IoT Coffee Maker

Final Report

The Brew Crew

Anthony Stem

Matt Hodges

Simon Garen

Table of Contents

1. Introduction

In such a fast pace society, caffeine is highly regarded and is an important staple in many peoples' diets all around the world. It provides people with energy to help them be more alert, focused, and refreshed as they take on their day-to-day activities and work. One of the most common ways people meet their caffeine needs every day is by drinking a beautiful cup of coffee.

People spend up to hundreds of dollars of their hard-earned money to purchase machinery just to make this valued drink. Other times, because of our world's face-paced lifestyles, people either run out of time to make their coffee or people simply skip the machinery altogether and end up spending profuse amounts of money to purchase this drink from vendors such as fast-food restaurants or quick-service places just to get an overpriced form of their beloved caffeinated beverage. Sometimes people even miss out on their daily caffeine intake because they might wake up late for work so they don't have time to make their morning cup of joe. Our product seeks to alleviate the daily struggle for people to get their favorite caffeinated beverage in an efficient and cheap manner by providing them with our newly innovated IoT smart coffee maker using the Raspberry Pi 4.

Our machine will come with a lot of functionality to help give customers the most efficient coffee-making experience they could ever hope to have. The machine will use a mobile application that allows customers to choose exactly when they want to have their favorite cup of coffee. The mobile application will allow customers to schedule their coffee whenever they want, whether that be as soon as possible, if they are in need of immediate caffeine intake, in twenty minutes, if they want to have their cup after they go about their morning routine, or even every Monday at 7:30 am. The mobile application will give users the ability to do just that, and it let them be able to find time to incorporate their favorite cup of coffee into their busy routines.

In addition, a touchscreen display on the machine will provide customers with the same functionality as the mobile application for times when they don’t want to use their phones and would like to interact with the machine manually.

The IoT smart coffee maker will ensure that users have an easy and pleasant experience making their favorite cup of coffee so they can spend less time worrying about getting their caffeine intake and focus more on the important day or night ahead of them.

2. Definitions, Acronyms, and Abbreviations

Internet of Things (IoT)	The Internet of Things describes physical objects or technologies that are embedded with a combination of sensors, processing ability, software, or other technologies, and they are able to connect to and exchange data with other devices and systems over the Internet or a communication network.
IoT	*abbr.* Internet of Things
Raspberry Pi 4	An inexpensive computer that is a platform for the fundamental basics of computer organization. The Raspberry Pi 4 will serve as the main functioning computer for the IoT coffee maker.
RPi 4	*abbr.* Raspberry Pi 4
Flask	A web framework in Python that allows the formatting of web pages.
Dynamic app	The construction of a web page so across all user devices the web page looks the same.
Responsive design	The formatting of a webpage that adapts the formatting based on the device accessed from.
Cron job	A Linux command that allows for job scheduling to run commands periodically at fixed times.
Crontab	A file that contains
MySQL	A database management system.
MariaDB	An open-source, community-developed fork of the MySQL database management system.

TRIAC	Three terminal AC switch that conducts current in both directions when a small current is applied to its gate terminal.
Solid State Relay	An electronic switching device that switches on or off when an external voltage (AC or DC) is applied.
SSR	*abbr.* Solid State Relay

3. Project Detail

3.1. General System Design Overview

3.3.1. General Design

Our prototype mobile Flask application will serve as one of the primary ways to interact with the IoT coffee maker. It will provide the user with the ability to brew their coffee at the press of a button. If a user wants to go about their morning routine before they want their coffee, they will have the option to set a time which will range from the time it takes the water to heat up in the coffee machine and the time to brew to any time after that the user wants their coffee to be ready. For example, if it takes a user twenty minutes to go about their morning routine, they can set the time to have the brew ready to be 20 minutes from when they want. In addition to this functionality, the app will also provide the user the ability to make regularly scheduled times to have their coffee ready. For example, if a user wants their coffee every Wednesday at 6:30 am before they leave for work, they will be able to use the scheduling feature on the app to make this happen.

In addition, the IoT coffee maker's touchscreen interface will be the portal that allows the user to manually interact with the machine. The customer can use the touchscreen when they do not want to interact with the machine via mobile phone or simply when it is more convenient to manually interact with the machine. The touchscreen will be mounted to the front of the machine's custom-built housing and connected to the RPi 4. The touchscreen interface will display the same Flask application used for their mobile device (responsive design will make the app display well on the touchscreen device) to allow the user to manually interact with the coffee machine. The touchscreen interface will improve customer satisfaction due to the increased ease of use of the machine since they can interact with the machine both manually and from afar using their phone.

3.3.2. General Design Changes

In our initial design, we planned to use motors for certain aspects of the coffee maker, and include a coffee ground hopper, as well as include an extra scheduling feature in the Flask application. However, we decided to change these ideas due to impracticalities and redundancies in our initial design.

To start, we moved away from using motors entirely. We planned to use motors to turn the coffee maker on and off, as well as dispense coffee grounds from a hopper into the coffee maker, but we all agreed that these ideas were too complex given the time frame, and they added unnecessary complexity to the project. While the coffee ground dispenser would've been a good addition to the project, it was impossible to incorporate it into our design without breaking the coffee machine that we had purchased to retrofit for this project. Additionally, we thought that using motors to turn the coffee maker on and off was needlessly complex and could be done using simpler methods. As a result, we chose to use a TRIAC and SSR circuit as outlined in *Section 3.2* to turn the coffee maker on and off using the Flask app running on the Raspberry Pi.

Secondly, we decided to remove the scheduling feature from the Brew Menu that allowed users to schedule a time only within the current day. We determined that this design was redundant since the Scheduler feature that was implemented in the application could handle the exact same requests while providing additional features such as setting recurring times and choosing specific dates to brew coffee. Instead of our initial design, we changed the Brew Menu to only include a button that allows the user to immediately start brewing coffee and bring them to the Brew page which will show a countdown until the coffee is ready.

Overall, we agreed that these design changes for our prototype IoT coffee maker were reasonable and made for a better prototype than the design we had previously envisioned given restrictions in budget, time, and practicality.

3.2. Circuit Diagram

Below in *Figure #1* is the circuit diagram of the solid state relay, TRIAC, and how it will interact with the coffee maker:

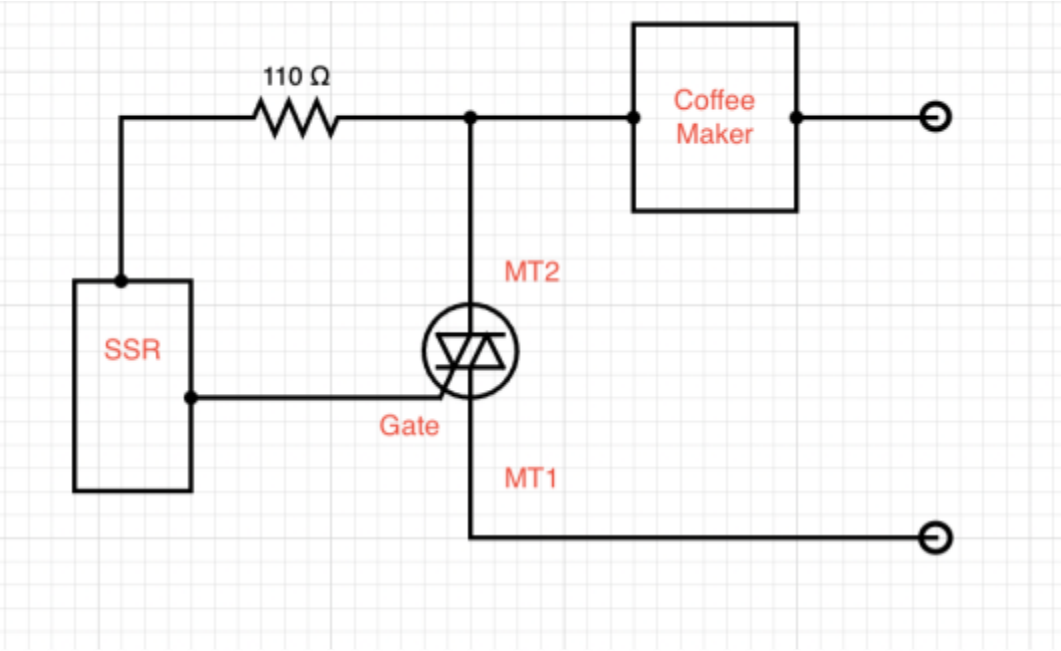

Figure #1: Circuit diagram that shows how the Raspberry Pi will be able to turn the coffee maker on and off.

This circuit diagram depicts a basic TRIAC circuit that will allow the coffee maker to be turned on and off with the Raspberry Pi. The portion of the extension cord plugged into the wall has its load wire attached to mounting terminal 1 (MT1). Attached to MT2 is the plug portion of the extension cord, with the coffee maker plugged into it. Also affixed to MT2 is a 110 ohm resistor, and attached to the resistor is a wire that goes to the solid state relay. Finally, the other outgoing wire from the solid state relay is connected to the gate terminal of the TRIAC. A relatively small electrical pulse sent to the gate from the Raspberry Pi will act as a switch for the high voltage coming from the wall's electrical outlet. This allows us to control the electricity being sent to the simple AC coffee maker,

which has the same effect as physically flipping the coffeemaker's power switch, it can be done using the Flask app on the Raspberry Pi.

3.3. Flask Application

3.3.1. Prototype Flask Application Design

The prototype Flask application will be one of the most crucial parts of the project. This will allow the user to directly interact with the machine via mobile device or touchscreen interface. To achieve synergy with the consumers and to ensure the coffee maker does not get overloaded with requests which could cause many problems for the machine and the user, it will be dynamic and include validation features that ensure the user can only schedule brew times at least 15 minutes from each other. These features will also prevent the user from brewing if a scheduled event is planned to take place during the time a full brew would happen. The app will ensure that across each of the user's devices (their mobile device and the touchscreen), the Flask app will update, ensuring the coffee maker is not overloaded with requests such as a user making multiple unwanted cups of coffee by accident. We will use cron jobs to have the Raspberry Pi 4 carry out the necessary functions to brew the customer's cups of coffee at their desired times. In addition to cronjobs, we will use MariaDB to store scheduled brew times so the Flask app can display scheduled times in the Scheduler calendar that the user can read or delete if wanted. If a user deletes a scheduled time from the Scheduler calendar, it will also remove the corresponding cronjob from the crontab. In addition, the application will also need to follow responsive design guidelines. This will allow the formatting of the Flask app displayed and formatted properly and beautifully across their devices. Having a responsive design will ensure the consumer's satisfaction is higher due to a friendlier and easy-to-use interface to interact with.

The general design for the Flask application can be seen below in *Figure #2* which shows a diagrammatic flow chart representation of our mock prototype application.

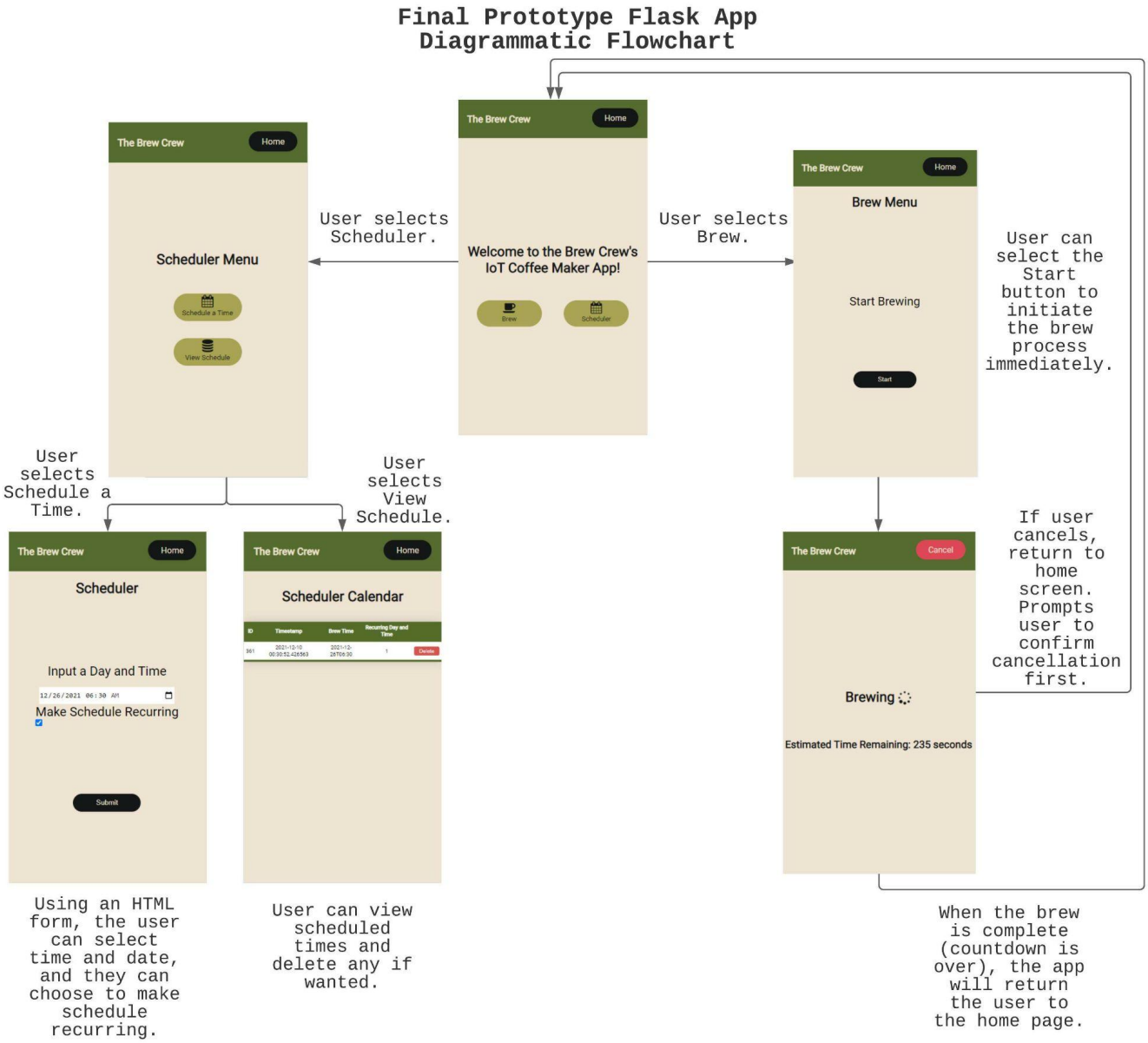

Figure #2: Diagrammatic flow chart of our final mock/prototype Flask application.

