ECE 445 Final Project: ShowerSync

By

Keshav Dandu

Edward Xiong

Reet Tiwary

Final Report for ECE 445, Senior Design, Spring 2024

TA: Nikhil Arora

1 May 2024

Group No. 21

Abstract

Our project introduces a novel self-adjusting shower knob system designed to automate the process of regulating water temperature in showers. The system incorporates a temperature sensor positioned within the shower stream, a motorized mechanism attached to the shower knob, and a microcontroller for processing temperature data and controlling the motorized adjustment. Through extensive testing and experimentation, we demonstrate the system's capability to accurately adjust the shower temperature to match user preferences, thereby enhancing comfort and convenience during showering. Additionally, the system operates on battery power, ensuring safety and portability without the need for electrical connections near water sources. Overall, our findings showcase the effectiveness of the self-adjusting shower knob system in improving user experience, promoting water conservation, and enhancing safety in bathroom environments.

Contents

1. Introduction

This report addresses the science and engineering problem of manual shower temperature adjustment, which often leads to inconvenience, water wastage, and safety hazards in wet environments like bathrooms. We introduce a self-adjusting shower knob system to enhance user experience, conserve water, and improve safety. It will detail the system's design, implementation, testing results, and cost as well as schedule implications. Additionally, it will discuss project achievements, challenges, future work, and ethical considerations.

1.1 Problem

Our problem focuses on how we can redesign the shower system in a way that minimizes the time taken by users to preset their shower through temperature settings and knob adjustment, but also reduces the water as well as energy consumption without compromising the comfort and effectiveness of the showering experience. Additionally, keeping in mind that the entire shower system mechanism should be cost-effective.

1.2 Solution

Our solution is a self-adjusting shower system designed for efficiency and sustainability. It features a motorized knob mechanism that attaches to existing shower controls and a temperature sensor located underneath the shower faucet that regulates water flow to ensure only water at the desired temperature is used. Excess water is diverted into a removable/portable container via a tubing system for reuse. The entire system is controlled by a microcontroller with Bluetooth connectivity, allowing remote temperature adjustments. It also includes a low-level mechanism to consider adjusting the faucet-pin for the shower system to officially start. This setup not only saves water and time but also offers a customizable shower experience through a user-friendly web/mobile based application, providing real-time temperature feedback.

1.3 High-Level Requirements

These are the High-Level requirements we aimed to solve with ShowerSync:

1. When the user requests, ShowerSync will rotate the shower handle and divert water until the user's desired temperature is reached **(within ±4 degrees)**.

2. The shower system notifies the user when the desired temperature is attained, initiating the faucet-pin subsystem **within 25 seconds**.

