



FUD

Project Report
CECS 490B - Senior Project
2/21/20

Raul Solorio | Geoffrey Grepo | Alex Agustin | Dennis Vivanco



Table of Contents

1	Introduction	3
1.1	Who Is Doing What?	3
1.2	Why FUD?	4
1.3	Problems To Be Solved	5
1.4	Technical Challenges	6
2	Design.....	7
2.1	The App.....	7
2.2	The Food Maker	10
2.2.1.1	Conveyor Belt	11
2.2.1.2	Bowl Holder	12
2.2.1.3	Fridge Doors	13
2.2.1.4	Fridge Design Flow	14
2.2.2	The Cooker	15
2.2.2.1	Cook Stands	16
2.2.2.2	Potholder and Cooking Mechanism	17
2.2.2.3	Induction Cooker	18
2.2.2.4	Claw	20
2.2.2.5	Cooker Design Flow	21
2.2.3	Circuit Schematics	22
2.3	Web Server.....	25
2.3.1	Database	25
2.3.2	API.....	26
3	Verification	29
4	Cost and Schedule.....	48
4.1	Cost Analysis	48
4.2	Schedules.....	Error! Bookmark not defined.
5	Ethical Implications.....	52
6	Local Contributions and Project Tools.....	52
6.1	Local Contributions	52
6.2	Project Tools.....	53
7	References	54



1 Introduction

1.1 Who Is Doing What?

Person:	Roles:
Alex Agustin	Food Maker: Fridge and Cooker (Design, Part Research, Manufacturing, Testing), Logo Design Example of tasks completed: -Created fusion 360 designs -Sent to be 3D printed and tested them. -Constructed Conveyor Belt.
Raul Solorio	Food Maker: Fridge and Cooker (Testing, Circuit Testing, Part Research, Manufacturing) Example of tasks completed: -Made all 11 motors move -Created code in C for motors. -Developed circuit for motors.
Geoffrey Grepo	Food Maker: Fridge and Cooker (Testing, Part Research, Manufacturing) Example of tasks completed: -Tested durability of motors -Tested and researched LV-MaxSonar-EZ1 Ultrasonic Sensor
Dennis Vivanco	App Development, Food Maker: Fridge and Cooker (Design, Parts, Manufacturing, Testing, Software), Obtaining Sponsors, Logo Design Example of tasks completed: -Developed app -Deconstructed fridge.



1.2 Why FUD?

We first came up with the idea of creating a food maker for our project due to our prior experience taking care of a disabled person or know someone with disabilities. We understood that someone with disabilities needs a lot of attention and sometimes have trouble preparing food to eat. As we continued to talk about the food maker, we decided to convert the food maker into something that anyone can use. The reason behind this change was because we noticed that many people don't have the time, energy, or method of enjoying a healthy homemade food. With this in mind, we decided to add components to the food maker in order to allow people to obtain healthy food even easier. This included adding an app where people can order food and a robot that can deliver the food. Below are some examples of how other people can use our product.

- Working individuals can use the FUD app to schedule food to be prepared at a specific time, so when they arrive tired from work, they can have a home cooked meal ready.
- College students can now eat healthier from home.
- Hospitals can have patients order food from their beds and then have the food delivered to their rooms.
- Senior centers can now serve food to the elderly no matter what time of day it is.
- Stay at home elderly people can now have a home cooked meal without being dependent on other family members.
- Everyone can meal prep at a given notice or push of a button within the app.



1.3 Problems To Be Solved

1) Convenience: Everyone must eat, but our design provides a more convenient way of getting cooked food, by behaving like a Keurig coffee machine but for food. All you need to do is select the food you want and done. Best of all, the food comes prepacked like Keurig coffee cups so all you must do is put the food container that were ordered into the machine and done.

2) Variety: FUD can also solve the problem of homes where different people will want to eat something else, by offering different recipes, and foods from different diet plan options like vegetarian, keto, vegan, and Atkins.

3) Meal Prepping: This can also eliminate meal prepping for a week, meaning you don't have to cook meals for the week when FUD can cook for you. Which is helpful since no matter the food, every recipe requires dishes, preparation, time, and cleanup. With FUD, we can reduce the number of dishes a person uses, nearly eliminate the preparation phase, reduce the amount of time needed to cook, and reduce the number of dishes that needed to be cleaned to 1.



1.4 Technical Challenges

From what we have seen so far there are several technical challenges that we going to face:

1) **Manufacturing:** A lot of parts will be required in this project. This has caused our total parts cost to be over \$700, which is more than expected. In order to help reduce this cost, parts can be manufactured via 3D printing, but this will require more time since the manufacturing process requires time for designing, 3D printing, and testing. To add on to this, if the manufactured part does not work, the whole process will need to be restarted thus taking more time. In the end, the big issue will be trying to find a medium of how much time we can dedicate to manufacturing parts and how much the group can afford to spend on parts. We must also deal with welding our cooking pot to a motor and test if the motor can handle the pressure it will go through as the cooking process completes.

2) **Microcontroller Communication:** This Project will require one TM4C123 development board and a Raspberry Pi thus an effective method of microcontroller communication will need to be implemented. Due to this project having different components, it's important that the microcontrollers can communicate with each other's with as little or no hiccups occurring, since a miscommunication can cause the whole cooking process to fail or be completed incorrectly.

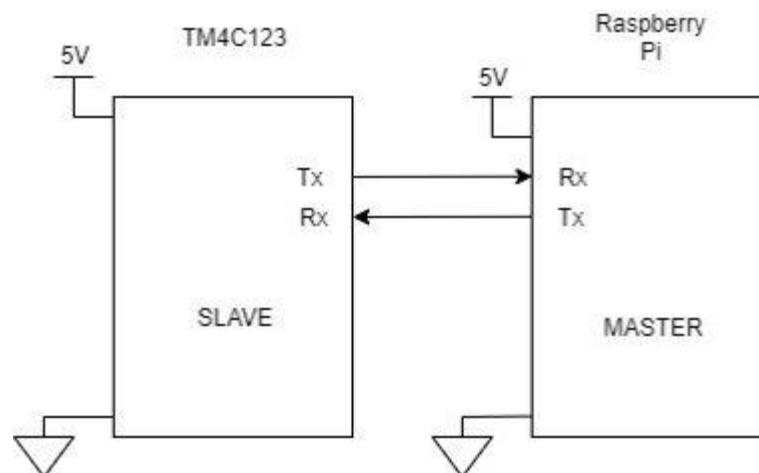


Figure 1. Wiring for UART communication between 2 microcontrollers

3) **Multiple parts project:** This project will incorporate many different components, especially in the food maker. Making all parts work together will be a heavy task, just in the fridge alone from the research that we have seen, the fridge will require around 20 parts, and the cooker will require 10 parts. To help tackle this challenge, we plan to order



extra parts to ensure we can replace parts that break/malfunction. We also plan to make sure that we have at least a working prototype of the fridge and cooker by the beginning of the spring semester in order to verify and test our design. Along with connecting the raspberry pi and mobile application to an api that will be in charge of placing orders on queue. The prototype will allow us to see how much time we need to allocate in order complete the finish product during the spring semester.

6) **Cooking process:** This project requires us to control the cooking process by using a microcontroller and various other parts. Being able to automate this process and figuring how to cook certain meals properly will be hard. If we have two different types of food that need to be prepared for the food, we will have to control how long each type is cooked since they may need different cooking times. We will have to properly mix, rotate, and heat up the food so that it can be eaten.

2 Design

Project consist of 3 main components, the app, the food maker, and the robot server. The group will mainly focus on the app and the food maker for now and make the food delivery robot as our stretch goal.

2.1 The App

The app's main goal is to order the food, and that will be the current focus of the app. If time permits, we will add other features, such as the ones shown below:

1. Display food nutrition information.
2. Keep track of food inventory and expiration dates.
3. View order history and reorder food.
4. Notify user when food is finished.
5. View food order queue.
6. Set up future food orders.

When ordering food from the app, the following actions will take place:

1. User opens FUD app.
2. App indicates the kind of food they can order, depending on what type of food the user placed into the fridge, and what inventory is available.
3. User selects the food that they want to order.
4. User confirms food order, and app displays estimated completion time.



5. App provides meal information to server which sends it to the raspberry pi.

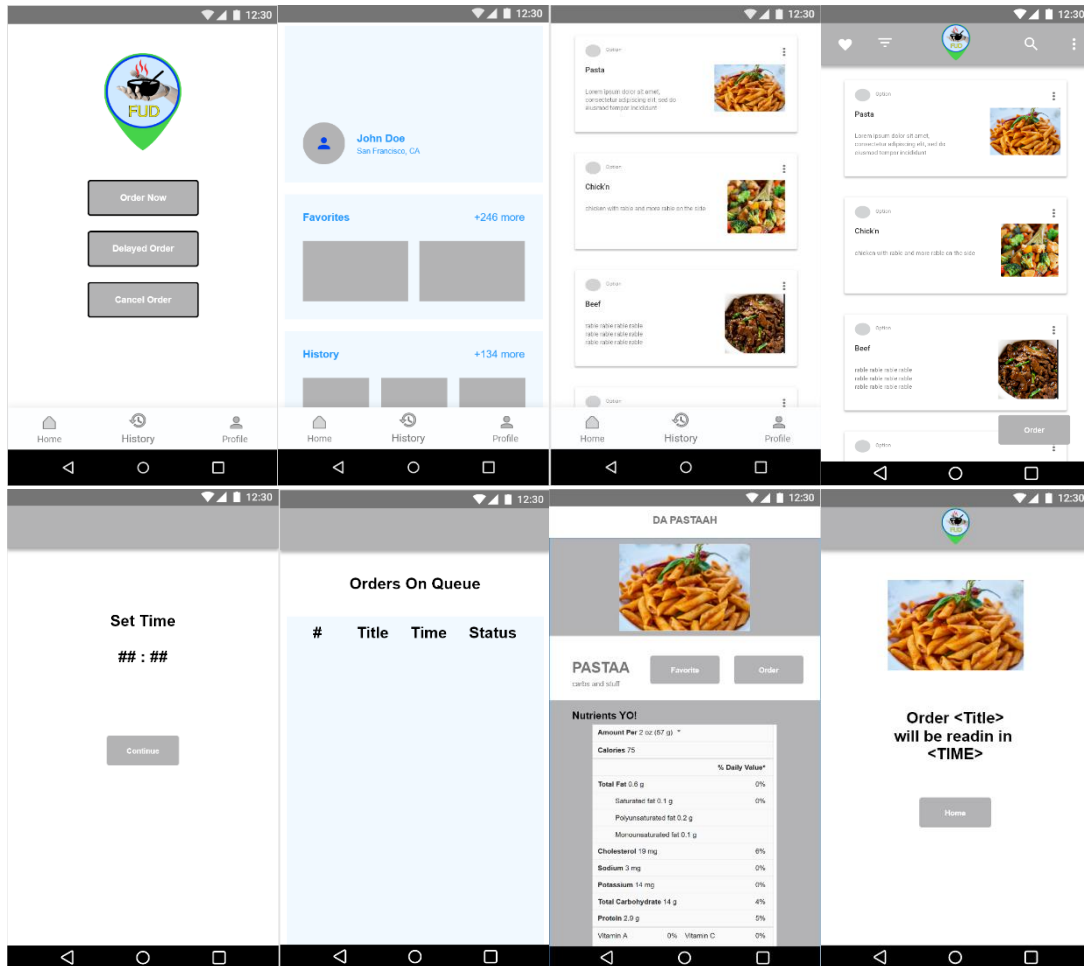


Figure 2. Screenshots of current app progress.



FUD APP FLOWCHART

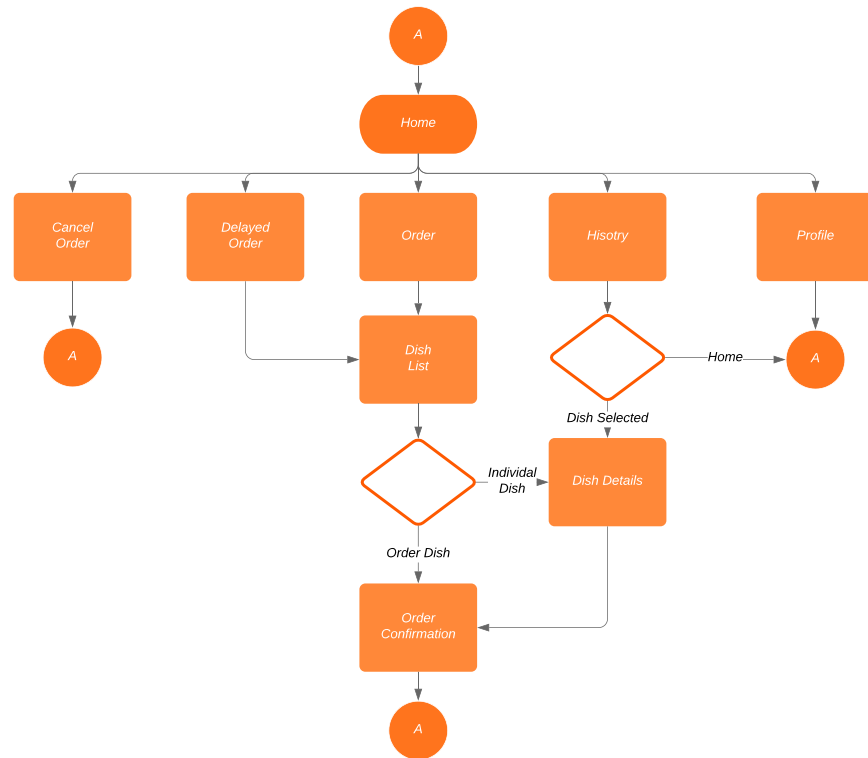


Figure 3. App Flowchart.



