SELF TEMPERATURE AND TASTE REGULATING TEA CUP

By

Anirudh Kumar

James Li

Lahiru Periyannan

Final Report for ECE 445, Senior Design, Spring 2025

TA: Rui Gong

7 May 2025

Project No. 44

Abstract

We propose a cup that can heat water to the optimum temperature for a given tea leaf for the best flavor and aroma extraction. The cup also performs other functions such as stirring the liquid regularly and monitoring the tea strength. The user has the capability to communicate with the device and obtain information about these various qualities in real time and also change the settings to suit their tea drinking preferences.

Contents

1. Introduction 1

  1.1 Motivation 1

  1.2 Solution 1

  1.3 Subsystem Overview 1

  1.4 High-Level Requirements 2

2 Design 3

  2.1 Subsystem #1: Power 3

  2.2 Subsystem #2: Control 4

  2.3 Subsystem #3: Sensors 5

    2.3.1 Temperature 5

    2.3.2 Total Dissolved Solids 5

  2.4 Subsystem #4: User Interface 5

  2.5 Subsystem #5: Heating and Stirring 7

    2.5.1 Stirring 8

    2.5.2 Heating 8

  2.6 Subsystem #6: the Cup 9

3. Design Verification 10

  3.1 Subsystem #1: Power 10

  3.2 Subsystem #2: Control 11

  3.3 Subsystem #3: Sensors 11

  3.4 Subsystem #4: User Interface 13

  3.5 Subsystem #5: Heating and Stirring 14

  3.6 Subsystem #6: the Cup 14

4. Costs & Schedule 15

  4.1 Labor Costs 15

  4.2 Parts Costs 15

  4.3 Total Costs 16

  4.4 Schedule 16

5. Conclusion 19

5.1 Accomplishments 19
5.2 Failures 19
5.3 Ethical and safety considerations 19
5.4 Future work 19
References 21
Appendix A Requirement and Verification Table 23

1. Introduction

1.1 Motivation

Tea is one of the most popular drinks in the world. In the U.S. alone, the estimated market size for the beverage is USD 25.1 billion as of 2024, with 80% of households having it at their homes. [1]. Furthermore, there are stated to be numerous health benefits to the consumption of tea. For instance, green tea can lower blood pressure and total cholesterol, while white tea can strengthen teeth and fight plaque [2].

Current methods to brew tea lack ways to handle different tea leaves and maintain temperature. Tea is usually brewed by adding boiling water to a cup of tea leaves. This is effective for tea leaves like black tea, however, for more delicate teas like green tea, this would bring out more bitterness as it burns the green leaves. This is because adding boiling 100°C water matches the preferred temperature range for black tea, but it is way over green tea's preferred temperature range of 70-80°C. In addition to inadvertently burning the tea leaves, temperature is important in brewing tea because different tea leaves require different temperatures to effectively bring out its aromatic compounds. The ability to heat different tea leaves to their optimal temperatures and maintain their warmness would provide the best possible tea drinking experience. However, it can be difficult to know the different times and temperatures needed for each type of tea leaf, considering there are hundreds of them.

1.2 Solution

We propose a cup that can heat water optimally to the type of tea leaf chosen and maintain it to the best temperature. Our system provides precise temperature control to combat inconsistencies in conventional tea brewing methods. Our cup integrates multiple subsystems to ensure optimal flavor extraction, temperature retention, and ease of use.

1.3 Subsystem Overview

Our system comprises six interconnected subsystems (see Figure 1):

- **Power:** Supplies energy for all subcomponents of our teacup
- **Sensors:** feed real-time sensor measurements to the microcontroller
- **Control:** processes data for subsystem control
- **Heating & Stirring:** receive commands from microcontroller to regulate temperature
- **User Interface:** receives input from user and sent to microcontroller for logic processing
- **Cup:** holds tea solution for the user and safely hold all electronic components

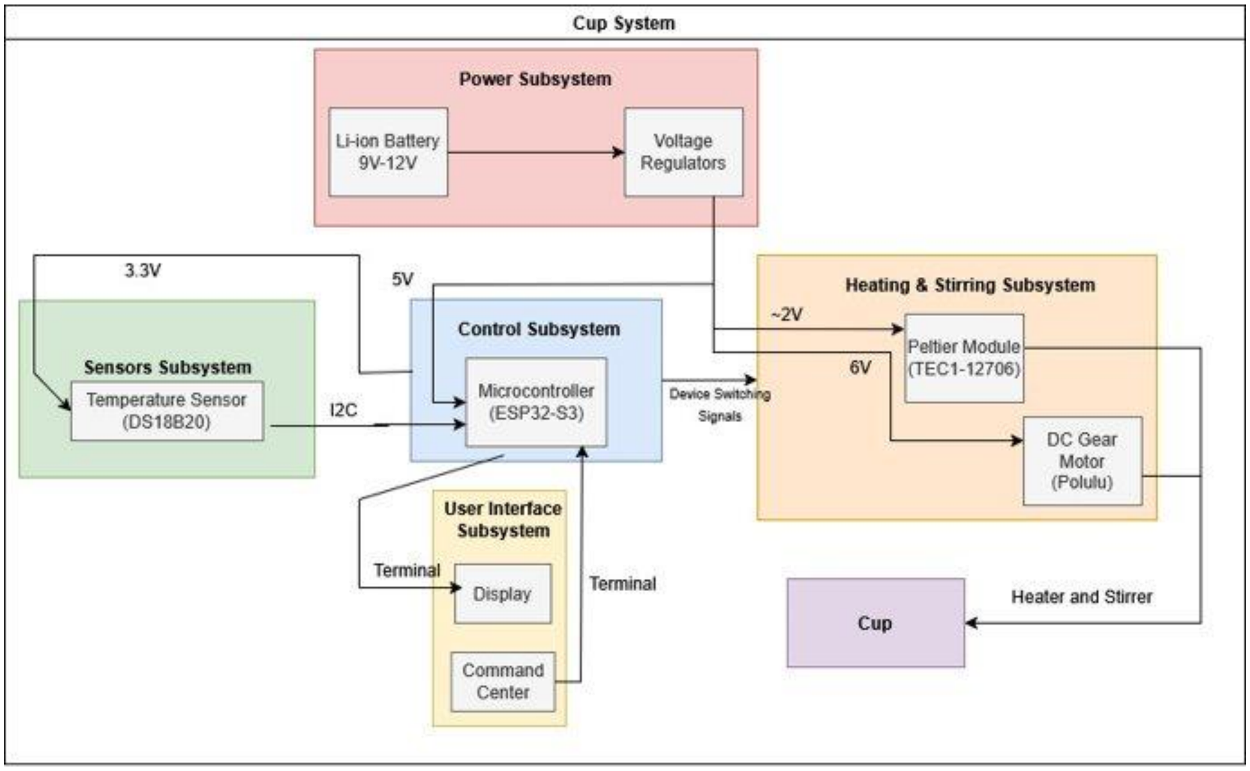


Figure 1: Block Diagram

1.4 High-Level Requirements

We believe our project must display these three high-level characteristics to solve the problem:

