EE175AB Final Report

B.R.A.D (Bread Recipe Autonomous Device)

EE 175AB Final Report

Department of Electrical Engineering, UC Riverside

Project Team Member(s)	Luis Herrera Nevin Kopp Felix Maass
Date Submitted	19/03/2023
Section Professor	Professor Roman Chomko Teaching Assistant: Christopher Eng
Revision	Version 2.0.23
URL of Project YouTube Videos, Wiki/Webpage	https://youtu.be/w2QDYGbPJTE https://github.com/Luckychief/Autonomous_Bread_Maker.git
Permanent Emails of all team members	Luis Herrera: lherr037@ucr.edu Nevin Kopp: nkopp002@ue.edu Felix Maass: fmaas001@ucr.edu

Table of Contents

Table of Contents **2**

1 * Executive Summary **5**

2 * Introduction **6**

2.1 * Design Objectives and System Overview 6
2.2 * Backgrounds and Prior Art 6
2.3 * Development Environment and Tools 7
2.4 * Related Documents and Supporting Materials 7
2.5 * Definitions and Acronyms 7

3 * Design Considerations **8**

3.1 * Realistic Constraints 8
3.2 System Environment and External Interfaces 8
3.3 * Industry Standards 8
3.4 * Knowledge and Skills 9
3.5 * Budget and Cost Analysis 9
3.6 * Safety 9
3.7 Performance, Security, Quality, Reliability, Aesthetics etc. 10
3.8 * Documentation 10
3.9 Risks and Volatile Areas 10

4 * Experiment Design and Feasibility Study **11**

4.1 * Experiment Design 11
4.2 * Experiment Results, Data Analysis and Feasibility 11

5 * Architecture and High Level Design **12**

5.1 * System Architecture and Design 12
5.2 * Hardware Architecture 12
5.3 * Software Architecture (only required if your design includes software) 13
5.4 * Rationale and Alternatives 13

6 Data Structures (include if used) **14**

6.1 Internal software data structure 14
6.2 Global data structure 14
6.3 Temporary data structure 14
6.4 Database descriptions 14

7 * Low Level Design **15**

7.1 **Auger Delivery* 15
7.1.1 Processing narrative for Auger Delivery *16*
7.1.2 Auger Deliver interface description *16*
7.1.3 Auger Delivery processing *16*
7.2.1 Processing narrative for Fluid Delivery *17*
7.2.2 Fluid Delivery interface description *17*
7.2.3 Fluid Delivery processing *17*
7.3.1 Processing narrative for Lid Control *17*
7.3.2 Lid Control interface description *17*
7.3.3 Lid Control processing *19*
7.4.1 Processing narrative for Bread Maker Interface *19*
7.4.2 Bread Maker Interface interface description *19*
7.4.3 Bread Maker Interface processing *19*
7.5.1 Processing narrative for Rumble Motor *19*
7.5.2 Rumble Motor interface description *20*
7.5.3 Rumble Motor processing *20*

8 * Technical Problem Solving **21**

8.1 * The Light Ingredient Problem 21
8.2 * Solving the Light Ingredient Problem 21
8.3 * The Coarse ingredient problem 21
8.4 * Solving the coarse Ingredient Problem 21
8.5 * The Liquid Ingredient Problem 21
8.6 * Solving the Liquid Ingredient Problem 21

9 User Interface Design **22**

9.1 Application Control 22
9.2 User Interface Screens 22

10 * Test Plan **23**

10.1 * Test Design 23
10.2 * Bug Tracking 24
10.3 * Quality Control 25
10.4 * Identification of critical components 25
10.5 * Items Not Tested by the Experiments 25

11 * Test Report **26**

11.1 * Test 1 26
11.2 * Test 2 26

12 * Conclusion and Future Work 27

12.1 * Conclusion 27
12.2 Future Work 27
12.3 * Acknowledgement 28

13 * References 29

14 * Appendices 30
14.1 * appendice A: Parts List 31
14.2 * appendice B: Equipment list: 32
14.3 * appendice C: Software List 33
14.4 * appendice D: User Manual 34

1 * Executive Summary

The Autonomous Bread Maker, also known as the Bread Recipe Autonomous Device (shortened here as BRAD), is a senior design project aimed at simplifying the bread making process and make it more accessible to the public by automating the dispensing, mixing, kneading, and baking stages. Our team's motivation behind this project was to provide a more convenient and accessible way for consumers to make and enjoy fresh, healthy bread at home.

Our team's overall goals of this project were mostly direct and to the point. Our team wanted to create a bread maker that is super easy to use, efficient, and produces high quality bread for consumers. Our team's design objectives were to incorporate several subsystems around a pre-existing bread machine, as to utilize bread makers that were already on the market. These subsystems include the solid dispensing subsystem, the liquid dispensing subsystem, the bread machine interface, and the logic circuitry. The key features of BRAD include a remote connection to the device through a virtual private network that can be accessible through the wider World Wide Web, a bread pan with a durable non-stick coating for easy cleaning, and the ability to add new recipes and ingredients once put to scale.

To test the effectiveness and stability of BRAD, a series of testing experiments were conducted with the specific subsystems as well as the wider system. These experiments and tests demonstrated that BRAD was capable of producing different types of breads at differing sizes and at different temperatures. Our experimental results showed that BRAD was successful in producing high-quality bread, with consistent flavor and texture across all tested breads.

Over the course of this senior design project, our team believes that BRAD is a successful endeavor and product that was developed and created with several electrical engineering skills that were learned and nurtured throughout our team's university stay. Our team's key achievements across this project were creating a functional prototype that interfaced directly with a bread maker's electrical logic board, designing a backend hosting site for the remote connection, as well as testing the entire system to produce high-quality bread. BRAD is an important and useful venture as it has the potential to provide individuals with a convenient way to make fresh, healthy bread at home without breaking the bank. BRAD made a great senior design project as it incorporated a wide variety of technical skills, such as microcontroller programming, mechanical design considerations, electrical integration, and electrical design considerations.

2 * Introduction

2.1 * Design Objectives and System Overview

B.R.A.D. is a device which enables the user to automatically and remotely make bread with the push of a button from any WiFi capable device. The device is a machine designed to automatically measure and dispense ingredients into a bread maker, for simple production of bread at home. The intended application of our device is to make baking bread more accessible to the public. This project encompassed different areas of electrical engineering, primarily embedded systems.

The machine is built around a finite state machine design, managing different subsystems. These subsystems include the solid dispensing subsystem, the liquid dispensing subsystem, the bread machine interface, and the logic circuitry.

Technical design objectives included 95% accuracy at dispensing ingredients[LH], 95% accuracy at dispensing fluids[FM], and response time of the webpage < 2 seconds[NK].

Responsibilities: clearly state which team member is responsible for which goals/objectives

2.2 * Backgrounds and Prior Art

Through an extensive literature and internet search, our team wasn't able to find distinct products that were similar to BRAD. The best our team could find was the Zojirushi Bread Maker Line, the Panasonic Bread Maker Line, and the Cusinart Bread Maker Line. As far as our team could tell as well as reference from the documentation provided by each company, each machine is unable to directly dispense all the different types of ingredients into the hopper.

These machines have the addition of a mix-in box which allows consumers to add toppings to the bread machine to top their bread as it finishes baking. None of these products have a feature which allows for all the ingredients to be dispensed into the hopper as well as a remote triggering feature. Several of these products have a fake "remote" triggering feature which is simply just a timer that counts down until start. BRAD has several advantages when compared to the leading retail products such that it can dispense different ingredients into the hopper, be remotely triggered from any mobile device with an internet connection, as well as have the potential to set up any recipe the consumer may want when the product is scaled out.

Additionally, BRAD is simply cheaper to produce than to purchase an "automatic bread maker" that really isn't automatic. These devices will still require you to place ingredients into the bread machine and press the start button. The only shortcoming that BRAD has compared to these "automatic bread machines" is that it is not portable nor is it a small device. BRAD measures to be roughly 900 mm tall, 400 mm deep, and 400 mm wide.

2.3 * Development Environment and Tools

The design environment surrounding the Bread Recipe Autonomous Device, or BRAD for short, required the utilization of several hardware components including Nema 17 stepper motors, an Arduino Mega 2560, solenoid valves, and the actual bread machine chamber. Our team utilized several software tools that aided us in our endeavor such as the Arduino IDE, AccelStepper.h which helped support the Nema 17 stepper motors, as well as Wifi.h to support the ESP32 module for wi-fi connections.

