1.68
5V Linear Reg	LM7805T	Mouser	2	$0.69	$1.38
3.3V Linear Reg	L78L33ACZ	Mouser	1	$0.42	$0.42
				Subtotal	$32.32

Table 7.4: Apparatus Bill of Materials					
Description	**Part Number**	**Vendor**	**Quantity**	**Unit Cost**	**Extended Cost**
80/20-10series 97” length	1010-97	Amazon	3	$31.30	$93.90
80/20-10series fasteners (x25)	3393-25pk	Amazon	1	$17.12	$17.12
80/20 10series Corner Bracket	4119	Amazon	8	$3.81	$30.48
Heating Element	Camco 02142	Amazon	1	$9.42	$9.42
Bulkhead fitting	1698T32	McMaster	1	$10.12	$10.12
Viton O-ring	5577K219	McMaster	1	$8.28	$8.28
Heat Exchanger	B3-12A	Duda Diesel	1	$48.87	$48.87
Hose Clamps	54155K11	McMaster	1	$11.13	$11.13
Submersible Pump	1E96208F54	Ebay	1	$6.99	$6.99
Solenoid Valves 24VDC	2WJ04008N	Duda Diesel	8	$28.49	$227.92
Carboy	-	Northern Brewer	1	$4.95	$4.95
2gal Kettle	-	Northern Brewer	1	$14.95	$14.95
Mashtun	-	Walmart	1	$14.95	$14.95
Sparge Tank	-	Target	1	$6.99	$6.99
Wort Pump	Chugger	Northern Brewer	1	$130.00	$130.00
				Subtotal	$636.07

Table 7.5: Total Bill of Materials	
Sub-System	**Subtotal**
Communication Bill of Materials	$13.72
Instrumentation Bill of Materials	$84.06
Power Supply Bill of Materials	$32.32
Apparatus Bill of Materials	$636.07
Total	$766.17

8.0 Administrative Content

The following information is on the budget and milestones set for this project. The initial budget planned for the project was $800 and then adjusted to $1000 when cost of the different components was considered. The most expensive portions of the system were the mechanical subsystems. The mechanical subsystems also caused issues elsewhere and is discussed in the milestone sections.

Budgeting was given more to the mechanical subsystem. All the subsystems taken into account are the mechanical, electrical, PCB cost, and power supply. The complete system expense of the project will be broken down at the bill of materials.

The milestone section will breakdown the group's schedule for completing the automatic brew extractor. It covers the overall plan for both sections of senior design. The milestones were set at the beginning of the project and made to change as the group continued on. The changing milestone goals moved the project along. Any issues that occurred with the schedule will be discussed along with the solutions that the team implemented.

8.1 Milestone Discussion

The group's overall milestone timeline for both senior design 1 and 2 is listed as the following. Using this timeline the team kept organized and steady as the initial project comes to an end and the construction of the brew extractor begins. Discussed in the outline are the expectations of each goal how the group planned on reaching it. Later on the paper discusses the issues reached and overcame through the semester. It will also cover what milestones were not reached and why.

1) Milestones: Fall 2014
    a) September 18th – Finalized preliminary system's structure.
        i) The mechanical subsystems are defined and divided amongst the members
        ii) The electrical subsystems driving the mechanical ones are discussed
        iii) Initial budget is decided.
    b) September 25th – Subsystems identified and divided
        i) The electrical subsystems are decided and extra attention is brought based on what each member wishes to bring to the extractor.
        ii) Preliminary research is done on the necessary components i.e. microcontrollers and operational amplifier research.
    c) October 2nd and 9th – Subsystem milestones set and research ongoing
        i) Research to be completed on microcontrollers and operational amplifiers within one week.
        ii) Research into effective ways of cooling wort for fermentation preparation.

iii) Research into effective single board computer for storing data and controlling a web server.
(1) Possible android application to go with web server.
iv) Research into effective ways of collecting analog data from the extractor that are necessary for the user to effectively make wort.

d) October 16th – Subsystem initial design collaboration and review
i) The operational amplifiers for the filter design and amplification stage are decided.
ii) The microcontrollers and web server host are decided.
iii) Sensors decisions are completed
iv) Power supply is to be discussed either built or bought.
v) Considerations about the budget should be addressed here.