1. The cup should be able to maintain the desired temperature for at least six hours. Given that this is designed to be portable, it would be important for the tea to be kept warm as long as the user is outside, which of course varies, but could be at least the length of a working day. This also means that we would want other subsystems to be as efficient as possible, to minimize the energy loss.
2. The cup should be able to maintain the temperature within 1-2°C of the desired user temperature. This requires precise sensors that can communicate effectively with the user application which in turn needs to communicate with the heating element to control its behavior.
3. Ideally, the cup should not be too heavy. Some of the heavier water bottles that people carry around are about 0.5kg [3], and ours should also fall within a similar weight range. One can consider that people might use this in situations when they are outside in a cold area, perhaps doing tourism, and it is important to travel a little light.

2 Design

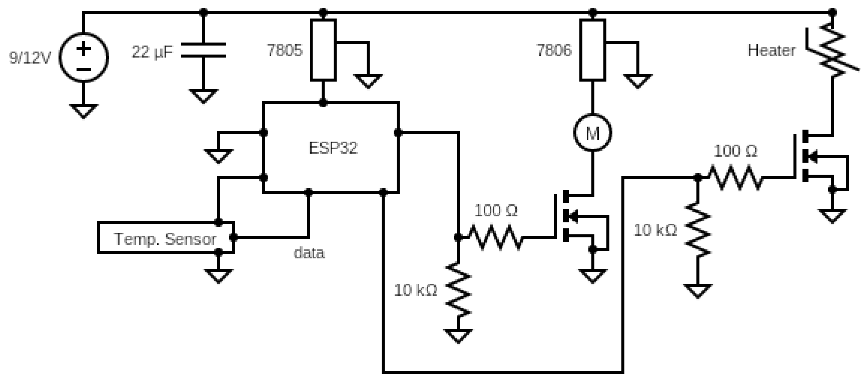


Figure 2: System Circuit Design

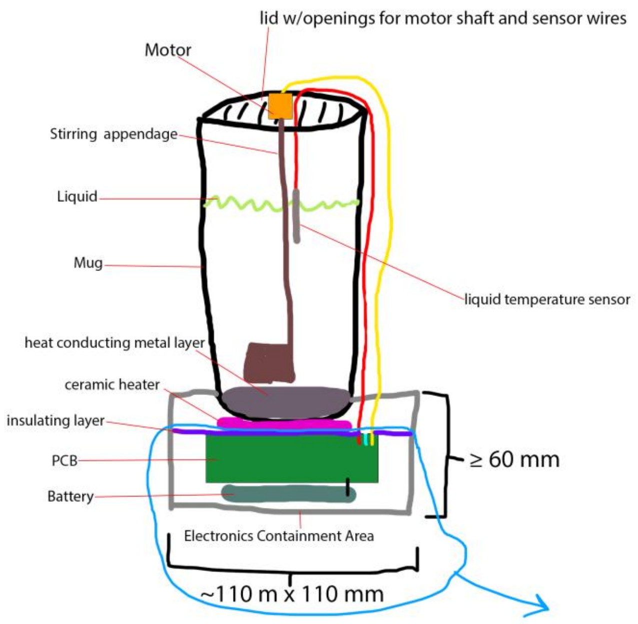

lid w/openings for motor shaft and sensor wires
Motor
Stirring appendage
Liquid
Mug
liquid temperature sensor
heat conducting metal layer
ceramic heater
insulating layer
PCB
Battery
≥ 60 mm
Electronics Containment Area
~110 m x 110 mm

Figure 3: Initial Schematic as submitted to machine shop

2.1 Subsystem #1: Power

The power subsystem is responsible for supplying power to all electronics used in the device, including the motor, the heater, the ESP32, and through that, the sensor. In the final implementation of the device, this was accomplished via a port for a battery pack. The trace widths on the PCB from the battery port were designed to support higher currents (that are drawn by the heater). The motor and ESP32 were supplied with respective voltages of 6 and 5 volts by voltage regulators.

The tea mug needed power to be supplied for at least 6 hours, such that it could be a part of daily usage. This factor is primarily limited by the battery chosen. The PCB power traces and fills were designed for higher voltages (up to 24V). This is, as discussed in the heating subsystem, due to the high voltage requirements for the metal-ceramic heater. To supply this 24V, it was initially considered that 24V worth of AA batteries (i.e. 16 series) would be used. However, this posed safety concerns (sparking). In the process of development, there was also the failure of the heating ring. This caused problems with our original plan of using the ceramic heater ring, however the switch to the Peltier module for heating, also opened us to some possibilities. The Peltier module did not require as high a voltage to heat at the same efficiency as the heater. However, this drew significantly more current than the PCB was designed for. As the PCB design had been finalized at this point, a voltage regulator would have been difficult to implement, so the makeshift solution was a voltage dividing/dropping (and thus current limiting) resistor. Quantitatively this was four parallel 27Ω resistors (6.75Ω combination) with 2W power rating. This yielded the following voltage drop over the Peltier module and power dissipation for each resistor, given that the resistance of the Peltier module was about 2Ω (as sourced from the datasheet) and a voltage input of 9V.

$$V_{Pelt.} = \frac{2\Omega}{2\Omega + 6.75\Omega}(9V) = 2.06V$$

$$P_{res.} = \frac{9V - V_{Pelt.}}{6.75\Omega} = 1.78W$$

This voltage was almost ideal for heating safely, and the power dissipation was within rating. However, the 9V battery being used as a supply did not have a significant lifespan. It was proposed that the two 8 AA battery holders could be used in parallel to supply 12V for the necessary lifespan as calculated in the verification. However, this results in a power dissipation of 3.17W, unsafe for the resistors. In an ideal case in which the device failures did not occur last-minute, a proper resistor(s) would have been sourced for ideal voltage dropping and power dissipation (or even better, a voltage regulator for the heater on the board itself. However, this was not possible in the limited time.

The temperature sensor is a very low power consumption device and thus was powered directly by the 3.3V pin of the ESP32.

