Autonomous Home Brewing System

Submitted by: Samuel Ward
Student ID: 17004205

Supervisors: Dr. Jeff Kilby (BEngTech)
Dr. Minh Nguyen (BCIS)

School of Engineering, Computer and Mathematical Science

A final year project report presented to the Auckland University of Technology
in partial fulfilment of the requirements of the degree of
Bachelor of Engineering

2021

Statement of Originality

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

……01/11/2021…… …….

Date Samuel Ward

Acknowledgements

First, I would like to express my sincere thanks and great gratitude to my employer the Royal New Zealand Airforce for funding my degrees and allowing my pursuit of educational development.

Secondly, thanks and appreciation go to my project partners, Mike and Boonie for their hard work over the year, and the good times had throughout.

I would like to send my appreciation to the AUT staff that have taught us over the years, and a special mention to our project supervisors Jeff and Minh for their support.

Acknowledgement and thanks to the Whenuapai Brew Club for their input and feedback at the early stages of this project.

Finally, I would like to thank my family Wenna and Leo for their unconditional support over the last three years of study.

Samuel Ward

October 2021

Abstract

The discovery and development of brewing techniques date back to about 12000BC and represent one of the most important technological achievements of humankind [1]. In more recent times the revolution of small batch breweries and craft beer has taken hold around the globe, making its way into the everyday household with the increase in popularity of hobbyist home brewing.

The Brew Buddy team consists of three home brewing enthusiasts with 25 years of electronics engineering experience between them, that through their own experience have discovered a niche gap in the home brewing industry for retrofittable brewing automation software.

Brew Buddy is an autonomous control system that allows home brewing enthusiasts to add automation to their setup utilising their existing equipment in a semi-modular fashion; and is designed to be adaptable and scalable from small to medium sized home breweries. It provides users with automation functions that are not currently available on the home brewing market, releasing them from the tedious process of manual brewing with a simpler implementation and without having to invest in an all-in-one solution.

This report covers the research, design, and construction involved in development of Brew Buddy.

Table of Contents

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project Introduction

This report details the design and construction of an autonomous home brewing system. It will cover:

- Design decisions of the system and the problems it aims to solve.
- Technical specifications of the system and its operation.
- The web-app and how it interacts with the system and end-user.
- Results of testing, operation, and a demonstration.
- Conclusions, challenges, and future work.

We have aptly named our system “Brew Buddy” as it is an aid to existing breweries, or every home brewer’s best friend.

The project team consists of:

- Samuel Ward – Author
- Michael Emmerson – Project Partner
- Jacob Boon – Project Partner
- Jeff Kilby – Project Supervisor (BEngTech)
- Minh Nguyen – Project Supervisor (BCIS)

Figure 1.1 Brew Buddy System

1.2 Purpose

Our project's aim is to develop an automated control system for hobbyist home brewing that is controlled via an app and minimises the requirement for user interaction as much as practicable. Home brewing can take a long time and requires frequent monitoring, we wish to provide a solution that eliminates the need for constant monitoring of the brew through the addition of automation to existing equipment, making the brewing process more enjoyable for experienced brewers, and less overwhelming for beginners.

1.3 Project Objectives

Our project goal was to deliver a viable market product over the course of two Semesters. This would be achieved through the following high-level objectives identified during our planning phase:

- Simplify the brewing process, requiring the least amount of user interaction as possible.
- Interface with the users Wi-Fi network and be controlled through either a mobile or web application.
- Continuously update with live sensor information and progress tracking.
- Allow the user to input their desired recipe information into the app which defines the timings, temperature, and flow rates that the system will operate within.
- To be semi-modular and be able to interface with existing home brew setups.

A significant amount of research was then carried out to find how to best achieve these objectives.

Chapter 2 Research

2.1 Background

Home brewing can be a tedious process which requires a great deal of time, experience, and interaction. Automation is becoming popular among the brewing community with a bunch of new systems having some form of automation. Most of these systems that we have researched are only semi-automated and still require a great deal of user interaction. Existing autonomous systems on the market are also generally an all-in-one solution, where you essentially have to discard your existing equipment and purchase theirs in order to reap the added benefits of automation. We consider this a less than ideal solution as most home brewers already have a lot of expensive and familiar equipment and believe there is a niche gap in the market for a system that adds functionality, automation, and control; enabling the brewer to get the most out of their existing equipment.

2.2 Research Topics

Based on our project objectives, the following research topics were chosen:

- Market survey in conjunction with the Whenuapai Brew Club to establish market needs and design decisions for our system.
- Current similar systems and equipment available on the market.
- System ideas and solutions to the frustrations faced with manual brewing.
- Research in to heating and heater options.
- Components required for building the system.
- Experimentation and testing.
- Power considerations and NZ electricity standards.

This research aided us in deciding what specific functionality we wanted to include in our autonomous home brewing system, and the hardware required to achieve this.

2.3 Survey

To help us determine our system requirements and target areas we engaged with the Whenuapai Brew Club to give us some market feedback in the form of a survey. This deemed to be very helpful and highlighted some areas of improvement we had not considered.

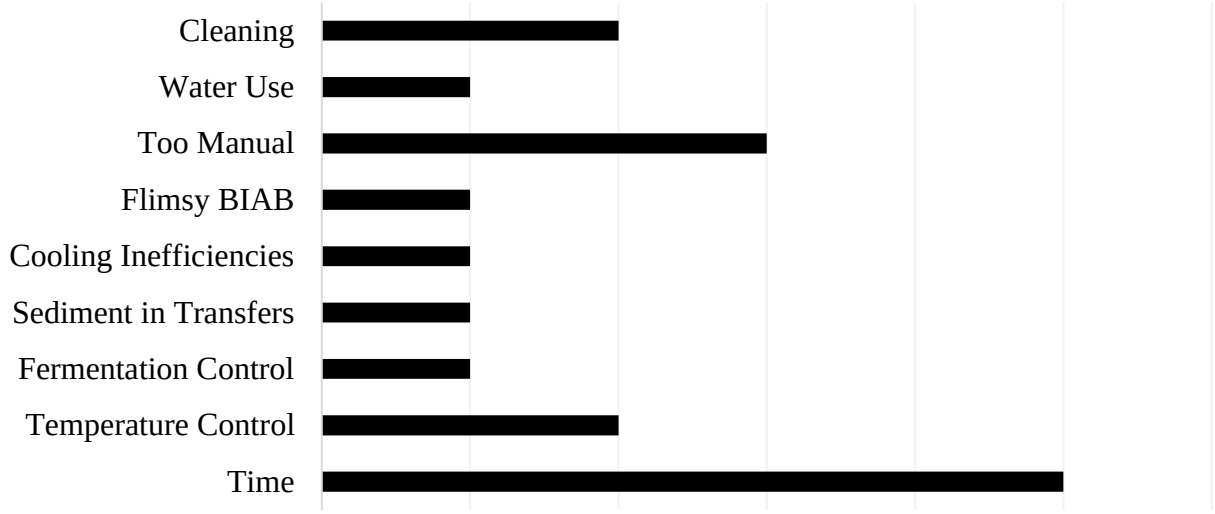


Figure 2.2 Common Frustrations among Home Brewers

Figure 2.1 Suggestions for Automation

As well as gathering feedback on suggestions and frustrations, we also covered some generic questions to help us establish common brewing practices which helped to steer the direction of our project by showing us equipment and technology to use in order to cover the biggest sector of the market.

Using this feedback along with our own experience we identified the following focus points of the project Brew Buddy: Safety by eliminating the need to move vessels containing hot liquids; time saving by being more hands off; and a further level of automation than currently available through temperature control, sparge automation, and cleaning automation.

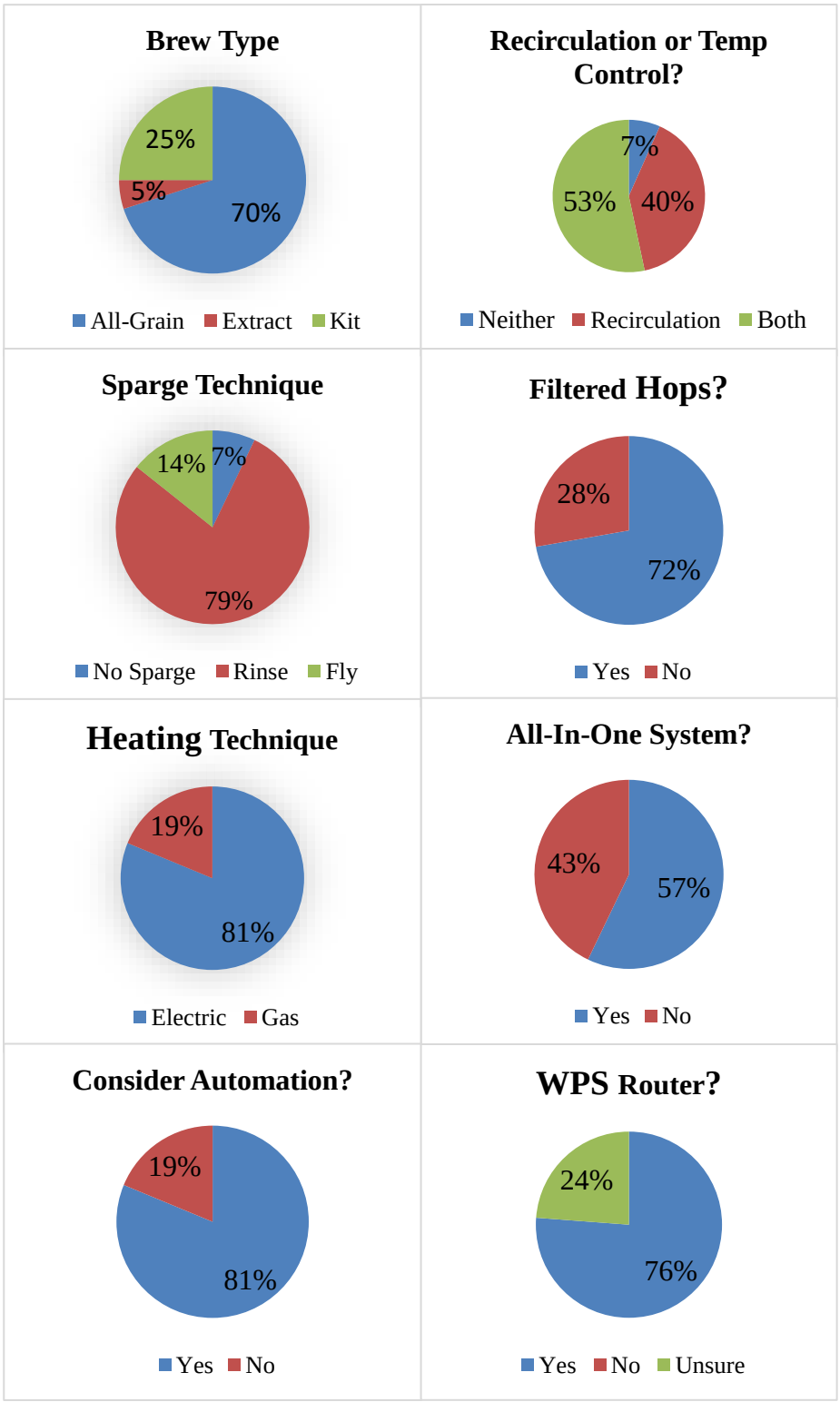


Figure 2.3 Generic Questions

2.4 Existing Market Products

Brand	**Brew Buddy (Planned Features)**	**Grainfather [2]**	**Robobrew [3]**	**Pico-brew [4]**
				
Price	TBA	$1,080.00	$595	$2734.00
Max Batch Size	50+L Depending on users' equipment	23L	30L	10L
User Interaction	Low	Moderate	High	Low
Programmable	✓	✓	✗	✓
Recirculates Wort	✓	✓	✓	✓
Automatic Sparging	✓	✗	✗	✓
All Grain	✓	✓	✓	✓
Temp Control	✓	✓	✓	✓
Self-Cleaning Function	✓	✗	✗	✓
Web/Mobile App	✓	✓	✗	✗
Wi-Fi Connectivity	✓	✓	✗	✓
Distilling Feature	✓	✓	✗	✗
Can be adapted to any kit	✓	✗	✗	✗

Table 2.1 Existing Products Comparison

Comparing features of other available brewing systems, and experiencing first-hand the RoboBrew, helped us to identify exactly what aspects we wanted to cover when designing our product, with the goal of being better than these systems.

2.5 Heating

2.5.1 Sources of Heating

For home brewing there are two commonly used sources of heating:

1. **Electric heating:** Electric heating is easy to control and automate which is already commonly done. For home brewers, electric heating is restricted by amperage limits of standard NZ household wiring, however some users may have higher amperage sockets installed. This limitation can cause problems when boiling a large volume as it will take a substantial amount of time.
2. **Gas heating:** Gas has superior heating capability in terms of energy; however, gas heating is far more dangerous and complex to automate.

Given the complexity involved in gas heating and the fact that electric is far more common, we chose to go with electric heating for our system.

2.5.2 Heating Methods

There are many different methods of electric heating for a home brewery, our objective was to find the most effective and commonplace methods for our system to be able to interface with.

2.5.2.1 Direct Heating

This is where the electric heating element is contained inside the boiler itself, either built in or in the form of an immersion element. This method is cheap and reliable, while minimising required equipment. This method is not the most effective, it can struggle to bring the contents to temperature uniformly, potentially resulting in hot spots and can throw off the automation algorithms, though it can still somewhat be automated by varying the power the to the element. A common method for beginner brewers, but generally upgraded to another method as experience develops.

2.5.2.2 RIMS

Recirculating Infusion Mash System. The RIMS system differs to the direct method in that the heating element is contained inside an external device, and the wort is recirculated through this with a pump [5]. This method ensures the heat is distributed evenly throughout the wort, eliminating the problem of hot spots incurred with the direct heating method. This system can allow for temperature control automation either by varying the power to the heater or varying the flow rate through the device. RIMS is able to convert 100% of its heat to energy, and since the element is immersed in wort it has the most efficient transfer of heat to wort. One of the negatives to this system is that a minimum flow rate must always be maintained otherwise scorching of the wort can occur. This system is much more effective than direct heating and is probably the most common among home brewers.

2.5.2.3 HERMS

Heat Exchange Recirculating Mash System. HERMS is similar to RIMS in that wort is pumped through the system, the difference is that instead of an inline element, the HERMS pumps the wort through a coil inside a hot liquor tank (HLT) with heat being transferred from the hot water through the coil [6]. The wort is never directly heated by the heat source therefore scorching won't occur. The temperature of the wort can be controlled by turning the pump on/off, altering the flow rate or by varying the temperature of the liquid inside the HLT. This method requires another brewing vessel with a heat exchanger so is far bulkier and costs more.

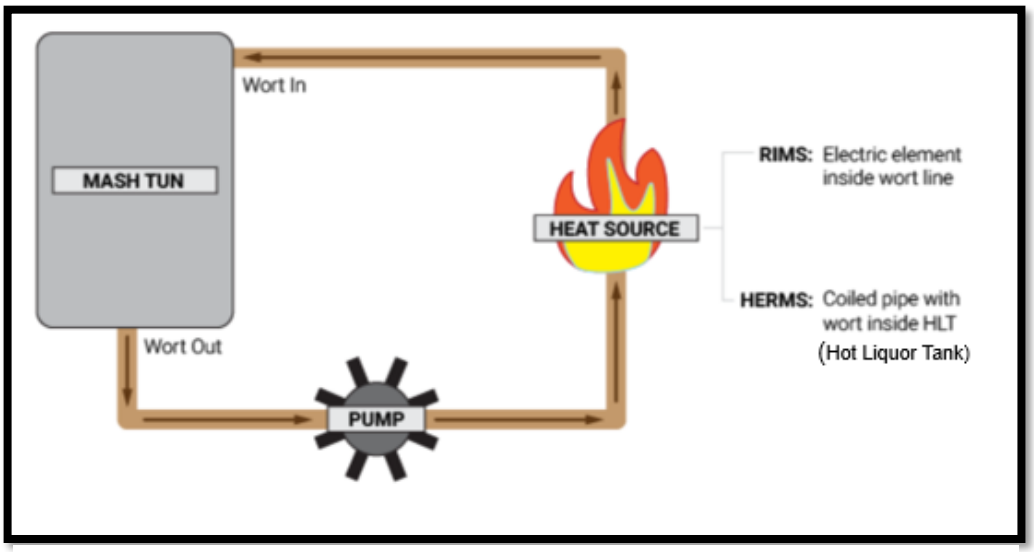


Figure 2.4 RIMS/HERMS Diagram

Calandria/Direct Steam Injection

Calandria and Direct Steam Injection (DSI) are two other methods of heating not commonly found in home breweries. These methods both utilise steam to heat the wort and while they are far more efficient than electric heating methods, they are generally reserved for larger breweries due to their size and cost.