e) October 23rd – Continued design review
i) Final power supply decision is made.
ii) User interface is researched.
iii) Initial power supply designs are made.
iv) Completed filter and amplification circuit designs integrated with sensors.

f) October 30th- Continued design review
i) Finalized design is made and plans for building prototype begins.
ii) Testing criteria is made with conjunction with specifications of the brew extractor.

g) November 6th – Order parts for prototyping
i) Testing criteria is finalized for each of the subsystems
ii) Any budget adjustments will be made here.

h) November 13th – Breadboard Testing (if parts are here), coding, and continued planning
i) The coding for communication and between microcontroller and single board computer is started.
ii) Actual testing of the sensors begin through the filter and amp stages.

i) November 20th – Breadboard Testing, coding, and continued planning
i) The coding for communication and between microcontroller and single board computer is finished.

j) December 15th – Breadboard Testing, coding, and design finalization
i) PCB layout begins to be laid out in Eagle.

2) Milestones: Spring 2015(dates subject to change)

a) January 8th – PCB layout done in CADSoft Eagle/all parts ordered
i) Final simulations are ran and hardware board testing is complete for individual stages of the subsystem
ii) The individual hardware stages are integrated together on PCB layout.

b) January 12th – Send design to board house
i) Troubleshoot any system integration issues.

c) January 15th – Begin assembling brewing system
i) The construction of support structure for mechanical subsystems begins
ii) Final dry runs are made using the breadboard before applying to the PCB.

iii) Full integration of hardware and software plans to begin.

d) January 22nd – Continue assembling system
   i) Continued dry run test are made if necessary
   ii) Web server to be set up and complete to be used in dry run test
e) February 5th - User interface integrated
   i) Dry runs are in conjunction with user interface to test software.
   ii) Coding corrected if necessary
   iii) Hardware tested for continued reliability
f) February 26th – System assembled and ready for testing
   i) Mechanical and Electrical subsystems are integrated
   ii) Final testing of the extractor begins
   iii) Troubleshooting if necessary.
g) April 2nd – Project complete and operational

8.2 Milestone Issues

The team's initial milestone goals were met rather well. The team was performing according to schedule until the subsystems were divided up. The decisions of components to be used took slightly longer because power supplies were not decided at an earlier time and decided.

The other issues were the power being single supply or double supply for the operational amplifiers. Creating a negative source was considered to be more of a hassle and thus single supply was chosen. Single supply source operational amplifier usually turn on at higher total voltages then a dual power supply. An example an operational amplifier operating with ±1.5Vcc dual supply would need at least 3.5Vcc single supply to operate. This made looking for operational amplifiers that could run at the desired criteria harder to find.

The I^2C protocol was another issue that slowed down the design process. The ATMEGA328P and raspberry pi were not communicating with each other. Signals were being sent to the pi and it was unresponsive. The pi was not verifying data was sent to it by sending a receive signal. After testing and looking into the code program and hardware specifications, it was found that the pi was sending back data at two low of a voltage. That is why a level shifting circuit was implemented into the I^2C bus.

There were problems with achieving the initial mechanical subsystems goal. The biggest being the heating of the boil kettle and chilling of the wort. The time and energy consumption for heating a ten gallon kettle was too much on the system. The team could not lower the amount of time the kettle needed to reach a rolling boil without drawing too much power. The same issue with power consumption was encountered with the chilling process.

The chilling subsystem was remedied by replacing the initial idea of using a coolant system with a plate chiller. Solving the heating problems became a bigger problem

as it drew more power to heat in twenty minutes. Solving the problem was tied in with the structural problem solution.

The structural problem was designing the overall frame of the brew extractor. The group noticed that the frame itself was too tall for the average user. The original scope of the project was more directed at an experience brewer looking to make larger batches. The design was changed toward smaller, more experimental batches.

It was determined that the focus should be on the electrical components; therefore, the overall mechanical subsystem was downsized to a tenth of the original plan. This solved both the structure height and heating problems. The heating problems were solved because a smaller kettle could be used; therefore, a smaller heating element will be used drawing less power.