2.2 Subsystem #2: Control

The control and communication subsystem acts as a bridge between the user interface (Subsystem #4) and the other subsystems. This is done by the ESP32-S3-WROOM microcontroller, which periodically collects data from the sensors. At the same time, it uses the real-time sensor data and any user-provided settings to determine the operation of the other peripheral devices. For instance, at the most basic level, if the current water temperature is lower than the temperature set by the user, the heating device should be turned on. Similarly, if the user wants the stirring to occur, the motor is turned on for 3 seconds and then off for 27 seconds; this cycle continues until the user turns it off or the tea has been brewed.

Each of the peripheral devices in our device has a MOSFET switch to allow them to be easily turned on or off. Thus, to allow the microcontroller to manipulate these devices, it has the capability of controlling the outputs of different General-Purpose Input and Output (GPIO) pins; these output signals are then connected to the MOSFET switches. One of the GPIO pins is also connected to the temperature sensor's 1-wire data bus, and this pin of course is used for input.

As mentioned, the microcontroller we chose for our project was the ESP32-S3-WROOM microcontroller. The ESP32 series of microcontrollers has extensive support and documentation from Espressif Sytems, a company that specializes in producing wireless microcontroller chips. Further, these microcontrollers allow for communication over both Bluetooth and Wi-Fi. Espressif also provides a Visual Studio Code extension called ESP-IDF that allows for easier programming of such devices. At one point, we short circuited our microcontroller by mistake and had to switch to an ESP32 DevkitC V4, but the ESP-IDF framework made this transition more seamless.

2.3 Subsystem #3: Sensors

The sensors of the cup device are essential to maintaining the two primary qualities of tea: taste and temperature. Of these two qualities, only one was measured using a sensor in the actual device, that being temperature. Due to the high-temperature non-functionality of the Total Dissolved Solids (TDS) sensor used to quantify tea strength, said sensor was only used in external experiments collecting data on tea strength, rather than active measurement in the cup.

2.3.1 Temperature

The sensor used was a DS18B20 temperature in a waterproof encasing. This sensor received power from the ESP32 and transmitted temperature data back to the ESP32 via its data pin. The sensor was inserted through the lid of the cup to be submerged in the liquid inside, reading its temperature.

2.3.2 Total Dissolved Solids

The TDS sensor was used experimentally to quantify tea strength. This process and its results are expanded upon in the corresponding verification.

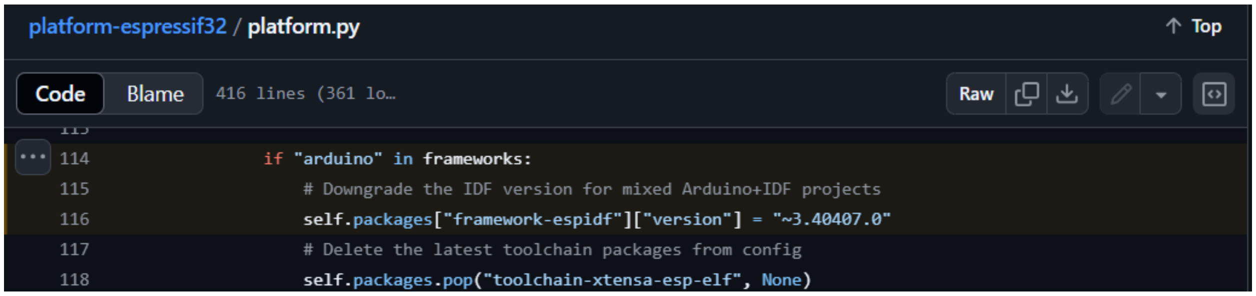
2.4 Subsystem #4: User Interface

The user interface subsystem, as the name suggests, allows the user to change the system settings and also receive real-time information from the sensors, specifically the current water temperature and tea strength. The tea strength values were measured in terms of the Total Dissolved Solids (TDS) but were converted to qualitative descriptors which would be easier to understand for the user. The user is able to set the desired temperature and turn the stirring on or off.

We of course have to run numerous tasks in parallel. To be precise, we need to constantly update the current water temperature, determine if we need to turn the heater on or off and handle the stirring capability. In addition, we must be ready to respond to any requests from the user. To accomplish this, we used Real-Time Operating System (RTOS) tasks. This allows us to specify a number of functions that are continuously running at the same time in the background, along with an allocated amount of stack memory.

Initially, we were using Bluetooth for the user interface. ESP-IDF supports two possible host stacks: Bluedroid and NimBLE. The former is capable of working with both classic and Bluetooth Low Energy (BLE) connections. The latter is only capable of working with BLE. We chose to use the latter, since we want to stick with BLE. A key requirement that we want our cup to satisfy is being able to run on its own power for a prolonged period. Further, given that we will never really be sending a very large amount of data, there is no real need for classic Bluetooth. Espressif also notes that NimBLE takes up less of a code footprint as well as less memory when running. Given that we have a limited amount of space for a program on the microcontroller, this is desirable.

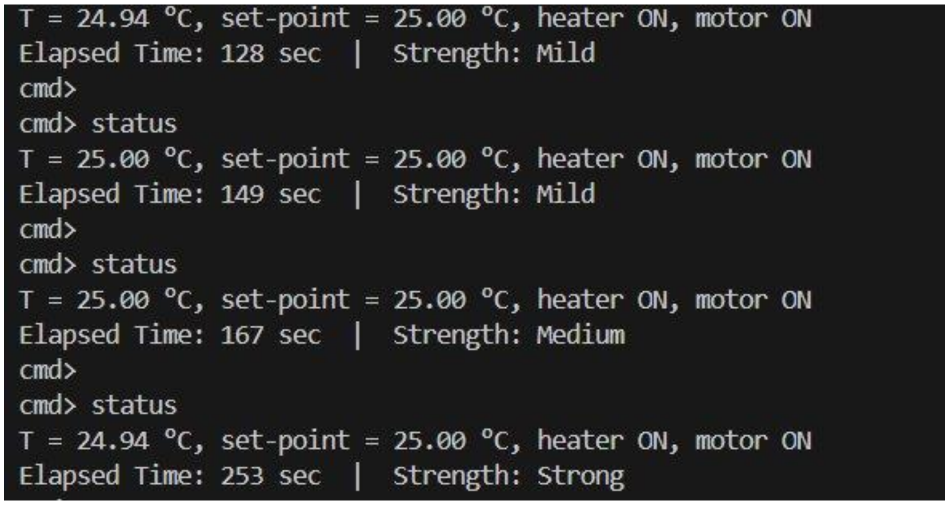
When we began incorporating the DS18B20 temperature sensor into the code, we needed to use the Arduino framework to use the relevant libraries, specifically DallasTemperature and OneWire. Thus, we pivoted to another software called Platformio, which allows for the usage of both the Arduino and ESP-IDF frameworks. However, we ran into compilation issues due to the following internal code in Platformio [4].



```
            if "arduino" in frameworks:
                # Downgrade the IDF version for mixed Arduino+IDF projects
                self.packages["framework-espidf"]["version"] = "~3.40407.0"
                # Delete the latest toolchain packages from config
                self.packages.pop("toolchain-xtensa-esp-elf", None)
```