2.2 The Food Maker

The food maker's main goal is to cook the food. This is done by housing the ingredients in a fridge then transporting the food to the cooker, where the food is cooked and placed in a bowl for the user to grab and eat.

2.2.1 The Fridge

The fridge will house the bowls that are filled with ingredients and ensure that the food ingredients do not spoil quickly. The fridge design generated on Fusion 360 is shown below.

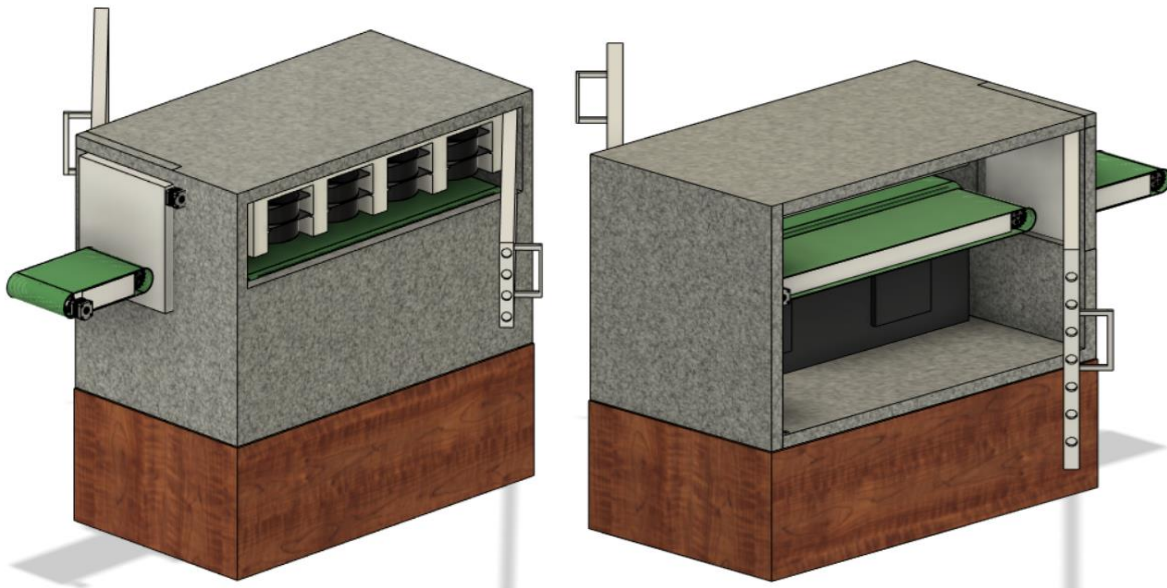


Figure 4. Fusion 360 fridge model

The fridge is composed of different components, and these components are shown below.



2.2.1.1 Conveyor Belt



Figure 5. Conveyor belt parts (left and middle) and finished component (right).

Purpose: Transport Bowls from the bowl holder to the food cooker.

Description: Motor will turn the axel, which holds the gears that latch to the tank treads. The tank treads once rotating will transport the bowl from the bowl holder to food cooker.

Part/Material	Purpose
7in Axels/Shfts	Aligns and acts as the backbone of conveyor belt.
Nema 17 Bipolar Motors	Rotates the axels which provide motion to the conveyor belt. Only 1 motor will be required since motors can provide up to 15 pounds of torque which is enough to transport 1-pound bowls.
Tank Treads	Will provide the grip needed to transport bowls and will be wrapped around the 18 tooth gears.
Shaft Collars	Hold the 18 tooth gears in place, thus preventing gear from shifting on axel.
5mm to 8mm Couplings	Help connect motor axel (cylinder face) to the 7in axels (rectangular face)
Support Block	Prevent tank treads from sagging and helps loosen tension on axels. Too much tension can cause axels to bend.
18 Tooth High Strength Gear	Tooth will latch to tank treads causing tank treads to move.

2.2.1.2 Bowl Holder

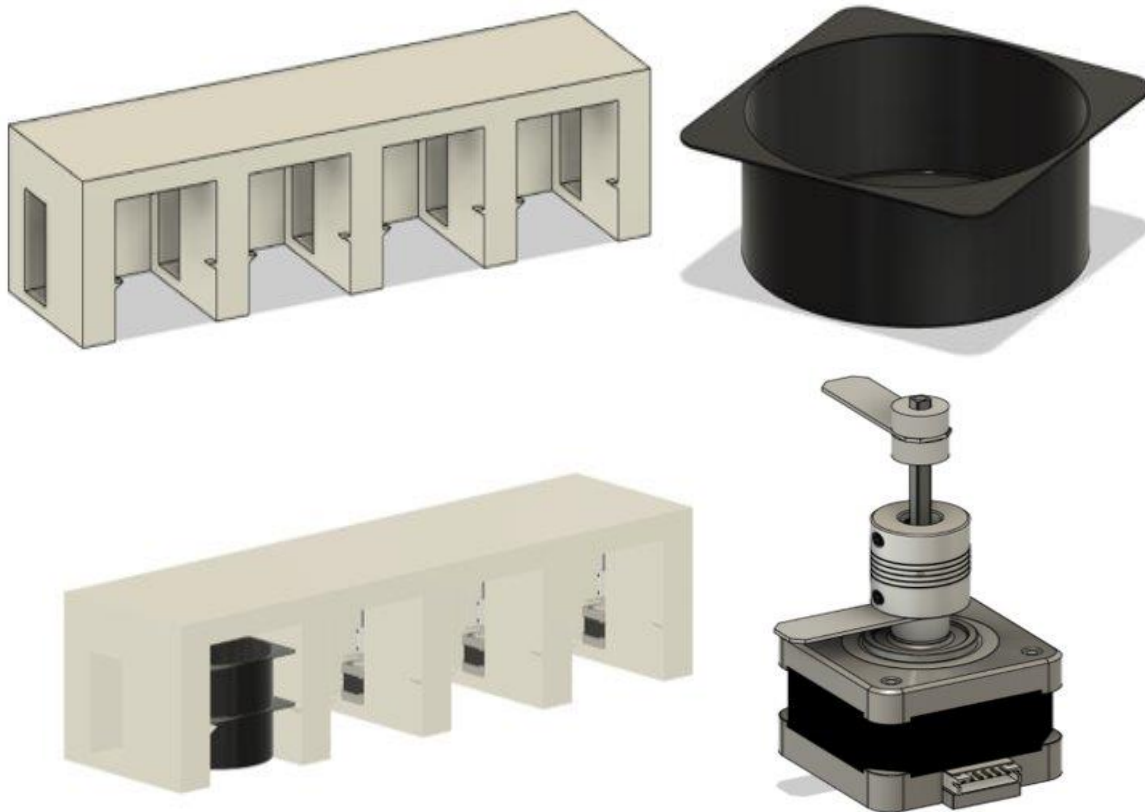


Figure 6. Bowl holder frame (top-left), bowl (top-right), motor with flaps (bottom-right) and finished component (bottom-left).

Purpose: Holds and dispenses 1 bowl at a time.

Description: The lower flap holds the first bowl. Once the motor rotates 90 degrees counterclockwise, the lower flap releases the first bowl while the upper flap holds the second bowl in place. After a couple of seconds, the motor returns to original place, causing the upper flap to release the second bowl, and is caught by the lower flap.

Part/Material	Purpose
2 in Axel/Shaft	Aligns parts.
Nema 17 Bipolar Motors	Rotates axel and will allow flaps to move.
Flaps	Will hold a corner of the bowl thus preventing them from dropping.
Shaft Collars	Hold the 18 flaps in place, thus preventing flaps from shifting on axel.
5mm to 8mm Couplings	Help connect motor axel (cylinder face) to the 2in axels (rectangular face)
6x2 inch Bowls	House the ingredients of the food.



2.2.1.3 Fridge Doors

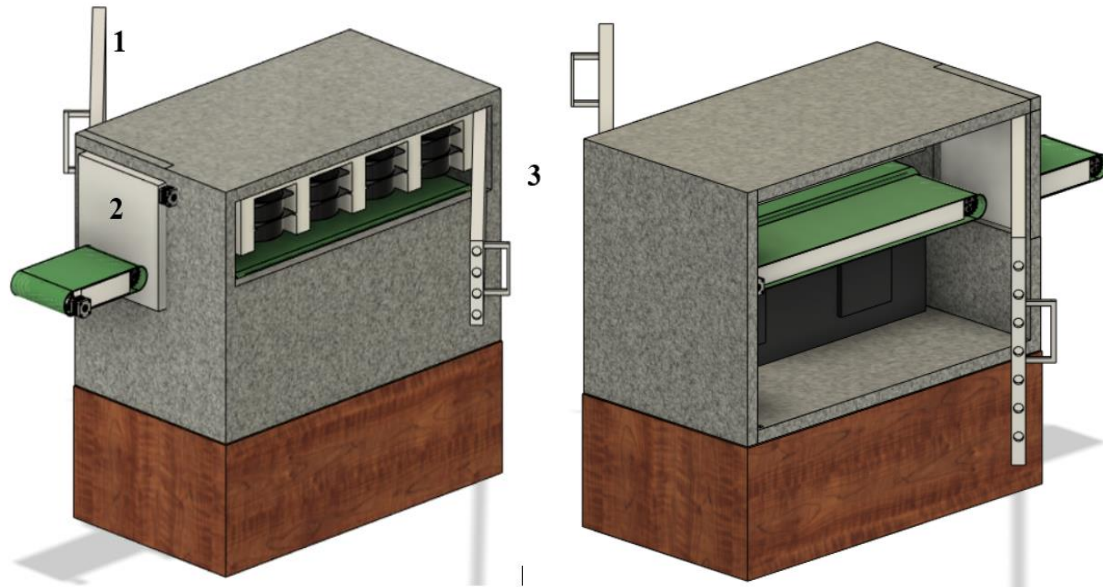


Figure 7. Fridge doors open.

Purpose: Bowls can enter and exit the fridge.

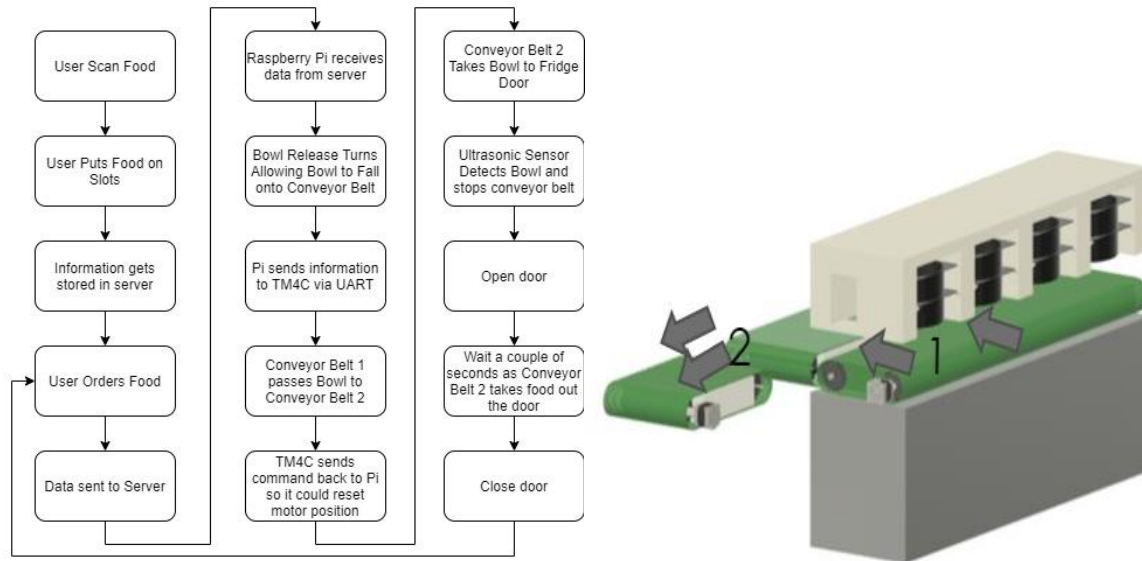
Description: The fridge will be composed of 3 doors. The biggest door (1) will allow the user and designers to debug or accesses the interior of the fridge. The longest door (3) will allow the user to insert/remove the bowls. The last door (2) will allow the bowls exit the fridge while traveling on the conveyor belt.

Part/Material	Purpose
Door Hinges	Connect the doors to the cover of the fridge and will allow the doors to shut and close.
1/5 th in metal + 1-inch insulation + 1/5 th in metal	The overall layering of the fridge doors. The 1/5 th in metal will provide a sturdy inside and outside frame for the fridge doors, and the insulation will help slow rate of heat transferring into the fridge.
1-inch circle magnets	Placed on the sides of the door and on the sides of the fridge, so when the door closes, the door snaps into place, thus preventing door from opening by itself.
Cabinet Handles	Handles so doors can be opened and closed.



2.2.1.4 Fridge Design Flow

With the design and components, figure 8 shows how each component will work together in order to take the bowl from the holder to outside the fridge.



8. Design flow and fridge component model.

Figure



2.2.2 The Cooker

The Cooker will take the ingredients from the bowls and then cook them using the induction.



Figure 9. Fusion 360 cooker model.

The cooker is composed of different components, and these components are shown below.

2.2.2.1 Cook Stands



Figure 10. Fusion 360 cook stands model.

Purpose: Supports cooking components such as the cooking pot and induction cooker.

Description: Holds the induction cooker via Velcro, and provides the support needed for the cooking pot.