3.3.2. Flask Application Design Changes

The original Flask application design featured a scheduling feature inside Brew Menu that allowed users to schedule a time only within the current day. However, we removed this feature because we agreed that it was redundant and added needless complexity for both users and the developer. The original design for the Flask application can be seen below in *Figure #3*:

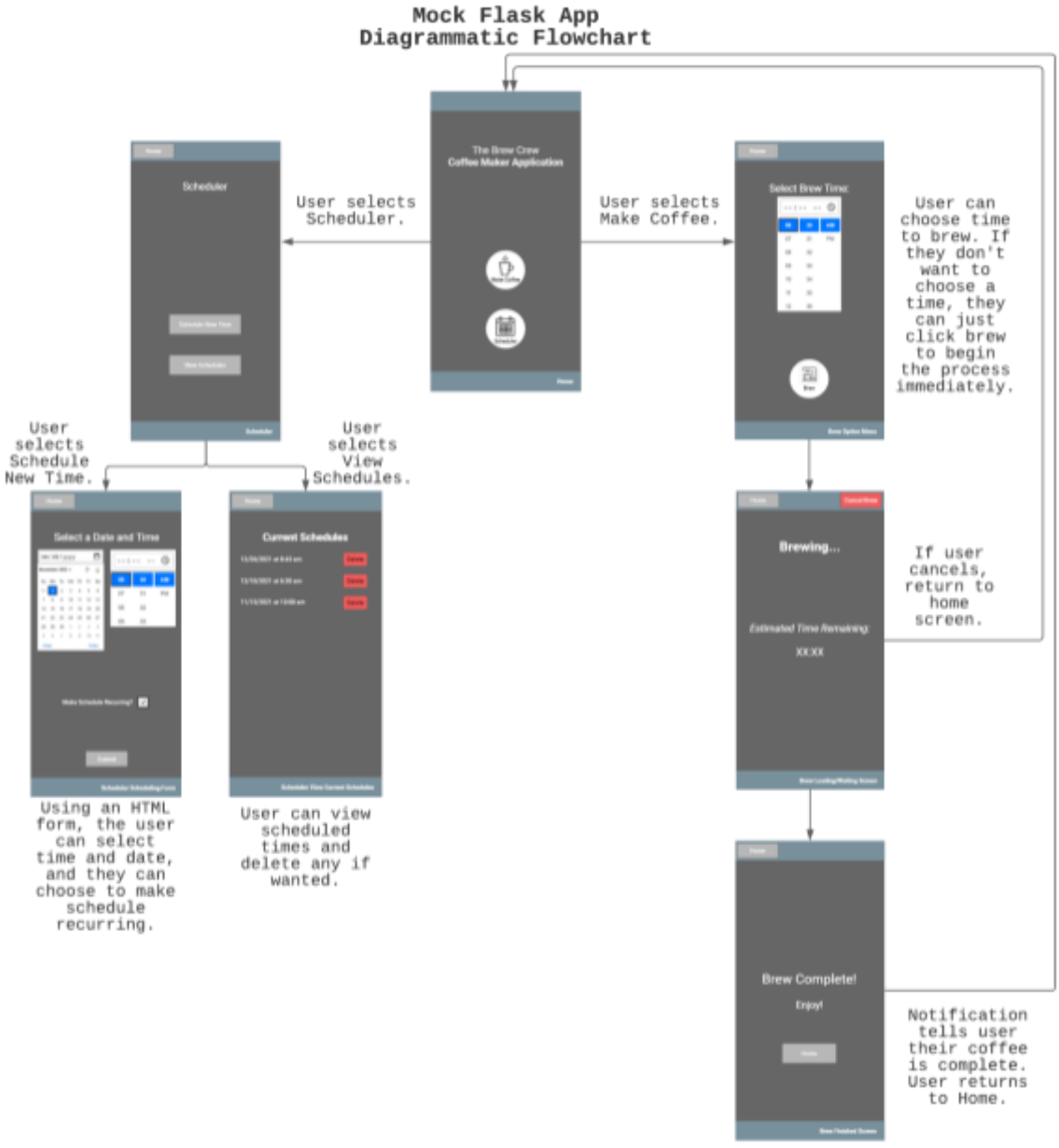

Figure #3: Initial diagrammatic flow chart of the mock/prototype Flask application.

3.4. Mobile Application to Completed Brew Process

3.4.1. Mobile Application to Brew Process

The mobile application to the completed brew process is outlined in the flow chart in *Figure #4* below. First, the user will open up the mobile application from their device and navigate to the home screen. Next, the user will make the decision to choose either “Brew” or “Scheduler.” If the user selects “Brew,” they will be brought to the Brew Menu where the user will be prompted to press start to initiate the brew process. If the user selects “Scheduler,” they will be brought to the Scheduler Menu where they can click “Schedule a Time” or “Scheduler Menu.” To initiate a brew, they have to click “Schedule a Time” which will take them to a HTML form to schedule when they want their coffee ready. Depending on the choice decided by the user, the coffee will either begin brewing immediately or postpone to a later scheduled time of the day. When the coffee begins brewing after starting an immediate brew request, the app will display the Brewing Screen, indicating to the user that the coffee is brewing and the estimated time that remains. During this screen, there will be a cancel feature, allowing the user to cancel the brewing process at any time up until the coffee is being poured out. Once their coffee is brewed, the application will redirect the user back to the home screen.