Image 1: Internal Code Snippet from Platformio

What this snippet does is check if Arduino is an included framework. If so, as it is in our work, it forcefully downgrades the ESP-IDF version down to 3.4. This prevents us from using the NimBLE functionality, which is only available on ESP-IDF versions of 5 or higher. We are also unable to use the Arduino framework's BLE libraries, as they are built off its older Bluedroid stack, which uses a large amount of flash memory, which is an issue for microcontrollers that only have a limited memory. Further, with the RTOS tasks mentioned above, we made attempts to minimize the amount of stack memory needed, but even then, we were using 80k bytes out of the available 100k. Attempting to use the Arduino BLE libraries meant we were exceeding that stack memory. Therefore, our current UI is done through the computer terminal, as shown here:



```
T = 24.94 °C, set-point = 25.00 °C, heater ON, motor ON
Elapsed Time: 128 sec  |  Strength: Mild
cmd>
cmd> status
T = 25.00 °C, set-point = 25.00 °C, heater ON, motor ON
Elapsed Time: 149 sec  |  Strength: Mild
cmd>
cmd> status
T = 25.00 °C, set-point = 25.00 °C, heater ON, motor ON
Elapsed Time: 167 sec  |  Strength: Medium
cmd>
cmd> status
T = 24.94 °C, set-point = 25.00 °C, heater ON, motor ON
Elapsed Time: 253 sec  |  Strength: Strong
```

Image 2: Example Console Interactions by the User

We can see that the 'status' command shows the current water temperature (T), the temperature set by the user, the current statuses of the heater and the motor, the elapsed time and the current tea strength. There are a few more commands, shown in the below code flow diagram.

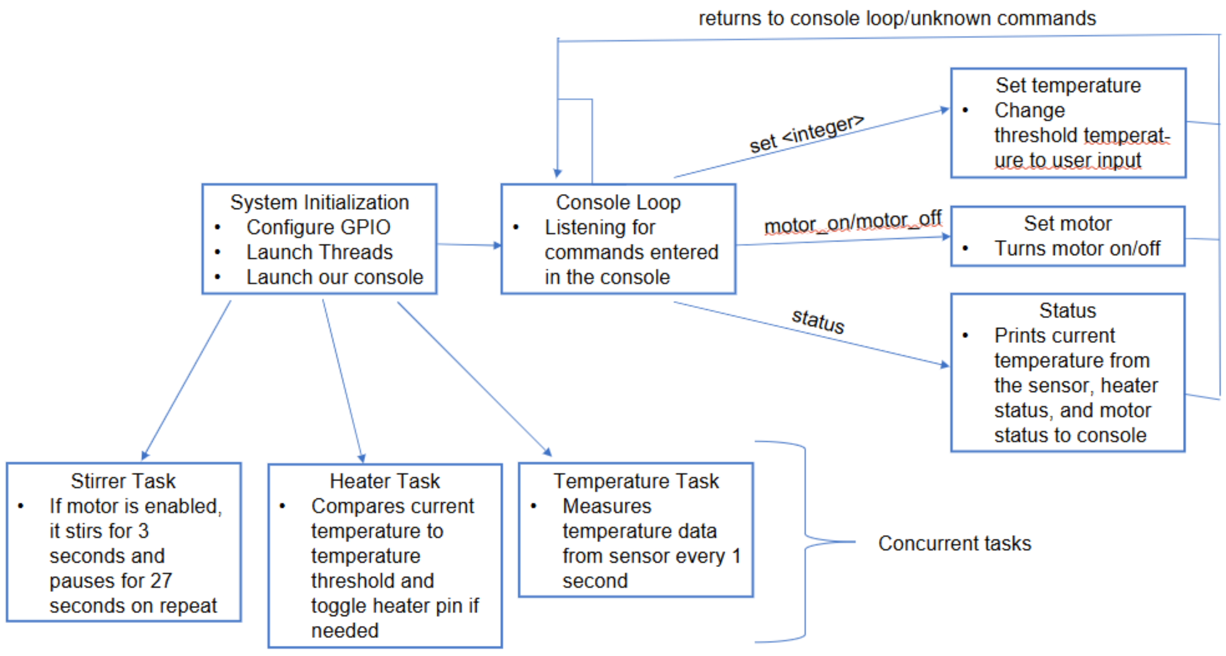

returns to console loop/unknown commands
Set temperature
• Change threshold temperat-ure to user input
set <integer>
Set motor
• Turns motor on/off
motor_on/motor_off
Status
• Prints current temperature from the sensor, heater status, and motor status to console
status
System Initialization
• Configure GPIO
• Launch Threads
• Launch our console
Console Loop
• Listening for commands entered in the console
Stirrer Task
• If motor is enabled, it stirs for 3 seconds and pauses for 27 seconds on repeat
Heater Task
• Compares current temperature to temperature threshold and toggle heater pin if needed
Temperature Task
• Measures temperature data from sensor every 1 second
Concurrent tasks

Figure 4: Flow Chart Summarizing Code

2.5 Subsystem #5: Heating and Stirring

The heating and stirring subsystem is required for temperature and consistency of the tea respectively. These functions were accomplished by a heater (specifics below) and a DC motor. Control of these devices was achieved by using a 30N06L N-Channel MOSFET for each device as a low-side switch, controlled from the ESP32's GPIO pins.

2.5.1 Stirring
The stirring component of this subsystem is centered around a DC Micro Metal gear motor. This motor is rated at 150 RPM at 6V with no load. This motor was attached to the center of the lid of the cup, and a flag-shaped appendage was attached to the axle for stirring.

Image 3: Motor and Attached Appendage (Partially Visible)

2.5.2 Heating
As partially discussed in the power subsystem, the heating portion of cup underwent various challenges and changes. In the initial planning stages, both a Peltier module and metal-ceramic ring heater were considered; these devices heat by using semiconductors and resistance respectively. The machine shop design for the system was created using the ring-heater. This also included a composite material below the heater to ensure minimal heat exposure for the PCB and batteries. Near the end of device development, an incident occurred in which liquid was spilled directly near the ring heater and likely contacted the electrical leads. This caused a short, and the ring heater was no longer functional. As the Peltier module had already been acquired earlier in the semester, adjustments were made, as detailed in the power subsection, to replace the ring heater.