2.5.2.4 Our System

For Brew Buddy we decided to include adaptability to the two more common methods in home brewing, direct heating and RIMS, with the focus for development being RIMS as this requires the most complexity, where direct heating just uses the same algorithms to a lesser extent. This allows Brew Buddy to cover from entry level to intermediate. Our own equipment and demonstration contains a RIMS and we carried out some temperature waterflow testing that is covered later in this report.

2.6 Pump and Flow Rate

2.6.1 Types of Pumps

Our system requires a pump capable of flow rate control, with a range of approximately 400ml/min (sparging) to at least 2.2 ml/min (mashing). The pump also needs to be made of food grade material and withstand temperatures up to 100°C as it will be pumping the hot wort. We investigated two pump options:

1. PWM controlled DC Pump
2. 230VAC Magnetic Drive Pump

Initially we wanted to go with a PWM controlled pump for varying flow rate, however when researching it was difficult to find something that had a suitable output flow and they tended to be quite expensive. This led us to investigate magnetic drive pumps, using a valve on the input for flow control. The magnetic drive pump is able to be run at full power, and with the valve

limiting the flow, it is possible to pump a very low flow rate. Being magnetically coupled and stainless steel it is food and temperature safe and doesn't require any seals.

2.7 Valves

In order to automate our system, we required electrically controlled valves, both for direction and flow rate. Research into various valves was carried out to determine the best options for our system.

2.7.1 Types of Valves

2.7.1.1 Ball Valve

Commonly available in all configurations and sizes and are a comparatively cheap option. For Brew Buddy we require these in a ½" 3-way "L" configuration. The most common type of valve for direction control.

2.7.1.2 Needle Valve

Needle valves allow for precise control over flow rate. Commonly available but are more expensive than ball valves. For Brew Buddy we proposed using one of these in ½" at the input of the pump to control the flow rate which ended up being too restrictive as explained later in this report.

2.7.1.3 Pinch Valve

Pinch valves are less common and more expensive than the other two mentioned. A pinch valve basically works like kinking a garden hose to stop the flow rate. We liked the idea of these valves for hygienic reasons as there are no moving parts in contact with the liquid, however we were unsure of their ability to precisely control flow and their longevity.

Figure 2.5 Pinch Valve

2.7.1.4 Gate Valve

A gate valve works by winding a gate down to shut off flow, allows for precise control and doesn't restrict flow when fully open. This valve would have been ideal for flow control but unfortunately, they are quite expensive, so we didn't go with this option.

2.7.2 Valve Control

Brew Buddy requires both direction control and flow control valves, the direction control valves can be a solenoid, but the flow control must be proportional. Off the shelf versions of these valves turned out to be very expensive, so we ended up engineering our own version of a proportional ball valve which is covered later in this report.

2.8 Power Considerations

Nominal NZ residential voltage is 230V ± 6% with the standard residential wall sockets being rated to 10A [7]. When designing our system, we had to ensure we kept our total current below 10A and built our circuitry to handle 230VAC. The main power draw for our system came from the heater element as we wanted to maximise the power to achieve rapid heating. We used the calculations below to help us choose a heater which would achieve this but would also be within NZ regulations and operate safely without exceeding 10A. We found that the largest NZ residential electric panel / column heaters available operated at 2.4KW which is approximately 38W below the limit as shown below.

Heater power calculation:

$$Pmax=Imax*Vmax$$

$$V=230VAC + 6\% = 243.8V \qquad I=10A$$

$$P=243.8 \times 10 = 2438W \text{ (Max Power)}$$

We decided to include some safety margin to ensure we did not exceed the 10A limit so used the below calculation.

$$P = Pmax\text{- }Pump\ Power - Micro\ and\ Sensor\ Power$$

$$P = 2.438 - 0.1 - 0.001$$

$$P \approx 2.3KW\ @\ 243.8V\ 10A$$

Heater Resistance Calculation:

$$R = \frac{V^2}{P}$$

$$R = \frac{243.8^2}{2300}$$

$$R = 25.84\Omega$$

From this we decided to ensure our heater element of choice would not exceed 2.3KW at 243.8V and was greater than 25.84Ω. When researching other market products, we observed that the power consumption of the pump was quite low. The Grainather G70 [8] which is a very large system only utilised a 32W pump, so we took this into consideration when choosing our own pump. The microcontroller and sensors consumption were less than 1W which was negligible compared to other loads in our circuitry.

We also investigated the option of building our system to withstand 15A in order to accommodate a 3.6kW element, giving more flexibility to experienced brewers with better equipment, at not much added effort.

2.9 Experimentation and Testing

2.9.1 Temperature Loss and Efficiency Testing

Jacob carried out some substantial testing on the temperature losses and the efficiency of his current standard homebrew kettle. During his testing he brought his kettle to boil (100ºC) then monitored the temperature as it cooled over a 15minute period. From this he was able to calculate the radiative and conductive heat losses a standard steel brew kettle dissipated. As radiative and conductive heat is lost at a rate proportional to surface area [9], he was able to

estimate the losses our recirculation system which was approx. 1/3 that of the brew kettle. The below formula was used to find the total energy loss during a boil.

Total energy loss during boil = Passive conduction and radiation losses of kettle + losses of recirculation line + state change losses

Figure 2.6 Brew Kettle Boil Off Testing

The conclusions from his testing and calculations were that we would need a heating element with a minimum power of 1.11kW to remain a boil. At any other temperature apart from boil we would not have to worry about the state change losses so the total energy loss would be a lot lower and subsequently the efficiency would be a lot higher. These results were very useful for us when calculating the time it would take for the wort to reach a specific temperature based on the wattage of the element and its efficiency. An example of this is below:

2.2kW element used during the boil $Efficiency = 1 - \left(\frac{1.11}{2.2}\right) = 49.6\%$

2.2kW element is used at 99ºC $Efficiency = 1 - \left(\frac{0.67}{2.2}\right) = 69.6\%$

2.9.2 RIMS Temperature Waterflow Testing

The purpose of this testing was to see if we could get an adequate consistent flow of water heated from ambient tap temperature through the RIMS system for sparging. The sparging required a flow rate of 500ml per minute at 75 °C. We also experimented with the flow rate to see if we could achieve boiling water. These were the results using our 2.2KW element:

- 10ml/s = 64.5°C
- 8ml/s = 74.8°C
- 5ml/s = 96.3°C
- 3ml/s = 100.6°C

Our conclusion from these results were that the 2.2KW element in the RIMs system was enough to heat 8ml/s (480ml/m) of water to 75°C as it passed through for sparging which aligned with previous calculated requirements and was deemed successful. We were also able to achieve boiling water flow of 3 ml/s. From this we decided that this element would function well in our system.

2.9.3 Electricity Regulations

Brew Buddy requires 230VAC and is powered by a standard jug connector, therefore we needed to ensure our system complied with the New Zealand electricity regulations. This includes ensuring our system is enclosed in a class II housing [10] and will require the device be PAT tested and certified.

2.10 Research Conclusions

2.10.1 Relevance

This research has shown us that Brew Buddy is a viable and possible project. It would be a welcome addition to the home brewing market of beginner and intermediate home brewers.

2.10.2 Key Decisions

Using the feedback from the Whenuapai Brew Club along with our own experience we identified the following focus points of the project:

1. **Safety**: The current status quo of lifting and moving vessels containing hot liquids is dangerous, a closed loop system with integrated cooling functionality will eliminate this danger.
2. **Time**: Brew days take 6-8 hours requiring a lot of focus, through automation we aim to make this process more hands off so the user can free up time to utilised elsewhere.

3. **Further Automation than Currently Available**: Current “automated” systems on the market still require a lot of manual input, Brew Buddy will go beyond this with the goal of a start to finish brew with no user input.
4. **Temperature Control**: Maintaining exact mash and sparge temperatures is difficult in a manual system, this will be automated through sensors and element control.
5. **Sparge automation**: Usually, sparging is done with pre heated water poured over the grain. Our system will heat the water as it is pumped into the mash tun and utilise the fly sparge technique.
6. **Cleaning Automation**: Cleaning equipment after a brew is a lengthy and painful process. Using a tap connection our system will heat, cycle, and discharge water through all the equipment in order to clean it.

Chapter 3 Design Philosophy

3.1 Brew Buddy Integration

Brew buddy is designed to integrate with home brewers existing equipment such as boilers, mash tuns, heating elements, pumps, coolers, and kettles.

It consists of flow and temperature sensor inputs that provide feedback to the control unit which then translates this feedback in to output signals to control servo valve outputs, pumps, and a variable power heater controller. All of this is controlled intuitively via our Brew Buddy web app.

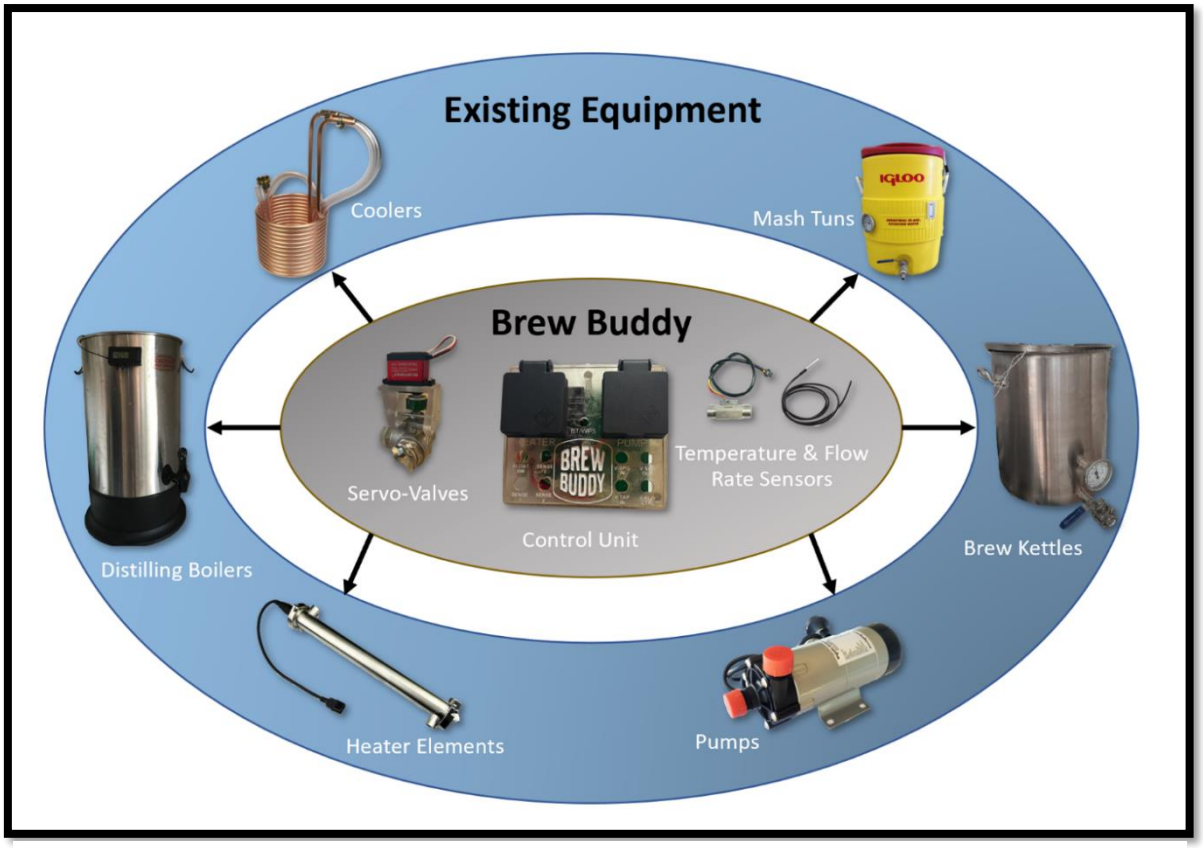


Figure 3.1 Brew Buddy Integration Diagram

3.2 System Functional Block Diagram

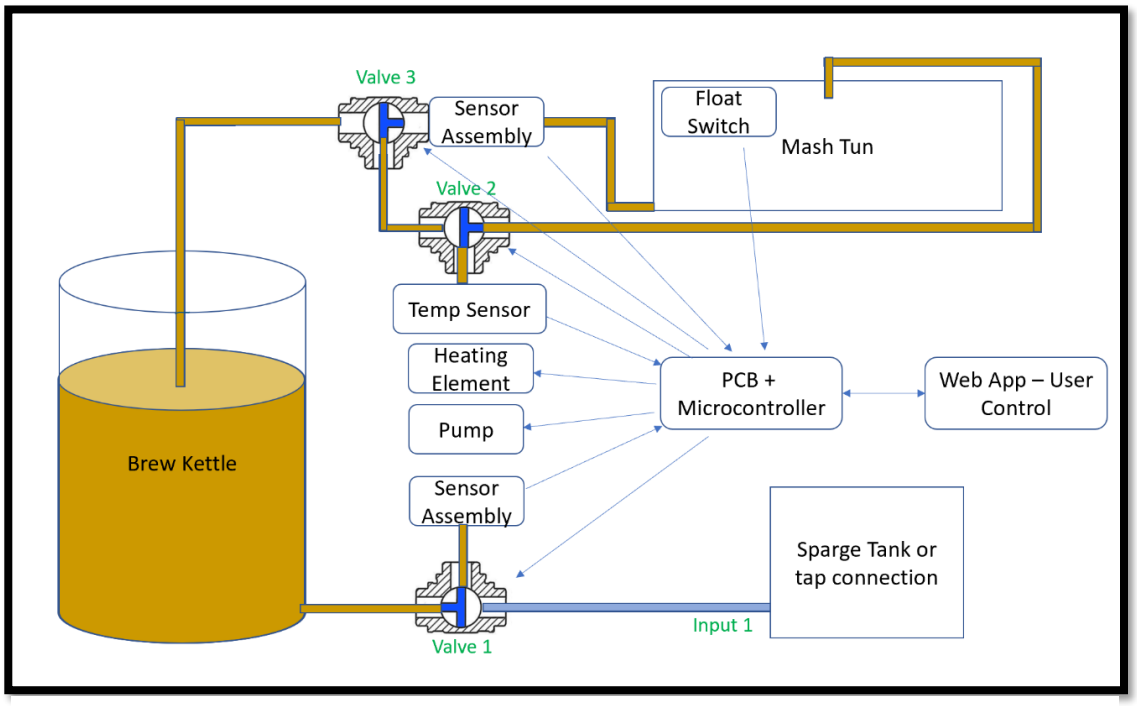


Figure 3.2 Functional Block Diagram

3.3 Automation Functionality

Brew Buddy's automation functionality includes seven main stages, each consisting of a range of separate steps. These are all detailed in the app, simplifying the brewing process, and making it easy even for the beginner brewer to follow. Details of these stages are below:

Cleaning: A cleaning cycle can be easily initiated from the main menu, pumping sanitized solution around the system then draining the wastewater upon completion. It is imperative that the whole system is sanitized and cleaned so no added bacteria enter the brew as this can be catastrophic and ruin the batch.

Mash: To begin the brew grains are manually added to the mash tun. Once the water of the system reaches the "strike temperature" which is generally around 75°C–80°C, valve

2 switches its output, allowing hot water to pass through the grains in the mash tun. Hot "wort" is continually recirculated, and the user defined temperature is maintained until the mash stage is complete.