Part/Material	Purpose
Wood	Provides a sturdy material for the stands, and it is easy to cut and buy when editing the overall design of the stand.
Velcro	Used to attach the induction cooker to the cook stands.



2.2.2.2 Potholder and Cooking Mechanism

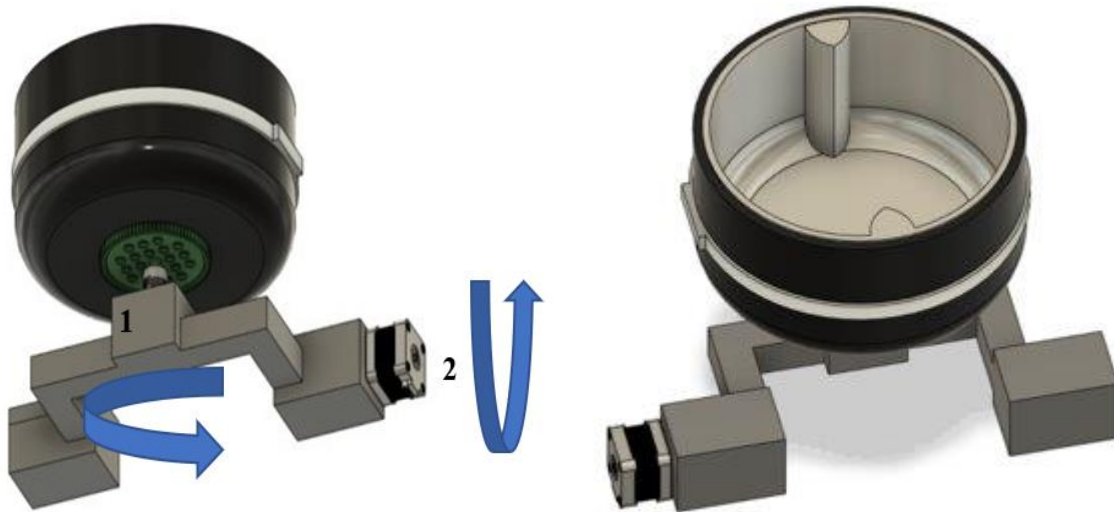


Figure 11. Fusion 360 potholder and cooking mechanism models

Purpose: Mixes and cooks the ingredients received from the bowls.

Description: The cooking pot with wedges will be rotated by motor 1, while motor 2 will tilt the cooking pot to a 45-degree angles. At an angle, the wedges will be to mix the ingredients.

Part/Material	Purpose
Hose Duct Clamps	Clamps flexible insulation sheet with cooking pot.
Nema 17 Bipolar Motors	Rotates axel causes the cooking pot to rotate.
Flexible Insulation Sheet	Ensure the heat from cooking pot does not transfer to surrounding parts/materials.
Magnetic Stainless-Steel Pot	Will hold and mix ingredients. Magnetic material ensures the induction cooker can heat up the pot.
5mm to 8mm Couplings	Help connect motor axel (cylinder face) to the 1in axels (rectangular face)
84 Tooth High Strength Gear	Connects the flexible insulation sheets to the motor via screws.

2.2.2.3 Induction Cooker



Figure 12. Fusion 360 induction cooker model (left) and Avalon Bay Cooker (right).

Purpose: Heats up the magnetic cooking pot

Description: As the cooking pot rotates, the induction cooker will heat up the sides of the bowl. The sides of the bowls will cook the ingredients due to the cooking pot being angled.

Part/Material	Purpose
Avalon Bay Induction Cooker	Heats up the sides of the bowl.



Technical Description: The induction cooking method was researched to be more efficient compared to an open flame, and a hot plate. It is also safer since no flames will be present, and the surface of the induction cooker will not heat up, but the cooking pot will. Induction cooking is performed by having a coil of copper wire, and then passing alternating current through the coil. The coil produces a magnetic field, which induces an electric current through the pot. This current flows through the pot which has a resistance, the resistance results in resistive heating.

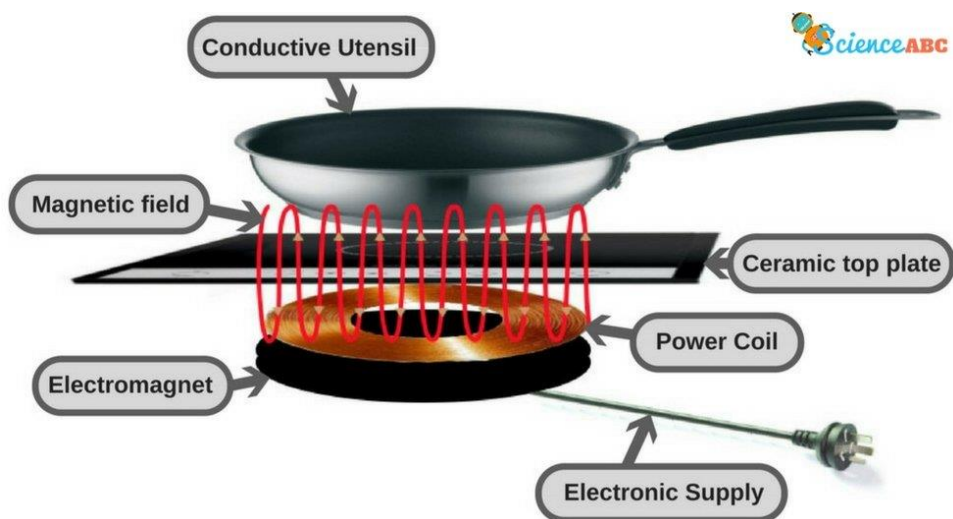


Figure 13. Induction Cooker Image

(<https://www.scienceabc.com/innovation/how-does-an-induction-cooktop-work.html>)

2.2.2.4 Claw

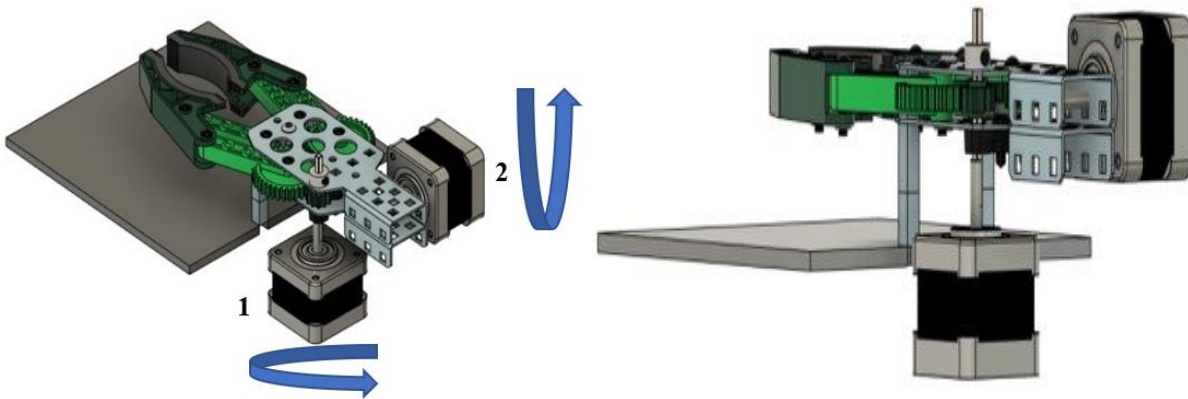


Figure 14. Fusion 360 claw model.

Purpose: Catches bowl and dumps ingredients into cooking pot.

Description: The claw is open when the bowl gets dropped onto the claw. The bowl is caught by the platform. The claw closes with motor 1 and causes the claw to grip the bowl. Motor 2 rotates and causes the claw to dip down and drop the ingredients to the cooking pot.

Part/Material	Purpose
32Tooth High Strength Gear	Connects to the claw and causes the claw to close when gears rotate.
12Tooth High Strength Gear	Connect to the motor and will turn the 32-tooth gear.
Nema 17 Bipolar Motors	Cause the claw to close or dip.
Shaft Collars	Will hold the axels in place when clamped to the metal sheets.
2in Axels	Will align 12 tooth gear with motor.
Metal Sheets	Provide metal structure to claw.
3D Printed Claw	Used to grip the bowl.



2.2.2.5 Cooker Design Flow

With the design and components, Figure 15 shows how each component will work together in order to cook the ingredients.

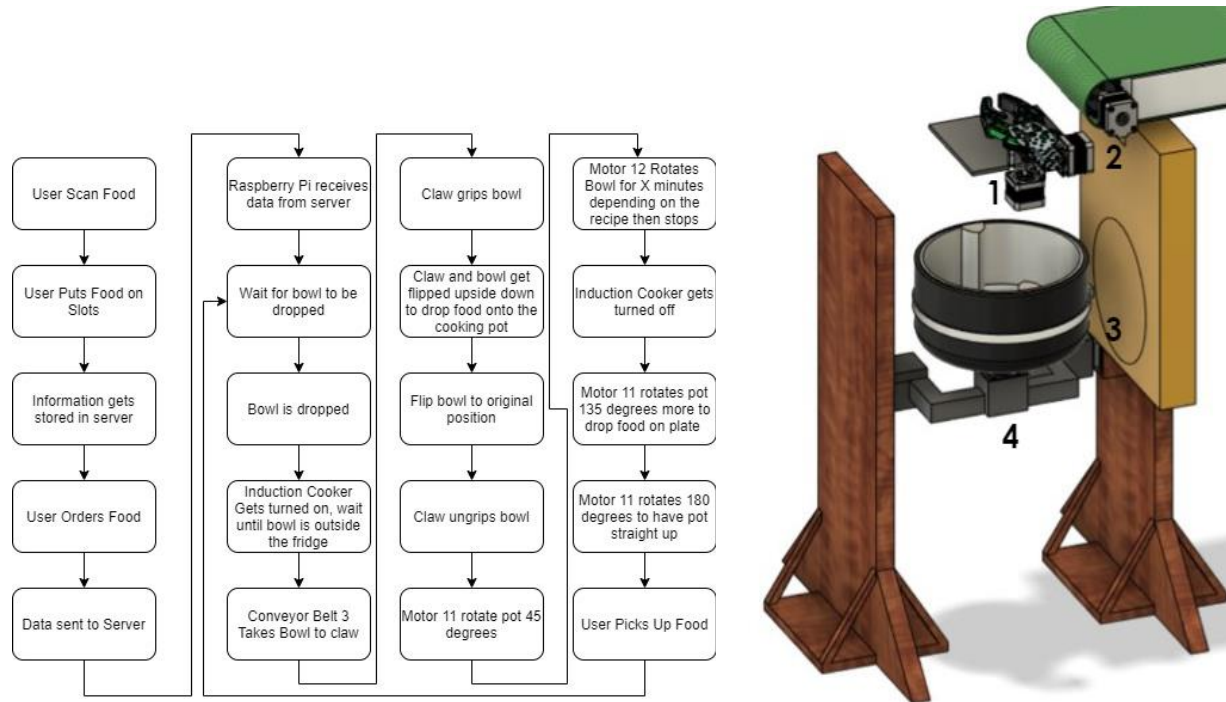


Figure 15. Cooker design flow (left) and fusion 360 cooker model (right).

2.2.3 Combined Design

With all the components of the food maker shown. Figure 16 shows the Fusion 360 image how each component is connected to each other.

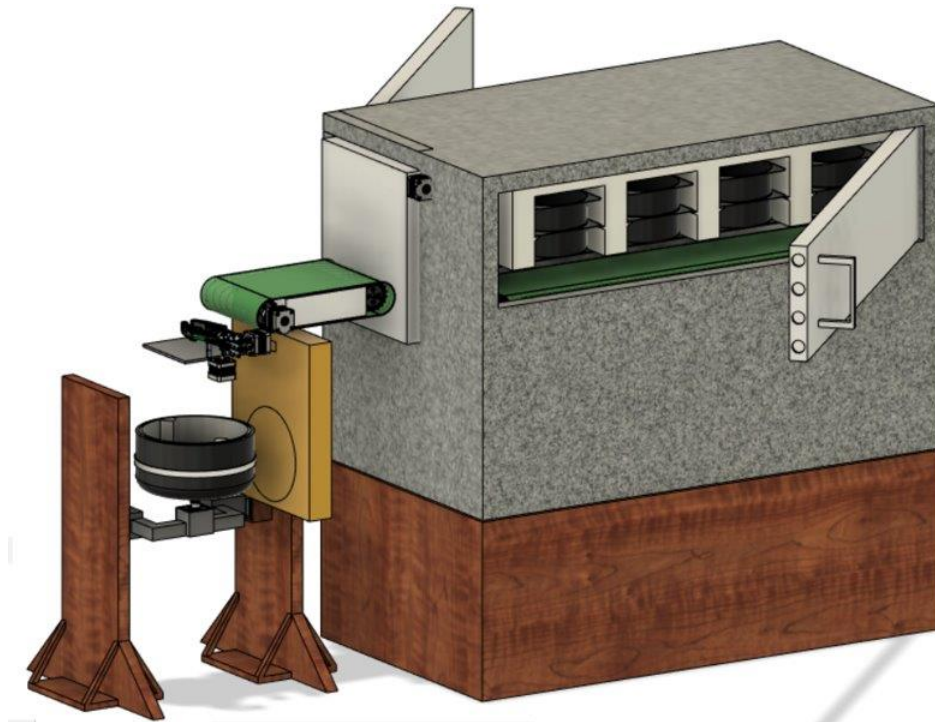


Figure 16. Complete food cooker design.