When testing BRAD, our team had to utilize several hardware tools to measure and analyze data from our prototype and the pre-existing bread machine. Our team utilized a standard multimeter to do continuity tests, as well as to check the varying levels of voltage, current, and resistance of electrical components. An oscilloscope was utilized to check what type of signals the logic board of the pre-existing bread maker accepted and sent so our team could reliably replicate it back to the logic board.

2.4 * Related Documents and Supporting Materials

IEEE 802.11(LAN)

2.5 * Definitions and Acronyms

B.R.A.D = Bread Recipe Autonomous Device

3 * Design Considerations

This section describes issues that need to be addressed or resolved prior to or while completing the design as well as issues that may influence the design process.

3.1 * Realistic Constraints

While undergoing this project, we had to adhere to several constraints during the development process. Since we wanted to interface with an existing commercial bread machine, we had to adhere to the voltages and power constraints that were part of the existing bread machine. This meant we needed ways to adjust the voltages sent to the machine so that we didn't damage the circuitry while still being able to send enough voltage to control the machine. There were constraints with the stepper motor as well, in our current setup we could only power one stepper motor at a time as there isn't enough power to have all the motors actuate together. Stepper motor speed was also another constraint. If the motors were too fast or too slow, then there would be inaccuracies in the dispensed ingredient amounts. Since we used a stepper motor to also open and close the lid, if that were too fast it could damage the bread machine.

In terms of the bread itself, we had several constraints with the ingredients and what bread we could actually make. Each bread type required different ingredients and our project used one container per ingredient. This means that if we wanted to add more ingredients, we would need to add a whole new container, auger and stepper motor, along with all the circuitry to control it. We were also limited to how big the bread loafs could be since the bread machine had a fairly small internal chamber. Even the largest setting loaf might not be enough to feed a full family for very long. Since the machine would be operating in residential homes, we had to make sure it wasn't too loud so we decreased the amount of steps the stepper motor did in order to make it quieter.

We wanted the machine to be internet capable as well, which introduced even more constraints. Our project relies on an internet connection to function, otherwise the automated features would not work. There are also concerns of security, we needed to use an internet protected SSID otherwise someone could have full control over the machine. Since we wanted the user to interact with the project, we needed an easy to use and simple interface. This was achieved with big, colorful buttons with simple labeling and even provided feedback when a button was selected. Additionally, we wanted the user to be able to control the bread machine with any device at their convenience. This meant if we wrote a specific app, say an android app, then it would have only been compatible with android devices. Using a web app instead means that the user only needs a web browser which increases compatibility.

3.2 System Environment and External Interfaces

N/A

3.3 * Industry Standards

One of the industry standards we used was IEEE 802.11 for internet access and control of the bread machine. One of the de facto standards we used was the Arduino IDE.

3.4 * Knowledge and Skills

Felix Maass:

Knowledge and skills required: Electronic Circuits, Fluid Dynamics, Embedded Systems

Engineering Technical Courses: EE100A&B, PHYS040B, EE128

New Skills: 3D Design and Modeling

Nevin Kopp:

Knowledge and skills required: Electronic Circuits, Embedded Systems, Circuit Soldering

Engineering Technical Courses: EE100A&B, EE128

New Skills: HTML, Internet of Things, User Experience Design (UX), Baking

Luis Herrera:

Knowledge and skills required: Electronic Circuits, Embedded Systems, Circuit Soldering, State Machine Design

Engineering Technical Courses: EE100A&B, EE128, CS120B

New Skills: 3D Design and Modeling, HTML

3.5 * Budget and Cost Analysis

The initial goal was to keep the budget underneath $250

-Our total was $550

The high cost came as a result of parts breaking or not working during the building process. Which would have brought the total cost down to closer to $400.

The other major cost came from the Aluminum Extrusion Frames, which would have brought the cost down to $270, if he had gone with another alternative solution. But because of the value it provided for the quick prototyping of parts, we decided to invest the money on them.

3.6 * Safety

The main safety consideration for this project is the issue of food safety. We addressed this by firstly, using safer ingredients that didn't need refrigeration such as milk or eggs. The ingredients we ended up using were all dry storage safe and didn't need to be kept in an airtight container. Another food safety concern is the materials used in the construction itself. We made sure to use plastics and other materials that wouldn't cause harm such as PLA plastics.

3.7 Performance, Security, Quality, Reliability, Aesthetics etc.

N/A

3.8 * Documentation

Our documentation process included documents such as design notes, technical specifications, flowcharts, code, schematics and user quick start guide. We used Google Drive to store and manage our documents and collaborate on them remotely.

3.9 Risks and Volatile Areas

N/A

4 * Experiment Design and Feasibility Study

Subsystems Feasibility Test

Ingredient Dispensing System <5% margin of error

Liquid Dispensing System <5% margin of error

4.1 * Experiment Design

One of our team's objectives was to determine if counting a stepper motor's steps is a reliable method for calculating the dispensed quantities of the auger system. To test, we had to build the auger mechanism, and test the quantities dispensed for a set number of steps. This was repeated multiple times, to collect quantitative data.(LH)

Another one of our team's objectives was to determine if adding a flow meter to the liquid dispensing system to shut off the solenoid valve was a steadfast method in determining the correct quantities of liquids dispensed. Additionally, our team wanted to test the generated pressure to allow for a more consistent and constant flow. In order to facilitate the testing for this, several containers were utilized to give differing types of volume and shape to generate enough pressure to allow for gravity to cause liquid to be dispensed. Our team expected to see results that would indicate a larger volume would indicate a larger pressure which would in turn cause a larger flow rate.(FM)

4.2 * Experiment Results, Data Analysis and Feasibility

For the first feasibility test, the results concluded that as long as there is a smooth flow of ingredients from the storage to the auger system, the dispensing amount deviates a maximum of 4 grams for every 100 grams of ingredients dispensed. This demonstrated that counting steps is a reliable method for dispensing specified quantities of ingredients. (LH)

For the liquid dispenser feasibility test, the results of this test concluded that with a larger weight of water above the opening, in addition to having more height in the containers would generate a better flow rate. Our team tested several strategies and compared them to the standard flow rate with a solenoid valve. With a ¼ inch pipe connecting a 2L bottle of water to a solenoid valve, our team was able to generate a 0.5 mL/s flow rate, which was suitable for our purposes. Our team originally tested several shapes to ascertain the best shape and volume for pressure and ultimately decided on a 2 liter bottle. Once happy with the flow rate provided by the solenoid valve, our team tested a flow meter to see if our team would be able to reliably detect how much water had passed through the flow meter. Unfortunately, our team found that with the addition of a flow meter in conjunction with a solenoid valve, the rotating turbine inside of the flow meter generated too much back pressure and caused the pressure to decrease substantially. When comparing it to the original, we found that the flow rate with the solenoid valve in conjunction with the flow meter would be around 0.1 mL/s which was substantially lower than the original flow rate. Our team opted to scrap the flow meter in favor of generating a better flow rate through the utilization of a gravity assist. (FM)

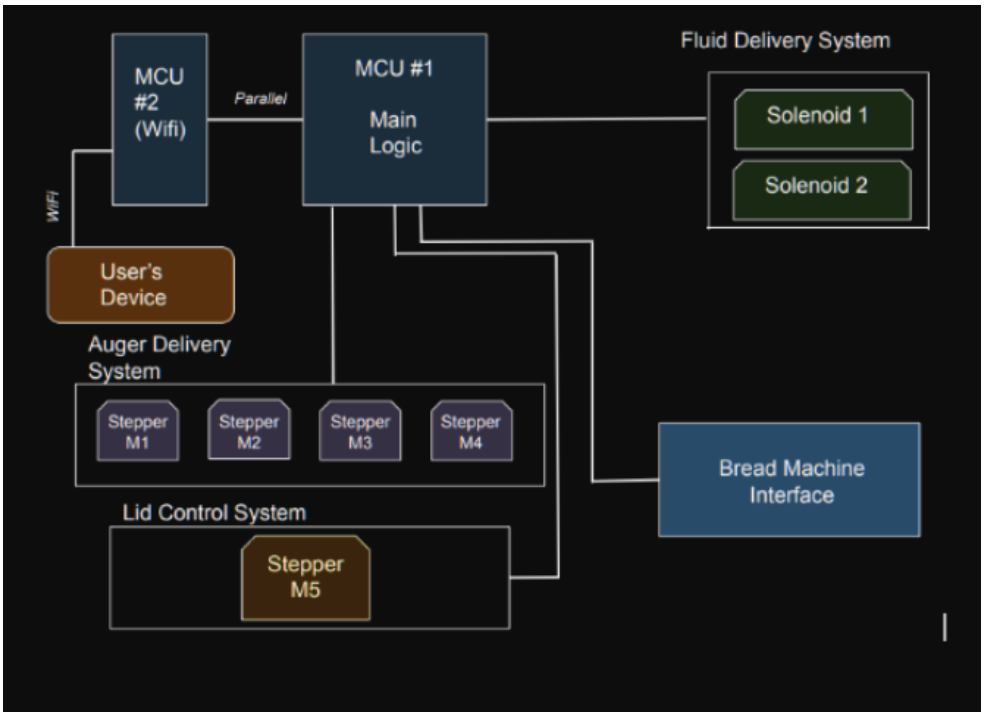
5 * Architecture and High Level Design

The architecture provides the top level design view of a system and provides a basis for more detailed design work. These are the top level components of the system you are building and their relationships.