Sparging: This step is required for achieving the desirable efficiency of sugar extraction from the grain. Hot water is sprinkled over the top of the grain allowing it to trickle through the bed. Sparge water flow rate is automatically adjusted. Once the desired amount of sparge water has been delivered to the mash tun, valve 1 switches inputs and the pump and element are turned off until all the sparge water has been drained from the mash tun.

Boil: At the beginning of the boil stage, valves 2 and 3 will switch outputs to isolate the mash tun. The pump and the heater are switched on and the wort is recirculated through the system and heated to 100°C. Once boiling begins it will continue for as long as the defined recipe calls for. Prompts are given to the user when the hops and adjuncts are to be added.

Cooling: Once the boil stage is complete, the wort needs to be quickly moved to fermentation temperature (20-22°C for most ales) to minimize the impact of any bacteria forming in the beer. Near the beginning of the boil stage, the user will be prompted to disconnect the mash tun and connect an external chiller in its place. Near the end of the boil stage valves 2 and 3 will switch allowing boiling wort to pass through the immersion chiller to sanitize it. The chiller is then immersed in a cold liquid tank (CLT) and the wort is recirculated until the temperature is lowered to the desire value.

Transfer: After the cooling phase is complete the wort can then be pumped into a fermenter where the yeast can be added. The batch is then stored for at least two weeks before bottling/kegging takes place.

Passive: At the end of the brew the system goes to a passive stage which zeroizes all active components and waits for user command. At this point the user would normally select the cleaning function.

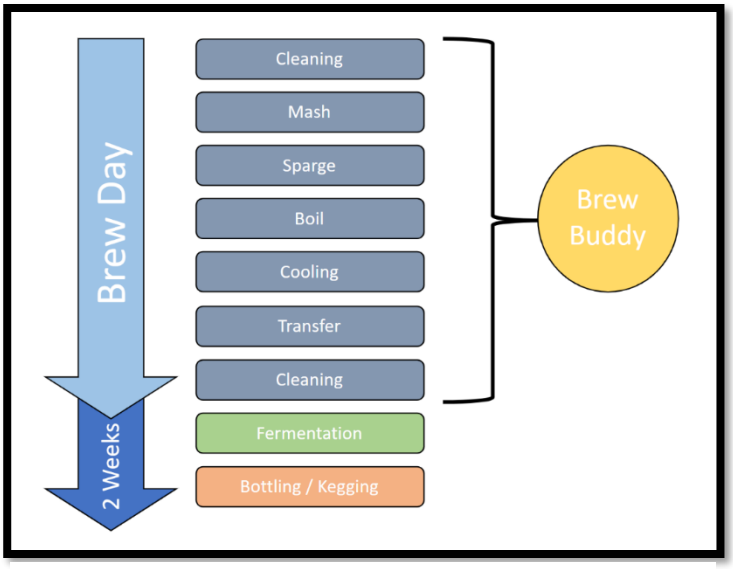


Figure 3.3 Brewing Stages

3.4 Project Management

For this project we chose to go with the Waterfall development approach, which follows a linear approach to project management. Waterfall is a methodology predominantly used in engineering projects, and while this project contains both Electronic Engineering and Software Development components, we decided that overall, it falls in the realm of an engineering project. The Software aspect of the project (the web-app) is designed around the electronic hardware rather than being feature driven and though there will be some iterative design elements in terms of the user interface, I did not consider this significant enough to justify an entirely iterative approach; instead breaking development of the app into two stages, initial development and then revision development after testing of the system. In terms of tools, we used Microsoft Teams for sharing information, GitHub for version control, and set up a Trello board for tracking project management and work allocation.

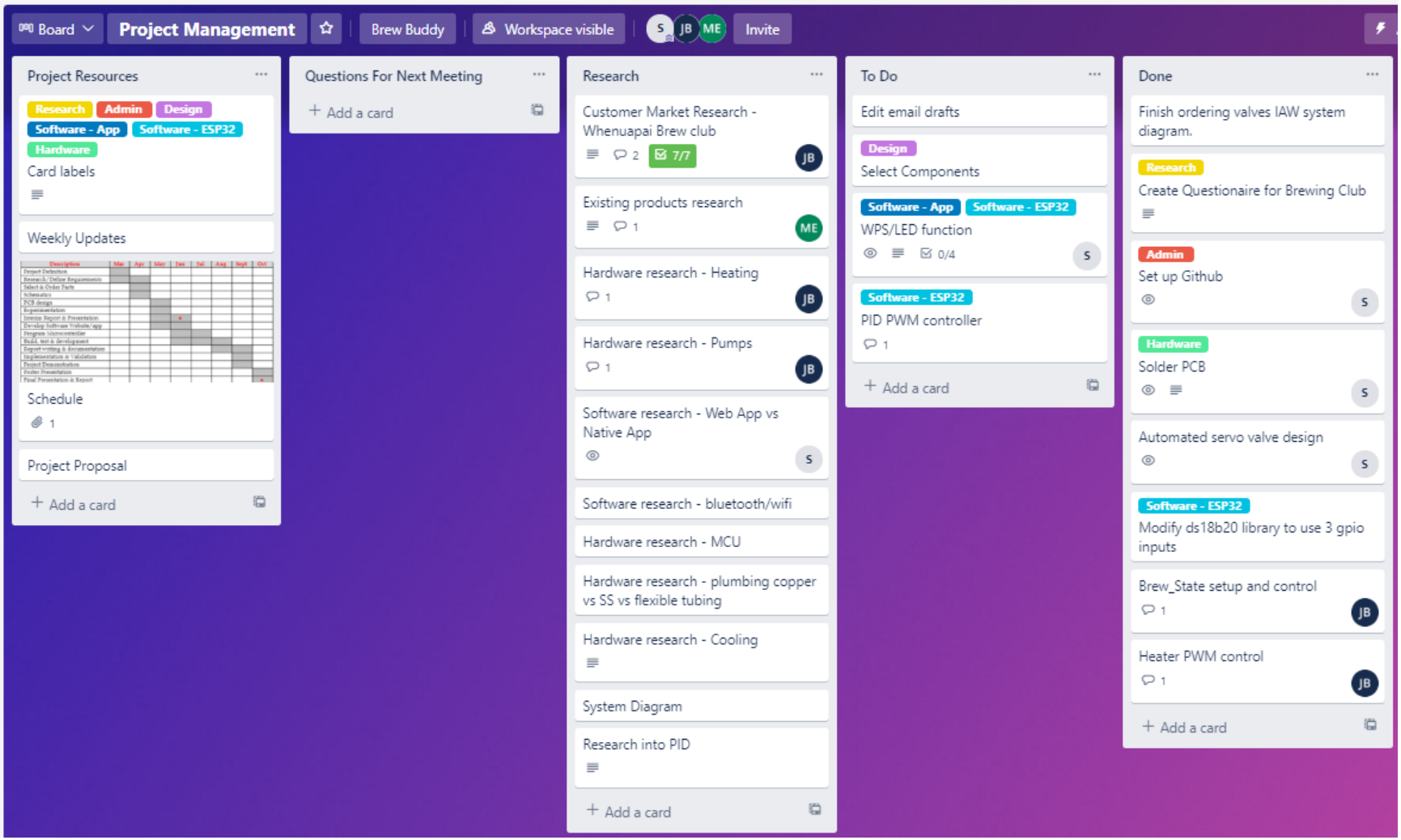


Figure 3.4 Project Management Trello Board

3.4.1 Project Timeline

The project Gantt chart helped us to track our progress throughout the year. We fell slightly behind schedule during Semester 2 due to COVID-19 lockdowns causing delays in shipping of components and not allowing us access to the AUT lab, meaning we were unable to achieve a fully finished product by the end of the semester.

DESCRIPTION	MAR	APR	MAY	JUN	JUL	AUG	SEPT	OCT
KICK OFF MEETING	◆							
PROJECT PROPOSAL	◆							
SURVEY BREWING CLUB								
BCIS PROJECT PROPOSAL			◆					
RESEARCH/DEFINE REQUIREMENTS			◆					
BCIS UPSKILLING								
SELECT & ORDER PARTS								
EXPERIMENTATION								
SCHEMATICS								
INTERIM REVIEW & PRESENTATION			◆					
PCB DESIGN								
DEVELOP SOFTWARE APP								
PROGRAM MICROCONTROLLER								
BUILD, TEST & DEVELOPMENT								
REVISE APP								
REPORT WRITING & DOCUMENTATION								
IMPLEMENTATION & VALIDATION								
PROJECT DEMONSTRATION								
POSTER PRESENTATION								◆
FINAL PRESENTATION & REPORT								◆

Table 3.1 Project Gantt Chart

3.4.2 Work Breakdown Structure

The work breakdown structure gives an overview of allocated tasks and work completed by each team member throughout the year.

Figure 3.5 Work Breakdown Structure

Chapter 4 Technical Specifications

4.1 System Block Diagram

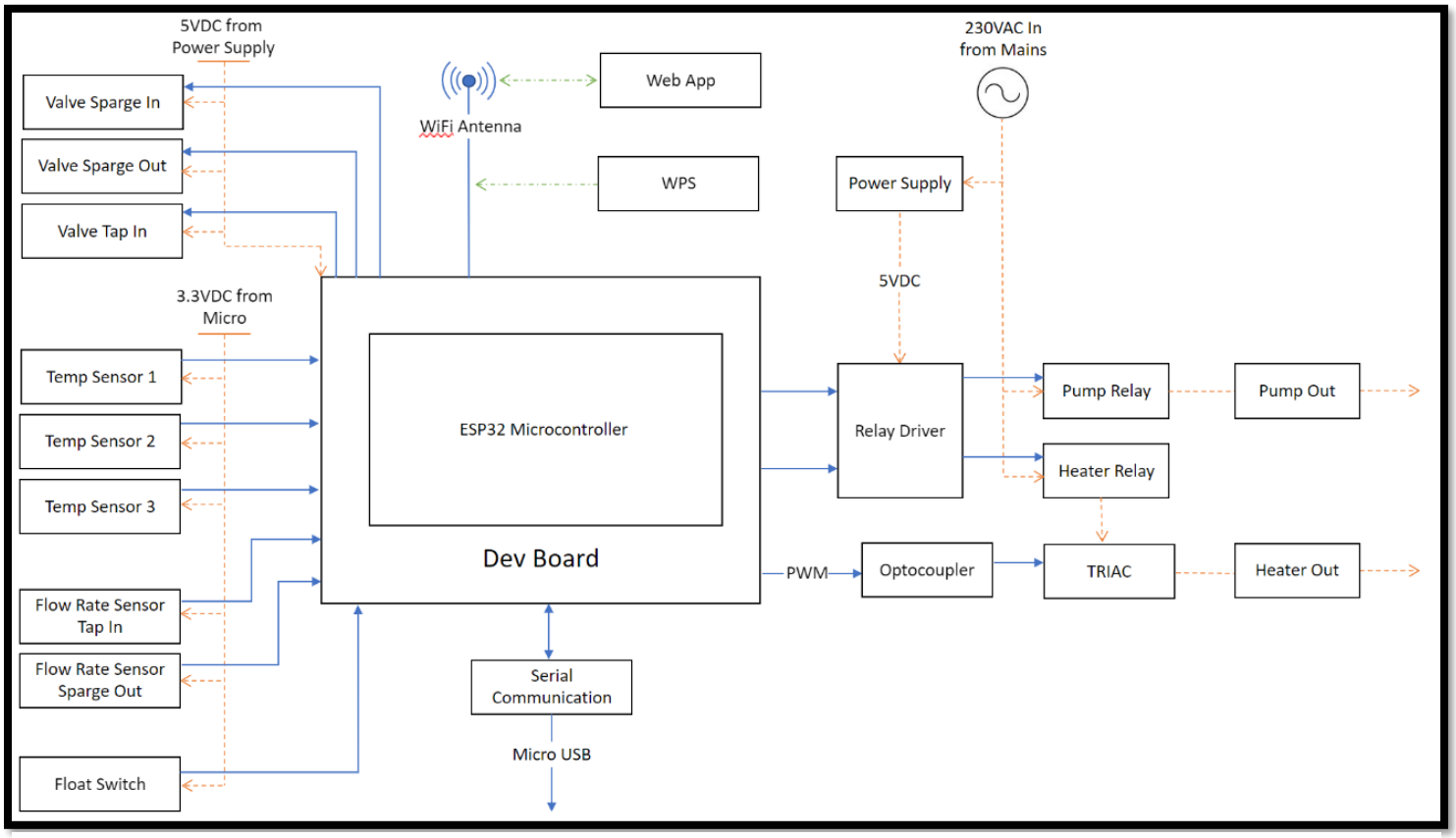


Figure 4.1 System Block Diagram

4.2 Circuit Operation

The Brew Buddy system is powered by a standard 230V AC outlet and consists of a high-power side and a 5V low power side, with an AC-DC converter and filtering circuit providing the 5V. The high-power side consists of relays for heater and pump control, and a proprietary power supply utilising PWM to provide variable power to the heating element.

Various sensors are employed to provide feedback to the microcontroller such as flow

meters, temperature probes, and float switches. This feedback is interpreted by Brew Buddy's control algorithms to control pumps, heater elements, and servo valves. The servo valves are an in-house custom designed and printed three-way servo-controlled ball valve that allows for changing of flow direction and proportional flow control in either direction.