Another issue encountered was parts not coming in on time. This slowed down the hardware testing process. This also affected the testing criteria implemented for the system. This made the hardware testing fall behind schedule and thus pull back the software integration. This was resolved by simulating the values assumed from the sensors and the software was tested. In the end the original plan was flipped with the coding being tested and mostly complete. The hardware will continue to receive testing to remedy the issue. Many of the systems have been simulated so any other problems were addressed in the previous sections of this paper.

9.0 Executive Conclusion

The proposed Senior Design project at hand made significant progress this semester towards effectively building, formatting, testing and engineering an automated home brewing system tailored to the specific needs of the home brewer and his/her recipe(s). The group discovered that there are several products on the market, some of which emulate the goals of this design team perfectly. Through gathering the bill of materials the group discovered that the design project was comparable in price to a lot of the existing systems on the market currently. Which was surprising to the group since this system incorporates quite a few electronic elements that most other systems do not carry on their own.

Nevertheless the team's main goal is not to present an affordable and fine-tuned piece of machinery, instead it is to bring a *tailored* finished product that can be freely manipulated by the home brewer for a more precise and genuine home tasting brew finish. Furthermore this project is mainly to demonstrate a proof of concept. The automated home brewing system, or A.B.E as it is to be known, remains to be comprised of a portable stable aluminum frame, along with the various sensors and circuits that are needed for the internal system, a central unit for data gathering and processing, two microcontrollers to ease the workload of programming communications and control, and finally a web server available for

the user of the Internet along with a very simple Android application for on-the-go monitoring.

As detailed previously in this document, there are several components that are being planned to be installed in this automated system. However, since the project was scaled down to produce significantly less product per cycle, the system will no longer be as complex as originally projected. The end user will lose no functionality in the system performance but will experience a decrease in the output per cycle. However, since the process output has been decreased, we are predicting an overall decrease in the amount of time required to perform a single brewing cycle. This will allow the user to theoretically produce the same amount of beer, but not necessarily all at once, and will in turn allow for a relatively quick turnaround time between batch cycles.

The group worked hard to find ways to keep the system interesting to the user as well, all throughout the design process. Since brewing beer is a hobby for most home brewers, the group wanted to keep the system interesting and enjoyable while maintaining the autonomy of the system and the hobbyist with more free time. Some of options sacrificed in the design stages were reworked into different features that the team wanted to provide to the end user. The group was able to focus on usability and smoothness of software structures as well as the overall aesthetic feel to the project. Furthermore, the group focused on improving features that had already been planned for this design project, mainly the electrical components that would drive the automation of the system.

As the group moves closer to creating the solution for this design project, the problem of time management versus home brewing quality remains at the top of the project's main objectives. Therefore this objective, along with demonstrating this project’s engineering capabilities, will be seen all throughout next semester’s progress, and of course in the live presentation of this one of a kind automated brewing system. There were several challenges for this semester that the group encountered. These challenges were met and overcome with a fantastic group effort and the team succeeded in meeting the milestone goals set forth at the beginning of the semester. As the group continues to move forward, the main goal as a group regarding this matter will be to stay very close to the proposed: design specifications, testing requirements, build schematics, circuit design layouts, software schemas, etc. that were documented in the previous pages of this document. The build time for this specific project has changed slightly considering the modifications and changes to the initial design that were met this semester. The group now estimates that the remaining time left for the build will be very close to ten weeks. The hardware testing and software debugging will be done towards the end of the second semester with the intention of completing a working system 3 weeks before the due date of the project. The finished product will in fact produce drinkable beer, but due to regulatory laws this part of the project will not be showcased in the Senior Design Showcase in April 2015, unless given special permission by the required University of Central Florida department.

Appendix

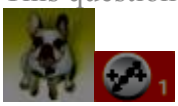
Copyright permissions

Element 14 Community Copyright Permission

Complete thread can be found at:
http://www.element14.com/community/message/131162/l/re-copyright-permission-to-use-a-picture-from-element14#131162

Copyright permission to use a picture from Element14

This question is Not Answered.

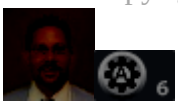


drodriguez128 Nov 11, 2014 8:37 PM

Hello,
I am currently a Computer Engineering student at the University of Central Florida, for our senior design project we are utilizing the RaspberryPi Model B+ and I was wondering if it would be ok to use the image that I found online belonging to Element14. I could not find anywhere on your website who to contact regarding this matter, so I turned to this. I would just like to know if it is ok for my team and I to utilize this picture of the GPIO layout for our project documentation.