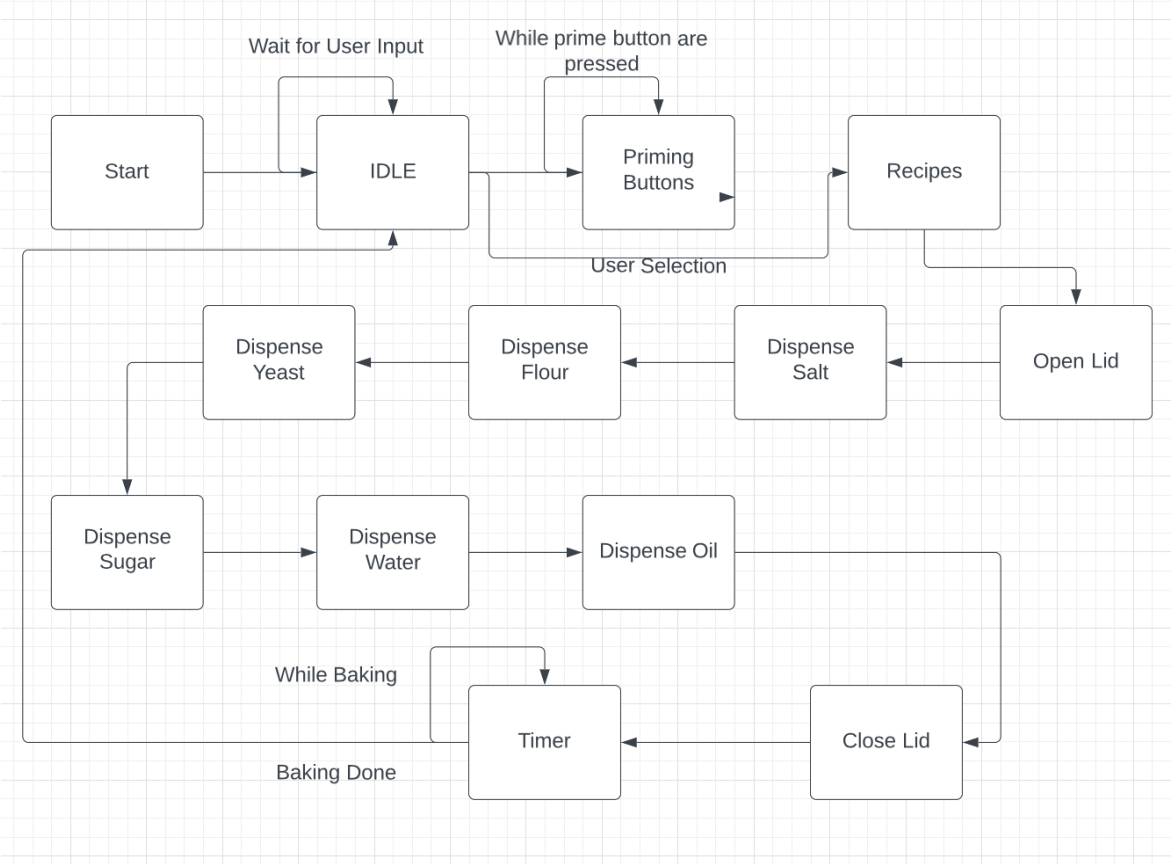
5.1 * System Architecture and Design

Our senior design project was decomposed into several subsystems. The reasoning behind why our team organized our setup in this way was because it was easier to modularize each subsystem, have individual people work on the subsystem, and finally integrate them all together into one big complex system. The entire complex system was composed of the fluid delivery system (FM), the auger delivery system(LH,NK), the lid control system(NK), the bread machine interface(LH,FM,NK), and finally, the wi-fi accessibility(NK). The fluid delivery system was developed to start and stop fluids using a valve and assess the amount of fluids dispensed. The auger delivery system was developed to start and stop delivery of solid ingredients using an auger based on the number of revolutions. The lid control system was developed to start and stop the open and closing mechanism based on the distance the lid needed to travel. The bread machine interface was developed to simulate the actuation of a physical button on the logic board. And finally, the wi-fi accessibility was created to remotely access the bread machine to start and stop the entire process.

5.2 * Hardware Architecture



5.3 * Software Architecture (only required if your design includes software)



5.4 * Rationale and Alternatives

Our team decided on this type of architecture and approach because it was the best step forward. Our team attempted to utilize a finite state machine but with Task Scheduler to simulate several tasks at the same time. Throughout testing and using the stepper motors with our power considerations in mind, our team was forced to look into alternatives to power each stepper motor independently and actuate them at specific but separate times. Our final conclusion was to utilize a finite state machine with a Moore machine setup. This would allow for several motors to be connected in and powered, but utilize a sleep mode to allow for only one motor to be powered.

6 Data Structures (include if used)

A description of all data structures including internal, global, and temporary data structures. State clearly who is responsible for which module/task

6.1 Internal software data structure

N/A

6.2 Global data structure

N/A.

6.3 Temporary data structure

N/A

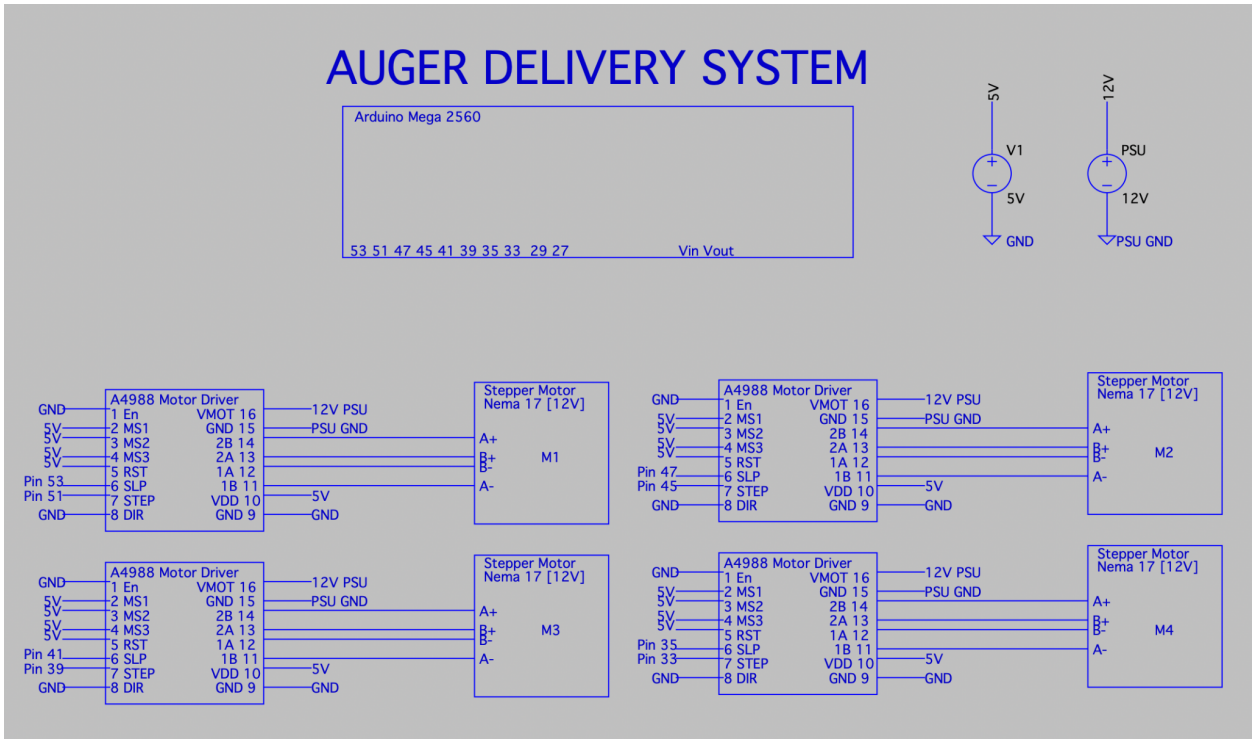
6.4 Database descriptions

N/A

7 * Low Level Design

7.1 *Auger Delivery*

The Auger Delivery module is responsible for controlling the 4 different stepper motors used to dispense and measure the quantity of ingredients being dispensed. (Luis H)