4.3 System Schematics

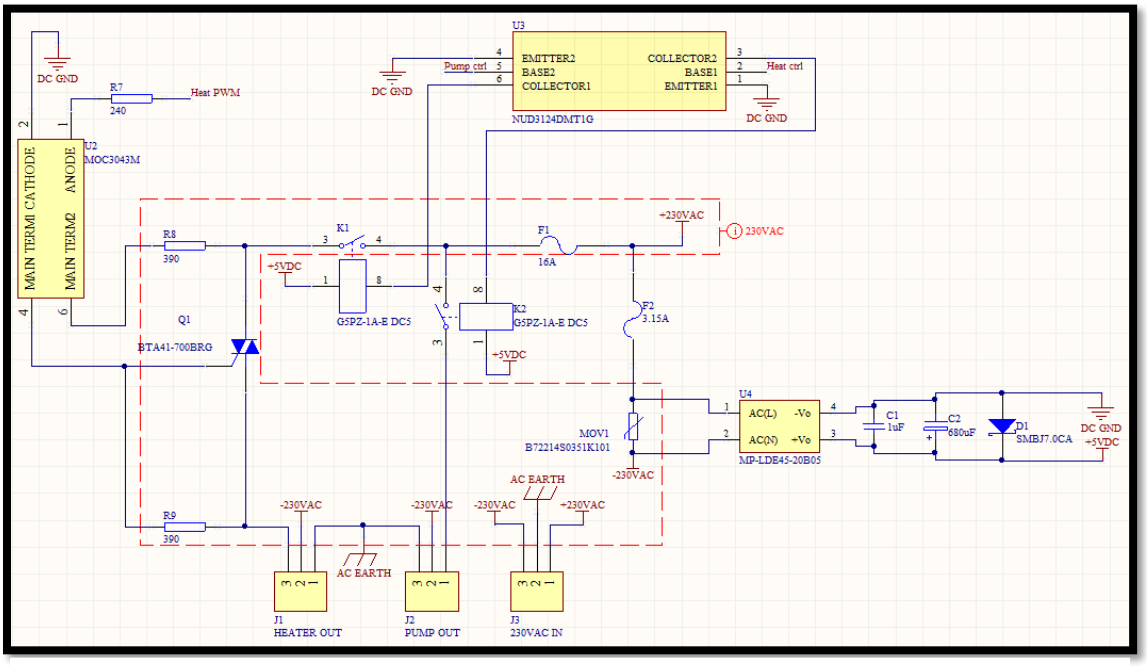


Figure 4.2 Power Schematic

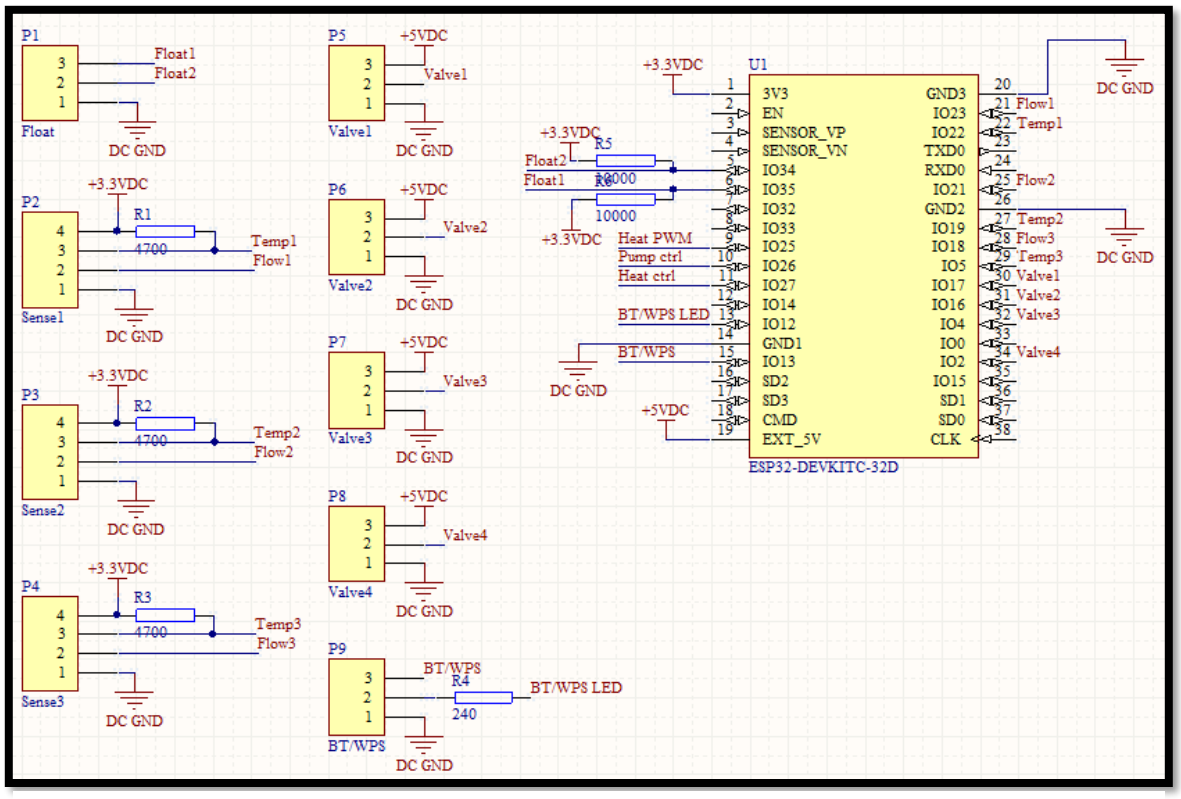


Figure 4.3 ESP32 Breakout Schematic

4.4 Power System

4.4.1 5V Power Supply

Brew Buddy requires 5V to run the ESP32 board and servos. Rather than building our own we decided to go for an off the shelf AC – DC converter, landing on the MP-LDE45-20B05 [11] as it met our specification requirements as follows:

- Input 85AC to 264VAC
- Output Voltage 5VDC
- Output Current 8A
- Over Voltage Protection
- Regulated output, low output ripple & noise
- High efficiency, low power consumption

Figure 4.4 MP-LDE45-20B05

The surrounding circuitry is a straight forward circuit taken from the datasheet [11] consisting of input surge protection in the form of a varistor, and on the output two filtering capacitors and a transient voltage suppressor (TVS) diode for protecting downstream circuitry from an overvoltage condition.

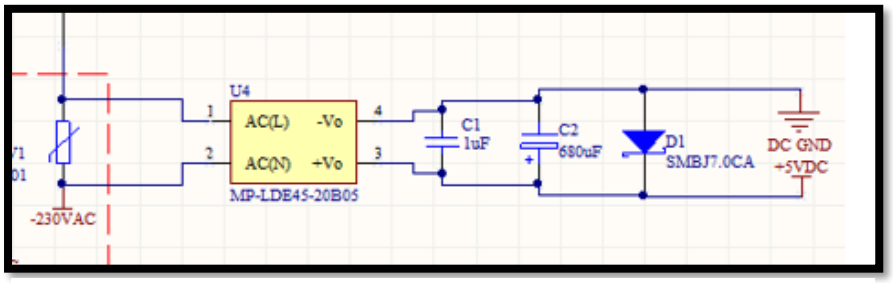


Figure 4.5 5V Power Supply Schematic

4.4.2 Relays

Two G5PZ-1A-DC5 [12] relays were used for switching of the heater and pump. Features include:

- 5VDC Coil Voltage
- Non-latching
- Max switching current 20A
- Switching voltage 250VAC

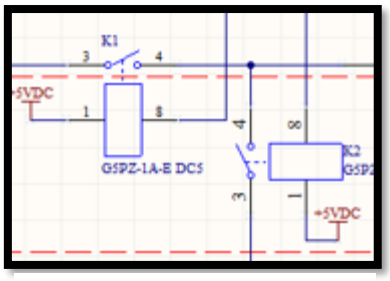


Figure 4.7 Relays Schematic

Figure 4.6 G5PZ-1A-DC5 Relays

4.4.3 Relay Driver

For controlling the two relays a relay driver was required. The NUD3124DMTIG [13] was selected for the following reasons:

- Provides Robust Interface between D.C. Relay Coils and Sensitive Logic
- Internal Zener diode eliminates noise from voltage spikes
- Simples circuit design, very small in size meaning a reduced PCB space required

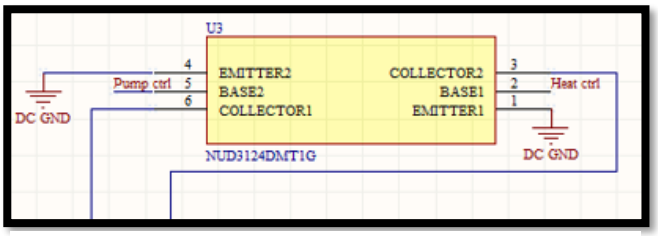


Figure 4.9 Relay Driver Schematic

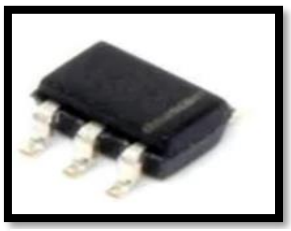


Figure 4.8 NUD3124DMTIG

4.4.4 TRIAC

The TRIAC is the main component of the proprietary PWM power controller for varying the power to the heating element. A large heatsink was fitted to dissipate generated heat, as shown the figure below. The BTA41-700BRG-TOP-3 [14] was selected for the following reasons:

- AC power switching device
- Max. off-state voltage: 700V
- Max Load Current: 40A
- Gate Current: 50mA

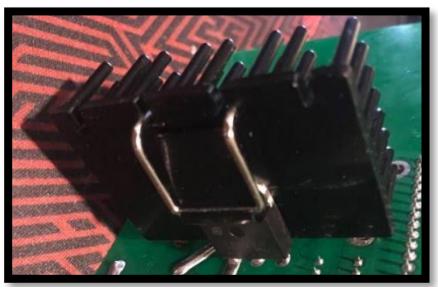


Figure 4.11 BTA41-700BRG-TOP-3 with Heatsink

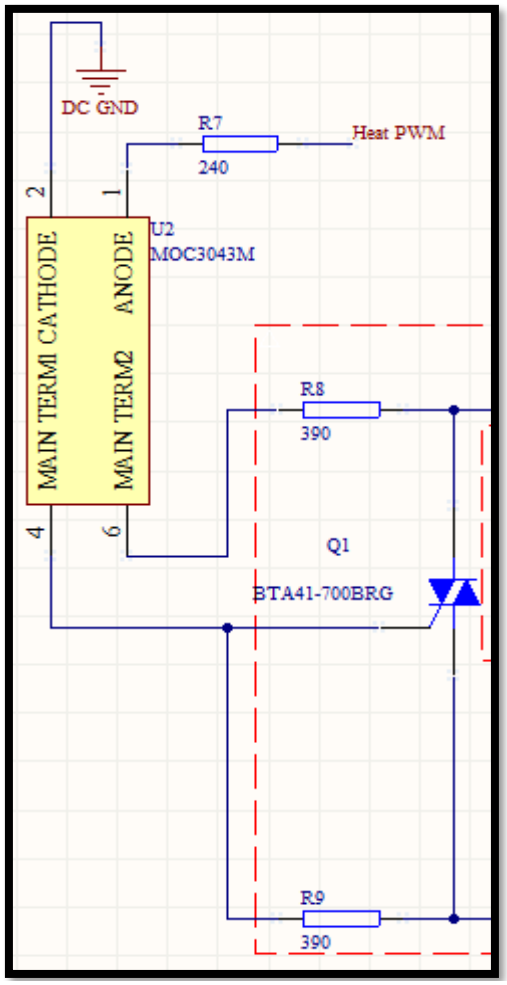


Figure 4.10 TRIAC + Optocoupler Schematic

4.4.5 Optocoupler

An optocoupler was required for electrically isolating the microcontroller from the mains voltage, and to provide the required current to drive the TRIAC. We selected the MOC3043M for the following features:

- Electrical isolation of the DC side from the AC
- Zero-crossing detection (ZCD) which was necessary for accurate operation of the variable power controller
- High isolation voltage of 7.5kV

4.4.6 Protection Circuitry

Brew Buddy's power system contains four protection components. Two cartridge fuses on the AC side, a 16A for the heater and pump, and a 3.15A for the AC-DC converter. A varistor is fitted on the input to the converter to protect it from voltage surges, and on the output is a transient voltage suppressor (TVS) diode to protect the 5V circuitry in case of overvoltage.

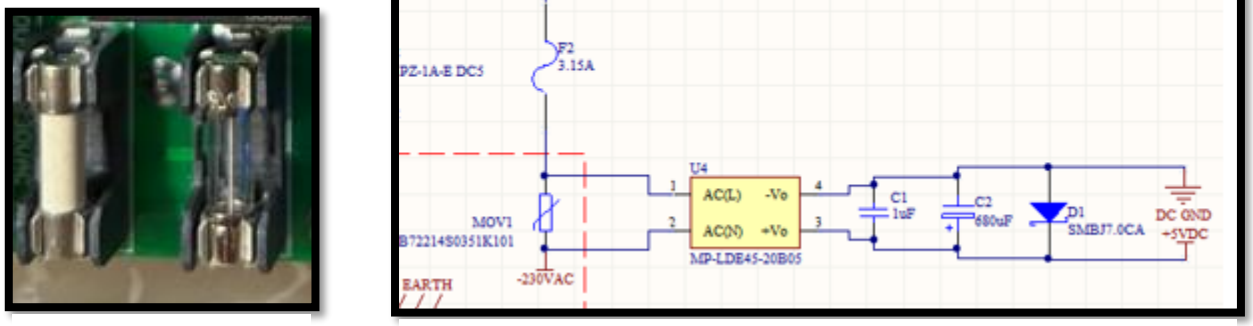


Figure 4.13 Fuses

Figure 4.12 Protection Devices Schematic

4.5 Microcontroller

The ESP32 microcontroller was selected for its powerful Xtensa® dual core processor and onboard Bluetooth/Wi-Fi [15], suiting it to the heavy requirements of running both the control algorithms and web server. We decided to go with a development board version with onboard programmer and 3.3v regulator for ease of integration. Features of the ESP32 development board include:

- 512kB RAM
- 4MB of flash memory
- Micro USB serial Interface
- WiFi 2.4 GHz ~ 2.5 GHz, 802.11 b/g/n
- Bluetooth v4.2 BR/EDR and BLE spec
- ADC 12-bit 18 Channel
- DAC 8-bit, 2 Channel
- 38 GPIO Pins
- PWM – 16 independent channels
- UART(3), I2C(2), SPI(4) + external interrupts
- 40 MHz crystal oscillator
- Cost-Effective with low power consumption
- 3.3V operating and input/output voltage
- 5V to 3.3V converter
- ESP-IDF (Espressif's IoT Development Framework)
- Strong function with support LWIP protocol and FreeRTOS
- Supporting three modes: AP (Access point), STA(Station), and AP+STA

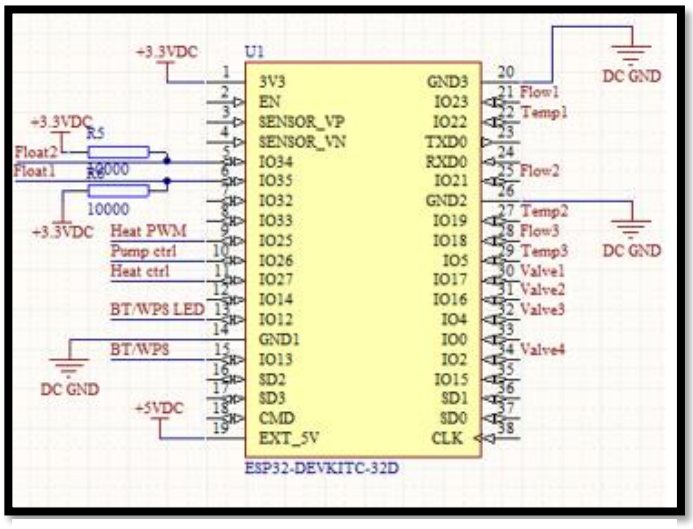


Figure 4.14 ESP32 Schematic

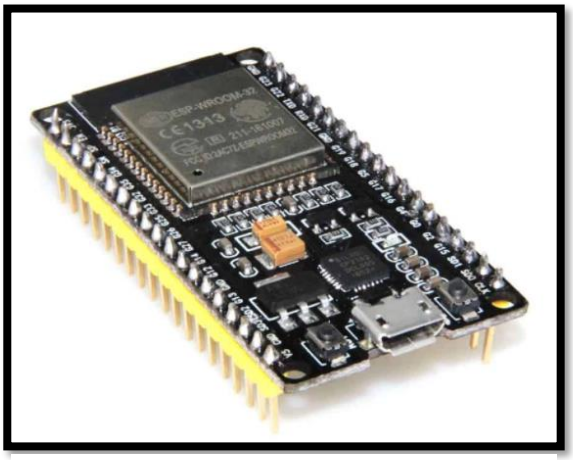


Figure 4.15 ESP32 Dev Board

4.6 Sensors

4.6.1 Temperature Sensor

Our system uses up to three temperature sensors depending on the configuration. These are at the tap input, sparge input, and sparge output, and are used for providing feedback to the heater PID control and temperature control algorithms. We selected the DS18B20 [16] for the following features:

- Input 3.0VDC to 5.5VDC
- Cheap, accurate and easy to use
- Can communicate over 1-wire bus communication (can use many sensors on same data bus). Each sensor has a unique 64-bit serial number.
- Operating range temperature: -55ºC to 125ºC
- Precision Control with an accuracy of +/-0.5 ºC between the range -10ºC to 85ºC
- 6mm watertight probe housing

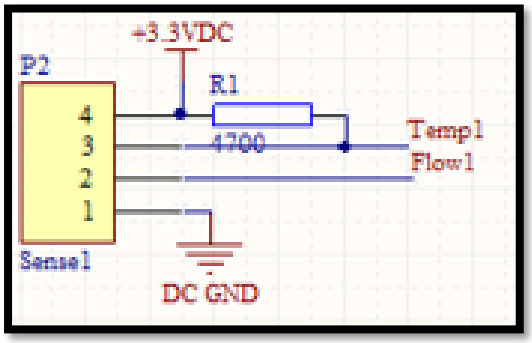


Figure 4.17 Sensor Assembly Schematic

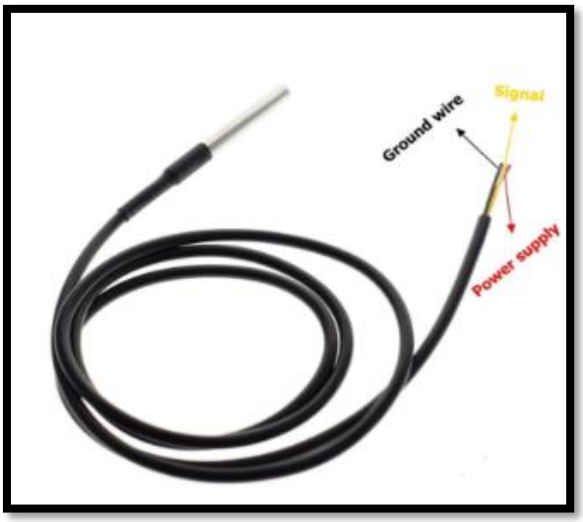


Figure 4.16 DS18B20 Temperature Probe

4.6.2 Flow Meter

Accompanying the temperature sensors are up to three flow meters in the same positions, combining to make a sensor assembly. This allows Brew Buddy to measure flow rate at different points in the system and adjust flow rate accordingly. The flow meters we selected are USS-HS21T1, these are a hall effect flow meter and were essentially the only type we could get at a suitable price. Due to these being from Ali Express the accuracy of them is unknown but we will monitor their suitability throughout testing. They have the following features:

- Temp range: -20 to 105ºC
- Flow Range: 1-30L/min
- Working voltage: DC3.3-24V
- Stainless steel body with a nylon turbine
- Pulse output duty cycle: 50+-10%
- ½ inch fittings

Figure 4.18 USS-HS21T1 Flow Meter

4.6.3 Float Switch

Stainless Steel reed float switches are used inside the mash tun for sensing an overflow due to a stuck sparge. There is provision for a second float switch at a low level that can be used for maintaining a minimum liquid level inside the mash tun.