Image 4: Peltier Module

Image 5: Metal-Ceramic Ring Heater

While this change caused inconveniences with the system design, it was noted that the heating power of the Peltier module was significantly higher at lower voltages compared to the ring heater.

2.6 Subsystem #6: the Cup

The cup/mug will consist of a generic stainless-steel mug. The bottom of this mug will be cut off and directly contact our heating subsystem to heat up liquid in the mug. Below that, there will be a sizable compartment to hold all electronic components. Additionally, a hole will be cut out of the lid such that the motor shaft from the stirring subsystem can be slotted through. Another hole will also be opened to place our temperature sensor into the cup. The cup will need to have a large enough radius to have enough space on the lid to place the motor and an orifice for the user to drink from, likely at least 3-4 cm. A physical design can be seen below:

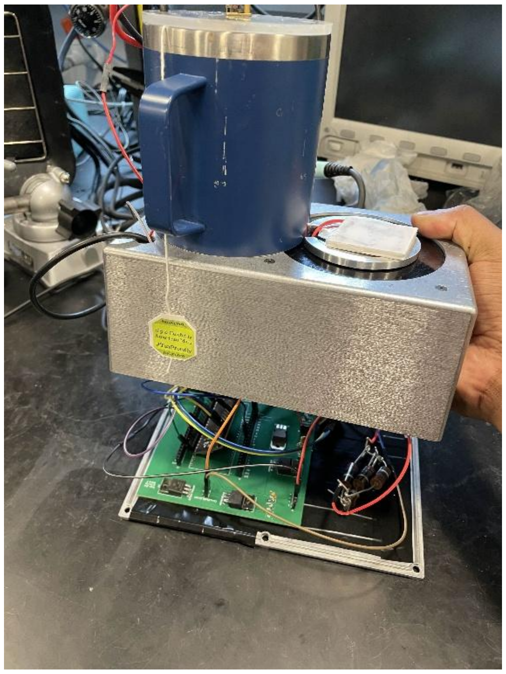


Image 6: Resulting Design with Exposed Electronics Enclosure

3. Design Verification

To determine the success of our project, we planned a number of requirements that each subsystem needed to satisfy as well as how we would test each requirement. We discuss the most important requirements for each subsystem below; a full table of requirements and verifications can be found in Appendix A. It should be noted that many requirements overlap over multiple subsystems.

3.1 Subsystem #1: Power

The power subsystem utilized multiple voltage regulators to drop the input voltage to levels necessary for devices such as the motor and ESP32. For system stability, it was necessary that these voltage regulators maintain voltages within a 5% margin of error. To ensure this, the output of the voltage regulators was attached to an oscilloscope probe, and the ripple was measured, as seen below.

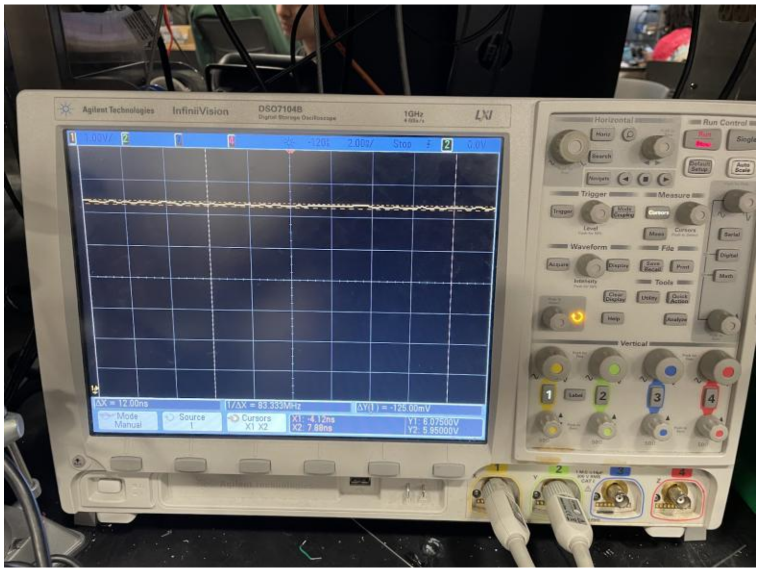


Image 7: Ripple waveform for 6V regulator (7806CT)

In this case, the ripple was 0.125V. In percent ripple: $\%\ Ripple = \frac{V_{max} - V_{min}}{V_{DC}} = \frac{0.125V}{6V} = 2.08\%$. This was safely within the 5% ripple desired.

Maximum possible usage duration for the system is calculated below using the data provided by Energizer for the Energizer MAX batteries used. An average current draw of 500 mA is assumed as per device testing.

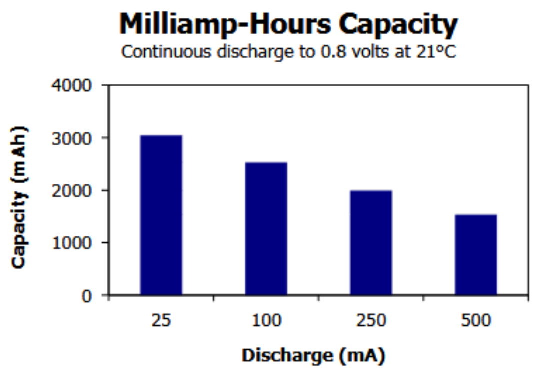


Figure 5: Battery Capacity at Various Current Discharge Levels

$$t_{life} = \frac{1600\ mAh}{500\ mA} \times 2 = 6.4\ hrs$$

Note the multiplication by two for the ideal case of two parallel 12V battery packs. This allows the device to achieve the high-level requirement requiring at least 6 hours of lifespan.

3.2 Subsystem #2: Control

The main requirement that is needed to be satisfied for the control system to be successful is for the heating to start and stop within a few seconds upon request by the control system. We tested this once we had the console-based UI established. We poured room temperature into the cup and used our UI to set the desired temperature to 40°C. Then, a thermometer was used to measure the water temperature and ensure it started rising immediately after the temperature was set. Meanwhile, we also monitored the temperature reading from the sensor to get a more precise value. When we were closer to 40°C, we then set the desired temperature back down to 20°C to make sure the heater turned off immediately and the temperature of the water, as seen in the thermometer, began decreasing as well.

3.3 Subsystem #3: Sensors

To ensure accurate readings for the user, the sensors must be well-calibrated and provide accurate data. The following experimental setup was used to verify the functionality of the sensors as well as collect data on tea strength.