7.1.1 *Auger Delivery* interface description

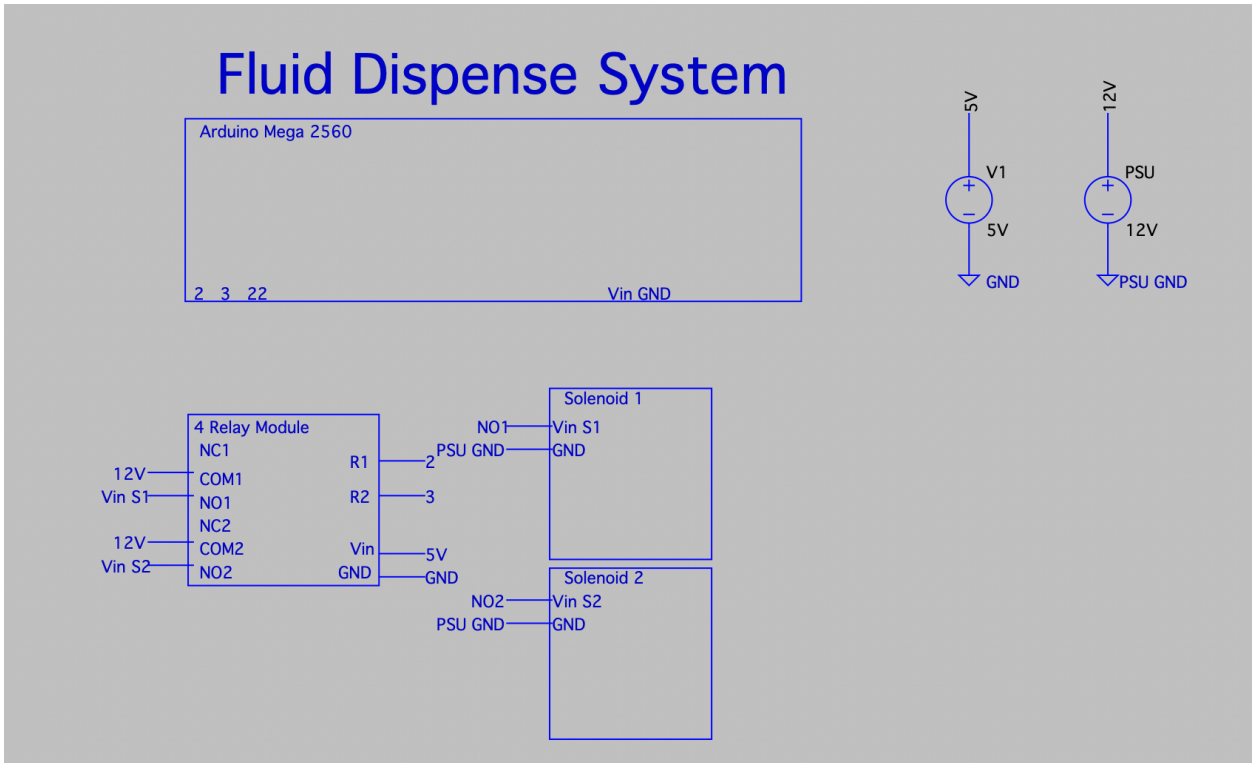
The Auger Delivery module uses the A4988 Motor Driver to control the stepper motors. In total we need to have two control signals per driver to interface with MCU#1(Arduino Mega). One of the signals provides step control, while the second signal is utilized to conserve power by turning off the driver and cutting power to the stepper motor.

7.1.2 *Auger Delivery* processing details

```
void stepper_dispense(int num, float amount){//Make sure to set sleep pin high, run the steper, then back low to enter sleep again
  switch(num){
    case 0:
    digitalWrite(8, LOW);                           //Used to control different module
    digitalWrite(stepperSleep[0], HIGH);            //Turns off stepper Driver Module
      stepper0.move(stepperRotations[0]*16*200*amount);   //Sets the number of steps needed to dispense ingridients
      //Serial.println(stepper0.distanceToGo());
      while (stepper0.distanceToGo() != 0 ){            //Moves stepper by specified number of steps
        stepper0.run();
      }
      digitalWrite(stepperSleep[0], LOW);           //Puts Motor Driver to sleep
      digitalWrite(8, HIGH);                        //Used to control different module
      break;
```

7.2 Fluid Delivery

The Fluid Delivery module is responsible for controlling the dispensing of Fluids . (Felix M)



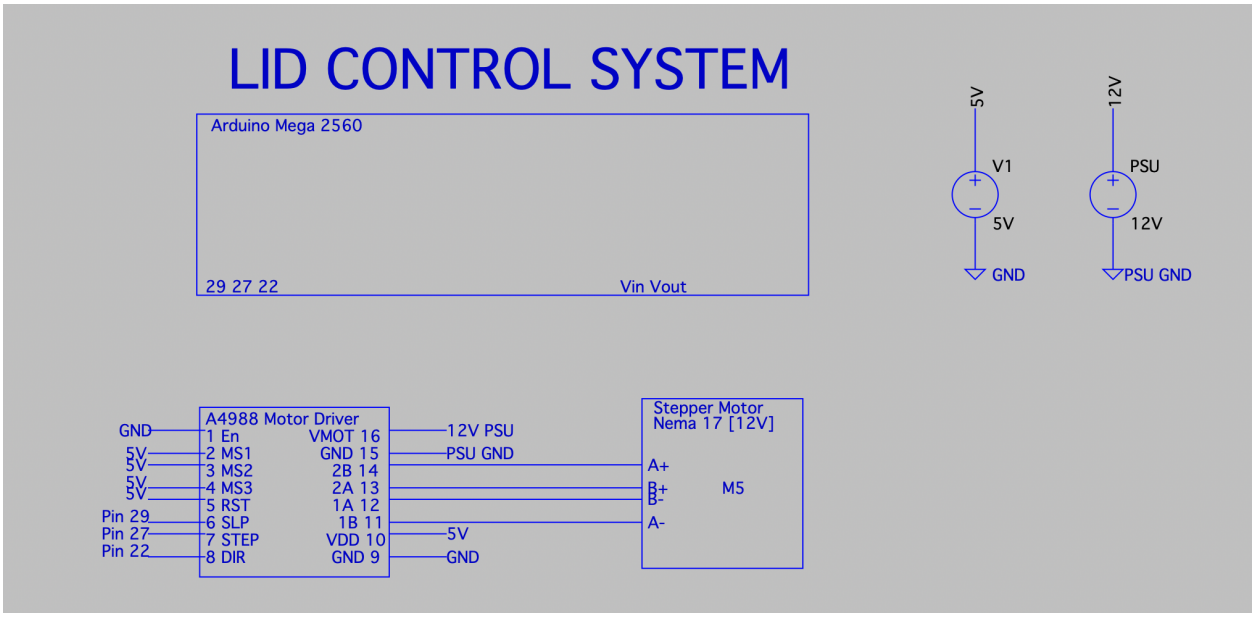
7.2.1 *Fluid Delivery* interface description

The module controls the flow of liquids by opening and closing solenoid valves with a measured time value, which corresponds to the amount of liquid the module should dispense depending on the recipe it is

following. Relays were implemented to control the solenoids, because of the power requirements they needed.

7.3 Lid Control

Module responsible for opening and closing the lid of the bread maker. (Nevin K.)



7.3.1 *Lid Control* interface description

The lid control module functions very similarly to the auger Delivery System, as it uses the same stepper Drivers to control the stepper motor. However, two important distinctions include the additional control pin required to control the direction of rotation, and the different sleep configuration the system requires. The driver must not be put to sleep after opening the lid, because it will cut power to the stepper, which would cause the lid to fall back down.

