Pour-over-and-over

Elijah Knupp, Rio Pacheco, Corrado Govea

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— A system capable of automatically brewing a barista-level cup of coffee utilizing the pour-over brewing technique. Typically, this process is difficult to learn and largely inaccessible to those with fine motor control deficiencies. Our machine will bridge this gap by requiring minimal input and effort from the user. It will have, at minimum, 5 different brewing presets (with customizable variables) and will be able to brew up to 300mL of coffee. The machine will have a precise and repeatable brewing process, with all variables being within 10% of what the user specified.**

***Index Terms*— Automation, Coffee, Gantry, Robot**

I. Introduction

THE world of coffee brewing is diverse and complex. There is a plethora of different brewing methods, each with their own advantages and disadvantages. However, there is a consistent pattern that emerges when one begins to explore these different options - quality and convenience do *not* typically go together. For the coffee connoisseur, they will focus on the quality of their coffee and will spend the time and effort required to make a premium cup of coffee. A popular choice for these individuals is the pour-over coffee brewing method, and for good reason. As the name suggests, this method requires an individual to pour boiling water over a bed of coffee grounds in a specific, time-intensive manner. They will repeat this process several times to brew a single cup of coffee. Their labor is rewarded with coffee that has full-bodied, highly specific, and differentiable flavors. Another advantage of this brewing method is the high level of customization it offers. Under normal circumstances, when the individual is using a more convenient and rigid coffee brewing process, their options to customize their cup of coffee are usually limited to choosing a different coffee bean. However, with the pour-over coffee method, users may change all the variables associated with the brewing process. This includes adjusting the flow rate, pour pattern, water temperature, and bed agitation. For the casual coffee drinker, who may just want a cup of coffee at the push of a button, this highly variable and detail-oriented approach to coffee brewing may sound like a nightmare. Additionally, this brewing method requires precise physical movements (with boiling water, nonetheless), which can cause accessibility issues for those with fine motor control deficiencies. This often forces those individuals to choose a more convenient coffee brewing experience.

We aim to bridge this gap between a convenient and quality cup of coffee through our machine: the Pour-Over-and-Over automatic coffee machine. After pouring in room-temperature water into the machine, pouring coffee grounds into the filter, and selecting the brewing option they desire, the user will not have to perform any further actions (other than removing the cup of coffee after the brewing process is complete!). Our machine will heat and pour the now-boiling water over the coffee in the exact manner the user specified. For the user looking fo a convenient experience, they can simply choose one of the 5 presets that will be programmed into the machine. For the coffee connoisseur, they may edit these presets or create an entirely new preset that will then be saved to the machine for later and repeated use. Our machine will prioritize precise repeatability, with us aiming for the capability of brewing coffee that is within 10% of quantifiable metrics of a cup of coffee brewed using identical variables. As the taste of coffee is highly subjective, this allows the user to determine what a good cup of coffee is to them, with our machine being able to reproduce that same cup of coffee using highly controlled quantitative variables and goals.

II. Use-Case Requirements

A. Capability:

To ensure we can make a full cup of coffee we want the device to hold 300ml of water. This will allow us to ensure that we have enough water to brew most if not all types of single cup recipes and ratios. Most pour-over recipes do not exceed 300ml of water, and when they do it ends up being a 2-cup recipe rather than 1-cup.

B. Parameter Accuracy:

To produce coffee to the specifications of a recipe the user desires we must ensure that individual parameters are accurate. These parameters are water temperature, poured water amount, and flow rate. It should be able to heat water up to a desired temperature, with a $\mp 5°F$ margin from the set temperature. The amount of water poured over the coffee grounds should be $\mp 10g$ from the desired amount. To ensure we can control coffee bed agitation we want the flow rate to be controllable, from 0g/s to 12g/s at most.

C. User Experience:

Given our goal of making this an accessible experience for people who are not familiar with specialty coffee or pour-over in general, we want to ensure that there is a plethora of options to choose from at the start. This means that the device should have a minimum of 5 preloaded presets to brew coffee with. Specifically, this means 5 different recipes, all with unique pour

patterns, timings, agitation, and end results.

D. Repeatability:

To ensure that users can trust that a recipe will produce the same cup of coffee every time (given the same coffee beans), we need to ensure that the amount of coffee extracted is consistent between cups. We will measure the TDS of the resulting cup and ensure that there is at most a ∓0.5% difference in TDS between cups of coffee.