2.2.3 Circuit Schematics

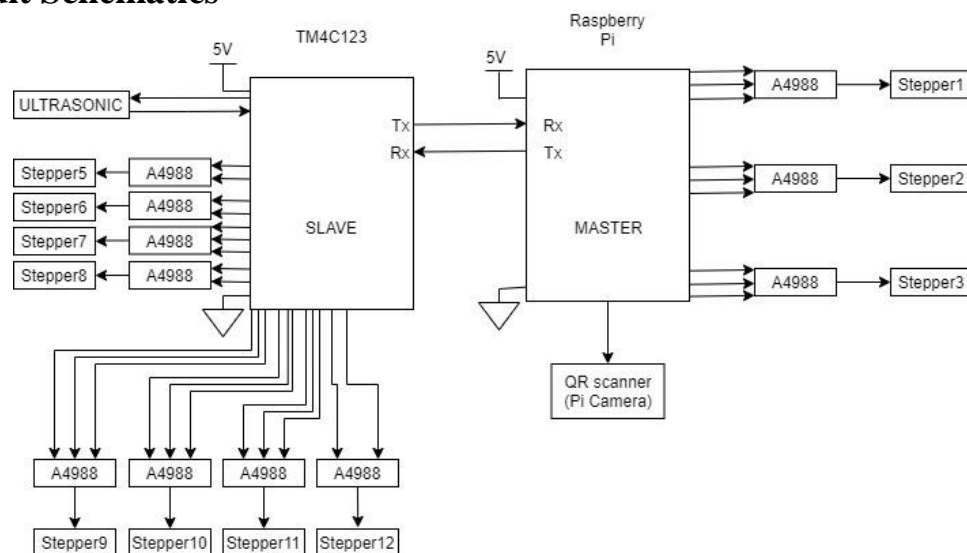




Figure 17. shows an overall block diagram on the connectivity for electronic components

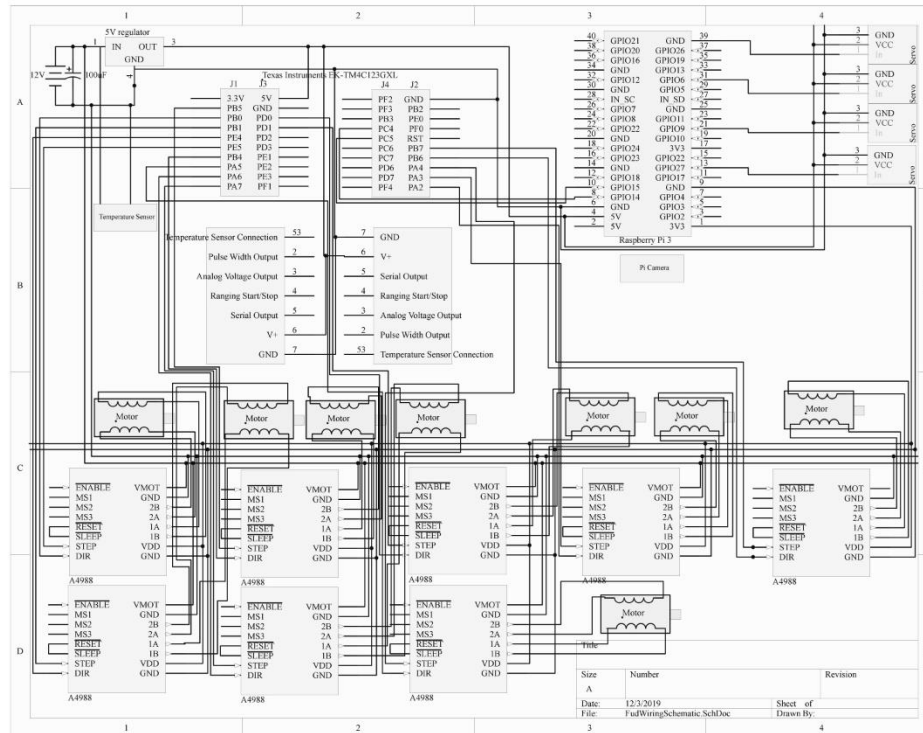


Figure 18. Full schematic of connections of microcontrollers to motors and sensors

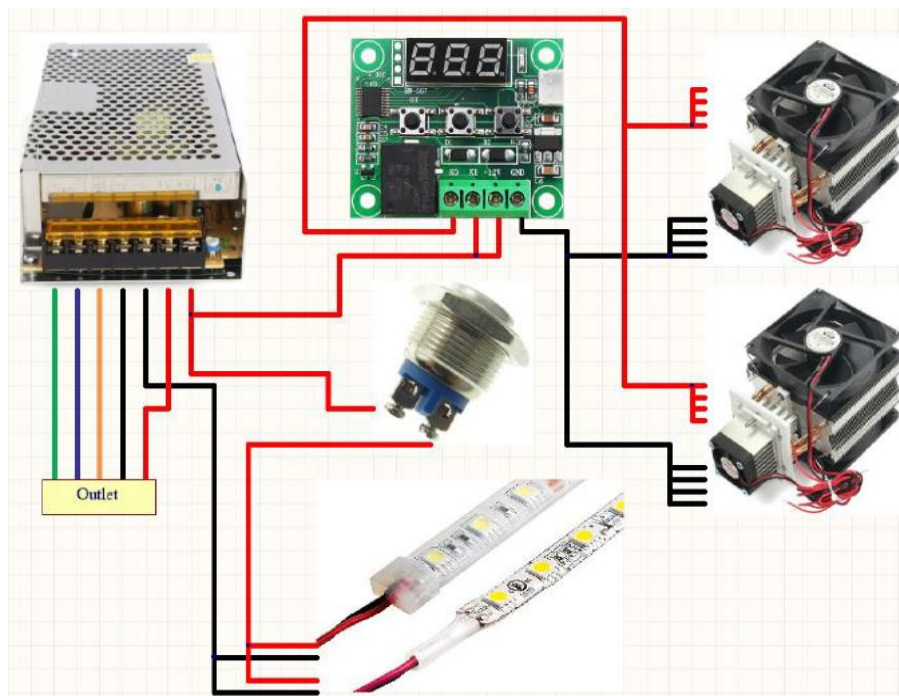




Figure 19. Full schematic of connections of fridge.

Even though the fridge schematic is created. We are currently also looking into getting the fridge cooling components from a mini fridge. This will provide us with an alternative method of cooling the fridge. Figures 20 show the progress made into obtaining the fridge cooling system.



Figure 20. Cooling system extraction progress.



2.3 Web Server

2.3.1 Database

The database contains information of the user and the meals that will be used for the cooker and the mobile application. The meals are still to be determined as we need to do trial and error to see which meals are going to be used. The profile only saves the basic information: Name, History meals, and favorites, anything more and we must consider encrypting user information (Refer to section 5 for ethical implications we must provide).

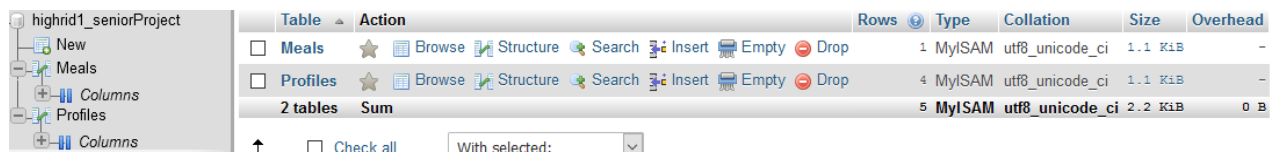
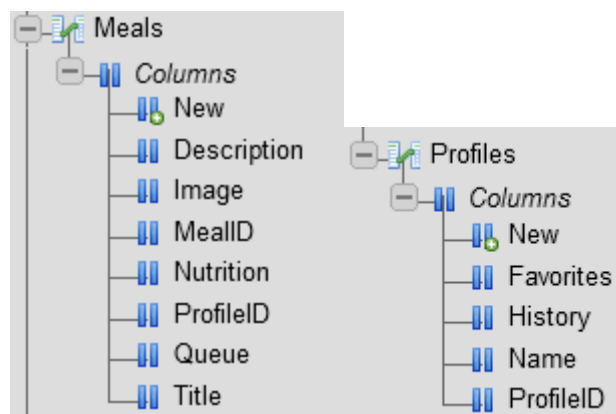


Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> Meals	★ Browse Structure Search Insert Empty Drop	1	MyISAM	utf8_unicode_ci	1.1 KiB	-
<input type="checkbox"/> Profiles	★ Browse Structure Search Insert Empty Drop	4	MyISAM	utf8_unicode_ci	1.1 KiB	-
2 tables	Sum	5	MyISAM	utf8_unicode_ci	2.2 KiB	0 B



Meals require basic dietary information, description of the meal, and an image of the finish meal. Along with details when it should be sent to the cooker, whether the user wanted a delay or not. Profile saves information to be displayed in the “Profile” section on the mobile application. The details of the meals will be displayed under the “Meal” section of the mobile application.

ProfileID	Name	History	Favorites
0	Dennis Vivanco	0,31,99,31,31,31,31,31	5,1,2,3
1	Bruce	1,3,3,3,1,6	3

ProfileID	MealID	Queue	Title	Description	Nutrition	Image
1	1	0	Pasta	pasta is food	some OG nutrition yo!	http://highridgeroofting.com/SeniorProject/pasta.jp



2.3.2 API

The API will be delivering the information of the user to the mobile application. The API connects to the database in the server and either display the user information or updates/inserts new user information. The meals are also displayed through this method. The API is created with PHP and SQL commands so the database can display the correct information.

```
<?php
$servername = "localhost";
$username = "*****";
$password = "*****";
$dbname = "highrid1_seniorProject";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

Connecting to the database

```
if ( $_SERVER['REQUEST_METHOD'] == "GET" ) {
    $name = $_GET['Name'];
    // -----
    // Display records
    // -----
    $sql = "SELECT * FROM Profiles WHERE Name='".$name."' ";
    $select_results = $conn->query($sql);

    if ( $select_results->num_rows > 0 ) {
        // output data of each row
        while ( $row = $select_results->fetch_assoc() ) {
            echo $row["Name"] . "|" . $row["History"] . "|" . $row["Favorites"] . "<br>";
        }
    } else {
        echo "0 results";
    }
    http_response_code(200);
    $conn->close();
}
```

Retrieve information about the user and display them accordingly so the mobile application can parse it

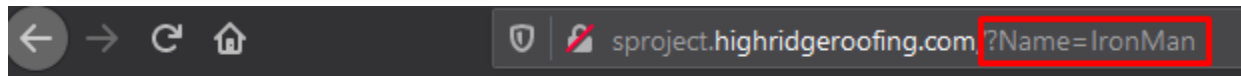


```
} elseif ( $_SERVER['REQUEST_METHOD'] == "POST" ) {  
    // -----  
    // Update record  
    // -----  
    if ($mode == 1) {  
        $sql = "UPDATE Profiles SET History=concat(History,','31') WHERE ProfileID = 0";  
  
        if ($conn->query($sql) === TRUE) {  
            echo "Record updated successfully";  
        } else {  
            echo "Error updating record: " . $conn->error;  
        }  
    }  
    // -----  
    // INSERT NEW record  
    // -----  
    elseif ( $mode == 2 ) {  
        // prepare and bind  
        $stmt = $conn->prepare("INSERT INTO Profiles (ProfileID, Name, History, Favorites) VALUES (?, ?, ?, ?)");  
        $stmt->bind_param("isss", $ProfileID, $Name, $History, $Favorites);  
  
        // set parameters and execute  
        $ProfileID = "33";  
        $Name = "Pikachu";  
        $History = "5,5,1,2";  
        $Favorites = "5";  
        $stmt->execute();  
  
        echo "New records created successfully";  
  