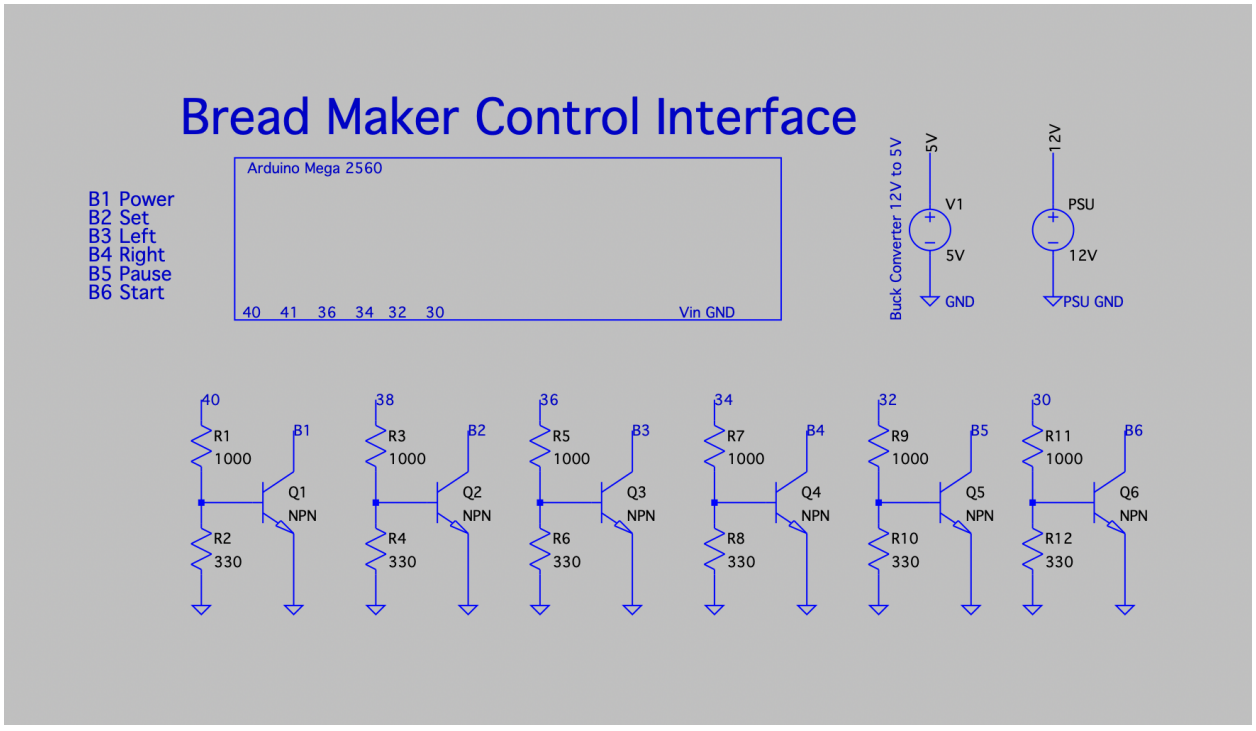
7.3.2 *Lid Control* processing details

```
void openLid(){
  digitalWrite(stepperSleep[4], HIGH);           //Turns stepper Driver
  stepper4.move(200*16*1.5);                     //Number of steps needed to open lid
  //Serial.print("Distance: ");
  //Serial.println(stepper4.distanceToGo());
  while(stepper4.distanceToGo() != 0){          //Stepper motor runs number of steps needed = 0
    stepper4.run();
  }
  return;                                        //steper Driver Remains On
}

void closeLid(){
  stepper4.move(-200*16*1.5);                    //Number of steps need to close lid
  //Serial.print("Distance: ");
  //Serial.println(stepper4.distanceToGo());
  while(stepper4.distanceToGo() != 0){          //Stepper motor runs number of steps needed = 0
    stepper4.run();
  }
  digitalWrite(stepperSleep[4], LOW);           //steper Driver is put to sleep
  return;
}
```

7.4 Bread Maker Interface

Bread Maker Interface responsible for sending input from MCU#1 to the Bread Maker's internal logic (Luis H.)



7.4.1 *Bread Maker* interface description

The Bread Maker interface is responsible for simulating button presses to the bead maker machine. This is accomplished with the help of npn BJT transistors.

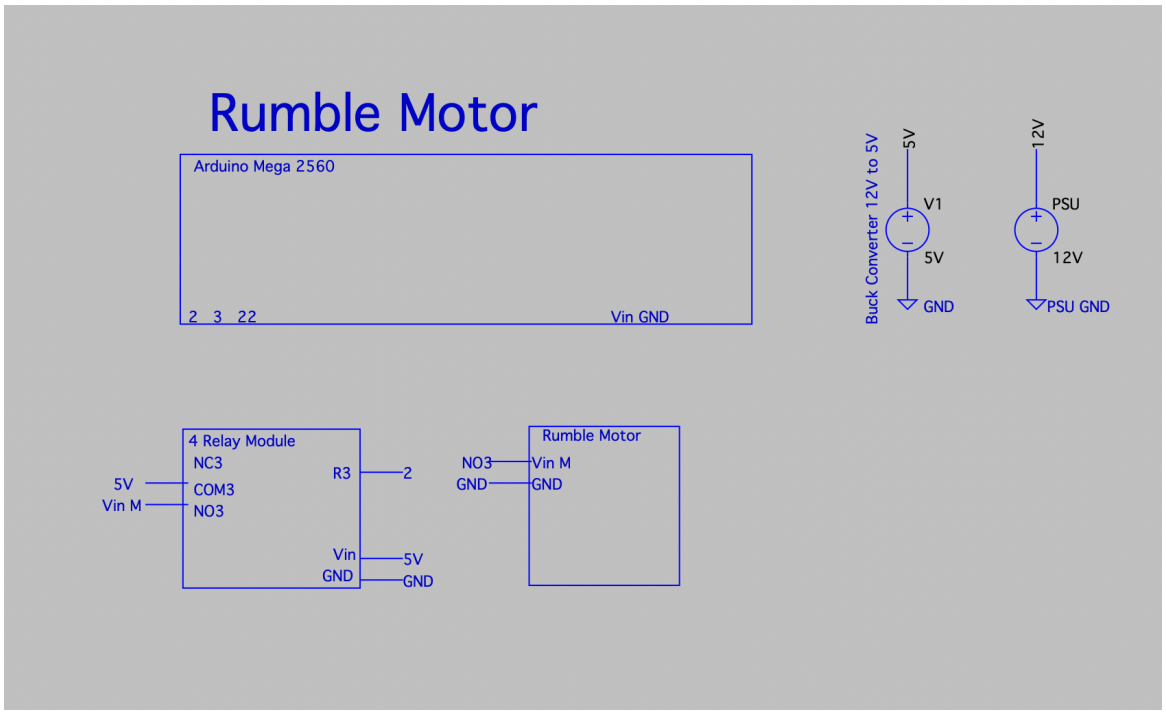
By applying a positive voltage to the base of the transistor, a button press is detected by the bread maker. Depending on the recipe selected, the right button sequences need to be pressed, to select the desired cooking settings. These include the level of toast, light, medium, dark. Alongside the desired batch size.

7.4.2 *Bread Maker* processing details

```
void press(int num1){
  digitalWrite(num1, HIGH);     //Button Press Simulated
  delay(200);                   //Button held down for 200ms, needed to be properly detected
  digitalWrite(num1, LOW);      //Button Realease Simulated
  delay(200);
  return;
}
```

7.5 Rumble Motor

Schematic:(Luis H)