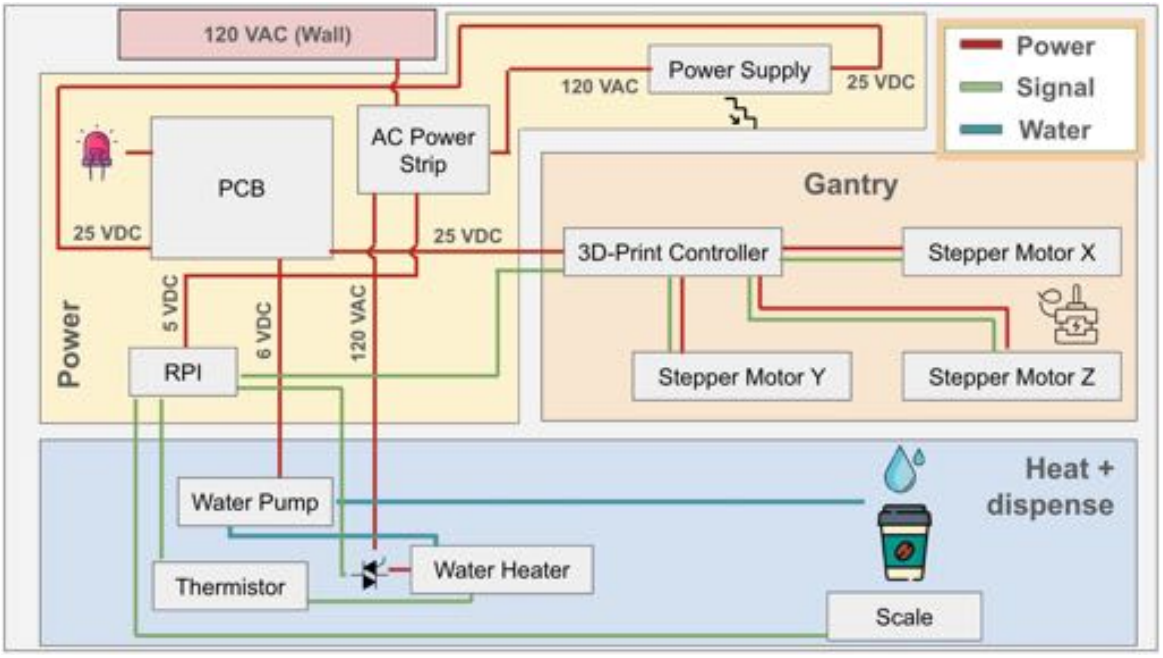


Fig. 1. Overall system diagram

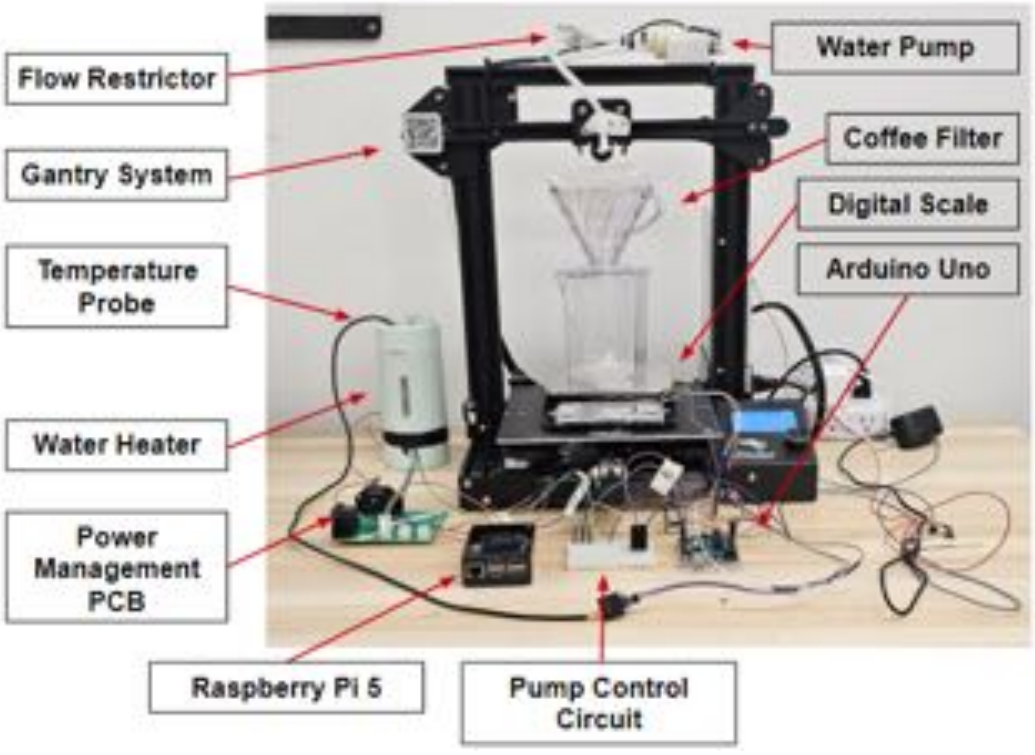


Fig. 2. Annotated Picture of Final Product

III. Architecture and/or Principle of Operation

Our system is made up of a 3-axis gantry system, a water distribution system, analog and digital microcontrollers, and a custom PCB for power distribution.