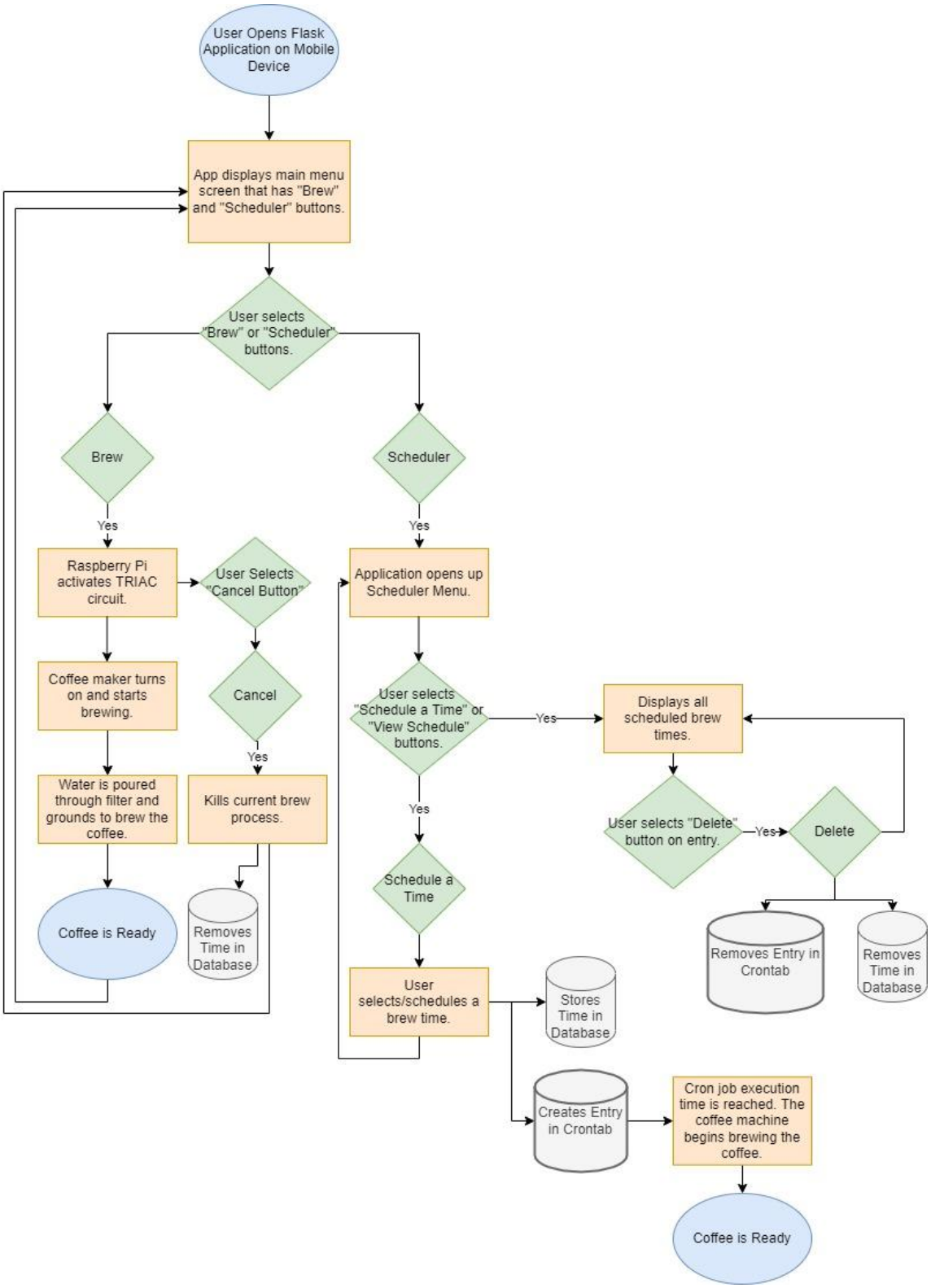

Figure #4: A flow chart depicting the functionality of the application when access from an external device not directly connected to the Raspberry Pi 4.

3.4.2. Mobile Application to Brew Process Changes

Our previously or initially planned mobile application to brew process is outlined in *Figure #5* below. In our updated touchscreen to complete the brew process, we scrapped the idea of using motors for any of the functionality. Instead, we switched over to a sturdier and less error-prone TRIAC and SSR circuit to turn the coffee machine on and off when desired. We updated the flow chart in *Figure #5* to show how the Flask application interacts with the Scheduler database and crontab in *Figure #4*.

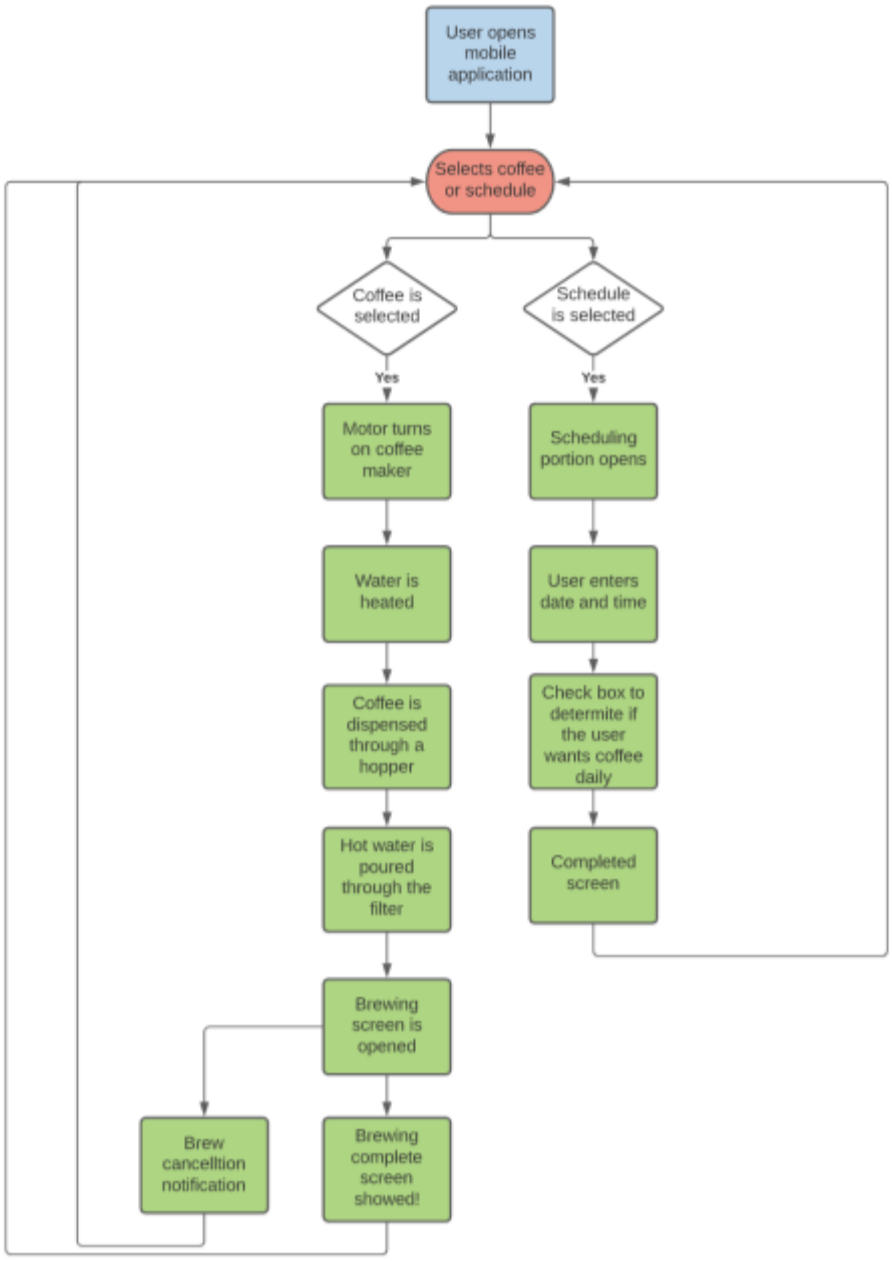

Figure #5: A flow chart depicting the initially planned functionality of the application when accessed from an external device not directly connected to the RPi 4 (e.g. mobile phone).

3.5. Manual Touchscreen Interface to Completed Brew Process

3.5.1 Touchscreen to Brew Process

The manual touchscreen interface to the completed brew process is outlined in the flow chart in *Figure #6* below. Similar to the mobile application, the touchscreen will present the Flask application to allow the user to control the coffee maker directly. First, the user will press the touchscreen and decide whether to brew a cup of coffee now or schedule a cup for later by selecting either the “Brew” or “Scheduler” button on the main menu. Depending on whether the user wants coffee now or later will change what the application does. If the user presses the “Brew” button, the program will prompt the user to click the Start button to initiate the brew process immediately. If the user chooses this option, then, while the coffee is brewing, the brewing screen will show with the estimated time remaining. While this screen is showing, the user will have an option to cancel the coffee for a specific amount of time. After the coffee is finished, the user will be pulled back to the home screen.