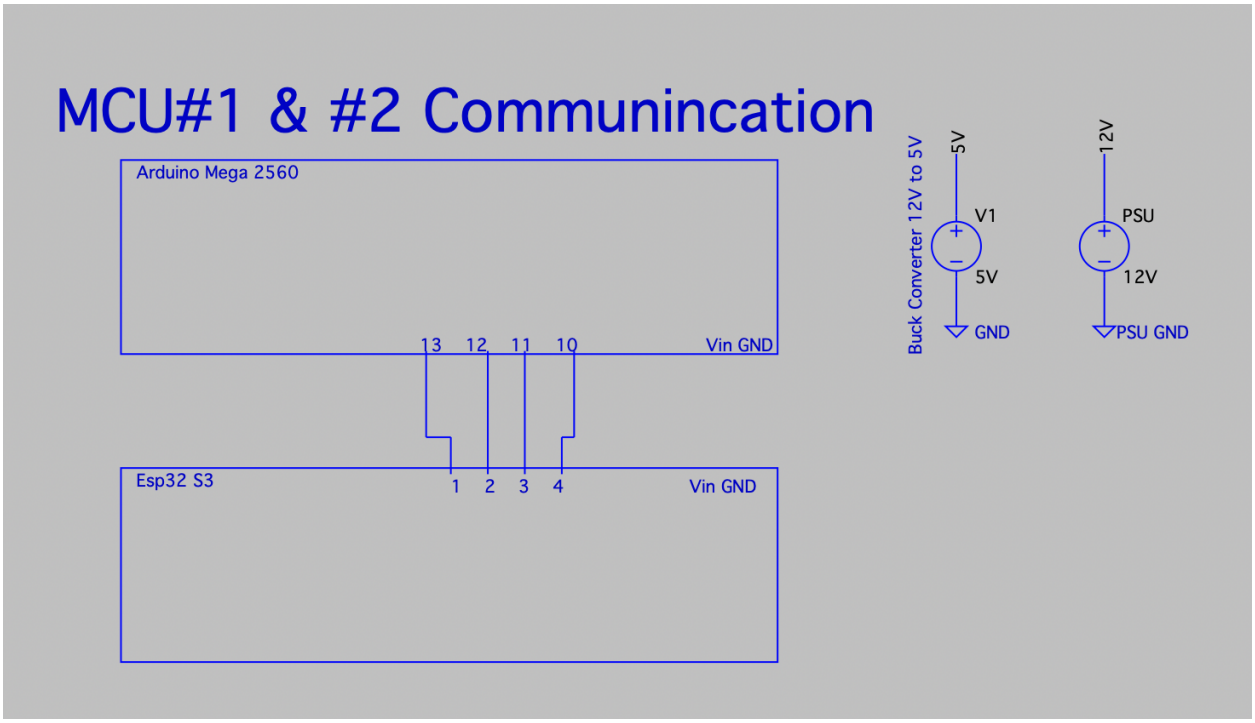


7.5.1 *Rumble Motor* interface description

The rumble motor is only used to prevent flour from clumping inside the flour container. Only flour had this issue. so only one rumble motor is being used.

The rumble motor is powered by a DC buck step down converter, which takes the power supply's 12V as input, to power the 5V rumble motor. A relay was added to the circuit connection to turn on and off the motor.

7.6 MCU Communication



7.6.1 *MCU Communication* interface description

At the moment, BRAD provides only 4 different recipes. Each connection represents one of those recipes. Ex: If MCU#1 detects Pin 13 to be high, then, recipes 1 is assumed to have been picked.

8 * Technical Problem Solving

8.1 * The Light Ingredients Dispensing Problem

The initial design for the auger system proved to have difficulty dispensing ingredients reliably. For light ingredients like flour, the main issue was inconsistent flow of the ingredients from storage to the dispensing system. This was because flour would tend to clump up within the storage containers, which would not feed ingredients to the augers responsible for dispensing. (Luis H.)

8.2 * Solving the Light Ingredients Dispensing Problem

The approach taken to solve this issue was to add a DC motor to the storage container walls, which would introduce vibrations to prevent the clumping of Flour. Fortunately, the issue was resolved, with the help of the Rumble Motor, taken from a broken PS4 Controller. (Luis H.)

8.3 * The Coarse Ingredient Dispensing Problem

The same initial design of the auger system frequently tended to jam with coarser ingredients like salt and sugar. This was caused because tolerances within the 3D printed parts of the dispensing system were small, and material tended to be squished against the moving bit and the outer body of the auger system. (Luis H.)

8.4 * Solving the Coarse Ingredient Dispensing Problem

To resolve this issue, we had to redesign some parts, and accommodate for bigger tolerances. This avoided ingredients being grinded inside the auger, which caused the system to stall in the initial test runs.(Luis H.)

8.5 * The Liquid Dispensing Problem

The liquid dispensing system within our design, is dependent on gravity to generate flow, and it did not account for not having enough fluid pressure. As a result, we can not reliably dispense accurate amounts of water or oil. (Felix M)

8.6 * Solving the Liquid Dispensing Problem

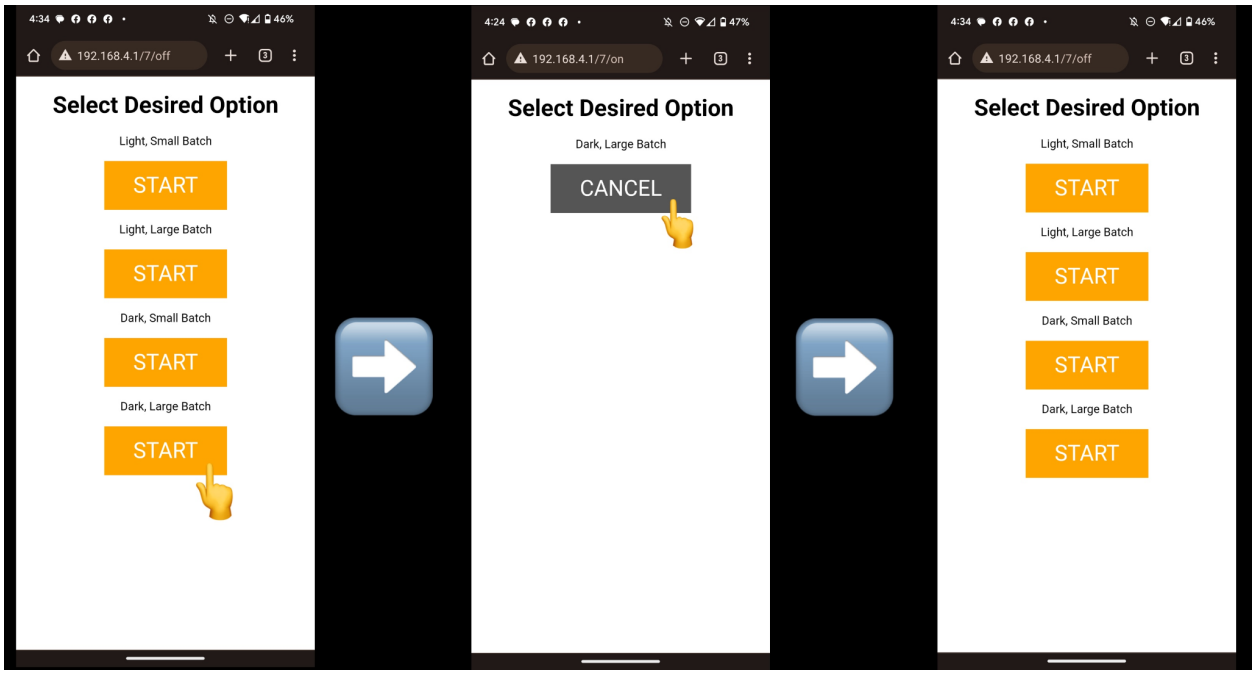
The liquid dispensing problem was not solved in time, but solutions like increasing the pipe diameter to decrease the resistance of the pipe, along with adding a liquid pump should be viable solutions for mitigating this problem. (Felix M)

9 User Interface Design

9.1 Application Control

The main menu as pictured below is the one the user will see initially. From here, the user will choose their desired menu option. Once the user taps a button, the second page will show (also pictured below). At the same time, the ESP32 will send a signal to the dedicated output pin telling the bread machine which recipe was selected. At this screen, the user has the option to press cancel, which will bring the user back to the initial screen. At the same time, the ESP32 pulls down the pin that was initially pulled high, telling the bread machine to cancel the operation. (Designed by Luis Herrera and Nevin Kopp)

9.2 User Interface Screens



10 * Test Plan

10.1 * Test Design

Test Case 1 (as performed by Nevin Kopp):

1. The objective of this experiment was to determine the number of stepper motor steps required to dispense a certain amount of ingredients. The steps didn't have to be pure integers, we were able to do fractions of a step to get precise measurements.
2. Experiment setup: The setup for this experiment involved a stepper motor and a scale to measure the amount of dispensed ingredients.
3. Experimental procedure: I first set up a scale underneath the auger to collect fallen ingredients. Using the software I then programmed a random number of steps to see how much ingredients were dispensed. If it was too much I would dial the steps back, if it were too little I would increase steps. Using this method I was able to determine a very accurate measurement for each dispensed ingredient. From there we were able to use these values to accurately dispense ingredients without a scale.
4. Expected results: The expected results were certain values per ingredient that we would use in the process of ingredient dispensing. Through experimentation I was able to acquire the values we were looking for.

Test Case 2 (as performed by Felix Maass)

5. The objective of this experiment was to determine whether or not the addition of a flow meter would impede the necessary flow rate to generate substantial fluid dispensing as well as how shapes and volume
6. Experimental Setup: The setup for this experiment involved several differing shapes and volumes of water to be dispensed through the solenoid valve and as well as a solenoid valve in conjunction with a flow meter.
7. Experimental Procedure: The fluid dispenser was attached to a measuring cup and a timer was set to ascertain the average flow rate from each volume and shape. The same tests were performed on a fluid dispenser with a solenoid valve and the flow meter.
8. Expected Results: Our team expected to see no importable change in the flow rate, but however, were surprised when the results indicated that the addition of the flowmeter would slow down the flow rate substantially.