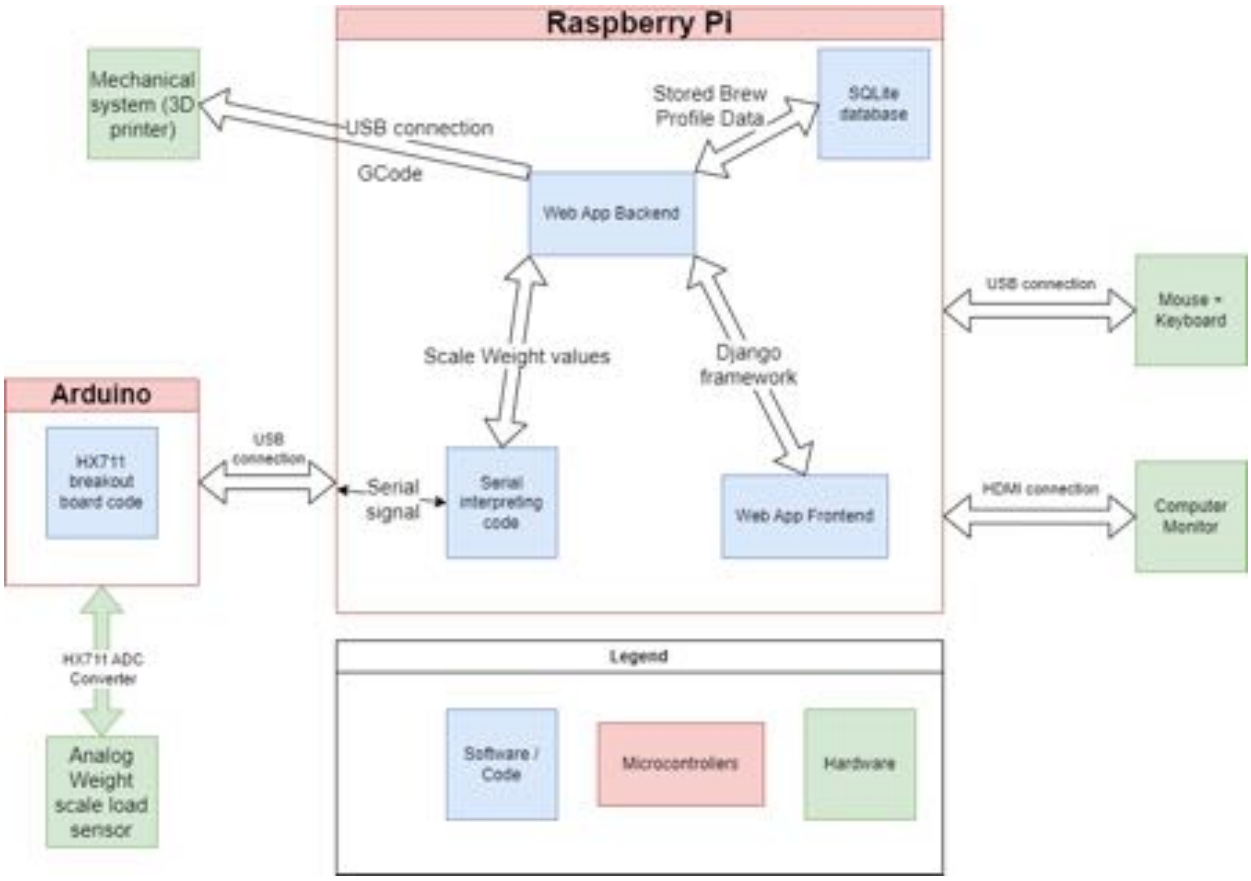


Fig. 3. Software Architecture

As per the hardware architecture plan, it can be divided into several subsystems: the water's heating and distribution system, the gantry and frame, and the scale. For the gantry and frame, the machine will use a re-purposed Ender 3 3-D Printer. All other components will either be integrated into the 3-D printer or mounted to its frame. The water will be heated in an electric kettle (which will also serve as the water reservoir) mounted to the frame of the 3-D printer. It will have a temperature sensor mounted to its lid to monitor and report the exact temperature of the water to the system. The water will be distributed using a food-grade, high-temperature water pump. The pump will be mounted to the 3-D printer's extruder head, replacing the fan that was previously mounted there. Concerning the scale, it will be integrated onto the hotbed of the 3-D printer. It should be noted that the hotbed will be disabled, so that it does not overheat and damage the scale. The scale will be used to provide information regarding the amount of coffee grounds in the filter and the total amount of coffee dispersed by the machine.

As per the software architecture plan (see Fig. 3) the Arduino will be running C++ code to interpret the analog signals from the weight scale. This will be sent over a USB connection to the Raspberry Pi, which will be running our Django web application. The Django web application will be written in Python, HTML, CSS, and JS. A JS script will take in the serial signal of the Arduino output and format the values to be sent to the web app backend code. The backend will handle interfacing with the database to retrieve and save brew profiles, serving the data to the front end, and sending printer movement details to the printer directly over USB. This will be in the form of GCode strings. The front end will be displayed via HDMI cable and interacted with via mouse and keyboard plugged into the RPI.

As per the power distribution plan, we take in 120VAC from the wall outlet into an off-the-shelf power strip, which will then power 3 things: the 25VDC power supply, the 120VAC kettle, and the 5VDC Raspberry Pi. The 25VDC will then connect to the power management board, which will step that down to 6VDC using an LDO circuit to then power the water pump. The 25V power supply will also connect directly to the gantry to power the 3-axis system through the 3D printer controlled. Lastly, we will route 120VAC from the power strip to the electric kettle. To control the AC current going into the kettle, we will use a triac circuit which will allow us to then regulate the flow rate coming from the pump.

IV. Design Requirements

A. Water Tank Capacity of 300mL

We require that our system can brew a full cup of coffee in order to be feasible for our market audience. In order to fulfil this requirement, the water tank, which in our case is the boiler, must be able to hold enough water to brew a full cup of coffee at once. One cup is equivalent to 236.6mL. However, some of the water will be absorbed by the coffee grounds – specifically, "about 0.5 ounce of fresh water is lost per cup of coffee" [1] which is equivalent to roughly 15mL. To have enough margin, our kettle is required to hold 300mL of water, which gives us a 19% safety margin. It is important to mention that from our testing, we realized that some of the total water dispensed goes into the pre-wetting stage, which doesn't contribute to the total water that goes into the final cup. However, thanks to this margin that we accounted for in the beginning, we were still able to maintain this requirement without needing to decrease the target or changing the design of the water heating system.

B. PID Temperature Control of +/-5 F

Part of our promise to our users is to be able to pour repeatable cups of coffee, and this can only be achieved with reliable parameter selection. For water temperature control, we have established that our PID heating system must have a +/- 5F accuracy from the set temperature throughout the pour.

C. Total Water Flow margin +/- 10g

An important parameter in pour-over coffee is the ratio of coffee grounds to water poured – this is because higher concentration of coffee leads to a richer end product, while a lower ratio leads to a lighter brew. To ensure the accuracy of our machine, we have set a design requirement of +/- 10g of water from the total set amount to be poured over the grounds – that is, total water poured before the grounds absorb any water.