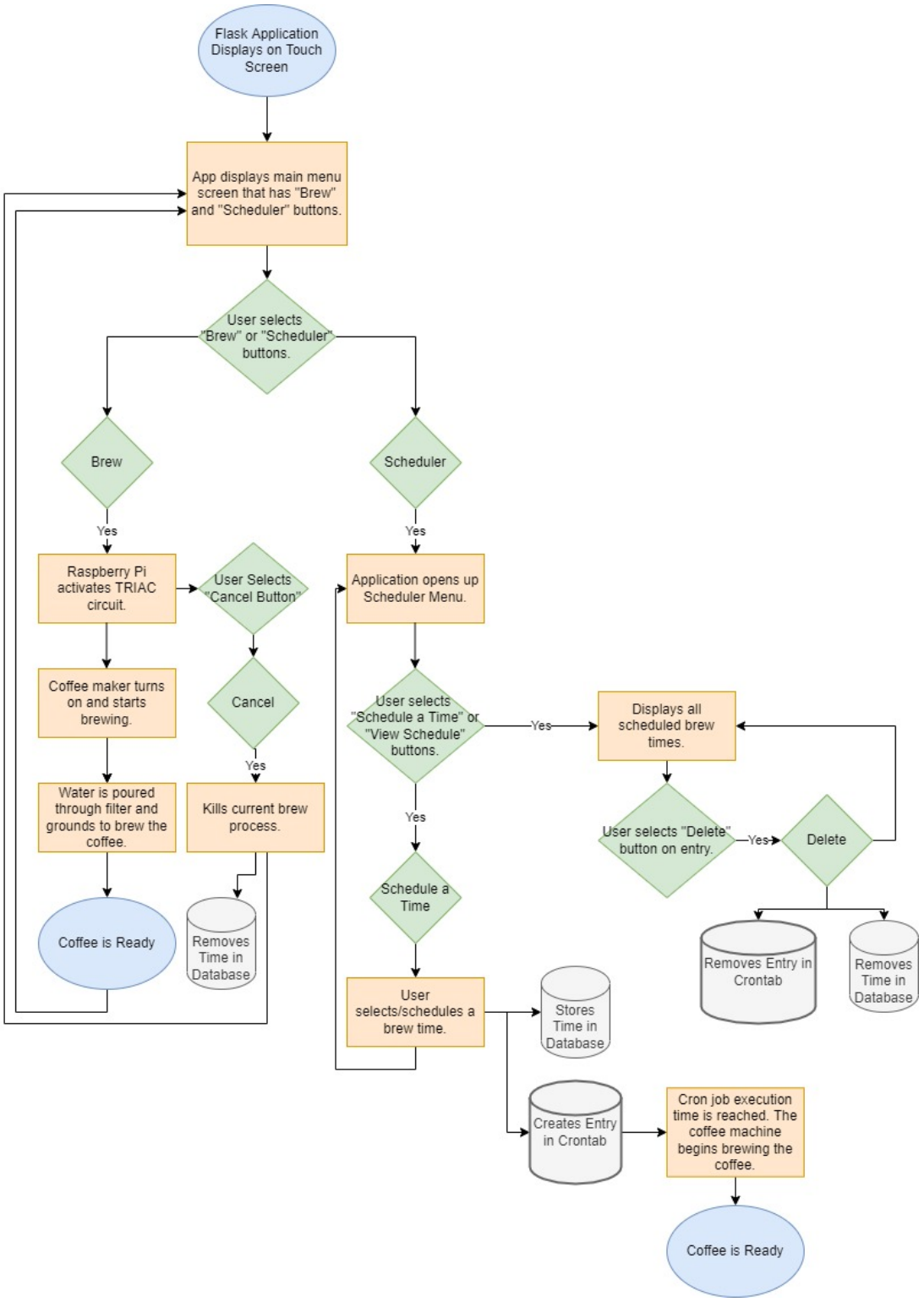

Figure #6: A flow chart depicting the functionality of the Flask application when accessed from the touchscreen interface.

3.5.2. Touchscreen to Brew Process Changes

Our previously intended (old) functionality and touchscreen to completed brew process is outlined in *Figure #7*. In our updated touchscreen to complete the brew process, we no longer use motors for any of the functionality. Instead, we switched over to a TRIAC and SSR circuit to turn the coffee machine on and off when intended. Also, we updated the flow chart in *Figure #7* to show how the Flask application interacts with the Scheduler database and crontab in *Figure #6*. As it stands, the current touchscreen to finished brew process matches the mobile/external device to completed brew process with the difference being that one using the touchscreen device plugged into the RPi 4 and the other uses a mobile or external device not connected to the RPi 4.

Figure #7: A flow chart depicting the functionality of the application when accessed from the touchscreen interface.

3.6. IoT Coffee Maker Housing Design

The housing for the Coffee Pi was a relatively simple design. The housing consisted of 6 rectangular pieces made of laminated spruce boards in various sizes to allow the pieces to fit together easily. The hardest piece of the project was the plunge cut of the front piece that would store the touchscreen. This piece had to be very precisely cut to allow the touchscreen to be inserted with very little room for error. The cut had to be placed 3.25 inches in from the side and 2.5 inches from the top and bottom of the piece. Overall the piece would be 7 inches by 10.5 inches. This piece was not the normal size of 16x12 because underneath there needed to be space available to place the coffee pot. In addition to the needed plunge cut, a 2-inch diameter circle was needed to place the coffee maker wires through. Although this cut was significantly less precise than the plunge cut, caution was still needed because of the location near the edge of the back cover.

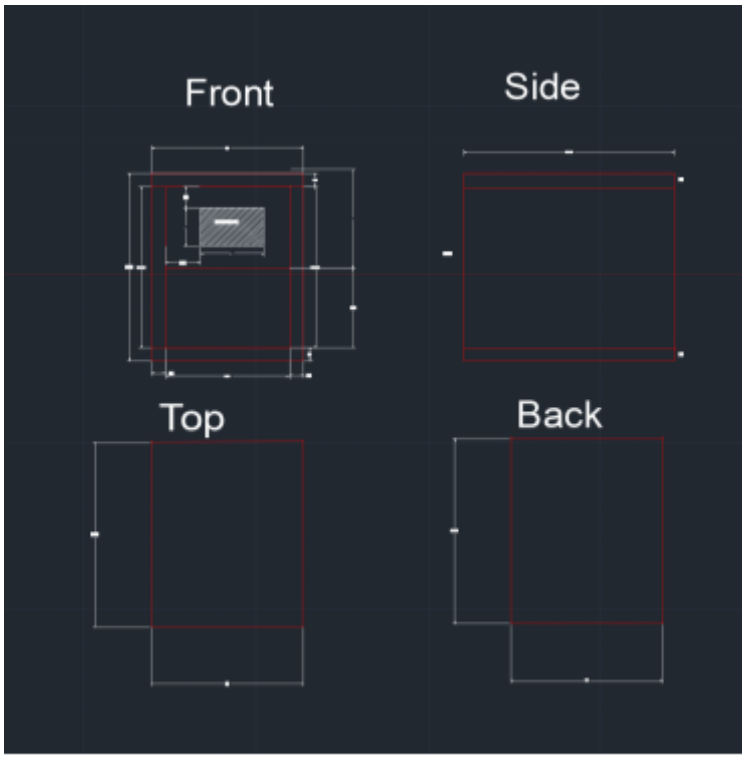

Figure #8: CAD drawings of IoT Coffee Maker housing.