        $stmt->close();  
        $conn->close();  
    }  
    else {  
        echo "incorrect mode option";  
    }  
} else {  
    http_response_code(405);  
}
```

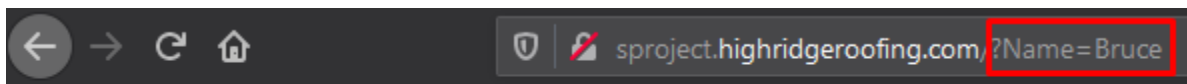
Updating or inserting new information to the database according to user profile



All attempts to update or request information from the database returns the proper response codes and connection to the database is always closed after being used. If the user is not in the database the API will display no information, but if there is the API returns the necessary data to be parse by the mobile application.



0 results



Bruce|1,3,3,3,1,6|3

If user is in the database the information will be displayed in this order: Name, History of meals, and favorite meals. This information can be displayed in the mobile application with the meals corresponding information (current data displayed is only for testing purposes and does not correspond to any meal).



3 Verification

Fall 2019:

We tested the Nema 17 Stepper Motor and the A4988 Motor Driver to observe how powerful the motor is and how the TM4C123 microcontroller interacts with it. Each Motor requires a certain amount of voltage and current in order to operate correctly since the Nema 17 has a variety of models. The datasheet contains the required voltage and current. The A4988 motor driver has a built-in potentiometer that limits the current going into the motor. For our motor, we noticed we needed a voltage of 3.6V and current of 1A. Before connecting the motor to the driver, we need to confirm the current going out of the driver is 1A. To do that we find the Reference Voltage using the formula $V_{ref} = \text{Rated motor current} * 8 * R_{sense}$. Different manufacturers have different R_{sense} values which the datasheet indicate. We did our calculations and our V_{ref} turned out to be 0.4V. However, we need to provide a 10% safety margin to not burn out any component. Thus $0.4 * 1 = .04$ and the final V_{ref} value is $0.4 - .04 = 0.36V$. To adjust the potentiometer to this value first put the positive probe of our multimeter to the potentiometer and the ground probe to ground on the bottom of VDD. Twist the potentiometer with a flathead screw to adjust the V_{ref} , which basically adjusts the current. We applied a 12V power supply to the motor since a higher voltage gives us higher step rates and a 100uF cap to smoothen out the voltage signal. The microcontroller sends the Step and Direction signal to the Driver. Motor driver requires a voltage of 3.3-5V and 5mA current which the microcontroller can provide so we hooked those wires up together to VDD. The rest were left unconnected since we wanted a Full Step Microstep Resolution. We connected the step from two A4988 to PC4 and PC6 and the DIR from the A4988 to PC5 and PC7. Both motors were continuously going one way drawing a total current of 0.22A. We do have a strange outcome when both motors are turned off drawing a current of 0.27A which at this point there is a noise coming out from one of the motors. We will need to investigate why we are obtaining this current and noise. If we apply a finger as the load on both motors the current goes up to around 0.31A. This load is not that useful since we need a real load to determine the total real current drawn from it.

MS1	MS2	MS3	Microstep Resolution	Excitation Mode
L	L	L	Full Step	2 Phase
H	L	L	Half Step	1-2 Phase
L	H	L	Quarter Step	W1-2 Phase
H	H	L	Eighth Step	2W1-2 Phase
H	H	H	Sixteenth Step	4W1-2 Phase

Figure 21: Microstepping Resolution Truth Table

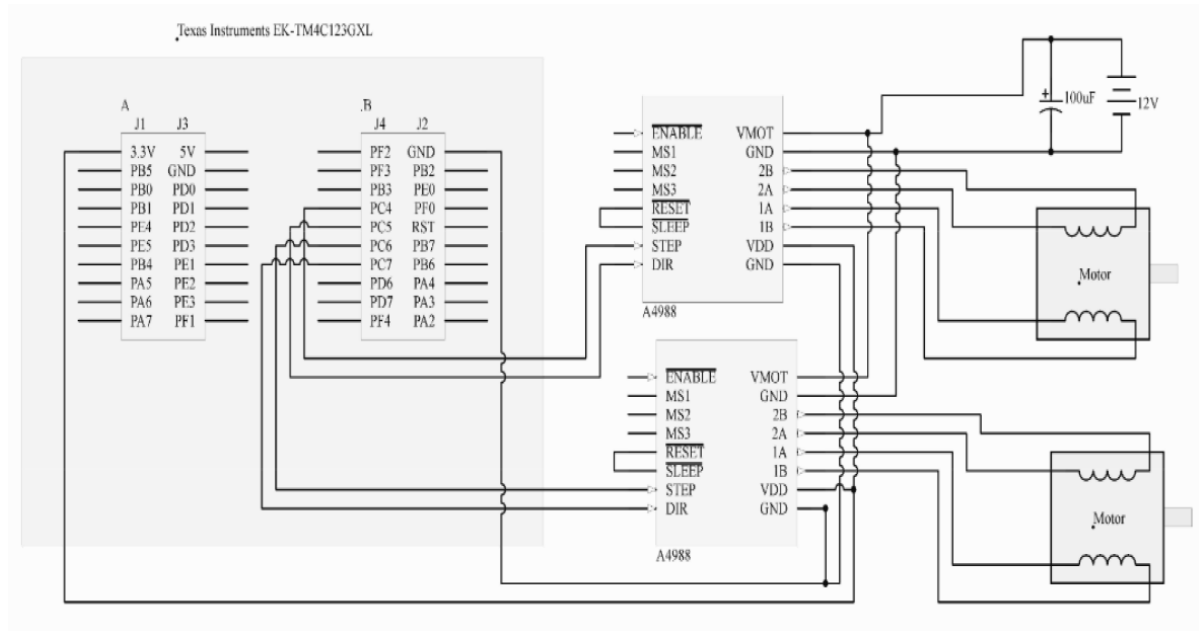


Figure 22: Schematic for two NEMA 17, two A4988, and TM4C connection

Our next test included making the two motors stop or rotate one direction with UART. We used UART0 within the TM4C123 to do a USB serial communication. The same schematic as above was used since UART0 is connected to PA0 and PA1 within the TM4C which the USB port uses. In our laptop we downloaded a program called RealTerm. Within the TM4C we configured our UART to be 115200 baud rate, 8 bits, no parity, 1 stop bit. We setup RealTerm to the same configurations. Within RealTerm we send the character 'r' or 'g'. 'r' turns the Red on-board LED on, and the two motors do not spin. If a 'g' is sent, the green on-board LED turns on and the motors continuously move 1 direction. We switch our clocks frequency from 16Mhz to 80Mhz to make the data transfer more accurate. We did not have time to set up the UART on the Raspberry Pi to make this exact functionality happen, however during the winter break we will get together and keep working on any tests we did not do this semester. We will make the Pi output the character 'g' and 'r' in certain time intervals as a starting point to make sure they are communicating with each other effectively.

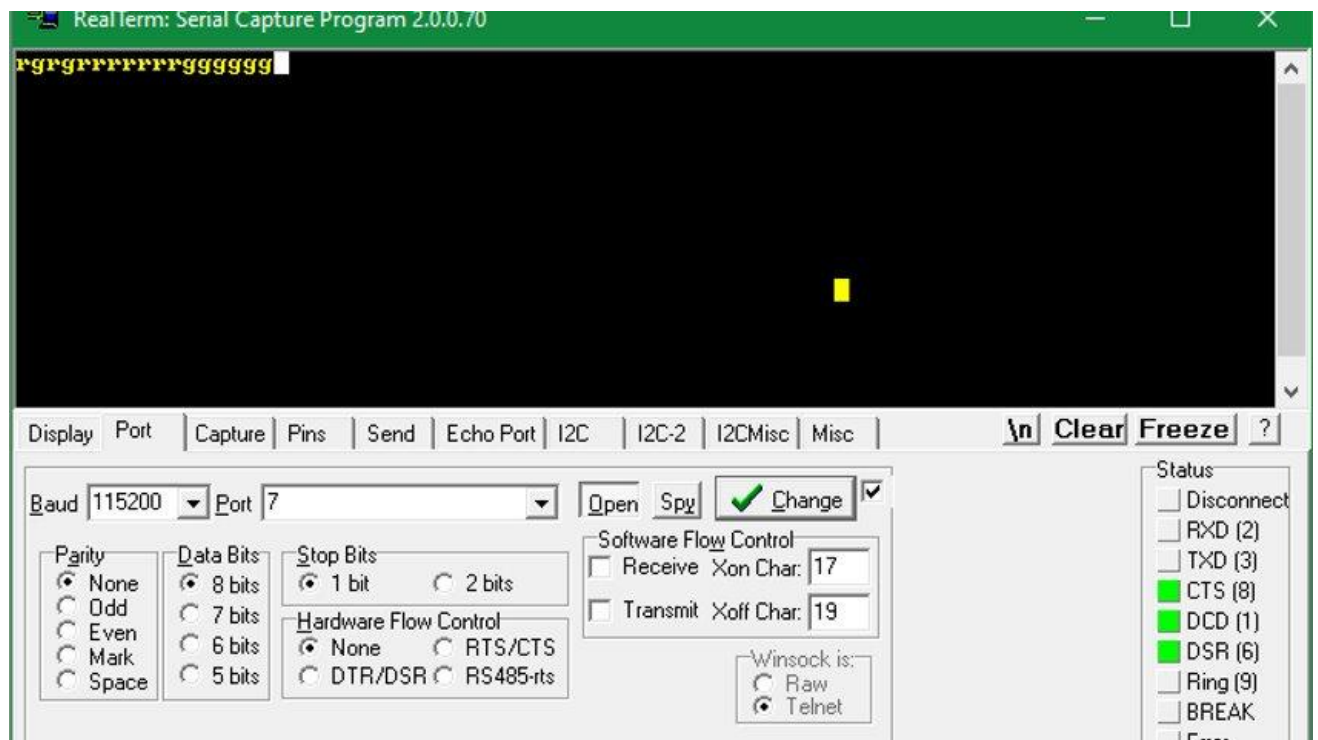


Figure 23: Showing characters inputted and the setup 115200 baud, no parity, 8 bits, 1 stop bit.



Figure 24: Green onboard LED on.



Figure 25: Red onboard LED on.

We proceeded to reducing the voltage from 12V to 5V using a LM7805. These 5V will be feed into the TM4C and Raspberry Pi. With a multimeter, we put the probes on the output of the voltage regulator and ground which read the 5V we wanted.

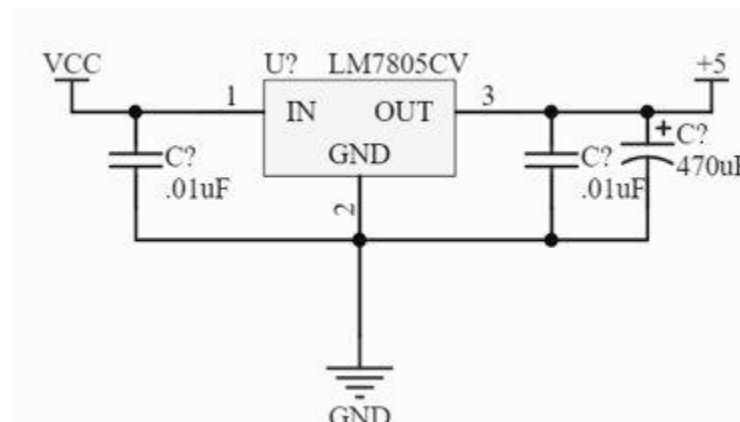


Figure 26: 5V regulator circuit.

Spring 2020:

Each Motor Drivers were configured to 0.36V or lower to provide the current necessary for the motors to move. Extra ports on the TM4C123 were used to connect the ~EN pin from the motor drivers. This pin allows the Motor to be ON or OFF, by default the motor stays on when not connected or when not provided a logic 1. The direction pin on four motor drivers will be connected to ground since all three conveyor belts are going one direction and the cooking pot will also rotates one direction. Using UART0, we tested the movement of each of the motors and adjusted how long it was going to rotate or how many degrees to rotate it. Once we got the motors to mimic the movements, we proceeded to setting up the Pi's serial and GPIO to make three motors move 90 degrees counterclockwise and clockwise. The three motors were put on the Pi due to the limited ports we have available on the TM4C and makes it more efficient since those motors will be the first to move. Both Serial and GPIO were set up with C++ and we used user input to test what motor we want to move either motor 1, 2 or 3. Any other will just turn the motors off with the ~EN pin. Since we knew the TM4C123 and the PI were both working individually we proceeded to combine both by making them communicate via UART. We changed the TM4C from using UART0 to using UART1 so it could obtain outside signals and not use the USB one and sent a string from the Pi to the TM4C containing a key character that starts the cooking process and numbers representing how long we want to cook the food in seconds. We were able to see the LED transition from RED to GREEN along with the movement of the motors meaning the communication and hardware are



working correctly. Each motor performs its operation in series meaning only one motor is moving at a time allowing the system to not consume a lot of current. The connection between the actual motor and driver are a bit weak so the motor stalls at times. This could be fixed by creating a PCB and soldering the header pin, so it is stable. We did a load test on the motors by connecting the pot, conveyor belts, and claw to the motors and see if they work. The test was successful.

The voltage regulator was put into test by attempting to power the TM4C and the Pi individually. The regulator was able to power the TM4C without a problem by connecting it to VCC and GND. The Pi on the other had was not booting up, however a second attempt was made after that day and the Pi booted up. We believe the current outputted by the voltage regulator is not enough since there is a yellow lightning bolt on the upper right-hand screen when booting up the Pi indicating not enough power is being provided.

```
#include <iostream>
#include <wiringPi.h>           // GPIO library
#include <wiringSerial.h>       // serial library

using namespace std;

int main()
{
    int fd;
    fd = serialOpen("/dev/serial0",38400); // opens AMA0 at a baud rate of 38400, 8N1
    wiringPiSetup();                  // set up wiring pi --> mandatory; found in his website
    pinMode(0,OUTPUT);                // CONFIGURE GPIO17 (0 for wiringPI) as output for step
    pinMode(1,OUTPUT);                // GPIO18 is DIR
    pinMode(2,OUTPUT);                // GPIO27 is ~EN (turn on/off motor) --> MOTOR 1
    pinMode(3,OUTPUT);                // GPIO22 STEP
    pinMode(4,OUTPUT);                // GPIO23 DIR
    pinMode(5,OUTPUT);                // GPIO24 ~EN ; MOTOR 2
    pinMode(6,OUTPUT);                // GPIO25 STEP
    pinMode(7,OUTPUT);                // GPIO4 DIR
    pinMode(8,OUTPUT);                // GPIO2 ~EN; MOTOR 3
```

Figure #: Library import, UART and GPIO setup

```
while(1)
{
    int slot;
    cout << "Which slot is the food you desire? ";
    cin >> slot;

    if(slot == 1)
    {
        /*
        CLOCKWISE
        */
```

Figure #: User Input



```

if(slot == 1)
{
    /*
    * CLOCKWISE
    */
    digitalWrite(2,0); // motor on
    digitalWrite(1,0); // motor spins clockwise
    for(int x=0;x<50;x++) // spin 90 degrees
    {
        digitalWrite(0,1); // step
        delay(1); // wait 1ms
        digitalWrite(0,0); // see the stepping transition (1->0)
        delay(1); // wait 1ms
    }
    digitalWrite(2,1); // turn off motor
    delay(1000); // wait 1000 milliseconds
    /*
    * COUNTER-CLOCKWISE
    */
    digitalWrite(2,0); // turn on motor
    digitalWrite(1,1); // motor spins counter-clockwise
    for(int x=0; x<50; x++) // spin 90 degrees
    {
        digitalWrite(0,1); // step
        delay(1); // wait 1ms
        digitalWrite(0,0); // see the stepping transition
        delay(1); // wait 1ms
    }
    digitalWrite(2,1); // turn off motor
    delay(1000); // wait 1000 millisecond
    serialPuts(fd,"f30\r"); // sends string "f30" via UART...TM4C needs to see 'ENTER' aka \r\n to work
}

```

Figure #: Spin Motors then send cooking starting command and seconds to TM4C