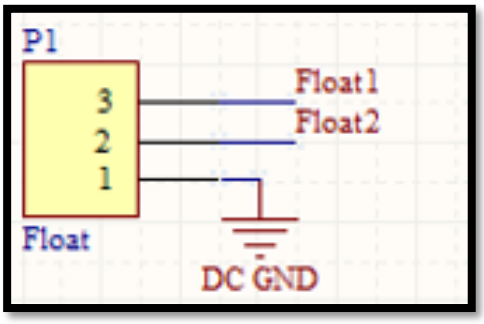


Figure 4.20 Float Switch Schematic

Figure 4.19 Float Switches

4.7 Servo Valves

As mentioned earlier in this report, commercially available off the shelve proportional control valves were difficult to find and were either very expensive or not fit for purpose. This prompted us to instead design our own servo valves, involving a lot of trial and error in wrong valve selection, servos not being strong enough, and multiple iterations of 3d printed housing design.

4.7.1 Valves

Brew Buddy has three ½" three-way stainless steel ball valves, positioned at tap input, sparge input, and sparge output. These are "L" configuration which allows for 0-100% flow in either direction with a 270° range of movement. Our original design contained a fourth needle valve specifically for accurate flow control but after testing this was deemed to be too restrictive to the flow, so we opted to adapt the three-way valves for flow control instead. An added benefit for this is that it opens up options in future development for having varying flow control rates at all three valve positions.

4.7.2 Servos

The servos used are 5V 35kg 270° coreless and are attached to the valves with a two-piece custom designed 3d printed housing. Some of the required specifications of the servos include:

- Voltage Range 4.8V-7.2V
- PWM Voltage 3.3V-5.0V, Pulse width range 500-2500us
- Advanced linearity and accuracy with precision potentiometers
- Waterproof
- Rated Current 1.4A

- Quiescent Current 100mAh
- Blocking Current 3.5A
- Running current: 140mA - 200mA Stall Torque 4.8V/29kg.cm

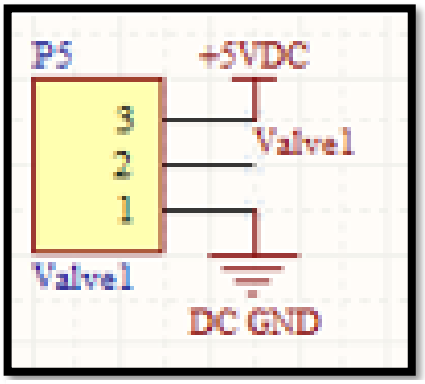


Figure 4.22 Servo Valve Schematic

Figure 4.21 Servo Valve

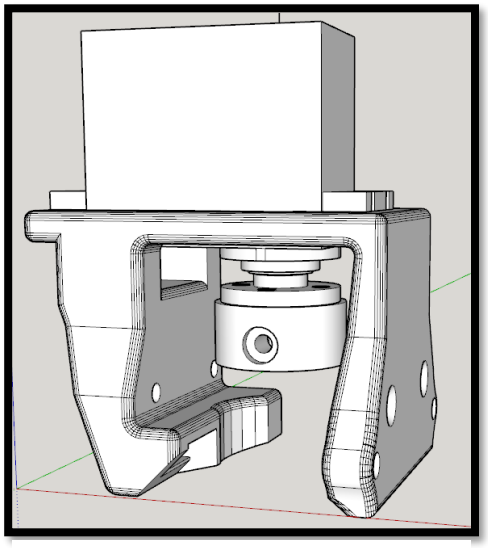


Figure 4.23 Servo Housing 3D Model

Chapter 5 PCB Design

5.1 Design Considerations

I was responsible for the design, layout, and manufacturing of Brew Buddy's PCB. A substantial amount of time was spent on the board layout in order to best fulfil design requirements. These requirements include:

- PCB must be reasonably small to fit inside a small enclosure
- The high-power sides trace width and separation must support 16A at 230V
- PCB must include adequate provision for a mounting system
- Convenient placement of components requiring easy access
- Clear separation of AC and DC sides
- Minimized noise interference to the ESP32 antenna

5.2 PCB Features

The PCB essentially consists of two electrically separate sides, the high-power(230VAC) side in the bottom left and the low-power(5VDC) side in the blue section with the ESP32 breakout. Some of the features of the board are:

- Two-layer board of 1oz copper.
- Hand soldered by me due to COVID-19 lockdown
- 110mm x 110mm
- M3 mounting holes for attaching to custom enclosure
- Large heatsink for the TRIAC on bottom layer
- High current AC path in as tight proximity as possible to decrease resistance
- High current AC traces are unmasked, and copious amounts of solder added to increase current carrying capacity
- Board cutouts used around the 230VAC terminals to increase creepage distance

- ESP32 oriented with antenna way from the AC-DC converter and TRIAC
- Easy access to fuses, peripheral connectors, and the ESP32
- Main DC trace width to support 8A
- Polygon pour connected to DC GND net on bottom layer