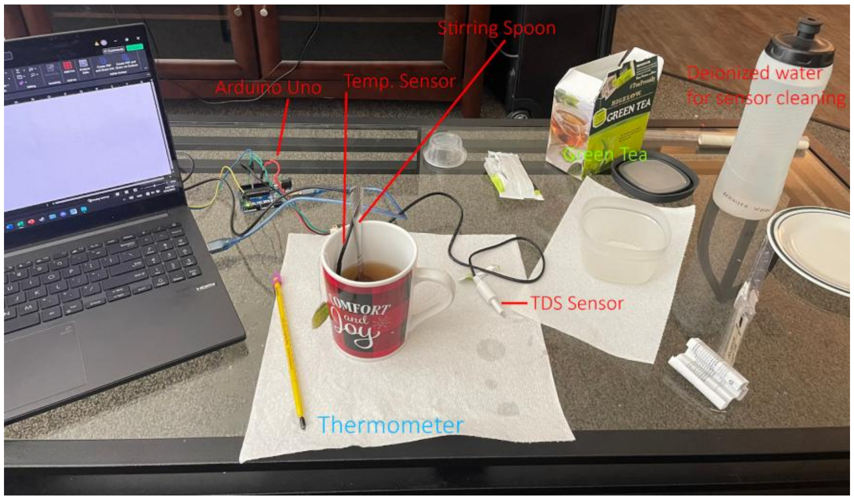


Image 8: Experimental Setup for Sensor Calibration and Tea Strength Testing

2.3.1 Temperature

The temperature sensor was tested for accuracy within 1-2 degrees Celsius by measuring varying water temperatures with it and comparing the reading to that of a lab grade thermometer. This yielded the following:

Temperature Sensor Measurement	Thermometer Measurement
78.5°C	79.5°C
25.94°C	25°C

Table 1: Temperature Sensor Calibration

In both cases, temperature readings were within a degree of accuracy, thus the requirement for the sensor was satisfied.

2.3.2 Total Dissolved Solids

The TDS sensor was first used to measure the innate TDS values of tap water. This was found to be on average 217 ppm. The TDS sensor was unable to be used in liquids above 55 degrees celsius, thus the following procedure was utilized to test tea strength.

1. Water was added to a mug and heated by a microwave to around 85 Celsius.
2. Water was then allowed to cool to the targeted tea steeping temperature. This temperature was monitored by both thermometer and temperature sensor.
3. After reaching the target temperature, a tea bag was added, and a timer was started for the desired steeping time. Liquid was continually stirred from this point until the end of the process, to avoid settling of tea particles.

4. After the steeping timer ended, the tea bag was removed, and the tea was allowed to cool to below 55 degrees Celsius.
5. After cooling, TDS readings were measured and recorded.
6. The TDS sensor was removed and then cleaned using deionized water to avoid contamination. This was repeated for various temperatures and steeping times.

The results of this process can be seen in the figures below.

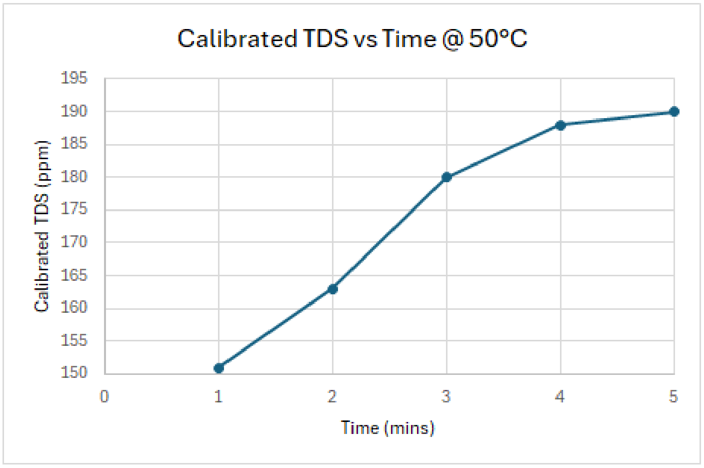


Figure 6: TDS for various steeping times at 50 Celsius

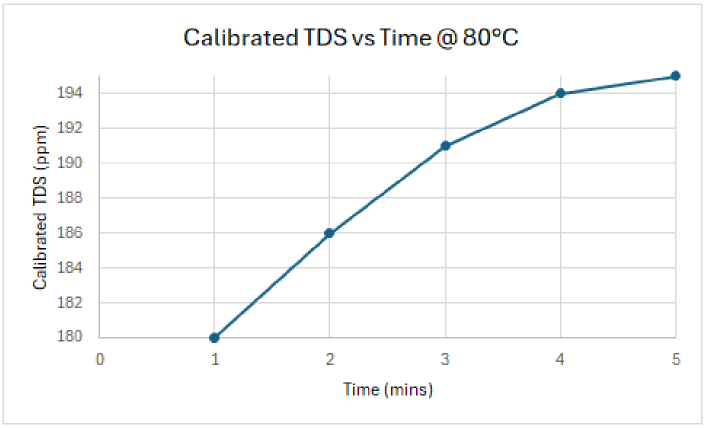


Figure 7: TDS for various steeping times at 80 Celsius

3.4 Subsystem #4: User Interface

A successful user interface allows the customer to change the desired temperature and turn the stirring on or off. They should also be able to get information about the system in real-time. The verification for these two requirements also involves the other subsystems. The first of these requirements can be checked easily via a thermometer and seeing if the motor turns on or off. The second needs information about the current water temperature and tea strength. The former was once again confirmed with a thermometer, while the latter was confirmed via the TDS value calculations as described in Section 2.3.

3.5 Subsystem #5: Heating and Stirring

Much of the requirements for different subsystems, specifically the control subsystem, and the verification done for those requirements cover those done for this subsystem. The main aspect of note was ensuring that our heating devices (whether the ceramic ring heater or the Peltier module) began correct operation once instructed to do so by the control subsystem. Again, this was confirmed via the use of the thermometer.

3.6 Subsystem #6: the Cup

For a successful cup, we require the weight of it to be less than 500g, which is a common teacup weight. When we measured our cup without liquid on a scale, we observed the cup to be 260g, satisfying our design requirement. Our cup is also made with a double steel layer and a handle to ensure that the cup's outside surface temperature is safe to hold for our user; the pain threshold for the human hand was found to 43.8°C [5], which ours is well below. The cup subsystem was also able to safely encapsulate all circuitry, so other subsystems' electronics is not at risk of contacting liquid.