```

else
{
    digitalWrite(2,1); // turn off motor
    digitalWrite(5,1); // turn off motor
    digitalWrite(8,1); // turn off motor
}

```

Figure #: Turn off motors

```

void degreeSpin(unsigned char ports, unsigned long volatile motor, unsigned long volatile motor1, unsigned long volatile steps, unsigned long volatile time)
{
    if(ports == 'B'){ // motors attached to this port
        for(x=0;x<steps;x++){ // 360 degrees = 200 steps. Each step goes 1.8 degrees
            GPIO_PORTB_DATA_R = motor; // STEP, DIRECTION and NOT ENABLE value. Dir = 0 (or 1 for the other direction)..Step = 1. NOT EN
            SysTick_Wait(time); // waits a certain amount of time. Ex:1000 microseconds (1000*10^-6) --> (1000*10^-6) /
            GPIO_PORTB_DATA_R = motor1; // motors goes same way. Dir = 0 (or 1)... Step = 0 --> needs to see a transition from 1 to 0 to
            SysTick_Wait(time); // Wait
        }
    }
    else if(ports == 'E'){ // motors attached to this port
        for(x=0;x<steps;x++){ // comment the same as above
            GPIO_PORTB_DATA_R = motor; // STEP, DIRECTION and NOT ENABLE value
            SysTick_Wait(time); // wait
            GPIO_PORTB_DATA_R = motor1; // Should only change STEP value
            SysTick_Wait(time); // wait
        }
    }
}

```

Figure #: Move motors

```

void Delay(unsigned int loops){
    for(i=0;i<loops;i++){
        SysTick_Wait(16777203);
    }
}

```

Figure #: Delays



```
int main(void) {
    PLL_Init();
    PortA_Init();
    PortB_Init();
    PortD_Init();
    PortE_Init();
    UART1_Init();
    PortF_Init();
    EnableInterrupts();
    SysTick_Init();
    while(1) {
        char InStringg[3];
```

Figure #: Initialize Ports

```
while(1) {
    char InStringg[3];
    char TimeInt[2];
    GPIO_PORTF_DATA_R = 0x02;
    GPIO_PORTA_DATA_R = 0x3C;
    GPIO_PORTD_DATA_R = 0x0C;
    GPIO_PORTC_DATA_R = 0x20;
    GPIO_PORTB_DATA_R = 0x02;
    UART1_InString(InStringg, 3);
    for(j=0; j<2; j++)
    {
        TimeInt[j] = InStringg[j+1];
    }
    sec = atoi(TimeInt);
    rot = sec / 0.5;
    rotDeg = 360 * rot;
    degStep = rotDeg / 1.8;
    UART1_OutString(InStringg);
```

Figure #: Stops all motors; Takes in UART string and splits the character plus number.



```
GPIO_PORTC_DATA_R = 0x20;  
degreeSpin('B',0x01,0x00,600,300000);  
GPIO_PORTB_DATA_R = 0x02;  
Delay(2);  
GPIO_PORTD_DATA_R = 0x08;  
degreeSpin('B',0x06,0x02,7200,80000);  
GPIO_PORTD_DATA_R = 0x0C;  
Delay(2);  
GPIO_PORTD_DATA_R = 0x04;  
degreeSpin('B',0x09,0x02,50,80000);  
GPIO_PORTD_DATA_R = 0x08;  
degreeSpin('B',0x06,0x02,1500,80000);  
GPIO_PORTD_DATA_R = 0x04;  
degreeSpin('B',0x1A,0x12,50,80000);  
GPIO_PORTD_DATA_R = 0x0C;
```

Figure #: Turn on and off certain motors from the fridge and cooker. Spins a certain amount of time.