5.3 PCB Layout

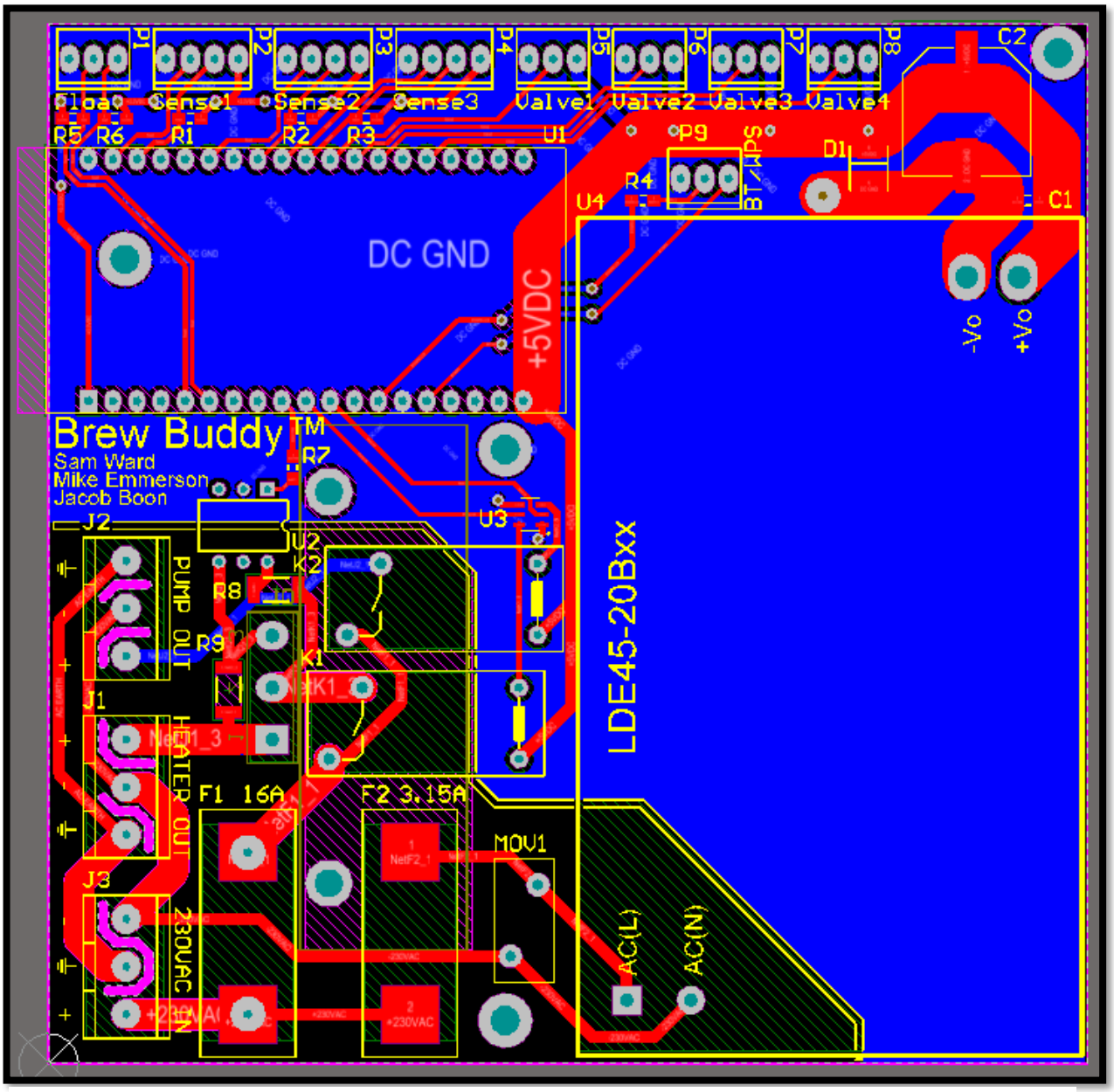


Figure 5.1 PCB Layout and Routing

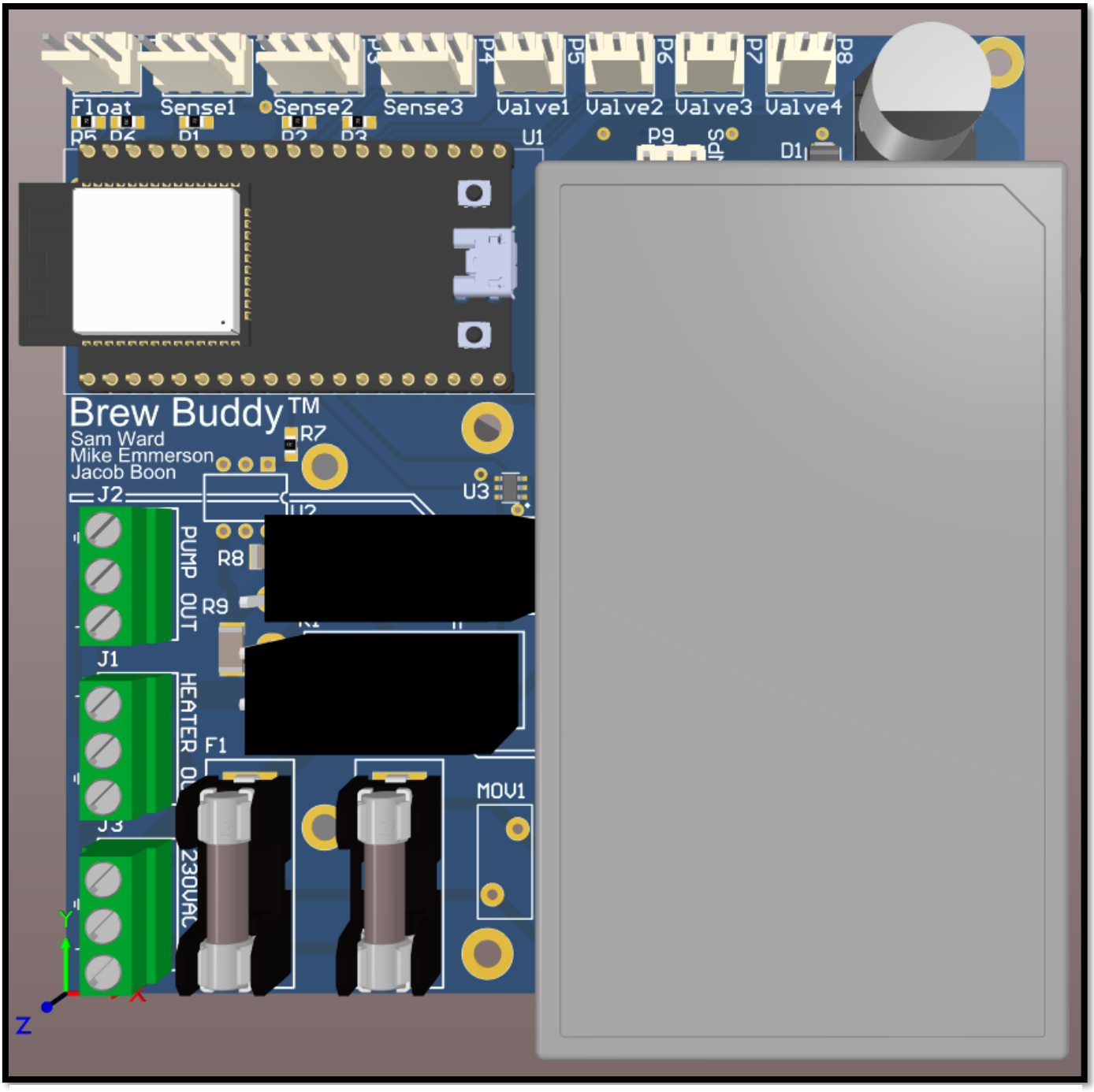


Figure 5.2 PCB 3D Component Layout

5.4 PCB Assembly

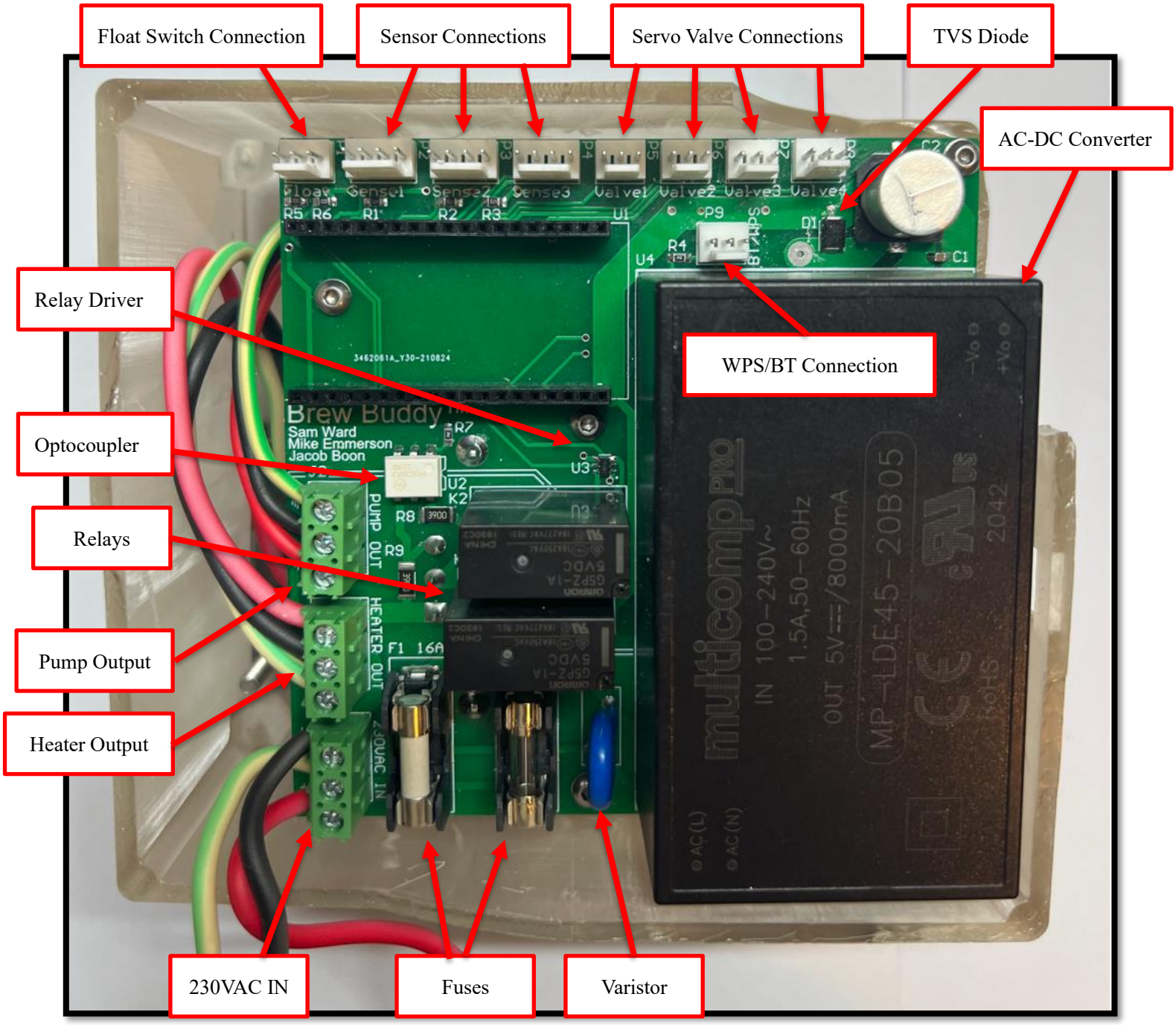


Figure 5.3 PCB Assembled Top View

Figure 5.5 PCB Assembled with ESP32

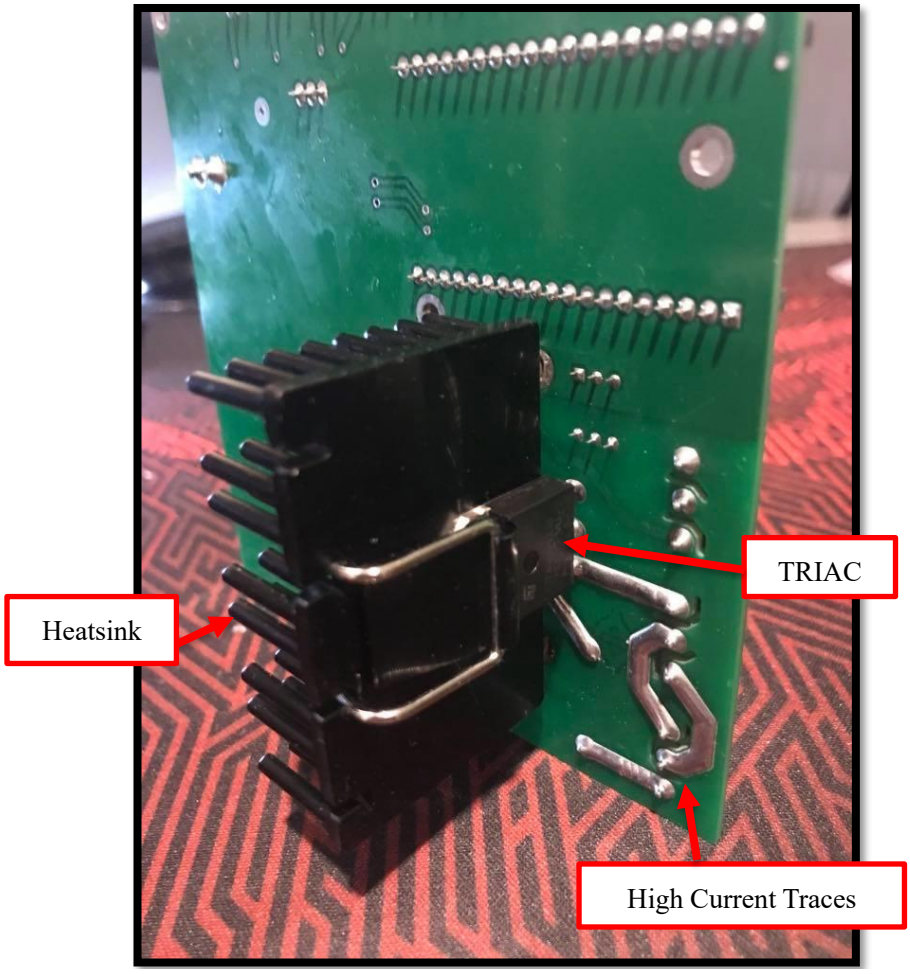


Figure 5.4 Assembled PCB Bottom View

5.5 PCB Enclosure

I designed a custom enclosure to house the PCB and internal wiring, and for mounting the various connectors. The enclosure is designed to be water and dust resistant and conforms to NZ class II electrical standards. A lot of time went into the layout of the enclosure as I wanted to keep it as small as possible, with the final result being 120x130x80mm. There is also room at the back to mount a 50mm DC fan if through further testing we find that the TRIAC is putting off too much heat. The enclosure was originally meant to be laser cut from 3mm acrylic but due to lockdown I as unable to access the laser cutter at either AUT or work, instead I ended up printing this at home in clear resin (not a very cost-effective way of manufacturing an enclosure) as we really wanted to be able to test our system.

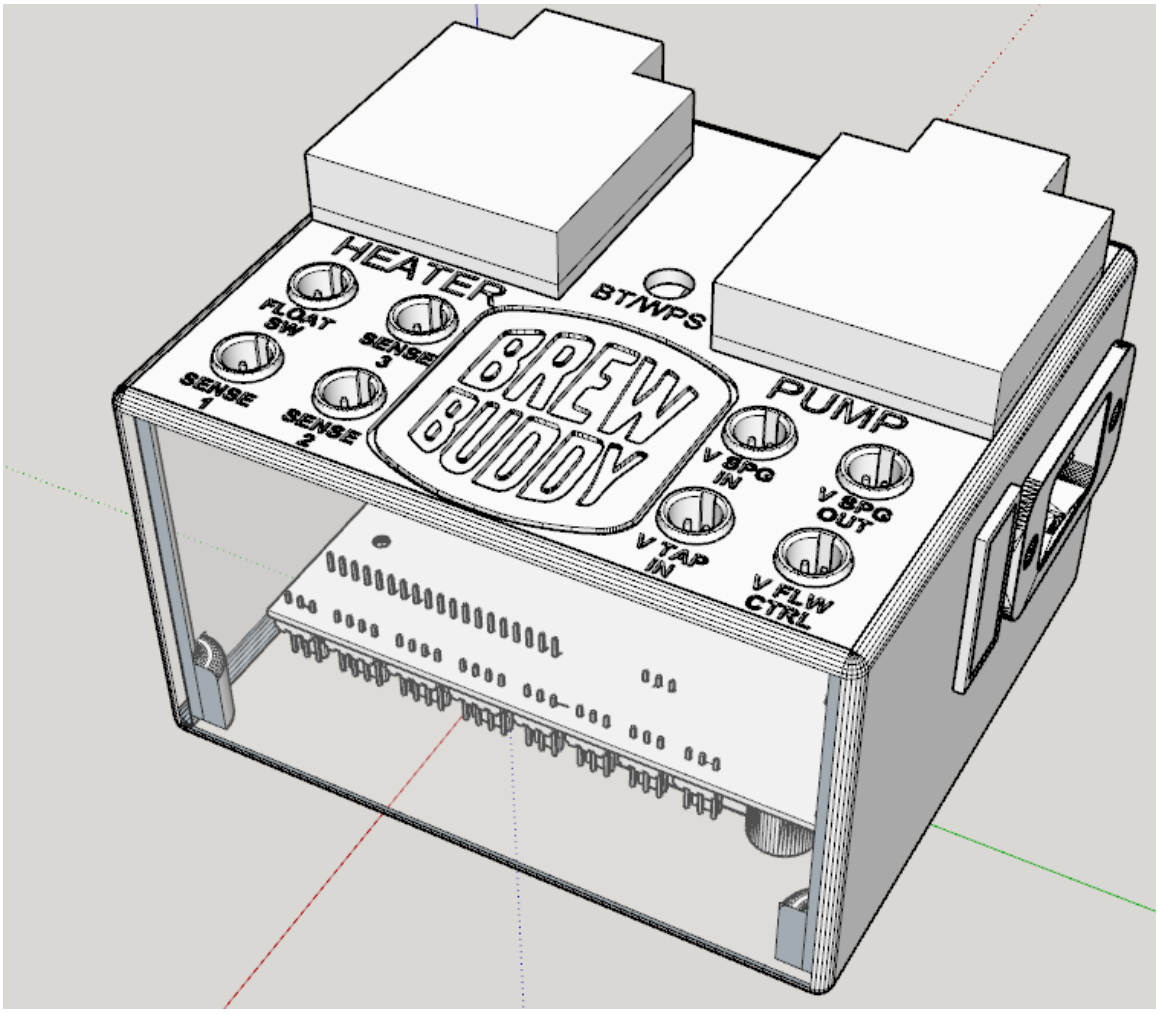


Figure 5.6 Brew Buddy Enclosure 3D Model

Figure 5.7 Brew Buddy Enclosure Printed

Chapter 6 Software

6.1 Microcontroller Architecture

Initial stages of the project involved brainstorming and developing various diagrams to cover the required functions and interactions of Brew Buddy's code.

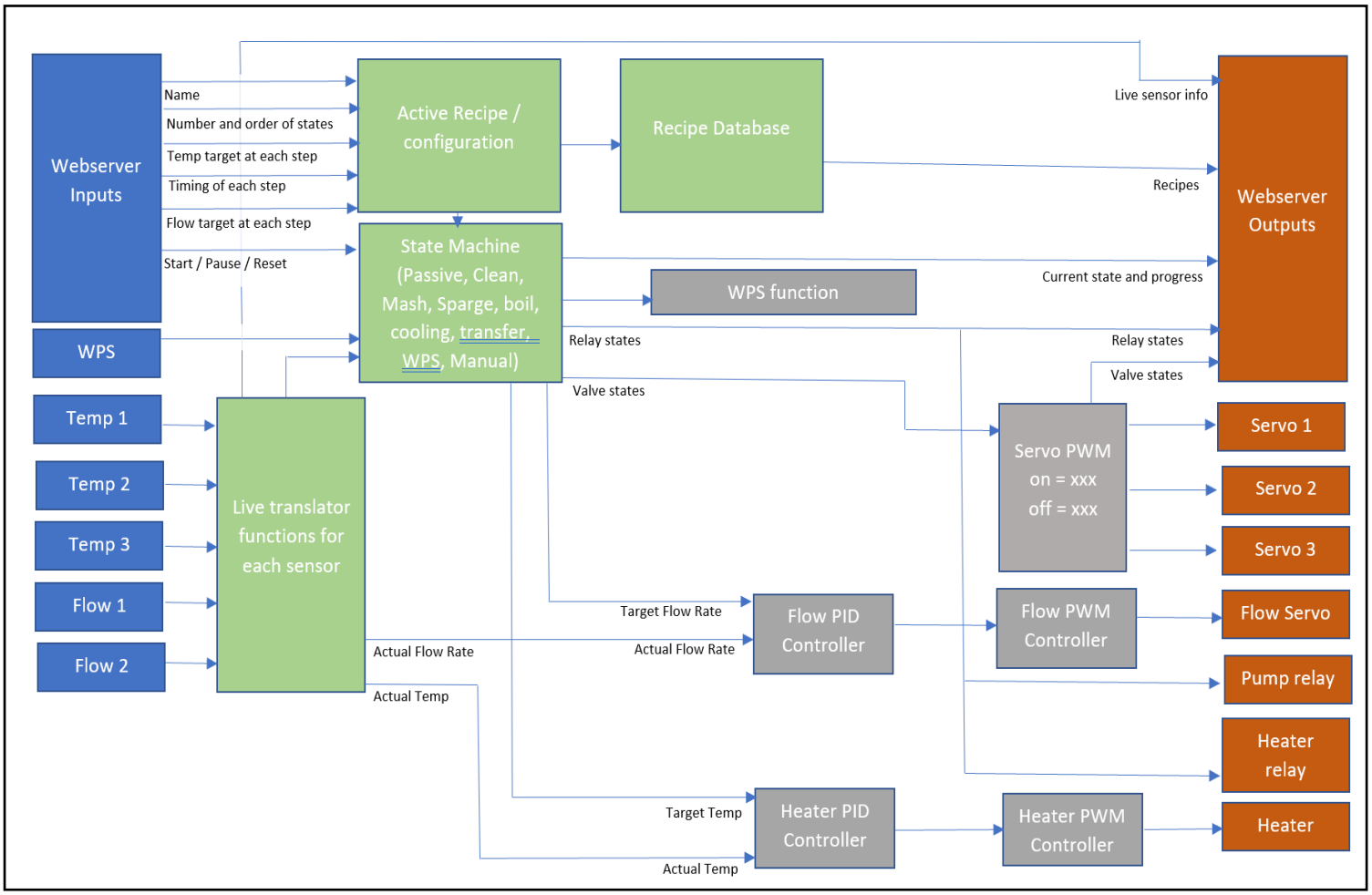


Figure 6.1 Microcontroller Block Diagram

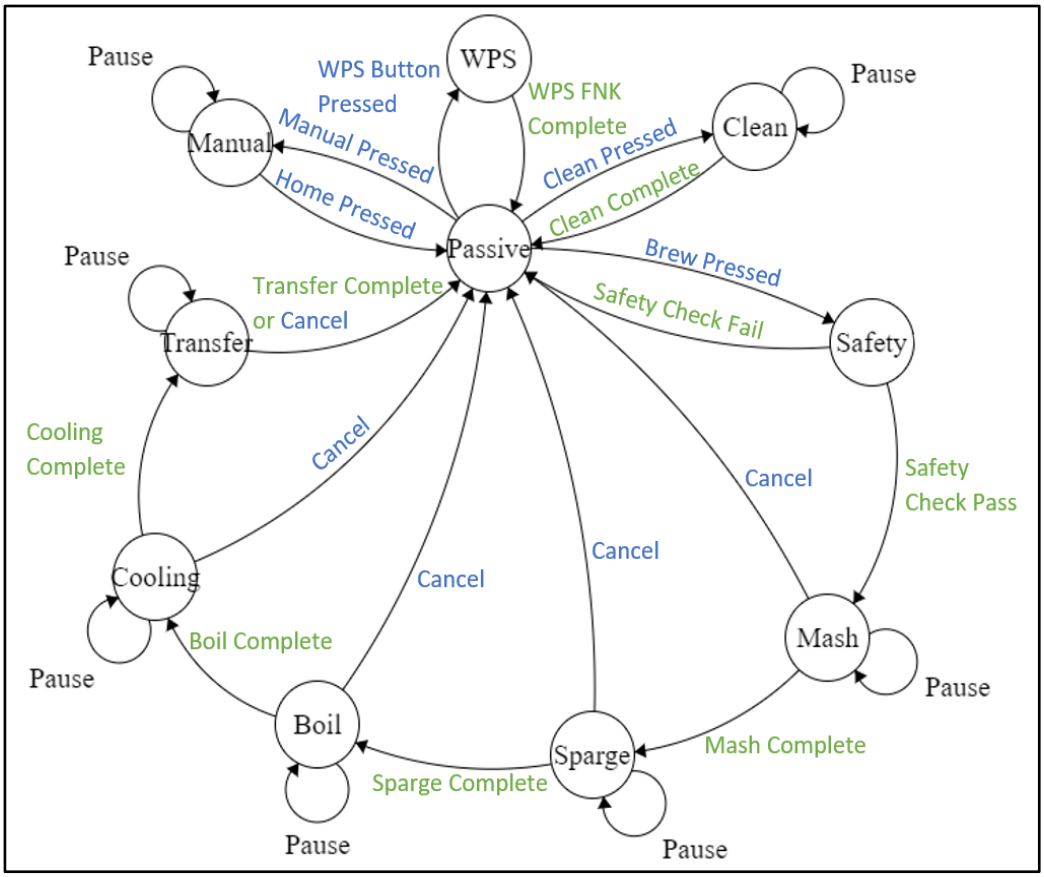
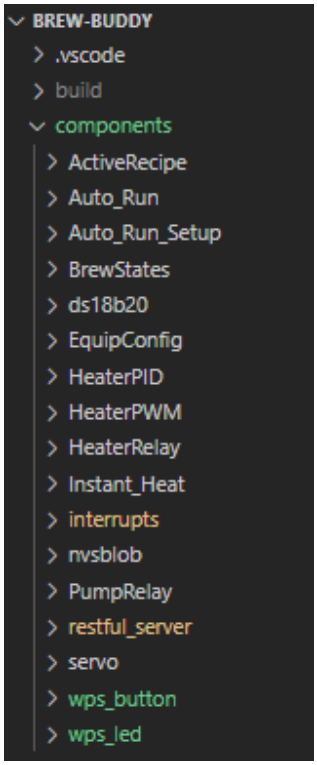


Figure 6.3 State Machine Diagram

6.2 Microcontroller Components



From this we were able to ascertain all of the various pieces of code, known as components, that needed to be developed in order for Brew Buddy to fully operate as intended. The benefit of breaking it into components is that we were able to more easily allocate and track who is working on what. My areas of responsibility were the components for WPS button and LED, servo, interrupts (flow meters), Non-volatile storage (NVS) binary large object (BLOB) for onboard memory management, and the RESTful server.