Thank you,
David Rodriguez

1. Re: Copyright permission to use a picture from Element14



element14jamie Nov 12, 2014 9:59 AM (in response to drodriguez128)

Hello David,
We have no issues with members using photo's from our site as long as they are for non-profit and as well as putting courtesy of element14 Community along with the picture.

Thank you,

Jamie

Banana Pi Copyright Permission

Dear David:

Ok, no problem.

Best regards,

Banana Pi R&D team

從我的 iPhone 傳送

drodriguez128 <drodriguez128@knights.ucf.edu> 於 2014/12/3 5:18 寫道：

Good evening,

My name is David Rodriguez, I am a senior at the University of Central Florida in Orlando, FL, and I am currently in Senior Design I (for computer & electrical engineers). For our two semester project we have decided that in our automated system we will be using the Raspberry Pi model B+ as our main controller. But we need to compare it to other similar “small single board computer” models, I was wondering if it would be okay to use some of the images and information I found on your website, more specifically the Banana Pi http://www.bananapi.org/p/product.html , this will be incorporated into our detailed report on our two semester project.

We will of course give 100% credit to Banana Pi(or the website) for allowing us to use the copyrighted material, in the report.

Thank you,

David Rodriguez

Raspberry Pi Copyright Permission

Hi

Thank you for your interest in Raspberry Pi. It is very important to follow the instructions on the Trademarks page at http://www.raspberrypi.org/trademark-rules/.

Regards

Nicola Early
Administrator
Raspberry Pi
nicola@raspberrypi.org

From: drodriguez128 [mailto:drodriguez128@knights.ucf.edu]
Sent: 30 November 2014 01:11
To: Admin
Subject: Image Copyright Permission

Good evening,

My name is David Rodriguez, I am a senior at the University of Central Florida in Orlando, FL, and I am currently in Senior Design I (for computer & electrical engineers). For our two semester project we have decided that in our automated system we will be using the Raspberry Pi model B+ as our main controller. I was wondering if it would be okay to use some of the images I found on your website, www.raspberrypi.org, since we also have to write a detailed report on our two semester project.

The images my group and I will use will only be the Raspberry Pi B+ images, and we will give 100% credit to the Raspberry Pi Foundation (or the website) for allowing us to use the copyrighted images.

Thank you,

David Rodriguez

Olimex Copyright Permission

Hi David,
no problem to use pictures and information from our web site if you make sure that you quote the source of this information.
Thanks
Tsvetan

On 12/03/2014 12:36 AM, drodriguez128 wrote:
Good evening,

My name is David Rodriguez, I am a senior at the University of Central Florida in Orlando, FL, and I am currently in Senior Design I (for computer & electrical engineers). For our two semester project we have decided that in our automated system we will be using the Raspberry Pi model B+ as our main controller. But we need to compare it to other similar "small single board computer" models, I was wondering if it would be okay to use some of the images and information I found on your website, more specifically the A13_OLinuXino-WiFi Enabled https://www.olimex.com/Products/OLinuXino/A13/A13-OLinuXino/resources/A13-OLINUXINO.pdf , this will be incorporated into our detailed report on our two semester project.

We will of course give 100% credit to Olimex (or the website) for allowing us to use the copyrighted material, in the report.

Thank you,

David Rodriguez

SYNEK Permission

Robert Bower
Nov 10, 2014

Hello Steve,

My name is Robert Bower and i am an electrical engineering student at UCF working on a home brew automation project for senior design. I would like to use some pictures of your system from the SYNEK website. The pictures would be used purely for illustrative purposes as a means of explaining existing products on the market. Any feedback would be greatly appreciated. Thanks in advance.

Robert Bower

Steve Young
Nov 10, 2014

Report Spam

Hi Robert, there are copious picture of the SYNEK on the website and www.syneksystem.com/press/, but note that many are not the final design. They are all for public consumption, though.

What is your project? What is home brew automation?

PicoBrew Image Use Permission

Robert Bower
Nov 10, 2014

Hello,

My name is Robert Bower and i am an electrical engineering student at the University of Central Florida, Orlando. I am working on an automated brewing system for my senior design class and would like to use some images from your website. The images would be used purely for illustrative purposes in describing systems that are currently available on the market. Any feedback is greatly appreciated. Thanks in advance.