```
GPIO_PORTA_DATA_R = 0x38;
degreeSpin('B',0x22,0x02,7200,80000);
GPIO_PORTA_DATA_R = 0x34;
degreeSpin('B',0x42,0x02,200,80000);
GPIO_PORTA_DATA_R = 0x3C;
Delay(3);
GPIO_PORTA_DATA_R = 0x2C;
degreeSpin('E',0x21,0x20,100,1000000);
GPIO_PORTA_DATA_R = 0x3C;
Delay(3);
GPIO_PORTA_DATA_R = 0x2C;
degreeSpin('E',0x23,0x22,100,1000000);
GPIO_PORTA_DATA_R = 0x3C;
GPIO_PORTA_DATA_R = 0x34;
degreeSpin('B',0xC2,0x82,200,80000);
GPIO_PORTA_DATA_R = 0x3C;
GPIO_PORTA_DATA_R = 0x1C;
degreeSpin('E',0x24,0x20,25,1000000);
GPIO_PORTA_DATA_R = 0x3C;
Delay(3);
degreeSpin('E',0x10,0x00,degStep,100000);
GPIO_PORTA_DATA_R = 0x20;
Delay(3);
GPIO_PORTA_DATA_R = 0x1C;
degreeSpin('E',0x24,0x20,75,1000000);
GPIO_PORTA_DATA_R = 0x3C;
Delay(3);
GPIO_PORTA_DATA_R = 0x1C;
degreeSpin('E',0x2C,0x28,100,1000000);
GPIO_PORTA_DATA_R = 0x3C;
```

Figure #: Turn on and off certain motors from the fridge and cooker. Spins a certain amount of time.

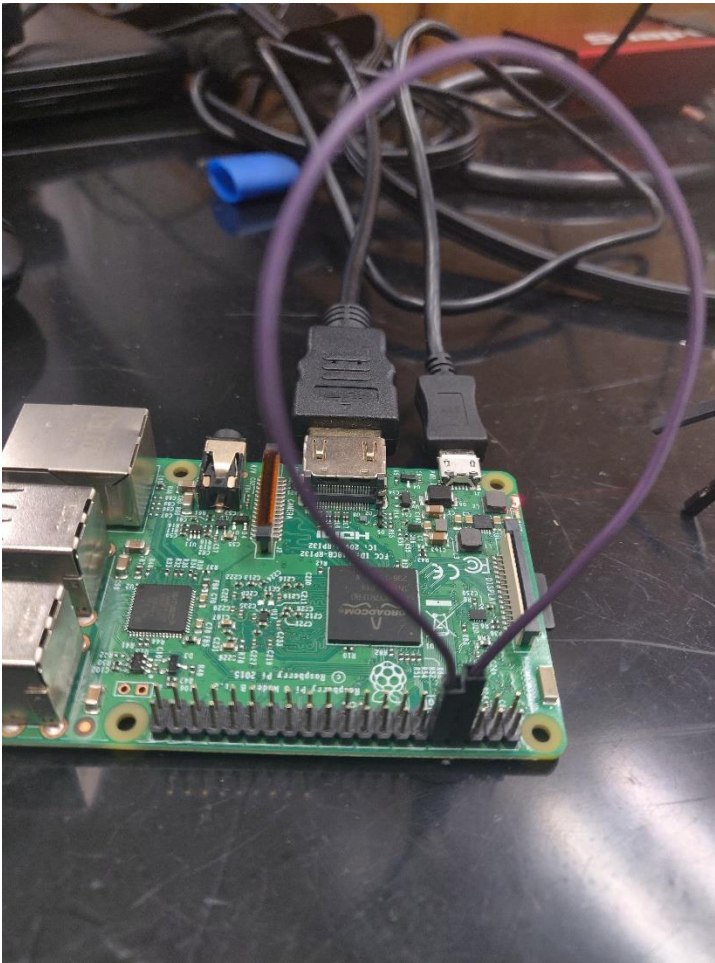


Figure #: TX and RX connected to transmit and receive data for testing purposes

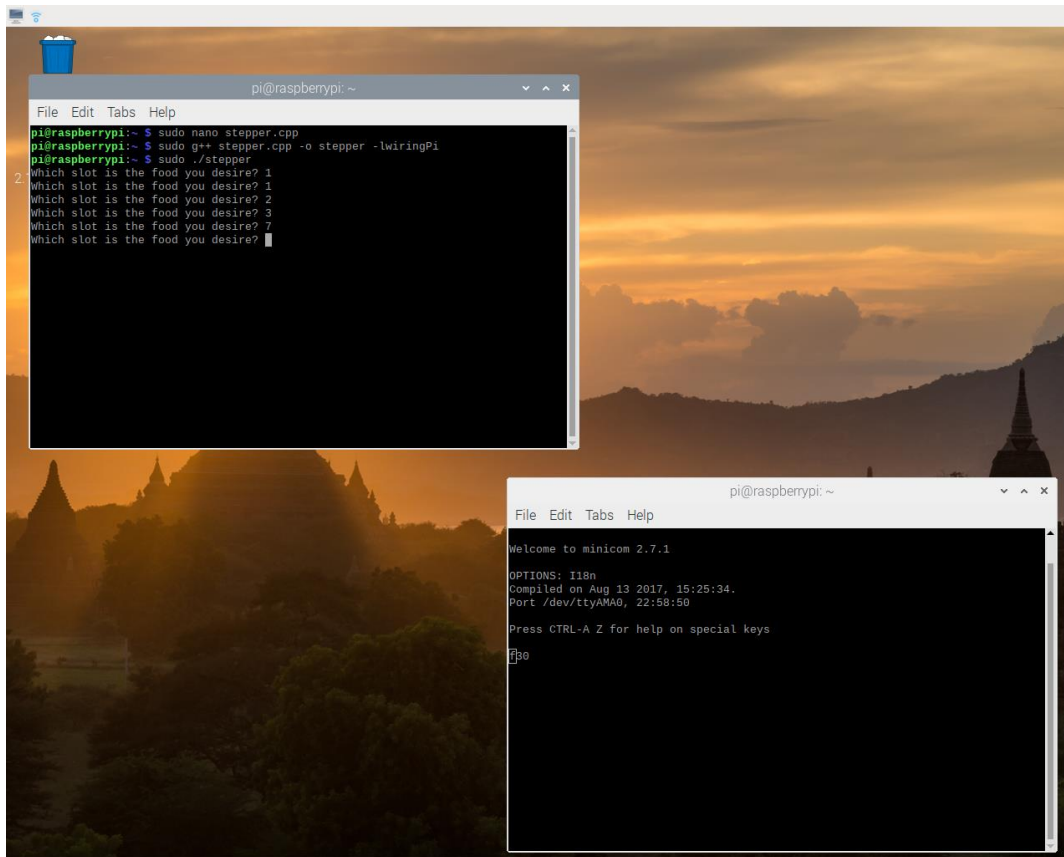


Figure #: UART test with minicom

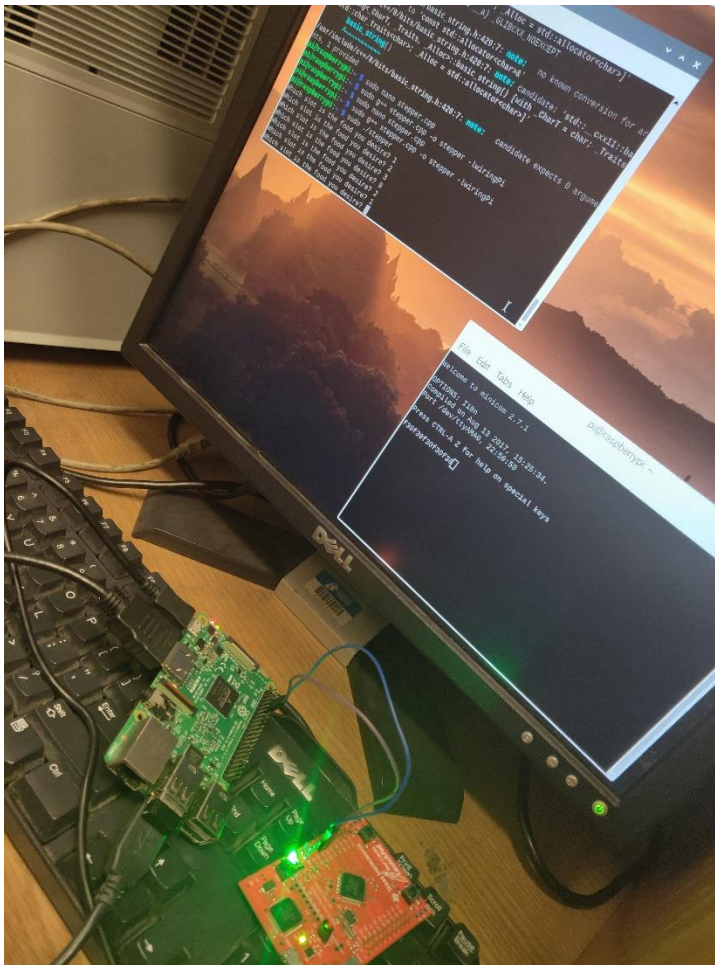


Figure #: Actual TX and RX connected; green LED means food is cooking

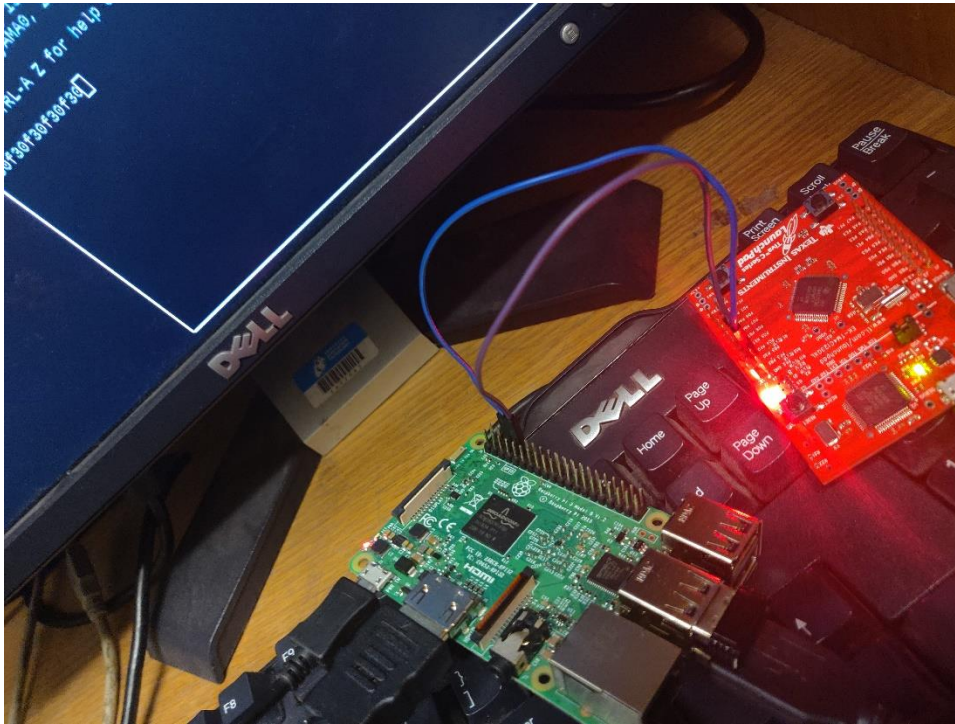


Figure #: Actual TX and RX connected; red LED means food is not cooking

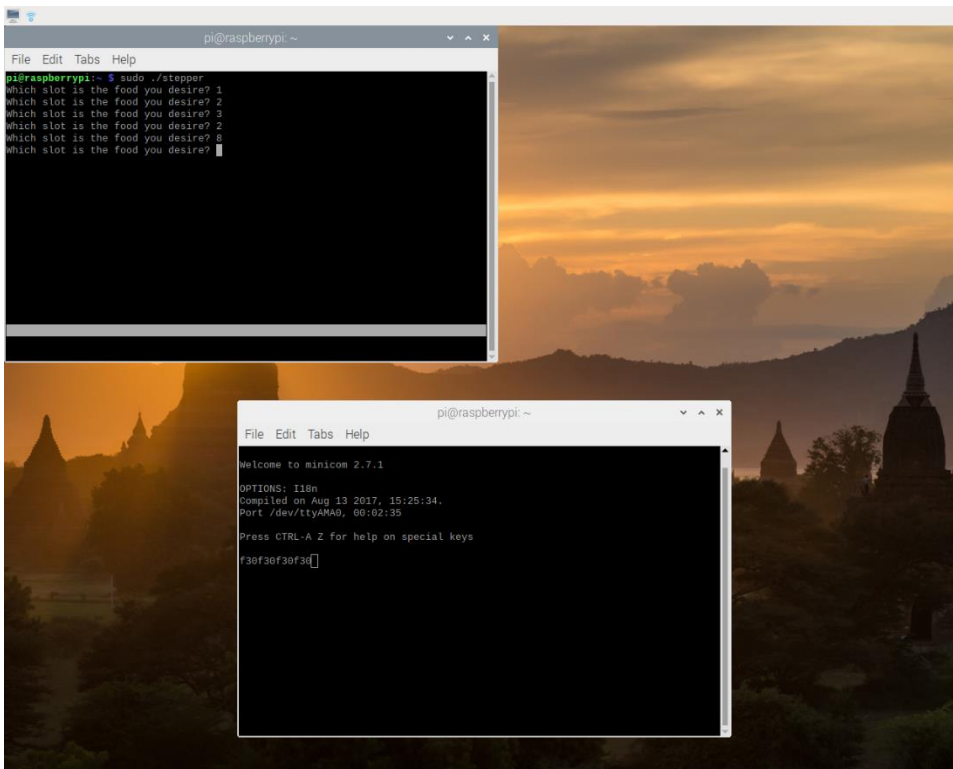




Figure #: UART test with minicom, TM4C sends data it receives back to Pi. Minicom captures it.



Figure #: Motor on Claw

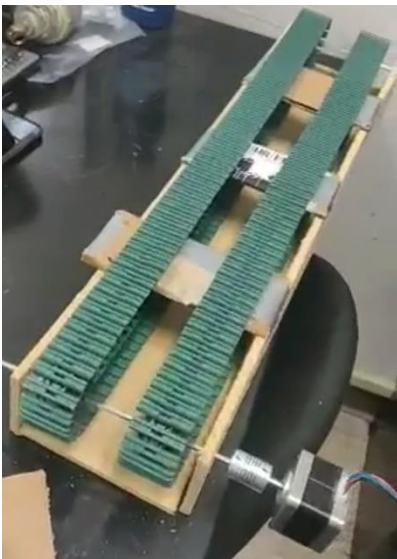
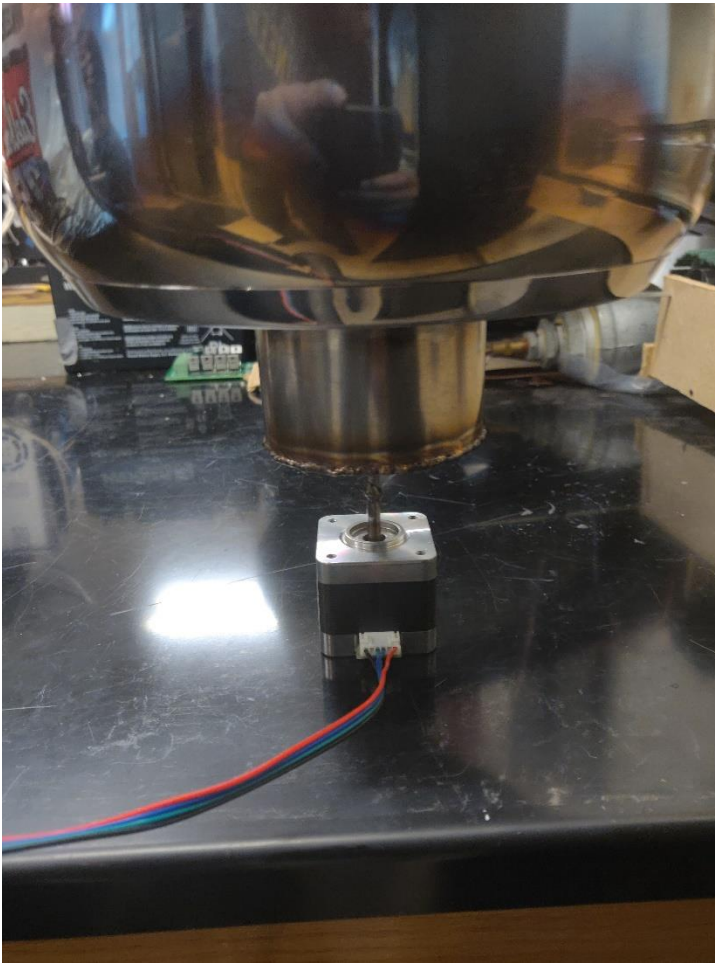


Figure #: Motor on Conveyor Belt



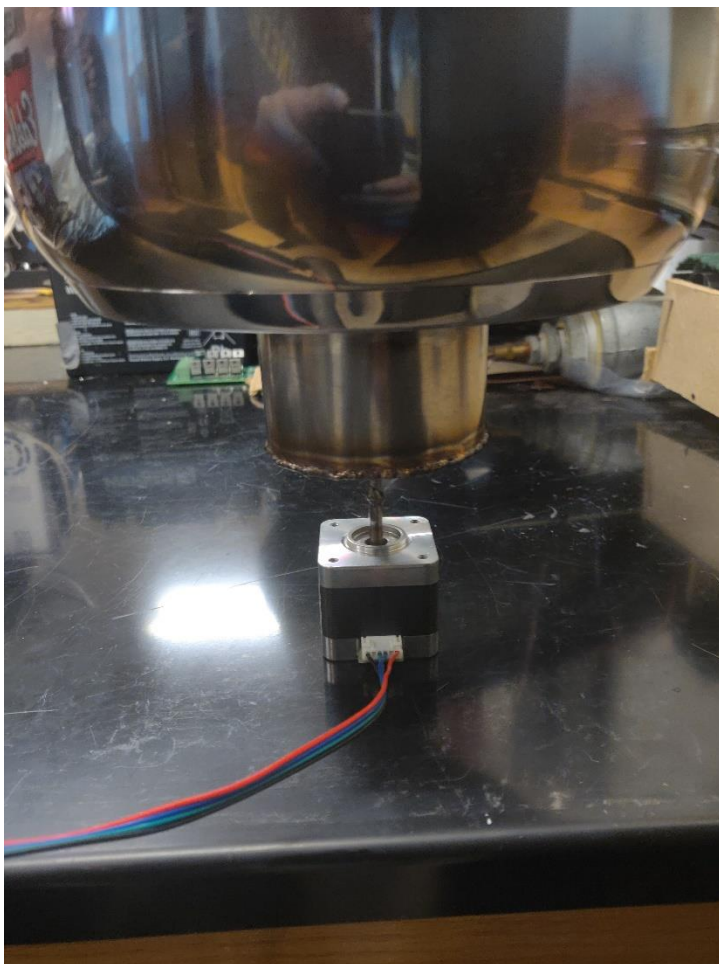
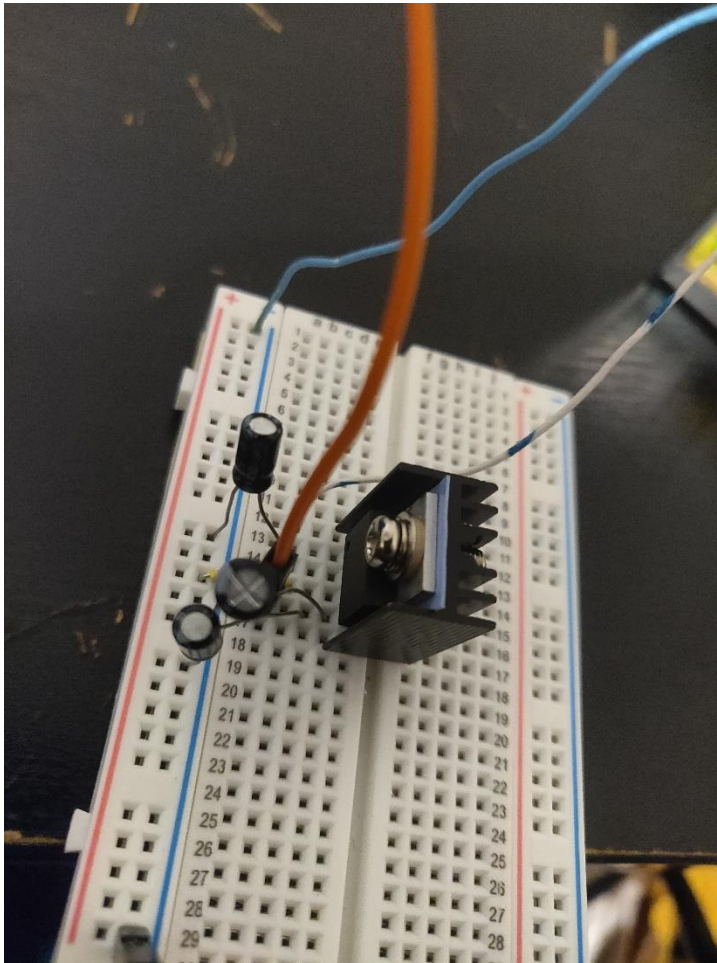


Figure #: Motor on Pot



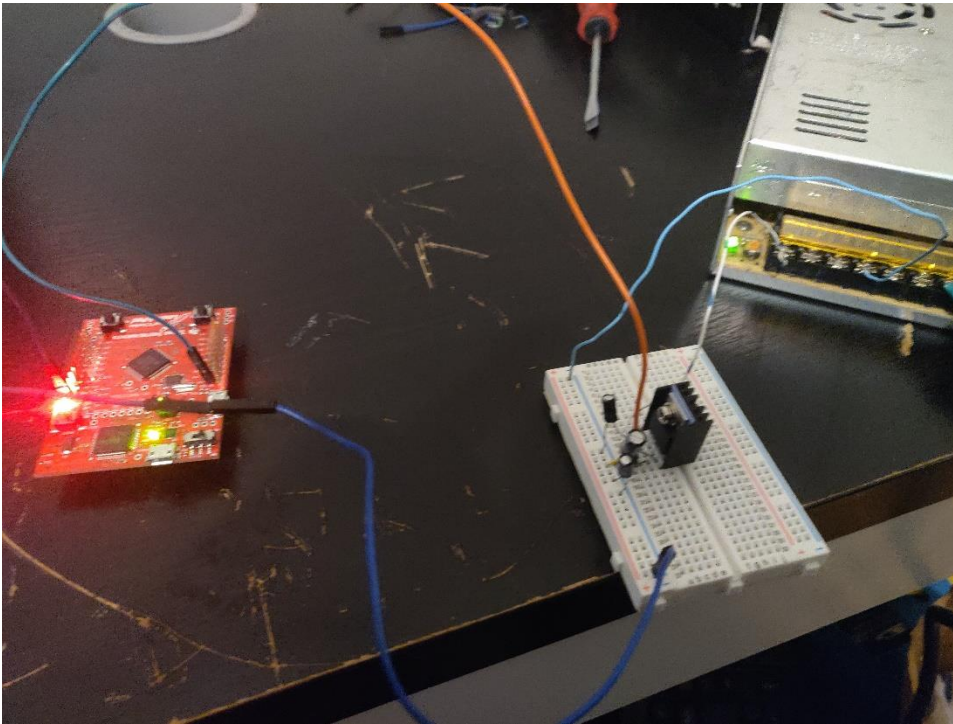


Figure #: Voltage Regulator powering TM4C

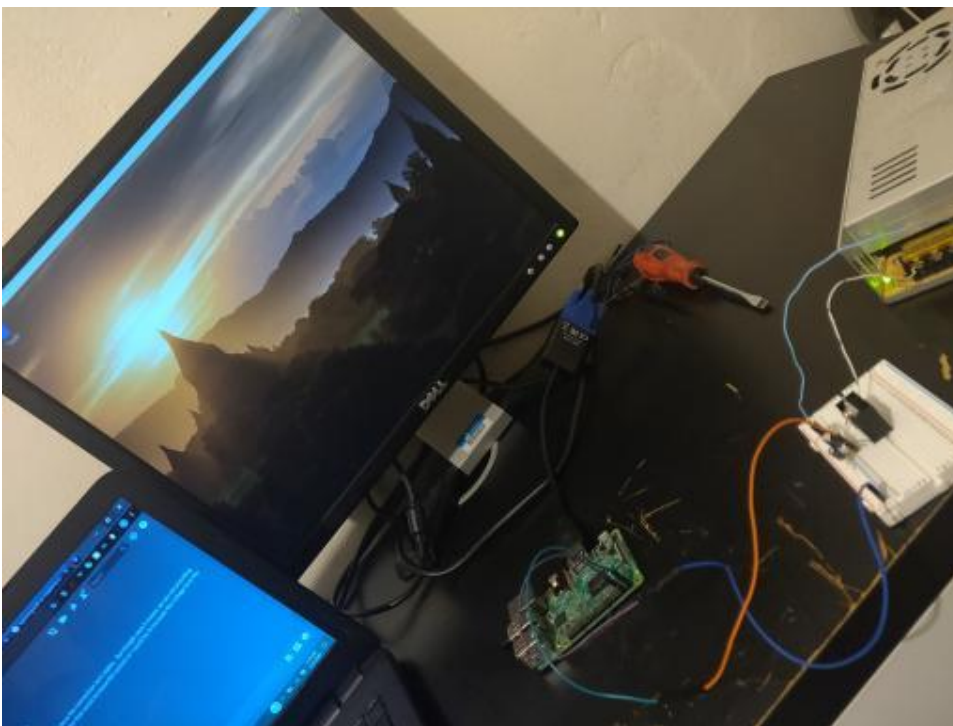


Figure #: Voltage Regulator powering Raspberry Pi





4 Cost and Schedule

4.1 Cost Analysis

a. Labor

- Raul
 $(\$35/\text{hour}) \times 2.5 \times 100 \text{ hours} = \8750
- Geoffrey
 $(\$35/\text{hour}) \times 2.5 \times 100 \text{ hours} = \8750
- Dennis
 $(\$35/\text{hour}) \times 2.5 \times 100 \text{ hours} = \8750
- Alex
 $(\$35/\text{hour}) \times 2.5 \times 100 \text{ hours} = \8750
- **OTAL LABOR = \$35, 000**

b. Parts

▪ Fridge:	
Adhesive Rubber Seal	\$8.00
Styrofoam	\$14.99
11x A4988 Drivers	\$22.77
Shrink Tubes	\$9.00
Capacitors	\$11.99
Wire Terminals	\$17.99
3x Cooling equipment	\$53.94
Power supply	\$17.98
2x Ultrasonic sensor	\$44.00
Pi Camera	\$29.95
5x Shaft Coller	\$27.45
3x Door Hinges	\$5.12
2x Door Handles	\$4.30
27x Circle Magnets	\$4.99
3x Tank Treads	\$98.87
4x axels	\$19.98
Lock Nuts	\$7.69
Total	\$399.01



- **Cooker:**

Induction Cooker	\$29.99
Cooking Bowl Bottom	\$1.89
Cooking Bowl Top	\$8.99
2x Motor Drivers	\$3.04
2x Insulation Sheet	\$43.50

Total	\$87.41
--------------	----------------

- **Microcontrollers**

TM4C123GXL	\$12.99
Wires	\$12.99
10x 5V regulators	\$13.81
10x Heatsink	\$4.49

Total	\$44.28
--------------	----------------