D. Water Flow Rate Control Range

On the same line of keeping parameters consistent across pours, another very important one is water flow rate. This is because higher flow rate causes more turbulence during the infusion process. To ensure the turbulence is kept consistent, we are controlling the water flow through our pump at a range of 0g/s to 12g/s.

E. Default-Loaded Presets

One of the biggest goals of our project is to make pour-over coffee more accessible, such that it is suitable for experts and beginners alike. We figure that the best way to do this is to have default, pre-loaded presets with carefully curated parameters such that users can simply power on the machine, select a preset, and begin pouring. To give enough flexibility to beginner users, we chose to have a minimum of 5 presets loaded into the machine.

F. Total Dissolved Solids Control

Measuring the quality of coffee, our end result, can be very subjective. Parameters like "flavor" and "bitterness" are very hard to quantify, especially for non-experts in coffee. For that reason, we have chosen to evaluate our end result by measuring the total dissolved solids (TDS) in the final product. Our requirement is for +/- 5% error in TDS throughout the pours.

G. Accurate Patterns

The highlight of pour-over coffee – and our project – is having users create their own patterns for water pouring. For this reason, we want our pouring patterns to be repeatable. Thus, we have chosen our design requirement to be for the gantry to be able to reproduce the same pattern 5 consecutive times without going off-path by more than 5mm. This will ensure pouring patterns are consistent.

V. Design Trade Studies

A. Espresso Machine Parts vs Hardware Hacking

Initially we had considered using off the shelf parts used for espresso machines to be our water heating and distribution system. Our train of thought was that these parts have been industry tested and would likely be more than enough for our purposes. However, it was found that these parts would not align with our use case requirements. Specifically, the boiler we were considering could only hold a maximum of 100ml of water. This would not be nearly enough to heat the amount of water we needed to brew one full cup of coffee. The pump, which was designed to work in tandem with the boiler, would also need to be scrapped. This led us to look into hardware hacking existing solutions instead since heating water safely and effectively to boiling is not as easy as it seems. We ended up choosing to take an off the shelf water bottle heater and modify it with a custom PID loop and temperature sensors. This would allow us to heat it to a desired temperature, rather than only boiling. Also, since the bottle was already insulated it would mean that we did not need to worry about heat or water leakage onto nearby electronics.

B. Custom 2-axis Gantry vs. 3D printer

During our ideation phase we looked into creating a simple 2-axis system that would allow us to control the direction of water flow. One of the pros of this design is that we would be able to spec it to the exact size that we needed, thus making the system much more efficient. However, when we looked into pricing for all of the individual components it ended up costing us a similar amount of money to an off the shelf 3D printer system, but with much less functionality than a 3D printer. We ultimately decided that our goal was to make good coffee, not design a 2-axis system from scratch. Thus, we ended up choosing the 3D printer as it better suited our needs and also gave us another axis of freedom to play with as opposed to the 2-axis system we initially thought of.

VI. System Implementation

This section will provide a more in-depth breakdown of how the Pour-over-and-over machine is going to be implemented. It

will be broken down into three main sections – software, hardware, and electronics – each with a detailed breakdown of their respective subsystems.

A. Software

Arduino

The Arduino will be running code for the HX711 breakout board, pulled from a publicly available GitHub repository [2]. This code is written in C++ and will allow the Arduino to process signals from the scale and send weight values to the RPI.

Raspberry Pi

The RPI will be running two separate pieces of software. In the foreground will be our web application which the user will interact with to choose their brew profiles and set parameters. The web application, running on the Django framework, will allow users to interact with the machine. Through this web application, they will be able to set parameters such as water temperature and pour pattern, as well as change and create existing brew profiles. The data for these profiles will be stored on an SQLite database. The web application will also have a portion of its codebase dedicated to interpreting the signals received from the Arduino to allow us to read scale weight measurements while brewing coffee.

Previously, we planned to use Octoprint, an open-source system to interface with 3D printers, to control the movement of the printer. However, through further research and testing, we found that sending GCode directly to the printer over USB allowed for a much more efficient and responsive method compared to having Octoprint as a middleman. Thus, we added an algorithm that would parse user pour patterns into GCode on the backend, allowing users to easily choose pour patters without needing to worry about how it would be made understandable by the printer. Although this did require more testing than using Octoprint, it was easier in the long run as it was not difficult to debug.

B. Hardware

Gantry and frame

Rather than crafting a gantry from scratch, our team has opted to repurpose a 3-D printer for our gantry system. Specifically, we decided on the Creality Ender 3 (Figure 4). This has several advantages. Firstly, due to the popularity and availability of 3-D printers, this was a cost-effective solution (Under $200). This model comes mostly assembled as well, with the motors, pulleys, and gears pre-mounted to the frame. This avoids many integration and construction related issues. Additionally, the 3-D printer's gantry was built to perform motions and actions that we needed our gantry to perform as well. By simply removing the nozzle and tubing from the 3-D printer's gantry, we can quickly integrate the tubing and pumps necessary for our pour-over machine. We will remove the fan assembly and nozzle from the 3-D printer's extruder head, and replace it with a custom, 3-D printed frame on which the water pump will be mounted onto. As noted previously, the Ender 3 will serve as the base on which all the other components will be mounted on. To attach these components, including the water heater, PCB, scale, and Rpi, custom mounts will be 3-D printed.