4. Costs & Schedule

4.1 Labor Costs

Cost is broken down into two parts. Labor costs and hardware costs. The average UIUC ECE undergraduate's hourly salary is $42.31. We can expect the labor cost for each partner to be 42.31 x 2.5 x 60 = $6346.5. Total labor cost is 6346.5 x 3 = **$19039.5**.

4.2 Parts Costs

Part	Manufacturer/Vendor	Quantity	Cost	Subtotal
6V 150RPM Micro DC Gear Motor	Lagogia/Amazon LLC	1	$7.99	$7.99
DS18B20 Liquid Temperature Sensor	Amazon LLC	1	$8.49	$8.49
TEC1-12706 12V 6A Peltier Module	HiLetgo/Amazon LLC	1	$9.79	$9.79
Total Dissolved Solids Sensor	CQRobot / Amazon LLC	1	$11.99	$11.99
Insulated Travel Mug	YINBAOGE/ Amazon LLC	1	$7.96	$7.96
Voltage Regulators	Texas Instruments	3	$2.12	$6.36
ESP32 WROOM-1	Espressif/Amazon LLC	1	$10.89	$10.89
19W Meal Ceramic Ring Heater with 10 kic Thermistor	Thorlabs	1	$69.17	$69.17
Parts Subtotal				**$132.64**

4.3 Total Costs

Category	**Cost**
Total Labor	$19039.5
Parts Subtotal	$132.64
Machine Shop Service	$50
Grand Total	**$19222.14**

4.4 Schedule

Week	**Dates**	**Tasks**	**Member Responsibilities**
Week 1	02/24-02/28	- Confirm hardware and software MVP - Start on PCB design	Lahiru: Order hardware components Anirudh: Validate hardware components with Lahiru James: Confirm mobile app platform All: Start on PCB design
Week 2	03/03-03/07	- Design document - Setup github - Setup breadboard for demo	Lahiru: Order hardware components Anirudh: Organize and submit design document James: setup github and prepare software environment
Week 3	03/10-03/14	- Second round PCB orders - Revisions to machine shop if needed - Revise bread	Lahiru: submit PCB design Anirudh: Research improvements/alternatives to our current hardware components

		board via demo feedback	James: confirm any revisions needed to machine shop All: revise breadboard based off demo feedback
Week 4	03/17-03/21	SPRING BREAK	SPRING BREAK
Week 5	03/24-03/28	- Assemble PCB - Software MVP - Revise/test breadboard	Lahiru: Assemble PCB design and mark any important notes Anirudh: Work on connecting ESP32 with mobile app James: Work with Anirudh
Week 6	03/31-04/04	- Assemble PCB - Software MVP - PCB order if needed - Revise/test breadboard	Lahiru: Order another PCB if needed and notify team if any problems come up Anirudh: Collect data transfer latency to account in liquid heating James: Confirm data transfer between ESP32 All: Work on connecting hardware component to software
Week 7	04/07-04/11	- Assemble PCB - Finalize Software UI - PCB order if needed - Revise/test breadboard	Lahiru: Finalize breadboard Anirudh: Finalize software and hardware connection James: Finalize user-friendly software UI All: make any necessary

			adjustments according to material uncertainty
Week 8	04/14-04/18	- Finalize Software - Finalize integration testing with software and hardware - Prepare other component for demo	All: Finalize project Prepare for demo and go through practices
Week 9	04/21-04/25	- Revise based off mock from feedback to prepare for final demo - Prepare for mock presentation	Lahiru: Make any necessary hardware adjustments Anirudh: Make any necessary changes to software James: Organize and setup mock presentation powerpoint All: fill in powerpoint and prepare for the final demo. Divide presentation into parts for each person
Week 10	04/28-05/02	- Revise presentation based off mock presentation feedback	All: make changes to presentation based off demo feedback Edit individual parts in the presentation
Week 11	05/05-05/09	- Final presentation - Submit final papers, lab notebook, lab checkout	Lahiru: Submit Final presentation Anirudh: Lab checkout James: Submit final papers

5. Conclusion

We now summarize our project and our findings throughout the process.

5.1 Accomplishments

We built a functional cup that satisfies all the high-level requirements we set out to achieve at the start of the process. The user is able to receive information in real-time and provide desired parameters for their tea; the cup accordingly controls various peripherals to achieve those desired tea properties. At the same time, we learned a great deal, specifically about PCB design, soldering and microcontroller programming.

5.2 Failures

While we did achieve a great deal of success and build a well-functioning product, there were some issues we ran into. As mentioned before, we were unable to get the user interface working through Bluetooth, which would have been more convenient for the user than the laptop terminal. We also ran into some issues with short circuiting some of our hardware, specifically the ESP32-S3-WROOM microcontroller and our ceramic ring heater. Switching to the Peltier module also added some heating inefficiency, since it is a square shape, while the base of the cup is a circular shape. The surface-to-surface contact is not as high as it was with the ring heater.

5.3 Ethical and safety considerations

There are of course a few safety concerns, specifically in relation to 1.1 from the IEEE Code of Ethics [6]. For instance, since this is meant to be for tea, we will be dealing with very hot water that can seriously scald or burn a person. The lid of the cup should only come off intentionally; in other words, it cannot accidentally come off. In essence, we want to make it spill proof. Another potential safety concern is washing the cup. The user will need to do this for hygienic purposes. We want to make sure this is possible to do without the water coming into contact with the electronics, which could be dangerous.

1.2 from the IEEE Code of Ethics is also important to our project [6]. This discusses the importance of improving individuals and society's understanding of smart devices. Our cup would likely fall under this category, especially with the initial communication method of Bluetooth. Such connections can be intercepted or interfered with, and it is important for the user to be aware of this. Also, we can envision the application eventually storing data such as user preferences. We need to store user data, no matter how inconsequential it may seem, securely. One last more ethical aspect is more about avoiding waste and proper design. Our cup should be made of a series of parts that can be replaced rather than one singular device. This can avoid waste of materials and make it cheaper to get one's cup functioning again if something goes wrong.

5.4 Future work

One of the high priorities for future work would be to implement the UI connection through Bluetooth rather than the laptop terminal. This is much more convenient for the user and also makes our product portable. Of course, we would have to be careful in terms of security, as remote connections can potentially be more susceptible to malicious actors. Regarding the portability, we would also want the electronics to be stored compactly in a more robust base to make it more easily handheld. We would also want to enhance the heating efficiency to allow the cup to operate for even longer. Finally, a more aesthetically pleasing UI would make the product more attractive to customers.