- **TOTAL PARTS = \$530.7**

c. **GRAND TOTAL**

- **TOTAL LABOR + TOTAL PARTS = \$35530.7**



4.2 Schedules

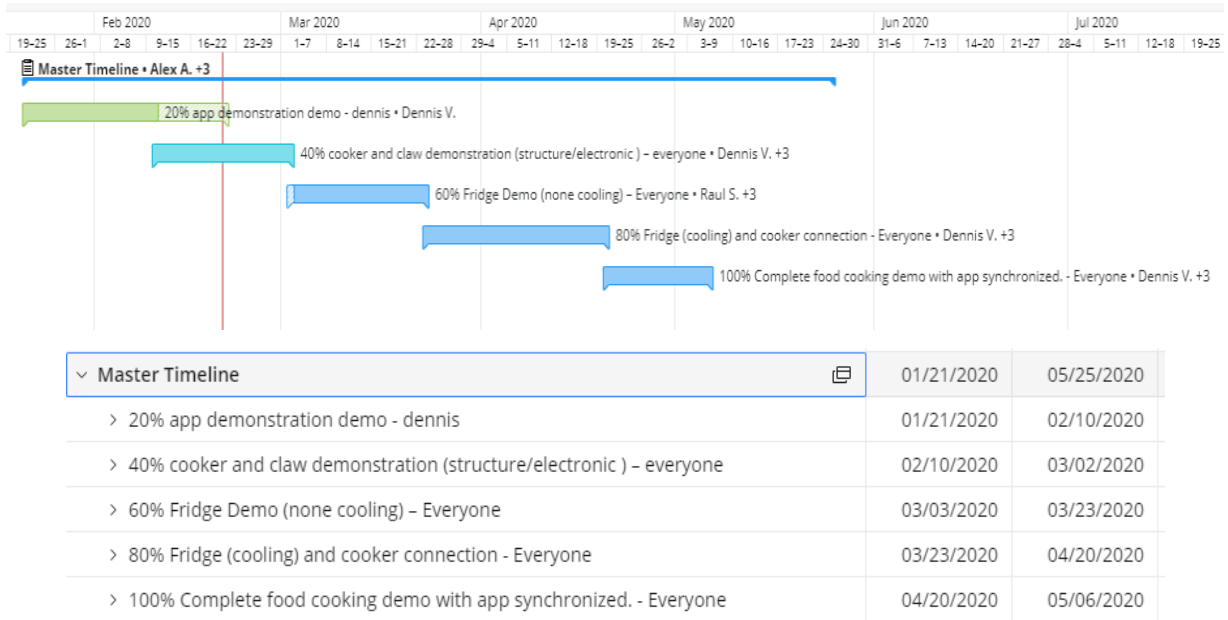


Figure 27: Master Timeline

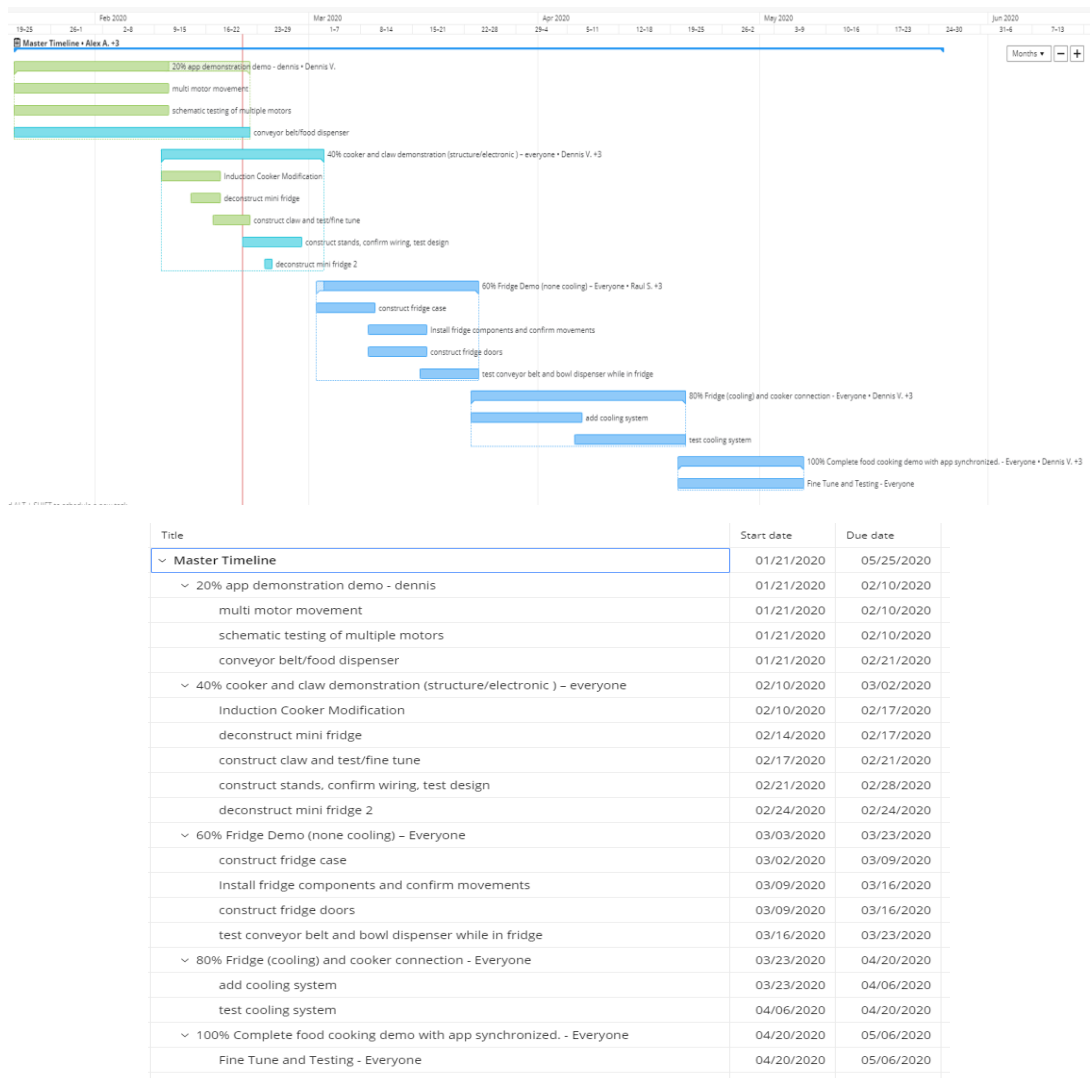


Figure 28: Progress Against Timeline



5 Ethical Implications

On September 7th, 2017, Equifax announced that a cybersecurity incident had occurred, where personally identifiable information or PII was accessed. Our project will incorporate an app which will ask for personal identifiable information or PII. The PII will include name and address information, which is used to ensure the user receives the bowls of food they order from the app. In order to ensure that data is not leaked like in the Equifax incident, the server used to hold the user's PII information must be secure and free of human security errors. Stanford University has some tips on how to prevent these human security errors, like to "[know everything that runs on the server, why, and which users have access.](#)" By following these tips, the server's databases can be more secure and free of human security errors.

To add on to this, computer engineers need to verify and ensure the software they create is compatible with the hardware they use. Just recently CBS News announced that "[Nissan recalled 1.23 million vehicles for faulty back camera.](#)" The faulty camera was caused due to a software error, which caused the cameras to not turn on when reversing. Specially for this senior project, since FUD will require a lot of physical components, a lot of software and hardware testing will need to be done to ensure everything works. For example, an error in the cooking process like a failure in the cooling system can cause the food to spoil, thus can cause human harm.

6 Local Contributions and Project Tools

6.1 Local Contributions

Long Beach Makers Society:

Makers Society has helped us by providing us with parts for our project and future 3D prints for free. As well as guidance to efficiently print our designs to make sure they fit properly in the end.

Irwin Jimenez:

Mr. Jimenez has provided his expertise with welding at no charge, including the labor and parts necessary to complete the bowl design we have created for this project.



6.2 Project Tools

Task Management:

Wrike



Communication:

Discord, Wrike Messaging, and Text



Software:

Circuit Maker (Schematics), TinkerCAD (3D Modeling), Fusion 360 (3D Modeling), Android Studio (App Development), Microsoft OneDrive (Document Storage), uVision IDE (TM4C programming)





7 References

- [Nema 17](#)
- [SPYCE](#)
- [Android Development](#)
- [Temperature Control Switch](#)
- [Danger Zone for Food](#)
- [Motors](#)
- [Motor Driver](#)
- [Ultrasonic Sensor](#)
- [Induction Cooking](#)
- [Measuring Vref](#)
- [OS download](#)
- [UART configuration](#)
- [C++ Setup](#)
- [WiringPi](#)
- [Direction Grounding](#)
- [TM4C datasheet](#)
- [Sensor Datasheet](#)



- [Pi Datasheet](#)
- [Motor Driver Datasheet](#)
- [Voltage Regulator Datasheet](#)
- [Power Supply Datasheet](#)