Water Heating and Distribution

The water will be stored and heated in a 300mL electric kettle attached to the frame of the 3-D printer. It will be mounted in such a way that the kettle will be easily accessible for the user, so that they can easily detach and fill the kettle. The lid of the kettle will be modified so that a temperature probe can be mounted to it. This will allow the system to constantly read and evaluate the temperature of the water as the kettle heats it. This will allow the user to select an exact temperature (within a reasonable range) for which they wish their water to be dispensed at. The dispensing will be accomplished through a food-safe water pump attached to the extruder head of the 3-D printer through a custom 3-D printed mount. It will replace the fan system that is normally mounted there. The tubing will basically replace the filament tubing (it will be larger, so larger zip ties will be needed to attach it to the wire bundle). The original filament nozzle will be removed and replaced by the water tubing. As of now, there are no plans to add a nozzle to the end of the tubing, as it assumed that the tubing is small enough to dispense the water in a precise manner. However, if this is not possible, a 3-D printed nozzle will be created and attached to the end of the tubing.

Fig 4. Creality Ender 3 3-D printer:
https://www.creality.com/products/ender-3-3d-printer

Scale

The scale will be utilized to allow for precise measurement of the ground beans in the filter and the total amount of water dispensed from the machine after the brewing process is complete. It will be mounted on the 3-D printer's hotplate. As mentioned previously, the heating function of the hotplate will be disabled to prevent damage to the scale. The scale, as ordered, is not integrable into our system. To do so, we will need to add some components so that the Rpi can take readings from the scale. Firstly, the analog signal from the scale will need to be amplified and converted to a digital signal. This will be done through the HX711 AD/C converter and amplifier. This signal will be routed to the Arduino R3. Finally, through a USB connection, this data will be sent to the Rpi, where the data can now be utilized by the system [3].

C. Electronics

This section of the project focused heavily on the safe, consistent supply of power to the heating and dispense systems. The overall design philosophy was to make the system as convenient as possible to the user, which translated into having a single plug coming out of the printers, and having internal electronics distribute power to each system accordingly.

Heater Power Management Board

The first major component of this area was the heating element's Power Management Board. The purpose of this PCB was to draw current directly from a wall outlet (at 120V AC), deliver power to the heating system, and lastly, switch it on and off in order to be compatible with our temperature control algorithm. To accomplish this, we opted for using a TRIAC AC switching device, which we were able to effectively use as a MOSFET but for switching an AC supply. Because we were working with 120V directly from the wall, we needed to ensure that safety was a top priority in design considerations, and this applied both in safety to the user, but also to protecting the rest of the system. When it comes to the user, we added two main, fast-blow fuses to the PCB: one at the input (closest to the wall outlet), and one at the output (closest to the kettle). This ensured that in the event of failure or shorting (for example, due to a malfunctioning of the dispense system, or other liquid spills common in kitchen activities, causing water to reach the power management board, the excess in current would immediately trigger these fuses, causing the circuit to open and provide immediate safety to the user.

Lastly, to protect the rest of the circuit, the Arduino, and other electronics from the high voltage/high current supply, we opted for using an optocoupler between the small signal control MOSFET (triggered by the Arduino) and the gate of the TRIAC. The optocoupler uses an LED and a receiver to translate the small signal from the Arduino into a fluctuation of the AC line from the wall. The result was a completely electrically isolated HV/LV system, with proper component separation in the PCB, both the user and the circuitry were safe from malfunctions on the HV side.

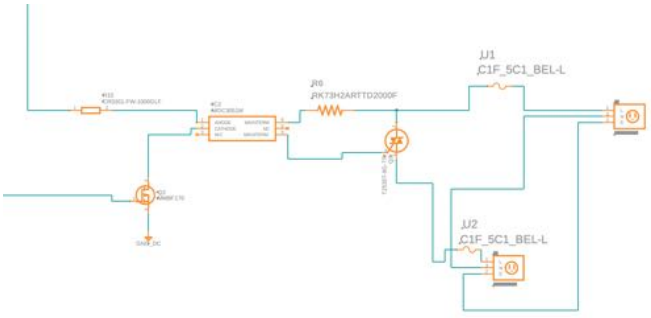


Fig 5. Heat Power Management Board Schematic

Pump Flow-Rate Control Circuit

The pump control circuit was one that changed significantly from our initial design, and this was due to the fact that we changed our pump selection for one with a much higher current draw (300mA to 2A). Several issues were tackled during the design and debugging of this circuit, which will be highlighted below.

The heart of this circuit is a power MOSFET, suitable for high current applications like switching our new 2A pump. The first issue that we encountered was that, even when turning the power MOSFET on with a power supply, the pump itself would only see a voltage drop of 1V as opposed to the expected 5V – not enough to turn it on. After thorough inspection of the power MOSFET's datasheet, we encountered that the internal resistance of the MOSFET was 2 ohms. This is usually not a problem with small signal transistors; however, since we were drawing 2A from the pump, by applying ohm's law to the MOSFET, we'd see that there would be a voltage drop of 2ohm * 2A. For this reason, we increased the voltage supplied to the series circuit from 5V to 9V to account for the voltage drop, and this allowed us to turn the pump on.

The second issue that we encountered was when migrating from switching the power MOSFET with a power supply to switching it with an Arduino PWM pin. When doing so, we were seeing a non-deterministic behavior of the MOSFET, where sometimes it would fully turn on, other times it would partially turn on (and get very hot), and other times it wouldn't turn on at all. Everything seemed to indicate that it was an issue at the gate, since we only changed the power supply. After much research, we determined that the issue was caused by the gate not getting enough current to fully charge (which is not as high of a requirement in small signal MOSFETs) and therefore couldn't be directly supplied by the Arduino's PWM pin. To mitigate this, we used the same 9V power supply that feeds the pump to feed the gate of the power MOSFET, and switched that signal with a small signal MOSFET (since the current is low enough). The result was a circuit that used a common 9V supply and two MOSFETs to drive the pump at the desired voltage level to control the water flow.