References

[1] Tajammul Pangarkar, “Tea Statistics By Country, Tea Consumer Behavior and Facts,” Sci-Tech Today, Oct. 04, 2024. https://www.sci-tech-today.com/stats/tea-statistics/#General_Tea_Statistics (accessed Mar. 05, 2025).

[2] Penn Medicine, “The Hidden Health Benefits of Tea – Penn Medicine,” www.pennmedicine.org, Mar. 04, 2022. https://www.pennmedicine.org/updates/blogs/health-and-wellness/2019/december/health-benefits-of-tea

[3] R. Ashurst, “The Weight of an Empty Water Bottle: Uncovering the Mystery - KitchenJournal,” KitchenJournal, Mar. 05, 2025. https://kitchenjournal.net/how-heavy-is-a-empty-water-bottle-in-grams/ (accessed Mar. 05, 2025).

[4] “platform-espressif32/platform.py at ad7b231ca3bb561eb71febef067c3a0f28787ee9 · platformio/platform-espressif32,” GitHub, 2025. https://github.com/platformio/platform-espressif32/blob/ad7b231ca3bb561eb71febef067c3a0f28787ee9/platform.py#L114-L116 (accessed May 08, 2025).

[5] “The Identification of User Centered Design Factors Required for Products Intended for Extreme Environments | International Conference on Environmental Systems (ICES),” International Conference on Environmental Systems (ICES), 2024, doi: https://doi.org/10.2514/MICES10;journal:journal:6.

[6] IEEE, “IEEE Code of Ethics,” ieee.org, Jun. 2020. https://www.ieee.org/about/corporate/governance/p7-8.html

[7] “Performance Specifications.” Available: https://peltiermodules.com/peltier.datasheet/TEC1-12706.pdfAvailable: https://peltiermodules.com/peltier.datasheet/TEC1-12706.pdf

[8] ThorLabs, “HT19R2-SpecSheet,” Thorlabs.com, Apr. 08, 2023. https://www.thorlabs.com/drawings/8714775760f50cc6-CCE7FA48-C94E-EC64-254EB0C32C5D3AEC/HT19R2-SpecSheet.pdf (accessed May 07, 2025).

[9] “DIYables DS18B20 Temperature Sensor for Arduino, ESP32, ESP8266, Raspberry Pi,” DIYables Products. ~~https://diyables.io/products/ds18b20-temperature-sensor~~

[10] “TDS (Total Dissolved Solids) Meter Sensor SKU: CQRSENTDS01 - CQRobot-Wiki,” www.cqrobot.wiki. http://www.cqrobot.wiki/index.php/TDS_(Total_Dissolved_Solids)_Meter_Sensor_SKU:_CQRSENTDS01

[11] “150:1 Micro Metal Gearmotor MP 6V,” Pololu.com, 2025. https://www.pololu.com/product/2368 (accessed May 08, 2025).

[12] Energizer, “MAX PLUS E91 Datasheet,” data.energizer.com. https://data.energizer.com/pdfs/ep91_eu.pdf (accessed May 07, 2025).

Appendix A Requirement and Verification Table

Table 1 System Requirements and Verifications

Subsystem	Requirement	Verification	Verification status (Y or N)
Power	The battery should supply 6 - 9V with current draw of up to 160 mA.	1. Measure voltage using a multimeter and ensure that it is in the desired range. 2. Measure current using the current setting of the multimeter to ensure that the current draw is not too high and the motor does not stall.	Y
	The voltage regulators should output voltages within 5% of the desired value	1. Connect an oscilloscope to voltage output and measure average voltage ripple of the signal. We can also measure the input voltage ripple to check that the device rated ripple rejection ratio is within values as stated by the datasheet.	Y
Control	When the control system learns that the temperature of the water in the cup has reached the desired temperature, it should signal to the heating device to power off.	1. Put room temperature water into the cup, and set the desired temperature as 40 °C. Firstly, verify that the control system tells the heating device to stop its operation. Monitoring the water temperature manually (e.g. via a thermometer), verify that once the desired temperature is reached, the control system tells the heating device to stop its operation.	Y
	The control system should periodically start and stop the motor for the stirring aspect of our cup.	1. Start the brewing process via the control system. Verify that the stirring occurs every minute and lasts ten seconds each time. 2. Send a message from the control system to turn off stirring. Repeat the brewing process, and just ensure that no stirring occurs.	Y

Sensors	Using the temperature sensor, we should be able to detect once the water temperature is within 1-2°C of the desired temperature.	1. Put water from a tap into the cup and check the temperature given by the thermometer. Verify that the temperature sensor reads the same. 2. Continuing from the previous test, measure the water temperature in the cup using a thermometer. Change the desired temperature to the exact temperature of that water and then test for desired temperatures within 1-2 °C of that water temperature. Verify that the heater is toggled off as displayed in console 3. Continuing from test 2, set the desired temperature to be just outside our acceptable threshold of error (i.e. 3 °C away from the temperature of the water in the cup). Verify that the heater is toggled on via console commands. 4. We should perform a similar test for boiling water. This will of course be a bit risky due to the high temperature which can cause serious burns so we will need to be careful. However, it is necessary to do this test because firstly, most of the tea brewing temperatures are in the higher range, and secondly, for many water based temperature sensors, their tolerance values are less well known for those high temperatures.	Y
User Interface	Console can send commands to our microcontroller	1. Entering “set <integer>” prints new threshold temperature in console 2. Entering “motor_on/motor_off prints “motor on”/”motor off” in	Y

		console 3. Entering “status” prints information about temperature sensor data, current threshold temperature, and motor status	
Heating & Stirring	The heating element should start and stop within a few seconds upon request by the control system.	1. Fill the cup with room temperature water. Send the message via console to begin its operation. Continuously measure the temperature of the water in the cup and ensure that it is rising. 2. In continuation of the first test, after the water is sufficiently above room temperature (perhaps around 40 °C), send the message to the heating device via console to cease its operation. Continuously measure the temperature of the water and verify that it starts to go down.	Y
	The stirring element is also operated through the control system. By default, we would want it to turn on every minute and last for ten seconds. However, there can also be the option for the user to turn off the stirring element, which should be tested as well.	1. It is likely that the temperature differences are very minute anyway, but measure the temperature at the surface and the bottom of the water before and after stirring. If there is a discrepancy, after stirring it should be reduced.	Y
Cup	Cup weighs less than 500g	1. Place cup on scale and observe its weight	Y
	Ensure cup temperature is below 35°C when liquid in cup is 60°C	1. Observe cup temperature is consistently under 35°C for 5 minutes	Y
	Electronic safe from liquid	1. Physically see all circuitry is hidden	Y