4. Projected Budget vs. Actual Budget

For both the projected and actual budget for this project, they are divided up into two parts: materials cost and labor cost. The overall projected and actual budget, the itemized budget for materials and labor, as well as the total budget overage is as follows:

Projected Materials Cost:
The current expected cost for the materials that will be used in the project is broken down as follows:

- Raspberry Pi 4 - $75
- Coffee Machine w/ Reusable Filter - $25
- Raspberry Pi Solid State Relay Board - $20
- EUDAX Mini Generator Motors 3V-12V - $10
- Smraza Raspberry Pi 4 Touchscreen - $42
- Netac Micro SD Card 32GB - $12

Total Projected Materials Cost: $184.00

Actual Materials Cost:
The actual cost for the materials that will was used in the project is broken down as follows:

- Raspberry Pi 4 - $75
- Coffee Machine w/ Reusable Filter - $25
- Raspberry Pi Solid State Relay Board - $20
- Smraza Raspberry Pi 4 Touchscreen - $42
- Netac Micro SD Card 32GB - $12
- Wood for housing - $50
- Coffee grounds - $16
- Rubber Grommets - $4

- Extension Cord - $9
- Zip ties - $3
- Clear box - $12
- TRIAC - $6
- Heatsink - $5

Total Materials Cost: $279.00

Projected Labor Cost:

The projected itemized cost for labor for each role in the project is as follows:

- Network Engineer - $43.55/hour for 25 hours - $1088.75
- Lead Developer - $47.88/hour for 25 hours - $1197.00
- Project Manager - $53.46/hour for 25 hours - $1336.50
- Hardware Guru - $42.89/hour for 25 hours - $1072.25
- Testing & Quality Assurance - $17.33/hour for 25 hours - $433.25

Total Projected Labor Cost: $5127.75

Actual Labor Cost:

- Network Engineer - $43.55/hour for 31.83 hours - $1,386.20
- Lead Developer - $47.88/hour for 90.08 hours - $4,313.03
- Project Manager - $53.46/hour for 31.83 hours - $1,701.63
- Hardware Guru - $42.89/hour for 31.58 hours - $1,354.47
- Testing & Quality Assurance - $17.33/hour for 6 hours - $103.98

Total Actual Labor Cost: $8,859.31

Projected Total Budget: $5,311.75

Actual Total Budget: $9,138.31

Total Budget Overage: $3,826.56

As seen above, we are over budget by $3,826.56. This is mostly due to the actual labor cost being significantly higher than the projected labor cost. The biggest discrepancy between projected and actual was the time the Lead Developer spent working on the project. In our projections, we projected it would take the Lead Developer only around 25 hours to finish their piece of the project project, which would have only cost a total $1197. However, the Lead Developer spent a total of 90.08 hours on the project, which totaled $4313.03. The Lead Developer had to carry out a lot more tasks and spend a lot more time developing the Flask application than initially planned, so we misprojected the time they would spend by 113.10%. The Lead Developer accounted for 81.4% of the budget overage and is the primary reason for theju budget overage.

5. Project Plan

5.1. Group Project Roles & Role Assignments

For this project, there are five primary roles that have been assigned to all three members of the group. These five roles for this project are defined as follows:

Project Manager: The leader of the project. The Project Manager is tasked with organizing the project meetings, submitting the project files, and monitoring the progress of the group to assure the group is not falling behind.

Network Engineer: The Network Engineer is tasked with setting up the wireless connection of the Raspberry Pi and connecting the device to the Raspberry Pi via Bluetooth.

Hardware Guru: The Hardware Guru is responsible for the assembly of the project. This will include the attaching of the Raspberry Pi, motors, and anything else that is needed

to achieve the project goal. The Hardware Guru will also need to work closely with the Lead Developer to make sure there is synergy with the hardware and the software.

Lead Developer: The Lead Developer is responsible for the programming of the Raspberry Pi. This will include working with the Hardware Guru to make sure that there is a connection between hardware and software.

Housing Designer: The Housing Designer is responsible for designing and building the housing that will contain all the hardware components including the Raspberry Pi and coffee maker.

Testing & Quality and Assurance: The Testing & Quality and Assurance will be in charge of assuring that the product is a high-quality product and works as intended.

The currently assigned roles for each member of the group are listed below:

Anthony:
Lead Developer, Network Engineer, and Testing & Quality Assurance.

Matt:
Project Manager, Housing Designer, and Testing & Quality Assurance,

Simon:
Hardware Guru, and Testing & Quality Assurance.

5.2. Major Goals

For the goals of this project, our next goals are to learn the functionality of the coffee maker. This will be accomplished by taking apart the coffee maker and examining the various wiring and mechanisms and reviewing the documentation. Doing this will allow us to construct a circuit diagram and establish any further hardware components needed for our design.

In addition to learning how the coffee maker works, we must begin the programming of the Flask app so we can ensure the app functions properly.

Lastly, we must design the custom housing of the coffee maker to ensure we have a set design for when the construction process begins. The project's housing is a very important aspect of the product because it is the physical design that customers and people will see.

5.3. Gantt Charts

5.3.1. Expected Gantt Chart & Project Timeline

The initially proposed and expected timeline for our project is depicted in the Gantt chart shown in *Figure #9* below:

THE BREW CREW PROJECT TIMELINE	10/29-11/4	11/5-11/11	11/12-11/18	11/19-11/25	11/26-12/2	12/3-12/5
Order and receive neccasary components	Matt					
Disassemble coffee maker and learn the various components and parts of the pre-purchased coffee maker	Simon					
Develop a circuit diagram for the product	Simon					
Begin coding the Flask application.	Anthony and Matt					
Continue coding and finish app design		Anthony and Matt				
Begin constructing the desired circuit for the hardware aspect of the IoT coffee maker		Simon				
Finalize IoT coffee maker inerds/circuit construction			Simon			
Build coffee maker housing				Matt		
Assemble the complete physical/hardware (including housing) of the IoT coffee maker				ALL		
Finalize coding and Flask app				ALL		
Run tests and finalize the project					ALL	

Figure #9: Initial Gantt chart showing the predicted timeline for each project step.

5.3.2. Actual Gantt Chart & Project Timeline

Below, in *Figure #10*, is the Gantt chart for our actual project timeline. This shows what our actual project timeline was for this project during the semester:

THE BREW CREW PROJECT TIMELINE	10/29-11/4	11/5-11/11	11/12-11/18	11/19-11/25	11/26-12/2	12/3-12/5
Order and receive neccasary components	Matt					
Disassemble coffee maker and learn the various components and parts of the pre-purchased coffee maker	Simon					
Develop a circuit diagram for the product	Simon					
Design the Flask web application pages.	Anthony					
Learn necessary programming intricacies and build and design Flask application		Anthony				
Begin constructing the desired circuit for the hardware aspect of the IoT coffee maker		Simon				
Finalize IoT coffee maker inerds/circuit construction			Simon			
Build coffee maker housing					Matt	
Assemble the physical/hardware (including housing) of the IoT coffee maker					Matt	
Finalize coding and Flask app.						Anthony
Install touchscreen interface						Anthony
Run tests and finalize the project						ALL

Figure #10: Finalized Gantt chart showing the actual timeline of the steps completed in the project.

6. Target Market

The target market for this product is the people who have busy lifestyles want to streamline their morning routine and get their cup of coffee in an efficient and timely manner. These consumers include college students and your everyday workers who rely on caffeine to stay focused and energized throughout the day.