10.2 * Bug Tracking

A database will be used to track defects found while performing the test cases. All defects will be logged as they are discovered. Defects will be assigned to Person A to fix, or to Person B to investigate.

Several defects were found when performing the various test as denoted below as a FAIL instead of a PASS. Throughout all of the solid ingredients, sugar was the least problematic as it had the necessary weight, texture, and size to be able to be pushed through an auger funnel. While testing for the salt and yeast, it was found to be problematic that the texture of the grains were too small and too light to be able to have enough weight to fall upon the auger system. Felix was put in charge of figuring out a solution, and as a team, we decided on utilizing a grinder to make the ingredients more finer, thereby causing less surface area to be on the particle, to allow it to fall faster and more consistent. The fluid dispenser had several design issues when trying to integrate and accommodate for ingredients other than water. When attempting to dispense water, the flow of the water generated was at a constant and consistent manner, but was definitively restricted by the addition of a flow meter. In order to achieve maximum flow through our ¼ inch piping, we decided to remove said flow meter to opt for the better flow rate.

10.3 * Quality Control

The completed test cases will be reviewed to ensure that all cases were run; that all were completed successfully; and that any deviations from the test cases were noted accordingly. Each step should be marked as Passed or Failed. Failed cases should be marked with the date and time of the failure, and the associated test track number. When a failed case is fixed, the date and time of the retest should be noted

Test #	Test Case	Case 1 2/10	Case 2 2/12	Case 3 2/20	Final Status
1	Tested Solid Ingredient (Salt)	FAIL	PASS	PASS	PASS
1	Tested Solid Ingredient (Yeast)	FAIL	PASS	PASS	PASS
1	Tested Solid Ingredient (Flour)	FAIL	PASS	PASS	PASS
1	Tested Solid Ingredient (Sugar)	PASS	PASS	PASS	PASS
2	Tested Fluid Ingredient (Water)	FAIL	FAIL	PASS	PASS
2	Tested Fluid Ingredient (Oil)	FAIL	FAIL	PASS	PASS

10.4 * Identification of critical components

The most critical component during testing was the stepper motor. Since we weren't using a scale, we needed to determine the amount of ingredients using a different method. We settled on doing some trials involving specific step amounts to determine the amount of ingredients dispensed. In the end we were able to achieve a very accurate process of ingredient dispensing.

10.5 * Items Not Tested by the Experiments

N/A All parts were tested and accounted for.

11 * Test Report

Carry out the experiments designed in the section above to test your modules and prototype and present the results. Present the results of the experiments and provide an analysis of the experimental test data.

11.1 * Test 1

Person performing the experiment: Nevin Kopp

1. Test results: For the large batch of bread I found that we needed 3 rotations of the stepper motor for the correct amount of sugar, 0.5 rotations for salt, 0.4 rotations for the yeast and 12.5 rotations for the flour.
2. Comparison with expected results: There were no expected results, I was determining how many rotations were needed for the correct amount of dispensed ingredient.
3. Analysis of Test results: The number of rotations I found were used for both the large and small loaf sizes since we just used certain ratios of the experimental numbers.
4. Corrective actions taken: When it dispensed too much ingredient, I reduced the amount of steps and if it wasn't enough ingredients I would increase the amount of steps.

11.2 * Test 2

Person performing the experiment: Felix Maass

1. Test results: For the flow meter in addition to the solenoid valve, our team found at maximum of a 0.1 mL/s flow rate, which was consistent with expectations. Without the flow meter, the solenoid valve would dispense water at a rate of 0.5 mL/s.
2. Comparison with expected results: We expected these results as the addition of the flow meter turbine would definitively cause a decrease in the water pressure derived from the bottle above.
3. Analysis of Test results: We decided to remove the flow meter in hopes of restoring proper flow.
4. Corrective actions taken: Flow meter was removed.

12 * Conclusion and Future Work

12.1 * Conclusion

In conclusion, the development of an automated bread making machine posed many technical challenges. Through the diligent efforts of all our team members we were successful in designing a machine that is able to automatically make fresh, high quality home baked bread. The project required our team to have a wide range of skills from mechanical engineering and software development to culinary skills. By bringing together team members with different backgrounds and skills, we were able to overcome many challenges and create a very useful product.

Take home lessons from each team member:

Nevin - I learned a lot from this project and went out of my comfort zone with many attributes. As an electrical engineering student, it was difficult to think in terms of mechanical engineering. I'm also not a great baker so learning how to make bread was challenging at first. This was also the first major project of mine and learning how to manage all the documentation and other project materials posed its own challenges. This was also the first time I deployed a major Internet of Things project so learning web server and HTML technologies was fun and I feel will play a major role in my future development.

Luis - I learned that quick adaptability to problems is an important skill to possess as an engineer. During the building process of the project, there were so many issues that arose, and being able to quickly resolve these problems, or come up with different alternatives was an indispensable tool. I also learned the importance of measuring multiple times when designing parts, for it could save one from potential time costly mistakes. And I learn the importance of taking tolerances into consideration when designing parts as well.

Felix - There were several new concepts and perspectives that I had to learn throughout this project. As an individual who prefers to steep himself in code, I was definitely taken outside of my comfort zone and thrust into a world of mechanical and electrical components. This project allowed me to develop the tools and habits necessarily to tackle full scale projects and be able to break them into smaller pieces. Not only was this indispensable towards my education, but I now know how to properly debug and test problems by taking the proper time to learn about each component and understand the capacity and restrictions placed upon each.

12.2 Future Work

One of the limitations to our design is the addition of a new ingredient. In order to add a new ingredient, we would need to add on an additional stepper motor, control circuitry and update the software. It would be much better if we were to use a single stepper motor on a sort of rotational base where the ingredients rotate and interface with an additional stepper motor that is used to dispense the ingredients. This way the addition of ingredients would be greatly simplified and more user friendly. Keeping the machine fully

autonomous would still pose challenges however, but many competing products do not offer the same level of autonomy.

12.3 * Acknowledgement

Be considerate and credit all those who have helped you in this project, especially credit the people who provided ideas or solutions.

Christopher Eng (our TA)
Roman Chomko (our Professor)
PantryChic (inspiration for auger system)
EW (Electronic Warehouse)

13 * References

List the references used in the design, including books, data sheets, technical documents, industry standard documents. References can be printed documents or online.

- Getting started with the ESP32 Development Board: https://randomnerdtutorials.com/getting-started-with-esp32/
- Installing ESP32 Board in the Arduino IDE https://lastminuteengineers.com/esp32-arduino-ide-tutorial/
- ESP32 Web Server - Arduino IDE https://randomnerdtutorials.com/esp32-web-server-arduino-ide/

14 * Appendices

*** Appendix A:** Parts List
*** Appendix B:** Equipment List
*** Appendix C:** Software List (URL to online drive or SVN server, with sharing set to Public. Can omit this appendix if your project didn't involving writing a program)
Appendix D: Special Resources
Appendix E: User's Manual - If your design requires instructions for future use, here is the place to put that information.
Appendix F to Z: Whatever Else You Wish To Add; for instance, here, you may include detailed solution methods or derivations, which you need for your future review of this report or whoever else is interested to pursue this study. Some side drawings and prints that are of value to people who will continue this work should be given herein. Some information about the vendors and how to locate parts for similar projects must be included herein. In other words, information that is important about the overall construction of the project should be given herein.)

APPENDIX A (Parts List)

Number of Parts	Part Name	Full Name	Subsystem	Cost	Where to Buy	Purchased?	Arrive Date	Total Cost
Number of Parts	Stepper Motor	Nema 17 Stepper Motor 1 A	Auger	10.99	Ebay	Yes	Arrived	54.95
10x	Driver Module	A4988 Driver Module Step	Auger	14.59	Ebay	Yes	Arrived	14.59
4x	Aluminum Frame	400mm Extruded Al	Frame	16.52	Ebay	Yes	Arrived	66.08
2x	Aluminum Frame	600mm Extruded Al	Frame	22.57	Ebay	Yes	Arrived	45.14
2x	12V PSU	12V 2A PSU AC Adapter	Auger	11.75	Amazon	Yes	Arrived	11.75
1x	L-Shape Interior Inside Corner Connector		Frame	13.64	Amazon	Yes	Ariived	13.64
1x	Flexible Couplings 5mm to 8mm		Auger	9.99	Amazon	Yes	Arrived	9.99
1x	Ball Bearings Skateboard Bearings 10pack		Auger	9.99	Amazon	Yes	Arrived	9.99
1x	3mm fastener and nuts set		Auger/Frame/Etc	18.99	Amazon	Yes	Arrived	18.99
1x	Bread Machine		Machine	0	NA	Yes		0
6x	Plastic Container		Storage					
6x	PVC Piping		Storage					
1x	Arduino Mega	Arduino Mega MCU	Controller	0	NA	Yes		0
2x	Wifi Module	ESP32 VROOM DevKit	Wifi System	$24.33	Amazon	Yes		$48.67
3x	Solenoid Valve	ESValve 1/4 12V Push Connect Solenoid Valve	Dispensing	$16.99	ESValves	Yes	1/22/22	$50.97
3x	Flow Meter	DIGITEN 1/4 Quick Connect 0.3-10L/min	Dispensing	$9.49	Amazon	Yes	1/19/22	$28.47
2x	Wires	Elegoo 120 pc Wire Kit	Wiring	$6.98	Amazon	Yes	1/21/22	$13.96
1x	4 Channel Relay	ELEGOO 4 Channel DC 5V Relay		17.22	Amazon	Yes	3/8/23	17.22
1x	DC-DC Converter	HiLetgo LM317 DC-DC Buck Converter		12.16	Amazon	Yes	3/3/23	12.16
	Total Cost	387.19						
		547.19	with the 2 bread makers					
		577.19	with the third bread maker					