Figure 6.2 Components

6.3 Development Environment

As I had the most experience in software development, I was responsible for setting up the development environment. I chose Visual Studio Code (VS Code) as our IDE due to its familiarity, and its ability to handle both the microcontroller code and front end code within the same project.

We used the ESP-IDF extension in VS Code for development of the ESP32. ESP-IDF is Espressif's official IoT Development Framework for the ESP32, providing a self-sufficient SDK for any generic application development using programming languages such as C and C++ [17]. Even though the ESP32 and ESP-IDF were new to me, my experience with other software and microcontrollers enabled me to quickly get up to speed with development, and set up the initial project before bringing the other team members on board to develop their individual components, and the automation algorithms that tie them all together.

Each of us had our own ESP32 board at home which we could use for running and testing code, however this was a slow process and difficult to debug due to not having a JTAG adapter on the development board.

I also set up the Git repository and helped the others to integrate this into VS Code for version control.

The entire code base can be view on my GitHub repo.

https://github.com/s-ward/brew-buddy

6.4 Brew Buddy Web App

Brew buddy is intended to be fully controlled via the user's phone or computer over their home Wi-Fi network, with no physical controls on the system itself aside from the button to connect to the Wi-Fi network via WPS.

6.4.1 High-level Requirements

Requirements were established early in the project to outline the operation of the app; these are broken into functional and non-functional requirements as follows:

Functional

- Allow users to save recipes
- Allow users to set up and save brewing system configuration
- Push notifications at various stages of brew, requiring confirmation before moving to next stage
- Provide error checking and configuration warnings when inputting recipe
- Allow users to change settings (units, lead times)
- Allow users to view current sensor information and brew progress
- Full integration with system hardware
- Allow users to select and initiate one of the modes (Brew, Clean, Manual)
- Allow users to cancel/stop/modify/pause during a brew in progress

Non-Functional

- Retain system Wi-Fi connection through power cycle
- Menus are easy to navigate
- Displays are visually pleasing and information easy to interpret
- The app shall be secure with any user input passwords
- The app must maintain good connectivity within range of the house

6.4.2 App Prototype

Using all the documentation created so far, I then developed an interactive prototype for the web app in Adobe XD. This helped to fast-track design decisions among the team, allowing us to easily modify features and layout before moving on to building the app itself.

This can be found at the following link (may eventually expire)

https://xd.adobe.com/view/2af8af27-7462-415a-91a0-3939c79c64c7-7fab/?hints=off

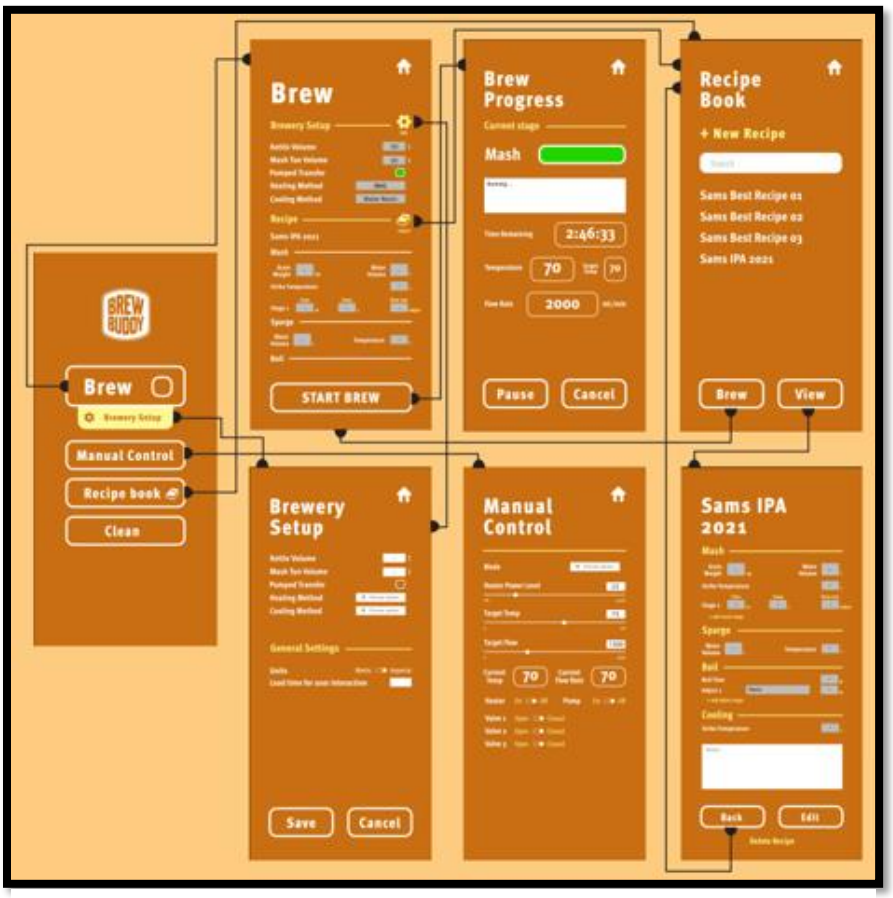


Figure 6.4 Brew Buddy app prototype

6.4.3 Frontend

The Brew Buddy web-app was developed in Vue.js using the Vuetify framework.

Vue was chosen as it is more lightweight than React and allows for easy development of cross platform native apps by porting the same component model through weex or capacitor.

Vue.js is a JavaScript framework for developing what are known as single-page applications, consisting of the main app and various views where essentially the different views are developed and the router switches between them. The store contains functions and data that is shared between views. For distribution Vue.js uses Node.js to build the distribution files which are then loaded on to the ESP32's memory where they can be accessed by the backend and delivered to the users screen. The frontend uses standard ajax queries and JSON data to communicate with the backend.

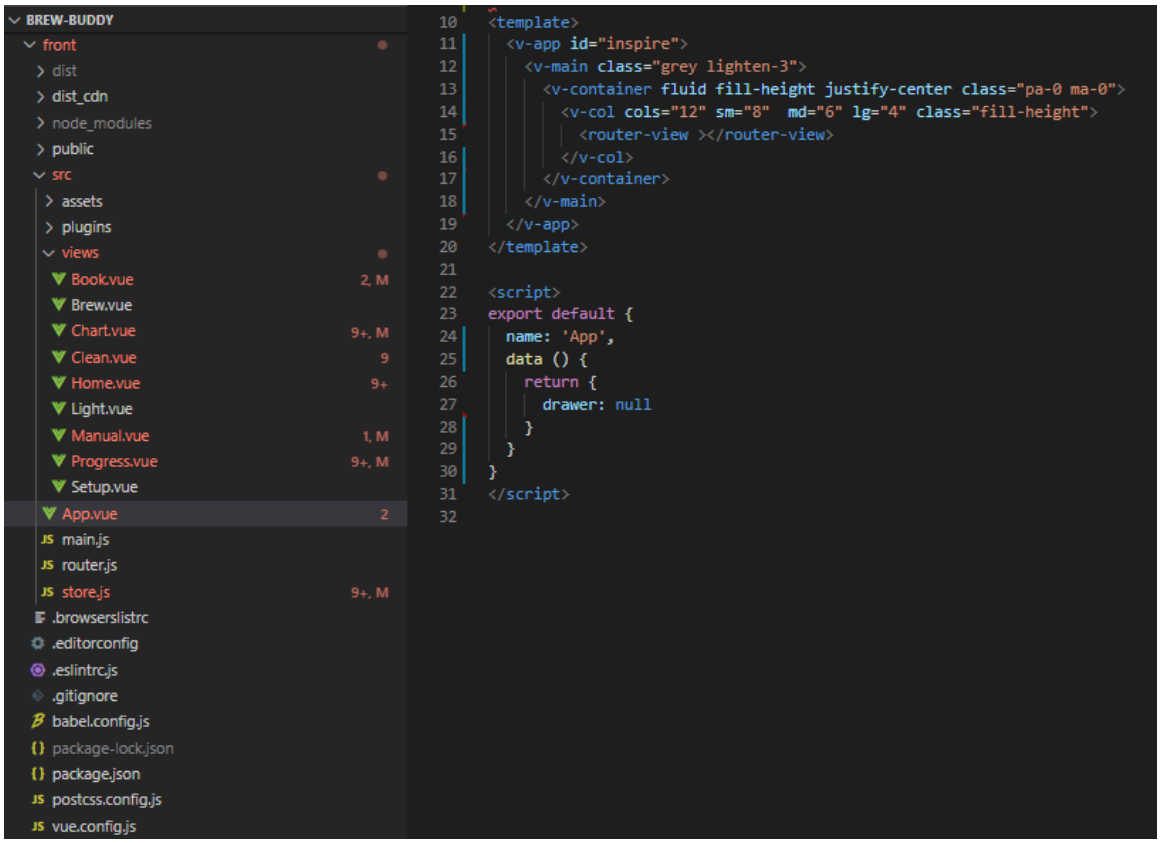


Figure 6.5 Frontend Views and Main App

Developing the Brew Buddy web app was an enormous task but a great learning experience, nonetheless. Vue.js was new to me and while I have done web development in the past it was nothing of this scale, with development of nine views ranging from 100 to 600 lines each. The biggest challenge would have been the concept of the store and dispatching functions or retrieving data from it, which took me a while to get the hang of.

One of the major differences between the prototype and then app is the change to recipe book, instead of having multiple pages for view/edit/new recipe, I went with a CRUD (create, read, update, delete) table on the recipe book page with a dialog popup for the recipe, resulting in a much sleeker approach.

```
methods: {
  set_manual: function () {
    this.$ajax
      .post('/api/v1/manual/set', {
        mode: this.mode,
        heaterpower: this.heaterpower,
        targettemp: this.targettemp,
        targetflow: this.targetflow,
        pump: (this.pump === 'On'),
        heater: (this.heater === 'On'),
        valve1: (this.valve1 === 'Internal'),
        valve2: (this.valve2 === 'Internal'),
        valve3: (this.valve3 === 'Internal')
      })
      .then(data => {
        console.log(data)
      })
      .catch(error => {
        console.log(error)
      })
  },
  gohome: function () {
    this.$store.dispatch('post_state_update', {
      brewint: this.$store.state.brewint,
      pauseint: this.$store.state.pauseint,
      cancelint: 1,
      cleanint: this.$store.state.cleanint,
      userintreq: this.$store.state.userintreq,
      adjunctreq: this.$store.state.adjunctreq

    })
      .then(() => (this.$router.push('/')))
  },
  minustargetflow () {
    this.targetflow--
  },
  addtargetflow () {
    this.targetflow++
  },
  minustargettemp () {
    this.targettemp--
  },
```

Figure 6.7 Example Ajax Query

```
set_brew_state (state, data) {
  state.pauseint = data.pauseint
  state.cancelint = data.cancelint
  state.cleanint = data.cleanint
  state.brewint = data.brewint
  state.brewstate = data.brewstate
  if (data.brewstate === 3) {
    state.status = 0
  } else if ([4, 5, 6, 7, 8, 9].includes(data.brewstate)) {
    state.status = 1
  }
},
set_brew_state_action ({ commit }, data) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      commit('set_brew_state', data)
      resolve()
    }, 1000)
  })
},

get_brew_state_action ({ commit }) {
  axios.get('/api/v1/getstate')
    .then(data => {
      setTimeout(() => {
        commit('set_brew_state', data.data)
      }, 100)
    })
    .catch(error => {
      console.log(error)
    })
},
add_message_action ({ commit }, data) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      commit('add_message', data)
      resolve()
    }, 100)
  })
}
```

Figure 6.7 Vue Store Functions

A hosted version of the front end only can be found at the following link.

http://bbfrontv7.avcl.co.nz/

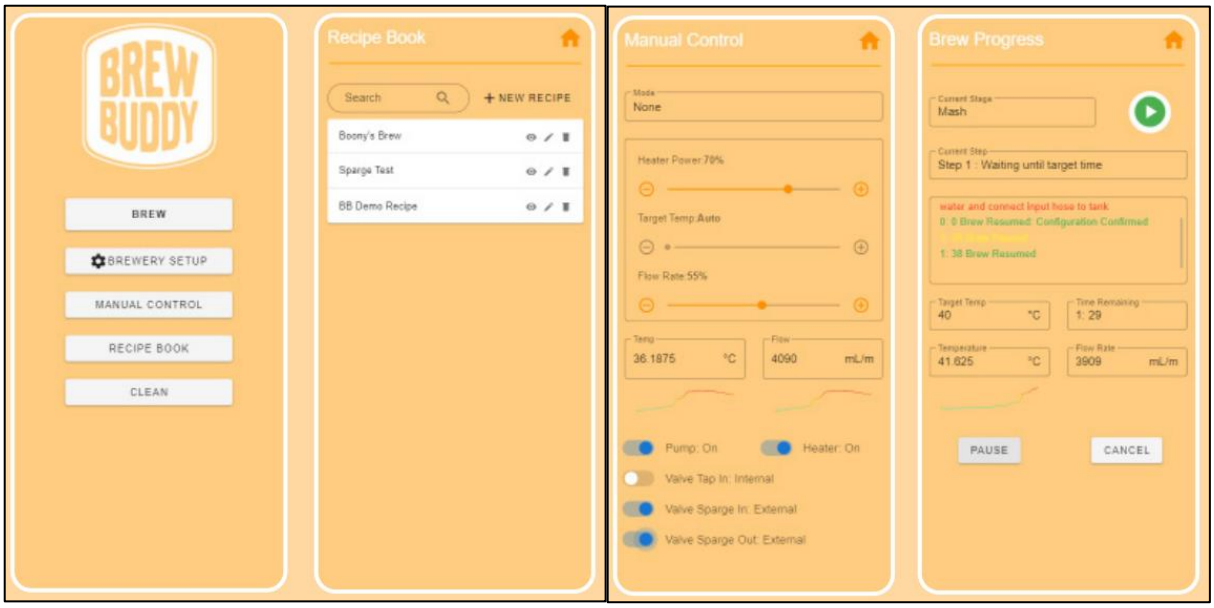


Figure 6.8 Brew Buddy App

6.4.4 Backend

The web server is hosted on the ESP32 and utilises the ESP-IDF REST server. It contains the various URI handlers which then call the appropriate control algorithms, using cJSON to pass data between the front and back end. One of the challenges I faced with the backend development was manipulation of cJSON data and converting to and from strings. Overall the back end development wasn't too bad as once you got the hang of coding a few handlers it was more or less rinse and repeat for adding more, with a total of 15 different get or post handlers.

```c
/* URI handler for fetching brewery setup */
httpd_uri_t setup_load_get_uri = {
    .uri = "/api/v1/setup/load",
    .method = HTTP_GET,
    .handler = setup_load_get_handler,
    .user_ctx = rest_context};
httpd_register_uri_handler(server, &setup_load_get_uri);
```

```c
/* Simple handler for getting brewery setup */
static esp_err_t setup_load_get_handler(httpd_req_t *req)
{
    httpd_resp_set_type(req, "application/json");
    cJSON *root = cJSON_CreateObject();

    cJSON_AddNumberToObject(root, "kettlevolume", brewery_setup.kettle_volume);
    cJSON_AddNumberToObject(root, "mashtunvolume", brewery_setup.mash_tun_volume);
    cJSON_AddBoolToObject(root, "pumpedtransfer", brewery_setup.pumped_transfer);
    cJSON_AddStringToObject(root, "units", brewery_setup.units);
    cJSON_AddNumberToObject(root, "leadtime", brewery_setup.lead_time);
    cJSON_AddStringToObject(root, "heatingmethod", brewery_setup.heating_method);
    cJSON_AddStringToObject(root, "coolingmethod", brewery_setup.cooling_method);

    const char *brewery_setup = cJSON_Print(root);

    httpd_resp_sendstr(req, brewery_setup);
    free((void *)brewery_setup);
    cJSON_Delete(root);
    return ESP_OK;
}
```