In the morning nobody wants to wait minutes for their coffee maker to warm up and wait while it slowly brews their coffee. Our device allows the user to immediately start the coffee maker in the comfort of their bed at the touch of a button so that when they get up, they can directly walk over and pick up a hot, freshly brewed cup of coffee. For users who do not drink coffee directly after waking up, this device is also a great match for them. The Scheduler feature of the app allows the user to schedule a date and time for the coffee maker to have their coffee ready. For example, if the user likes to drink their coffee after they shower and get dressed, they could set a timer for 1 hour, go about their routine, and then pick up their freshly brewed coffee when they're finished. Also, they can schedule recurring times. So, if, for example, a user likes to have their coffee every Wednesday at 7:30 am, our app offers the functionality to let the user have their coffee ready at exactly that time just by simply selecting a date and time and checking the box that says they want this scheduled event to be recurring.

These features make the product excellent for those living busy lives and who need caffeine to ensure they are ready for the busy day or night ahead of them.

7. Work Summary

7.1. Anthony's Contributions

Anthony contributed to the project by designing and creating the Flask application, as well as creating the software responsible for driving the IoT Coffee Maker's functionality.

To start, he created the website part of the Flask application using HTML and CSS for the formatting and design. In addition, he used JavaScript to create the countdown feature on the Brew screen that displays how long a user has left until their coffee is ready, as well as AJAX to send POST requests to turn the coffee maker on and off at certain points. He also used MariaDB to create the scheduler database to store scheduled brew times and display them in the scheduler calendar.

Not only did he design and build the website for the Flask application but he also developed the Flask application using Python. The Flask application interfaces with the website to form one unified web application. It includes a variety of features that allow the IoT coffee maker to function as designed. Firstly, it includes validation features so that users cannot schedule brew times on top of each other. If a user tries to schedule a time or brew during an event, a message will appear on the bottom of the screen telling them that they cannot. Additionally, the Flask application uses python-crontab to allow users to schedule cronjobs on the Raspberry Pi that initiate the brewing process for the coffee machine. If a user schedules a one-time event, a cronjob will be created and an entry will be inserted in the database to be displayed on the Scheduler Calendar; however, once the event is complete, it will be removed from both the crontab and the database. If a user schedules a recurring event, the scheduled time will not be deleted from the crontab or database unless the user manually removes it from the Scheduler Calendar.

He also programmed the flask app and an additional script for scheduled events to interact with the TRIAC circuit to turn the coffee maker on and off to brew coffee.

In addition to developing the software aspect of the project, Anthony installed the touchscreen on the physical coffee machine housing so that the user can interact with the coffee maker manually.

Anthony also took part in testing and finalizing the software end of the project, as well as the physical project itself so that it would be ready for presentations.

Finally, outside of the development of the software of the project, Anthony played a major role in creating and writing up the initial and final project proposals.

Overall, Anthony played a major role in the project by designing and developing the software piece of the IoT Coffee Maker, as well as ensuring it was ready for presentations and use.

7.2. Matt's Contributions

Matt's contribution was building the IoT coffee maker housing, assembling the coffee maker, and keeping the project on track.

Matt and the group initially began designing a project with a hopper incorporated. After numerous group meetings and discussions, the hopper design was decided to be impractical due to multiple reasons including, the bulky design, the need for a sensor and the difficulty of changing the coffee filter if installed. One aspect of the project that was agreed upon from the beginning was the aspect of the touchscreen. Obviously, with this coffee maker, a design was needed that gave the space for the touch screen, along with space for the coffee maker and the hardware.

A design was made 16inx16inx12in that incorporated all the aspects discussed in the meetings. The hole for the touch screen was 4inx2in. Using pinewood, Matt used a table saw to cut the larger pieces into various sizes to match the dimensions needed. After cutting the wood needed, the plunge cut was needed for the touchscreen. To make the plunge cut, Matt needed to use a drill press to drill a 2 inch diameter hole in the center of the wood. Matt also used the drill press to drill another 2 inch diameter hole on the back covering inorder to run the extension cord through. After making the drill press cut, Matt used a jigsaw to cut the corner out of the circle. To make the board look more visually appealing Matt sanded the corners with a power sander with a lower grit sandpaper, then going at the piece with high grit sandpaper to make it smoother.

While assembling the pieces of the housing, Matt had to pre-drill the holes to assure the wood did not split. To achieve this, Matt used a drill bit that was the same size the screw, drilled the hole the same length (⅝ inches in some locations and 1 ½ inches in others) then inserted the screws with a screwdriver to make sure there was not too much pressure against the wood to once again assure no splitting. Inserting the hinge was difficult because Matt had to use two different screw sizes and had to directly drill

to a piece four times one the same edge. While doing this, Matt was very careful as splitting the wood would mean a restart was needed.

While assembling the interior coffee maker, Matt used double sided velcro strips to allow the coffee maker to be taken out and inserted with ease. Along with this, the triac needed to be placed in a way that the wires could still reach the solid state relay connected to the pie. After all of this, the wires were placed through the holes on the back covering allowing the coffee maker and the Raspberry Pi to be connected to an outlet with ease.

7.3. Simon's Contributions

Simon's contributions were getting the Raspberry Pi to control the coffee maker. This was done through the use of a solid state relay and creating a simple TRIAC circuit. Simon initially planned to directly control the coffeemaker with a GPIO pin from the pi, but as the coffee maker is AC, 120V from the wall could potentially kill the Pi and or start a fire. His next idea was to control the remote of a switchable outlet from the solid state relay, but the solid state relay could only control ac devices, so eventually, he went with a TRIAC design.

As the solid state relay did not come with any manual or guide, Simon figured out how it worked from a circuit diagram on the Amazon store page, eventually figuring out GPIO pin 26 controlled the first relay. He then tested the functionality of the solid state relay by attaching a TRIAC to a lightbulb and writing a simple python script to control the light. After ensuring the reliability of the solid state relay in conjunction with a simple TRIAC circuit, Simon went on to design a smart outlet for the coffee maker to plug into.

Simon started by getting a clear plastic softball display case, in which the TRIAC would be visible, but sealed off so no one could accidentally be shocked by 120V, which can

potentially be fatal in certain circumstances. He took a 16 gauge extension cord, cut it in half, using drilled holes in the display case, along with zip ties and rubber grommets, to feed both ends of the severed extension cord into the case. The zip ties fastened around the wires and the bottom of the case prevented someone from tugging on the wires, to protect them and the TRIAC. Inside the case, Simon used speed connectors to reconnect the ground wire and neutral wire. He soldered the load wire that came from the end with the outlet plug to mounting terminal 1 of the TRIAC, and the other portion of the load wire to mounting terminal 2. Also on mounting terminal 2, Simon soldered a 110 ohm resistor, and soldered to that was one of two wires that went to solid state relay output 1. Simon also soldered the other wire to the gate of the TRIAC, which went to the second spot of relay output 1. Simon used a screw through the base of the case with a rubber spacer to raise the TRIAC and prevent its movement. He also applied a thermal compound to the TRIAC and affixed a heatsink in order to dissipate some of the heat generated from the current flowing through the TRIAC. To further ensure user safety, Simon used a series of zip ties on the outside of the clear box to make it impossible to unintentionally come into contact with the high voltage components. Simon's TRIAC design in combination with the pi and the solid state relay allows the coffee maker to be plugged into the extension cord and controlled with the Raspberry Pi, effectively creating a smart outlet.

8. Reflections

8.1. Anthony's Reflection

I was the Lead Software Developer in my group and so I allocated a majority of my time to designing the Flask web application and ensuring that the IoT Coffee Maker was fully functional and operational. For me, my task was difficult but it served as an amazing learning opportunity, and I am completely delighted at the outcome.