APPENDIX B (Equipment List)

Our team utilized several pieces of equipment throughout the testing and debugging of this project:
Oscilloscope - to find the signals of each of the interfaces with the bread machine
Multimeter - to find and check continuity in the electrical components as well as to be used as an ohmeter.
Soldering Iron - to solder in connections to the actual bread machine
3D Printer - to print and prototype different parts of the system

APPENDIX C (Software List)

Program Code can be found here:

https://github.com/Luckychief/Autonomous_Bread_Maker

Softwares that were Utilized in this Senior Project

Wifi.h - used to serialized the access point for the Wifi ESP32 Module

Accelstepper.h - used to interface with the stepper motors and the PSU boards

LiquidCrystal.h - used to debug error codes from the MCU to a display module

APPENDIX D (User's Manual)

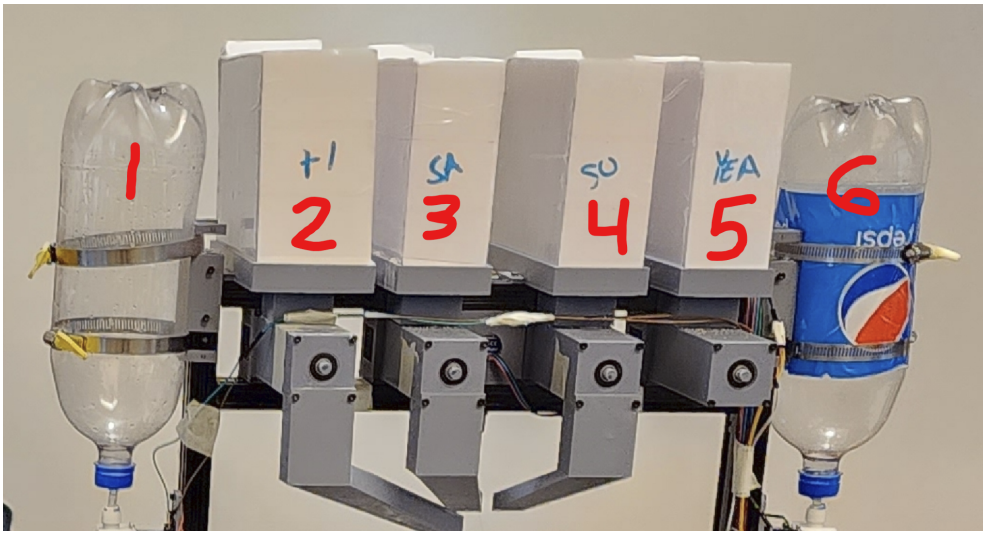
B.R.A.D (Bread Recipe Automated Device)
Quick Start Guide
ver 2.0.23

Getting Started

Congratulations on your purchase of brad, a state of the art automated bread making machine! In the quick start guide you will learn everything you need to know to start making delicious, fresh homemade bread. Refer to the product picture as shown on the first page of the guide to get an idea of what the final setup looks like. Once you have your machine set up and powered, you can now insert your ingredients.

Adding the Ingredients

Now we can start to add the ingredients. As shown below, brad has 6 containers for ingredients, 2 to hold fluids and 4 to hold dry ingredients such as flour.



Each container must hold a specific ingredient, otherwise your bread may not come out as desired. Refer to the following list for specific ingredient placement:

1. Oil
2. Flour
3. Salt
4. Sugar
5. Yeast
6. Water

You can place as much of the ingredients as can fit, just make sure they are in the correct container.

Connecting Over the Internet

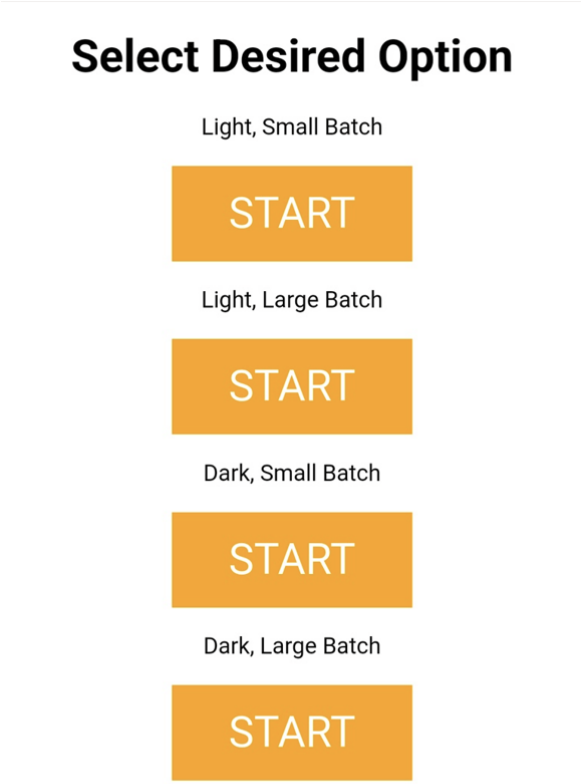
One major advantage of brad is that you can make fresh bread over the internet. Follow the steps outlined in this section to get you up and running.

1. Make sure brad is connected to power. Once brad has power it will be ready to connect to.
2. Open your wifi settings and search for nearby networks. You should see one listed as “ESP32 Access Point”.
3. Using your designated password, connect to “ESP32 Access Point” and open your web browser.
4. In your search bar, type in “192.168.4.1”.
5. Congratulations! You should now be connected to brad.

Using the Webapp

Now that you are connected to brad, you can start making delicious, fresh, homemade bread! Follow the instructions as shown below to have brad automatically start a bread cycle.

You should currently see something like the following picture:

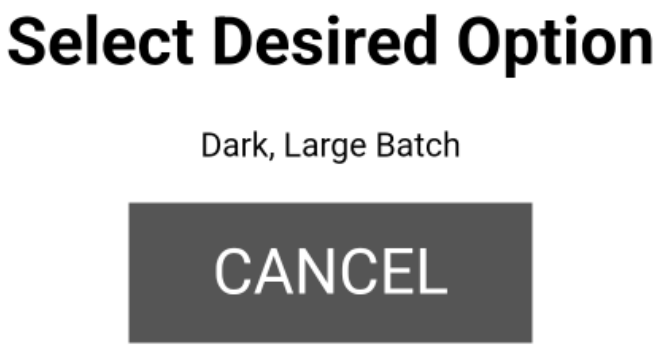


If not, please refer to the “Connecting Over the Internet” section.

Once you see the previous page, making fresh bread is one simple click away! Below are the current bread options brad is capable of:

- **Light, Small Batch**. Choose this option if you want your white bread toasted slightly and only want a small loaf.
- **Light, Large Batch**. Choose this option if you want your white bread toasted slightly and want a bigger loaf.
- **Dark, Small Batch**. Choose this option if you want your white bread more toasty and want a small loaf.
- **Dark, Large Batch**. Choose this option if you want your white bread more toasty and want a bigger loaf.

Once you press an option, brad will immediately start preparing your bread. If you wish to cancel and return to the main page, press the following button:



Care and Maintenance

Once brad is completed with your bread, remove the loaf from the main cooking/mixing chamber as shown below:

Main cooking/mixing unit

You should see a small paddle at the bottom of the chamber, if not, it might be inside the loaf so make sure to remove it. Once the loaf is removed, hand wash the inner metal chamber and the paddle. Make sure to fully dry both parts before placing them back into the machine. Once the inner chamber is back in the machine, you are ready to make another loaf!