Robert Bower

PicoBrew LLC
Nov 26, 2014

Report Spam

Hello Robert,

Feel free to use the images in this way and good luck on your project.

-Douglas White
Customer Relations Manager

Duda Diesel Copyright Permissions

You may use the photos and send me an address and contact information for the heat exchanger.

CHRISTOPHER BOLTON
DUDA ENERGY LLC
1112 BROOKS ST SE
DECATUR AL 35601
256-340-4866

-----Original Message-----
From: aubilla@knights.ucf.edu [mailto:aubilla@knights.ucf.edu]
Sent: Tuesday, December 02, 2014 10:15 PM
To: technical@dudadiesel.com
Subject: DudaDiesel Question

I apologize for using this contact form for something other than its intended use, but I could not see any other method of contact.

Hi, I am an engineering student at the University of Central Florida. I am currently writing a report about designing an automated home brewing system for personal use. I will be using your B3-12A heat exchanger for the wort chilling process and I was wondering if you could grant me permission to use the following two images in my report.

http://www.dudadiesel.com/img/HXflow2.jpg
http://www.dudadiesel.com/img/items/HX1210_ID648-S.jpg

I was also wondering if you would be interested in donating the heat exchanger in exchange for dudadiesel.com advertising during our senior design showcase where hundreds if not thousands of engineers and DIY's alike will see the product in operation.

Thank you for your time.

WEBstaurant Copyright Permission

Hello,
Thank you for your interest in our website! You may use those images but you will need to site out store as the place you got the images. Just let me know if you have any questions. Thanks!
Tricia Wilkerson
Kentucky Customer Solutions Specialist
Customer Support @ WEBstaurantstore.com
"Like" us on Facebook
Follow us on Twitter, Google+, Pinterest

On Dec 2, 2014 5:22 pm, aubilla wrote:
Hi, I am an engineering student at the University of Central Florida and I am currently writing a report considering possible designs for standalone beer brewing systems. In this report I detail different methods of heating liquids which are necessary throughout the brewing process.
I would like permission to use some of your product images in my report. Please note that none of these products will be spoken about in a negative light, I simply need examples of products in which I will compare the technologies represented by each product.
These are the images which I intend to use for my report:
http://www.webstaurantstore.com/images/products/main/36606/40239/backyard-pro-square-single-burner-outdoor-patio-stove-range-sq14.jpg
http://www.webstaurantstore.com/images/products/main/32777/83092/avantco-eb100-single-burner-countertop-range-120v.jpg
http://www.webstaurantstore.com/images/products/main/108158/132588/sink-heater-element-208v-3000w-1-3-4-oc.jpg
http://www.webstaurantstore.com/images/products/main/17169/129455/avantco-icbtm-20-countertop-induction-range-cooker-120v-1800-watt.jpg?
Thank you.

CaseID: 0D18834D-C35B-0180-5B506C1E47BFE941

DATASHEETS

Atmel

ATmega48A/PA/88A/PA/168A/PA/328/P

ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH

DATASHEET

Features

- High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
  - 256/512/512/1KBytes EEPROM
  - 512/1K/1K/2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C[1]
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix® acquisition
  - Up to 64 sense channels
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change

- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 4MHz@1.8 - 5.5V, 0 - 10MHz@2.7 - 5.5.V, 0 - 20MHz @ 4.5 - 5.5V
- Power Consumption at 1MHz, 1.8V, 25°C
  - Active Mode: 0.2mA
  - Power-down Mode: 0.1µA
  - Power-save Mode: 0.75µA (Including 32kHz RTC)