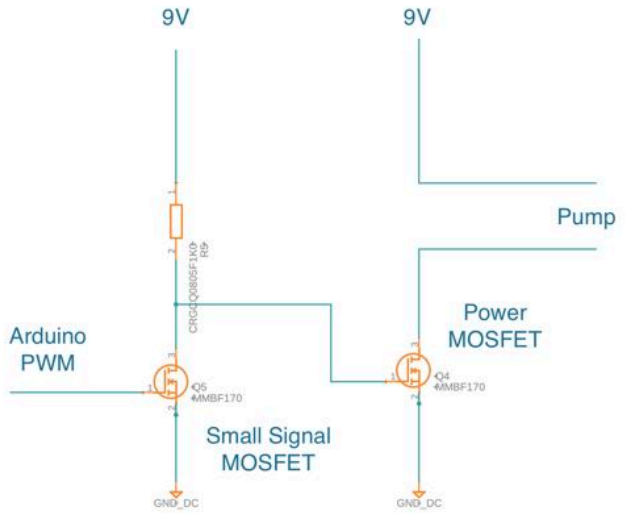


Fig 6. Pump Driver Circuit Schematic

VII. Test, Verification and Validation

A. *Results for Water Tank Capacity*

This was a simple test to confirm that the kettle we purchased was able to hold 300mL of water as per our design requirement. In order to account for water loss due to evaporation, we conducted this test with hot water using a test script. We then ran a full profile and measured the total water dispensed using the integrated scale, to ensure the water poured was within bounds. The result was that we were able to pour the full 300mL before the kettle emptied out. One thing we highlighted during this test was that part of the 300mL dispensed were part of the filter pre-wetting stage, a crucial step in the brewing process. With this information, we had to decide between removing this step and allowing the full 300mL to go into the pour, or to slightly limit the actual poured water over the beans. The decision was to keep the pre-wetting stage, since, as discussed in section B, enough water was still poured to meet our use-case requirement.

B. *Results for Total Water Poured to Fill Cup*

After validating the system design requirement in section A, we ran a full profile using coffee grounds in the filter, and observed the total water poured into the cup. As per our use-case requirement, we confirmed that enough water was poured into a standard (300mL) cup to fill it up by more than 2/3, which meets our requirement.

C. *Results for PID Temperature Control within 5deg F*

To conduct the experiment and test our PID loop system, we inserted a thermocouple into the kettle with water. We set a desired temperature in the system, and let the water reach the desired temperature. After the machine confirmed the water had reached steady state and went into "ready mode", we started measuring the water temperature for 4 minutes (which is longer than the expected pour time of 3:30 minutes). We would then confirm that the temperature didn't go more than 5F off of the set temperature. We repeated this test at 90C, 95C, and 100C, with the following results:

Temperature **Target** (°F)	Temperature **Min** (°F)	Temperature **Max** (°F)
180	177	183
195	190	200
205	203	207
212	212	212

This test confirmed that our design requirement had been met by directly measuring the water temperature in the kettle. The results above directly confirmed that we had met our use-case requirement, since we were able to provide the confidence to the user that the system could provide consistent temperatures throughout the pour, and therefore the brew quality and extraction levels wouldn't be affected by major fluctuation in the water temperature.

D. *Results for Total Water Flow Margin*

To test total water flow, we selected a desired pour volume and let the machine pour into an empty cup (since this requirement refers to water amount before it's absorbed by grounds). We then measured the total mass poured with a kitchen scale and validate the desired margins. We repeated this test with 180g, 200g, and 220g of water, which are around the expected uses for the machine. This test was performed as we continuously tuned the system's parameters to account for any physical margins in the algorithm. After completing the tuning stage, we were able to validate that all of our pours, across the different presets tested, were able to consistently pour a total amount of water within +/- 10 grams of the set desired volume.

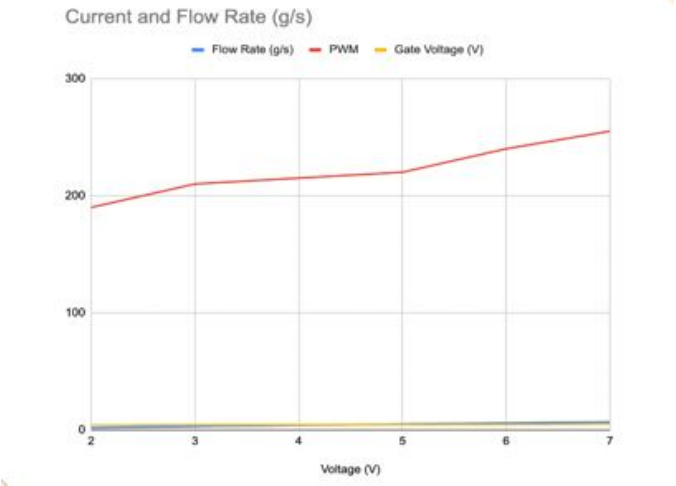
We know from our studies that our users value the richness of the coffee, and this is what motivated us to have a requirement for total water dispensed. From this test, we validated that we met our use-case requirement of pouring a precise amount of water over the coffee grounds, to ensure that no excess water (which would make the final product less rich) was poured, or that too little water was poured (which would cause an incomplete extraction from the coffee grounds).

E. *Results for Water Flow Rate Control Range*

This was a similar test to that of water volume, but with an added time component. We selected a constant flow rate and turned the pump on. Initially, we had considered using our integrated scale to manually calculate the flow rate from the

volume dispensed / dispense time. However, we were able to use a specialized scale that had a pre-programmed flow rate measurement, which have us more precise values than our original test design. The results of this test are laid out in the table and graph below:

Measured **Flow Rate**	**PWM** Applied to MOSFET Gate
7 (MAX)	255
6	240
5	220
3	210
2 (MIN)	190



In the end, helped us confirm that our parameter selection for flow rate is accurate, and thus would give the users the confidence that they will reliably get the desired extraction rate out of their grounds, since this is so affected by the turbulence controlled by water flow rate.