Figure 6.9 URI Get Handler for Loading Brewery Setup

```c
/* URI handler for saving setup */
httpd_uri_t setup_save_post_uri = {
    .uri = "/api/v1/setup/save",
    .method = HTTP_POST,
    .handler = setup_save_post_handler,
    .user_ctx = rest_context};
httpd_register_uri_handler(server, &setup_save_post_uri);
```

```c
/* handler for saving brewery setup*/
static esp_err_t setup_save_post_handler(httpd_req_t *req)
{
    int total_len = req->content_len;
    int cur_len = 0;
    char *buf = ((rest_server_context_t *)(req->user_ctx))->scratch;
    int received = 0;
    if (total_len >= SCRATCH_BUFSIZE)
    {
        /* Respond with 500 Internal Server Error */
        httpd_resp_send_err(req, HTTPD_500_INTERNAL_SERVER_ERROR, "content too long");
        return ESP_FAIL;
    }
    while (cur_len < total_len)
    {
        received = httpd_req_recv(req, buf + cur_len, total_len);
        if (received <= 0)
        {
            /* Respond with 500 Internal Server Error */
            httpd_resp_send_err(req, HTTPD_500_INTERNAL_SERVER_ERROR, "Failed to post control value");
            return ESP_FAIL;
        }
        cur_len += received;
    }
    buf[total_len] = '\0';

    cJSON *root = cJSON_Parse(buf);
    int kettlevolume = cJSON_GetObjectItem(root, "kettlevolume")->valueint;
    int mashtunvolume = cJSON_GetObjectItem(root, "mashtunvolume")->valueint;
    bool pumpedtransfer = cJSON_GetObjectItem(root, "pumpedtransfer")->valueint;
    char *units = cJSON_GetObjectItem(root, "units")->valuestring;
    int leadtime = cJSON_GetObjectItem(root, "leadtime")->valueint;
    char *heatingmethod = cJSON_GetObjectItem(root, "heatingmethod")->valuestring;
    char *coolingmethod = cJSON_GetObjectItem(root, "coolingmethod")->valuestring;

    save_brewery_setup(kettlevolume, mashtunvolume, pumpedtransfer, units,
                       leadtime, heatingmethod, coolingmethod);

    ESP_LOGI(REST_TAG, "Brewery Setup Saved");
    cJSON_Delete(root);
    httpd_resp_sendstr(req, "Post control value successfully");
    return ESP_OK;
}
```

Figure 6.10 URI Post Handler for Saving Brewery Setup

Chapter 7 Demonstration

Luckily toward the end of the Semester we managed to assemble a complete system using our own brewing equipment and undertake some testing and calibration.

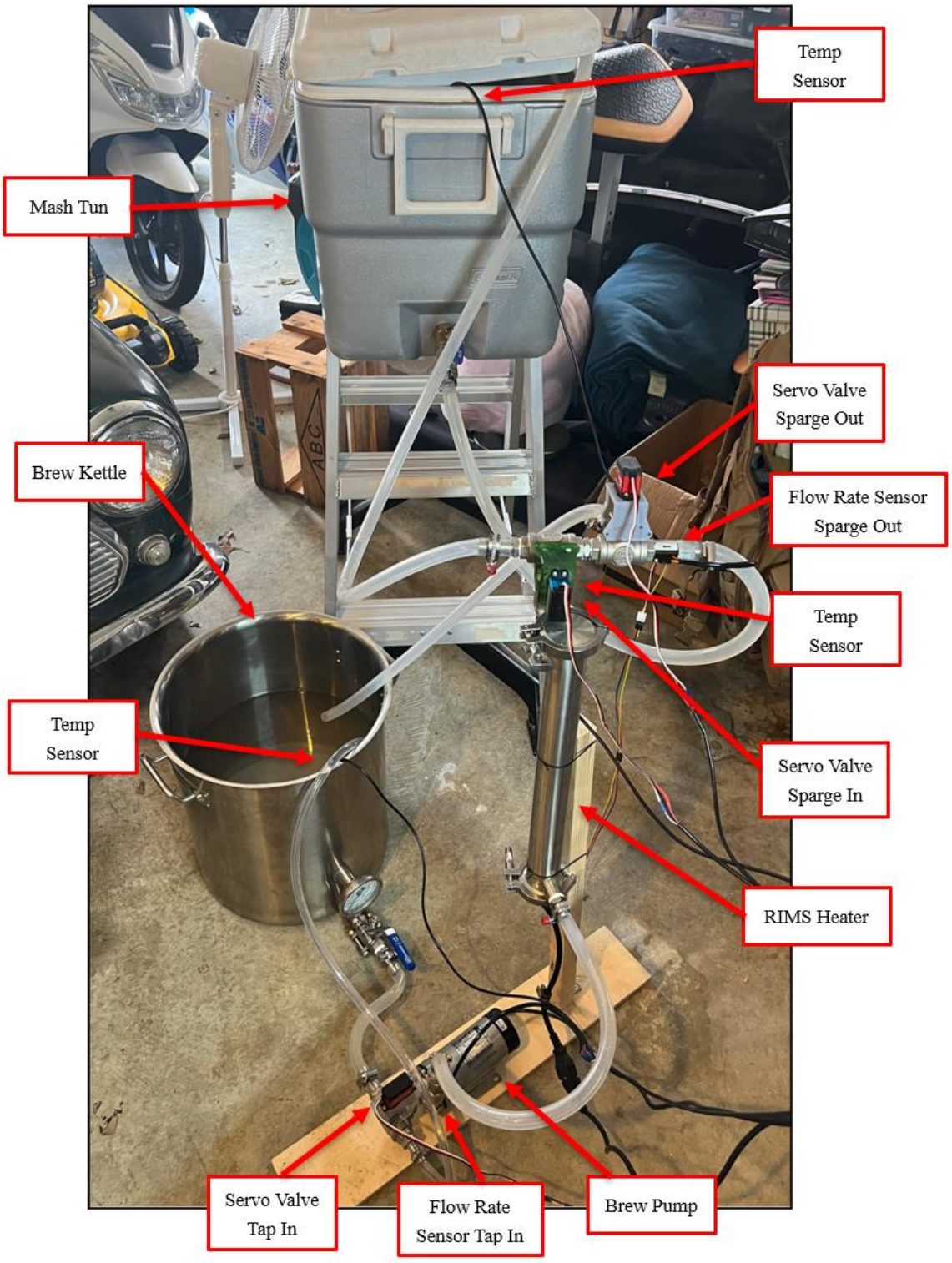


Figure 7.1 System Setup

After a series of bug fixes, a few servo hot swaps, and some rough PID calibration we were able to run through an entire simulated brew and film a demonstration video which can be viewed at the link below or by scanning the QR code.

https://www.youtube.com/watch?v=f-N0NDuo76k&list=FLas1kC1m3BJX_He1mauJ9qw

Figure 7.2 QR Demonstration Video

Chapter 8 Conclusion

8.1 Results

We started this project at the beginning of the year with the following objectives:

1. Simplify the brewing process, requiring the least amount of user interaction as possible.
2. Interface with the users Wi-Fi network and be controlled through either a mobile or web application.
3. Continuously update with live sensor information and progress tracking.
4. Allow the user to input their desired recipe information into the app which defines the timings, temperature, and flow rates that the system will operate within.
5. To be semi-modular and be able to interface with existing home brew setups.

Arguably with a lot more work required than initially anticipated, and with COVID-19 up our back hindering our progress at every opportunity, we remarkably managed to meet our objectives and finish the semester with a functioning product. Some of the key milestones throughout the project were:

- Market research and survey results evaluated, assisting with key design decisions.
- Research and selection of hardware components completed.
- Circuit design and PCB manufactured
- Microcontroller code base developed including specific component functions and automation algorithms
- Brew Buddy web app developed and tested, completed to a high standard
- Full operational system was assembled, and demonstration carried out

8.2 Future Work

8.2.1 Mobile App Development

Developing a native mobile app that the user can download from the app store rather than using the web browser would be a nice feature for further development. This coupled with the addition of push notifications would give the Brew Buddy user a true mobile experience.

8.2.2 Unfinished Work

There is still some outstanding work required before Brew Buddy can be considered a viable marketable product. This includes PID tuning of the four PID algorithms, completion of the enclosure and PAT testing for electrical certification, and finding a solution to mounting the temperature probes in the system.

8.2.3 Fermentation Control

This was one of the more requested features of automation from our survey. However, the system required for this would be completely different from a system that would automate the other aspects of making beer or distilling and therefore was deemed out of scope for our project. Future development ideas could be proprietary fermentation control design implemented into Brew Buddy, or integration of Brew Buddy with other market fermentation control systems available.

8.2.4 Distillation

Originally part of the initial project proposal it was discarded due to low amount of interest from the community. This would be a relatively simple addition to Brew Buddy as it utilises the same control systems, just requires another algorithm to be developed.

8.3 Challenges

8.3.1 Technical

- COVID-19 hindered our progress toward the end of the Semester by not allowing us to meet up or use the AUT laboratory, as such we were unable to get Brew Buddy electrically certified for the 230V.
- Had an issue with the ESP32 running out of flash memory. To remedy this I moved storage of the front-end files to an external web server acting as a content delivery network (CDN), unfortunately resulting in the app no longer being offline as originally proposed.

8.3.2 Non-technical

- Being the sole front and back-end developer presented its challenges. It would have been helpful to have another software developer with experience in the team to bounce ideas off, get software specific feedback, and to help with problems.
- Balancing the combined electronics and software project proved to be harder than expected. I found myself at times focusing too heavily on one aspect when another part of the project needed attention.

8.3.3 Lessons Learnt

- We didn’t establish thorough coding standards at the beginning of the project, this led to a bit of confusion when helping or reviewing other’s code due to different coding styles among the team.

- Steep learning curve for both ESP-IDF and Vue.js. More research into these at the project initiations stage would have been beneficial for establishing upskilling requirements.

8.4 Conclusion

This report contains the research, design, and construction involved in development of our autonomous home brewery system, aptly named Brew Buddy. We believe that our systems simple operation and ability to interface with existing equipment fills a sought-after niche in the home brewing market, allowing for a hands-off brew day through robust automation, at a cost-effective level.

The combination of electronics and software development aspects amalgamate seamlessly and leave us with a product we are proud of, albeit with a few short comings requiring work before we can consider this project truly completed.

I’m proud of the Brew Buddy team and what we have achieved, we all put an immense amount of work in and look forward to developing Brew Buddy further and enjoying the fruits of our labour.

References

[1] I. Cabras and D. Higgins, "Beer, brewing, and business history", *Business History*, vol. 58, no. 5, pp. 609-624, 2016. Available: 10.1080/00076791.2015.1122713 [Accessed 1 November 2021].

[2] Grainfather, "G30 Brewing System", *Shop.grainfather.com*, 2021. [Online]. Available: https://shop.grainfather.com/au/g30-brewing-system.html. [Accessed: 01- Nov- 2021].

[3] "Robobrew Brewzilla - 65 Litre Gen. 3.1", *Brewshop.co.nz*, 2021. [Online]. Available: https://www.brewshop.co.nz/brewzilla-65litre-gen3.html?gclid=EAIaIQobChMI3vGJlJ348wIVW5hmAh1HTwfuEAAYASAAEgIA8vD_BwE. [Accessed: 01- Nov- 2021].

[4] "PicoBrew Z series modular all-in-one", *Engadget.com*, 2021. [Online]. Available: https://www.engadget.com/2018-02-12-picobrew-z-series-homebrew-beer.html. [Accessed: 01- Nov- 2021].

[5] F. Lockwood, "RIMS or HERMS: Understanding Mashing Equipment", *Homebrew Talk - Beer, Wine, Mead, & Cider Brewing Discussion Forum*, 2017. [Online]. Available: https://www.homebrewtalk.com/threads/rims-or-herms-understanding-mashing-equipment.679057/. [Accessed: 28- Oct- 2021].

[6] B. Smith, "RIMS and HERMS – Recirculating Infusion Mash Systems for Beer", *Beersmith.com*, 2011. [Online]. Available: http://beersmith.com/blog/2011/08/11/rims-and-herms-recirculating-infusion-mash-systems-for-beer/. [Accessed: 30- Oct- 2021].

[7] "Electricity Regulations 1997 (SR 1997/60) (as at 03 September 2007) 53 Voltage – New Zealand Legislation", *Legislation.govt.nz*, 2021. [Online]. Available: https://www.legislation.govt.nz/regulation/public/1997/0060/1.0/DLM229459.html. [Accessed: 01- Nov- 2021].

[8] Grainfather, "G70 Brewing System", *Grainfather.com*, 2021. [Online]. Available: https://grainfather.com/wp-content/uploads/2020/03/Grainfather-G70-specifications_ONLINE.pdf. [Accessed: 01- Nov- 2021].

[9] "The Physics Classroom Tutorial", *Physicsclassroom.com*, 2021. [Online]. Available: https://www.physicsclassroom.com/class/thermalP/Lesson-1/Rates-of-Heat-Transfer. [Accessed: 01- Nov- 2021].

[10] "New Zealand Electrical Code of Practice for homeowner/occupier's electrical wiring work in domestic installations (NZECP 51:2004)", 2004. Retrieved 01-Nov-2021 from:

https://www.worksafe.govt.nz/laws-and-regulations/standards/electricity-standards-and-codes-of-practice/

[11] Multicomp Pro. (2020, September). *45W AC to DC Converter - PCB Mount Datasheet* (V1.0). https://www.farnell.com/datasheets/3155177.pdf

[12] Omron Electronics, 2021. *G5PZ PCB Power Relay Datasheet.* [Online]. Available: https://omronfs.omron.com/en_US/ecb/products/pdf/en-g5pz.pdf. [Accessed: 01- Nov- 2021].

[13] On Semiconductor. "NUD3124 - Automotive Inductive Load Driver Datasheet", *Nz.mouser.com*, 2021. [Online]. Available: https://nz.mouser.com/datasheet/2/308/1/NUD3124_D-2319507.pdf. [Accessed: 01- Nov- 2021].

[14] STMicroelectronics. "BTA40, BTA41, and BTB41 Series 40A TRIACs" Datasheet, *Farnell.com*, 2021. [Online]. Available: https://www.farnell.com/datasheets/80810.pdf. [Accessed: 01- Nov- 2021].

[15] *ESP32 Series Datasheet*. (2021). https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[16] Maxim Integrated Products, Inc. (2019). *Programmable Resolution 1-Wire Digital Thermometer - DS1820B Datasheet* (No. 19–7487; Rev 6; 7/19). https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[17] *IoT Development Framework I Espressif Systems*. (2021). Espressif.com. https://www.espressif.com/en/products/sdks/esp-idf