1. Pin Configurations

Figure 1-1. Pinout ATmega48A/PA/88A/PA/168A/PA/328/P

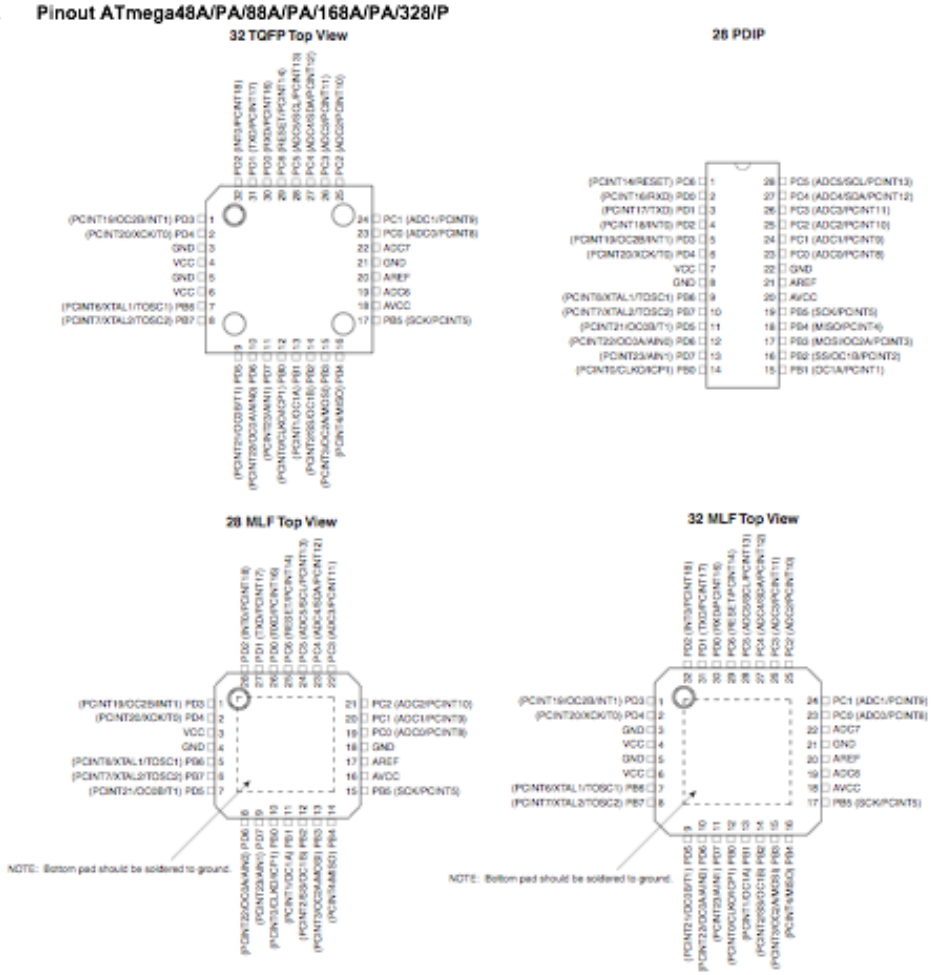


Table 1-1. 32UFBGA - Pinout ATmega48A/48PA/88A/88PA/168A/168PA

	1	2	3	4	5	6
A	PD2	PD1	PC6	PC4	PC2	PC1
B	PD3	PD4	PD0	PC5	PC3	PC0
C	GND	GND			ADC7	GND
D	VDD	VDD			AREF	ADC6
E	PB6	PD6	PB0	PB2	AVDD	PB5
F	PB7	PD5	PD7	PB1	PB3	PB4

Raspberry Pi

MODEL B+

Product Name	**Raspberry Pi Model B+**
Product Description	The Raspberry Pi Model B+ incorporates a number of enhancements and new features. Improved power consumption, increased connectivity and greater IO are among the improvements to this powerful, small and lightweight ARM based computer.

Specifications

Chip	Broadcom BCM2835 SoC
Core architecture	ARM11
CPU	700 MHz Low Power ARM1176JZFS Applications Processor
GPU	Dual Core VideoCore IV® Multimedia Co-Processor
	Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode
	Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
Memory	512MB SDRAM
Operating System	Boots from Micro SD card, running a version of the Linux operating system
Dimensions	85 x 56 x 17mm
Power	Micro USB socket 5V, 2A

Connectors:

Ethernet	10/100 BaseT Ethernet socket
Video Output	HDMI (rev 1.3 & 1.4)
	Composite RCA (PAL and NTSC)
Audio Output	3.5mm jack, HDMI
USB	4 x USB 2.0 Connector
GPIO Connector	40-pin 2.54 mm (100 mil) expansion header: 2x20 strip
	Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
JTAG	Not populated
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane
Memory Card Slot	SDIO