F. Results for Default-Loaded Presets

To test for this design requirement, we first confirmed that all 5 presets were accessible on the web application without any further modification (which represents the idea that these presets are loaded into the machine upon startup, when a user wither buys or builds the product from the project's repo, since it is open source). Then, we actuated the presets to make sure that they all worked properly based on the rest of the design requirements. The result of this test was that all presets loaded into the web application were accessible, and all produced the desired outcome based on the design requirements.

G. Results for user experience

Focused now on the use-case requirement of providing an intuitive, straightforward and accessible experience to the user, we conducted a survey with a sample size of ten randomly selected students which assessed our product in 4 categories which we found to be crucial to meet this use-case requirement. The results of this survey, which was heavily focused on the web application design, are shown in the table below:

Question Asked	Average Result
How useful were the features (tare, profile creation, sorting)?	**4.8**
Was it easy to navigate?	**5**
How satisfied are you with the functionality?	**4.3**
Do you see yourself using this at home?	**4**

The quantifiable target we set for this requirement was scoring at least a 4.0 on all four categories. The results showed that we were able to meet our use case requirement. One thing that we noticed from these results was that the last question, asking whether users would see themselves using our product at home, was closer to the lower bound for our test. We concluded that this was likely due to the fact that when we conducted the survey, only the webapp was at a stage of development where it was attractive to users – the rest of the system (gantry and electronics) were still scattered into their subsystems, and thus we could attribute concern in the users surveyed to this.

G. Results for Accurate Pattern Reproduction

To test that our gantry can produce accurate pouring patterns, conducted a simple test. We attached a dry-erase marker to the pouring head of the machine, and then we had it produce the same pattern 5 consecutive times on the same white board. Then, we visually inspected the patterns produced with a ruler, to make sure that no drawing deviated by more than 5mm from the original path. This simple –yet obvious– parameter of being able to produce accurate patterns is something we need to ensure in our design. After all, this is what differentiates pour-over coffee from many other brewing alternatives. By confirming that the patterns are repeatable, we are closing the set of variables that can greatly impact the final product and ensure that our design produces consistent cups of coffee for users to enjoy. The results showed that we were able to produce the same pattern within the bound constraints.

H. Results for Accurate Water Dispense Pattern

In order to finally validate that our gantry was able to repeatably pour consistent patterns using hot water (as opposed to the dry-erase marker), we repeatedly ran the filter pre-wetting step as a test script. This step is expected to go around the perimeter of the filter, having water reach the entire surface. By visually inspecting the filter in these trial runs, we were able to conclude that our water dispense system, with the full gantry integrated, was able to meet our use-case requirement of pouring consistent patterns across brews, which ensures to the user that their brewing quality, extraction levels, and turbulence created would be consistent across pours.

I. Project Management

a. Schedule

The schedule remained quite consistent since the design presentation. The only major change was that, due to time constraints, we decided to keep the first revision of the PCB for the high voltage circuit, and opted for using a proto-board for the LV pump circuit. This meant that the time initially allotted to PCB lead time was used for circuit debugging, which proved to be critical time for completing the debugging stage. Please refer to table II in the appendix for the final schedule.

b. Team Member Responsibilities

Overall, for our project, the task division was as follows: Rio was in charge of software development, which includes implementing a user interface for creating and selecting presets, enabling saving presets into the machine's storage, and communication between the RPI to the gantry. Elijah was responsible for developing the gantry, as well as the scrips and algorithms to control the actuators in the system. This included nozzle movement, dispense, and heating, as well as all necessary sensors like weight and temperature. Lastly, Corrado worked on the custom electronics, which focused on power electronics design – this included a custom power management PCB for AC heater control, as well as high-current pump flow regulation.

c. Bill of Materials and Budget

Refer to Table I in the appendix of this report to locate the bill of materials and budget specifics. Since the design report, we ordered several parts that replaced the originally ordered parts. Parts that were no longer used are written in *red.* Parts that are new to the project since the design report are written in *blue.*

d. Risk Management

The main risks that we anticipated for this project were related to the compatibility of off-the-shelf components. Although this was a calculated risk, we understood that there was the chance that there could be variables or ratings that we didn't account for, and could lead to the need for a re-design or re-ordering of components.

From the design standpoint, we made sure to release all designs needed for the MVP within the first 4 weeks of development. This sprint allowed us to identify several issues early on – for example, we noticed that our pump of choice did not have enough power to prime itself, and that meant that we needed to order a new pump, but also completely redesign the control circuit in order to handle almost 10 times the originally expected value.

In terms of budget, our initial orders amounted to less than $400, which left us 33% of the total budget left for eventualities. This margin enabled getting the new pump, coffee beans and cups for demo day (which we hadn't initially accounted for), and other miscellaneous items used in development and debugging.

Lastly, regarding the schedule, we were impressed by how crucial it was to have an initial plan with several weeks of slack before the final demo – specific issues in debugging and integration, like learning how to properly drive power MOSFETs for the new pump, took upwards of 40 total man-hours of work and literature review, which wouldn't have been possible without this pre-determined slack time.

All in all, the combination of schedule and budget margins, along with the decision to release all designs within the first month of the project, allowed us to deliver a completed project that met all of the specifications laid out in our ideation stages.