One major challenge that I faced while working on this project was having to learn new programming languages or improve in ones I already knew in order to make certain aspects of the project function as designed. For example, prior to this course, I had very limited knowledge of Javascript, particularly working with AJAX. One feature of the app is a countdown timer that uses AJAX to send POST requests to the Python Flask app to turn the coffee maker on and off. When a user presses the Brew button in the Brew Menu, I had to program it to send a POST request to turn on the coffee maker and switch to the Brew screen which displayed a countdown that ran also using Javascript. The timer was difficult for me to implement since I did not know how to make a program sleep for 1 second between function executions (counting down). So, I had to learn how to do this using online sources. Once the countdown timer reached 60 seconds, I had the program send another POST request using AJAX to turn the coffee maker back off so that the coffee maker would have enough time to stop and so the machine would stop dripping. Luckily, I was able to use the labs we did in class to get a baseline understanding of how AJAX works and then use outside resources to get a better understanding of it.

In addition, I had to further improve my knowledge of Python, including Flask and Jinja, and learn new and different libraries to create some of the intricacies of the project. In this project, I had to use the *datetime* and *python-crontab* modules so that I could

implement certain features such as validation for scheduled brew times and scheduling cronjobs on the Raspberry Pi that will execute the brewing process for the coffee maker.

Also, I had to learn how cron jobs and crontabs work on the Raspberry Pi. This involved learning the syntax for them so that I could program them properly with the Python module I used, as well as making sure they executed the proper commands (which was annoyingly difficult due to file paths).

Another challenge, and what was likely the hardest one that I faced while working on this project, was learning to take steps backward. At a certain point during this project, I hit a roadblock because of the additional scheduling part that we had planned on the Brew Menu. We wanted the user to be able to select a time within the current day to have their brew be ready. However, I realized that this feature was redundant given that we have the Scheduler feature that lets users do the exact same task. Therefore, I made the difficult decision of removing code that I had already written for this feature, and I had to convert it into a button that allows users to just immediately start brewing their coffee. For me, taking one step forward and two steps back was a very hard thing for me to do as it was something I never have considered doing when making a project. However, it saved me from spending unneeded time and having unnecessary stress to implement additional parts of the code to make this feature work. Also, I think the result (having a button that simply starts the brewing process immediately) is more fitting for the project and makes it easier to use.

Finally, another challenge that I faced was needing to wait for the physical coffee machine to be finished so that I could program it as needed. To make up for this, I had to exercise patience and use my time to finish up or refine other parts of the project until I had the missing piece that I needed to finish the software side of the project.

Overall, this project was an exceptional learning experience. During this project, I not only expanded upon my programming knowledge and skills but also improved my

communication, risk management, and project skills. It was a difficult assignment, but I certainly think that the skills I learn will have a lasting impact on me as I move forward in my Computer Science career.

8.2. Matt's Reflection

A challenge I faced in this project was the carpentry for the IoT coffee maker housing. One of the most difficult parts of this was the plunge cut needed to place the touch screen. The plunge cut took me multiple tries and hours to complete. One consistent problem I had with the plunge cut was because of the generally low-quality wood, the wood had a problem splitting if too much force was applied while cutting the wood with my jigsaw. I ended up solving this problem by using a drill press to drill a hole in the center of the wood, then using a jigsaw to get the rectangular cut needed. Another challenge faced in this project was learning how to work in a group setting again. I spent the previous year at home taking online classes at UVM, so it has definitely been an adjustment being back in Burlington. Given that I hadn't been in a group project for some time, I was definitely nervous about this one. Not only have I not worked in a group-oriented way in over a year, but this project was also worth 35% of the final grade, which was very intimidating. Alas, once I got reacclimated to working with a group, the project, although challenging, did not have the added challenge of having a dysfunctional group. Arranging times to meet and assuring the group was on pace to complete everything on time was a stressful experience. Arranging times to meet were difficult because everyone's schedule is different, even with us all being computer science majors. Some people have class, some people have work, but in the end, we met many times and ended up having enough time to complete the project. Overall, the project was a great experience and gave me the ability to learn and make something completely of our group's own imagination.

8.3. Simon’s Reflection

Looking back at the final result of the project, there were many unexpected challenges and takeaways. The most significant takeaway for myself is the disconnect between the simplicity of an idea and the hardship of actually implementing it. One of the initial ideas for the project was to create a coffee/tea hybrid machine where motors would release tea bags into a cup. While this idea didn’t make it farther into development, it showed how important it is to keep your ideas feasible with your current skill level, and achievable in a reasonable amount of time. An idea in the initial proposal that got axed was for a hopper to hold coffee grounds to sit on top of the coffeemaker. This wasn’t possible as the filter needs to be cleaned in between each use, which would mean you’d have to move the hopper, and there would need to be some sort of sensor to prevent it from dispensing with the lid open. Also, we could not actually fit the hopper without breaking the coffee maker. Moving forward I think it's important to draw a schematic before implementing a feature, as it brings to light problems that you didn’t imagine when it was just an abstract thought.

My portion of the project, remotely turning on and off the coffeemaker, turned out to be much harder than I initially imagined. At first, we thought we could directly connect the GPIO pins to the coffeemaker, but as it's a 650 watt AC device, doing so would instantly kill the raspberry pi or create a fire. After that my next idea was to use a remote outlet and attach the solid state relay to the remote, but since the solid state relay was only compatible with ac devices, this was not possible. Eventually, I split an extension cord and used a TRIAC in combination with the solid state relay to control the current flowing to the coffeemaker. This process made me realize how complicated circuits can be and how potentially dangerous dealing with 120V house voltages is and caused me to grow a deeper appreciation for electricians and electrical engineers.

I’ve really appreciated the time spent working on this project. It was interesting seeing how software can control physical devices, in our case a coffee maker, to make life slightly more convenient. In our group, everyone worked on their specialized portion of the project, and fortunately, everything came together well, culminating in a prototype I’d be glad to use every day.

9. References

AJAX Introduction. Ajax introduction. (n.d.). Retrieved December 10, 2021, from https://www.w3schools.com/xml/ajax_intro.asp.

Datetime - basic date and time types. datetime - Basic date and time types - Python 3.10.1 documentation. (n.d.). Retrieved December 10, 2021, from https://docs.python.org/3/library/datetime.html.

Format, L. (2016, November 3). How to build your own smart coffee machine. TechRadar. Retrieved from https://www.techradar.com/how-to/how-to-build-your-own-smart-coffee-machine

Indeed.com. (n.d.). Home. Job Search. Retrieved October 28, 2021, from https://www.indeed.com/career/network-engineer/salaries.

Indeed.com. (n.d.). Home. Job Search. Retrieved October 28, 2021, from https://www.indeed.com/career/lead-developer/salaries.

Indeed.com. (n.d.). Home. Job Search. Retrieved October 28, 2021, from https://www.indeed.com/career/project-manager/salaries.

Indeed.com. (n.d.). Home. Job Search. Retrieved October 28, 2021, from https://www.indeed.com/career/hardware-engineer/salaries.

Indeed.com. (n.d.). Home. Job Search. Retrieved October 28, 2021, from https://www.indeed.com/career/quality-assurance-manager/salaries.

Open source database (RDBMS) for the Enterprise. MariaDB. (n.d.). Retrieved December 10, 2021, from https://mariadb.com/docs/reference/.

Python-Crontab. PyPI. (n.d.). Retrieved December 10, 2021, from https://pypi.org/project/python-crontab/.