J. Ethical Issues

The main ethical consideration for this automatic pour-over coffee machine is its potential impact on public health and well-being. Easier access to quality coffee could lead to overconsumption of caffeine, causing sleep problems or anxiety. This is similar to existing pod-based machines, but since this machine wouldn't grind beans, its impact might be slightly less.

Another concern is public safety. Caffeine can counteract the effects of alcohol, and easier access to caffeine could enable people with substance abuse issues to drink more alcohol. This could be mitigated by a tracking system limiting daily coffee intake.

Finally, there's a potential negative impact on local coffee shops. Easier home brewing could lead to fewer people buying coffee at cafes. However, this machine requires coffee beans, and users could still support local shops by buying beans there. The machine could even suggest local shops to users.

K. Related Work

Some other related work to our idea is the Xbloom and the Poursteady. The Xbloom looks to serve a group of people who already use pod-based machines such as Keurig or Nespresso but allow them to experience and experiment with specialty coffee and pour-over. The issue with this is there is no specificity in choosing brew parameters, as you can only change them based on an arbitrary visual scale. The expense of the machine as well as the pods proves to be unsustainable in the long term. On top of this, the added material waste of

buying pods as well as the packaging for them proves to be much larger than just making pour-over coffee the traditional way. Poursteady looks to automate pour-over coffee for a cafe setting. This machine costs a whopping $11,000 at its cheapest, making it completely impossible to buy for an average consumer. While it can be useful for cafes it doesn't allow for customization of pour parameters at all. There is only one button on the machine which is to start the brew process.

We believe that compared to our end product, we have achieved a similar brewing experience as an Xbloom and other options at a fraction of the cost, with even more customizability. Users are allowed to use their own kettle, customize pour patterns as well as profiles, and utilize many different parameters in the brewing process that Xbloom does not allow for.

L. SUMMARY

Our goal was to create an open-source pour-over coffee machine that aimed to bring specialty coffee to any user's home. By automating the process, we aimed to open the door to consistent, high-quality pour-over coffee to experts, enthusiasts, and beginners alike. Our key success metric was repeatability, which seems to be our biggest differentiator from traditional techniques of pour-over.

Overall, we are excited to report that we met our goal. In a recent demonstration, we brewed eight batches of pour-over coffee over the course of three hours (with some people coming back for seconds!). With that being said, our machine still has its limitations. The pump was too powerful to reach the range of flow rates that we initially aimed for. With more budget, we could have purchased a better pump to solve this. Additionally, we would have like to add a couple more user features, like learning module that informed users how to make their own profile (and why changing certain variable changed the attributes of the coffee). Finally, we wished we would have ordered a larger water kettle to hold more water.

a. Future work

We believe that this machine has a lot of potential beyond the scope of this class. Although there are no current plans to continue developing this product, the natural next step (after adding the features mentioned previously in the summary section) would be to work with a vendor to source parts, potentially market and raise funds, and then produce this machine on a large scale.

b. Lessons Learned

We learned so many valuable lessons as both individual engineers and how to work on a team of engineers. Perhaps one of the biggest lessons we learned during this project was to read the data sheets of all of the components we integrated into our project. There were several times in which we could have saved ourselves many hours of troubleshooting if we took the time to read the datasheets - even the parts that did not seem pertinent at the time.

Concerning the team aspect, we learned that respect for each other and timeliness played a key role in keeping our team chemistry strong!

GLOSSARY OF ACRONYMS

BOM - Bill of Materials
PCB – Printed Circuit Board
RPi - Raspberry Pi
TDS - Total Dissolved Solids
RPi – Raspberry Pi

REFERENCES

1] KitchenAid "COFFEE MAKER MAKING LESS COFFEE THAN SELECTED", 02 Mar 2024. https://producthelp.kitchenaid.com/@api/deki/pages/2512/pdf/Coffee%2bMaker%2bMaking%2bLess%2bCoffee%2bthan%2bSelected.pdf

[2] BOGDE on Github "An Arduino library to interface the Avia Semiconductor HX711 24-Bit Analog-to-Digital Converter (ADC) for reading load cells / weight scales." Web, 17 Mar 2022 https://github.com/bogde/HX711

[3] Scalafiotti, Edo. "Build a Scientific Scale and Process Its Live Data Using Arduino, RaspBerryPi and Socket.io." Medium, 15 July 2018, medium.com/@edoardo849/build-a-scientific-scale-and-process-its-live-data-using-arduino-raspberrypi-and-socket-io-13a5671011d9.

TABLE I. BILL OF MATERIALS

Item Name	Purpose	Manufacturer	Quantity	Cost @	Total
Raspberry Pi	Control	Raspberry Pi Foundation	1	Free	$0
3-D printer	Gantry	Creality	1	$168	$168
Water heater	Water heating	DREAMOSA	1	$20	$20
Water pump	Water transport	Lightobject	2	$14	$28
10mm Silicon tubing	Water transport	Quickun	1	$10	$10
12mm Silicon tubing	Water transport	ANPTGHT	1	$17	$17
Temperature probe	Sensing	BOJACK	2	$9	$18
Refractometer	Verification	Xindacheng	1	$20	$20
HTPLA filament	Printing custom parts	Protopasta	1	$30	$30
Custom PCB	Power management	PCBWay	1	$70	$70
AC power strip	Power management	4 LEAF	1	$15	$15
Scale	Weight measurement	BAGAIL	1	$15	$15
AD/C Converter	Weight measurement	Arduino	2	$18	$36
Arduino R3	Weight measurement	ELEGOO	1	$17	$17
Water Pump	Water Transport	Delinx	1	$17	$17
				Grand Total	**$481.00**

TABLE II. TEAM SCHEDULE