3. The entire device process begins **within 10 seconds** of remote command.

2 Design

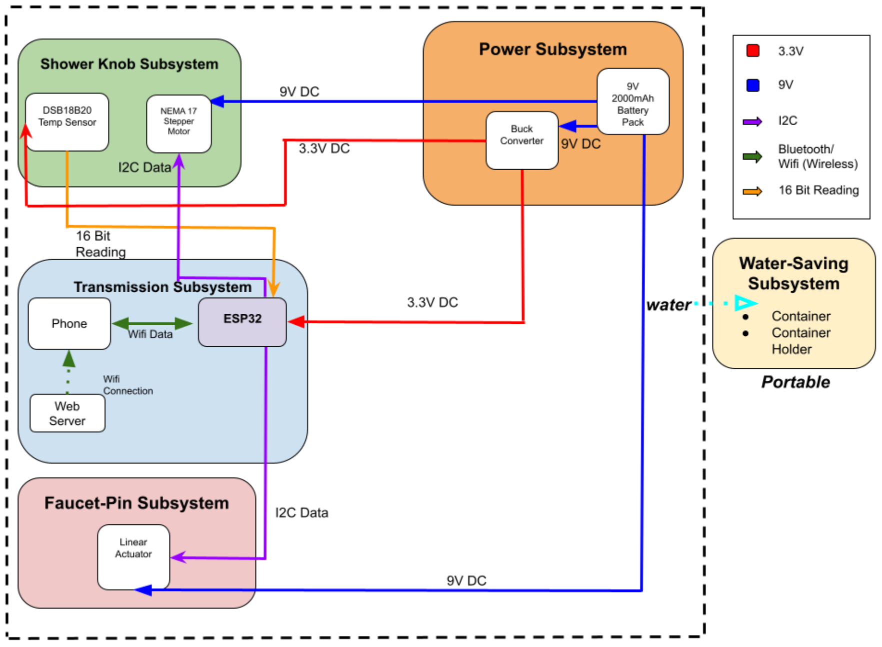

Figure 1: Block Diagram of ShowerSync Subsystem Design

2.1 Block Diagram

The Power Subsystem energizes shower components, supplying 9V to the NEMA17 Stepper Motor and linear actuator and 3.3V to the DSB18B20 Temperature Sensor.

The Transmission Subsystem, interfacing with a web/mobile app via WiFi, coordinates temperature requests using ESP32 and I2C data transfer.

The Shower Knob Subsystem, driven by a stepper motor and temperature sensor, regulates water temperature, integrating the Water-Saving Subsystem when needed. Upon temperature attainment, the Microcontroller triggers the Transmission Subsystem to activate the Faucet-Pin Subsystem via a linear actuator for showering.

**Our project focused on the Microcontroller, Transmission, Power, and Shower Knob subsystems, with the Faucet-Pin subsystem's integration pending due to PCB issues.

2.2 Visual Aid

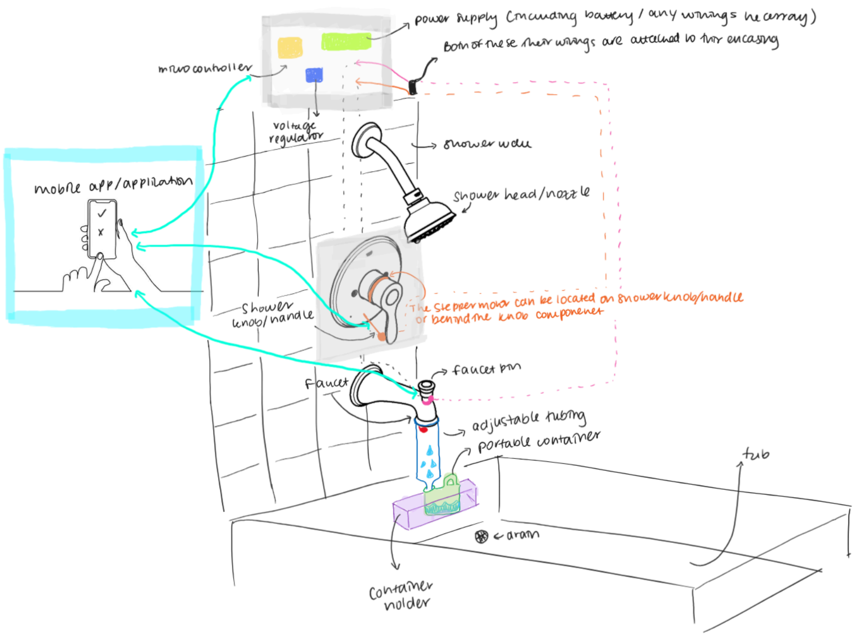
power supply (including battery / any wirings necessary)
Both of these their wirings are attached to this encasing
microcontroller
voltage regulator
shower wall
mobile app/application
shower head/nozzle
Shower knob/handle
The stepper motor can be located on shower knob/handle or behind the knob componenet
faucet pin
Faucet
adjustable tubing
portable container
tub
drain
Container holder

Figure 2: Visual Aid of ShowerSync

We showcased a schematic of our proposed shower system. The waterproof power subsystem is situated above the shower head. The shower knob subsystem features an adjustable knob handle and a temperature sensor beneath the faucet. The faucet-pin subsystem coordinates with an actuator adjacent to the faucet-pin. Furthermore, the water-saving subsystem consists of tubing, a portable container, and a holder for storing excess water for reuse.

2.3 Physical Design

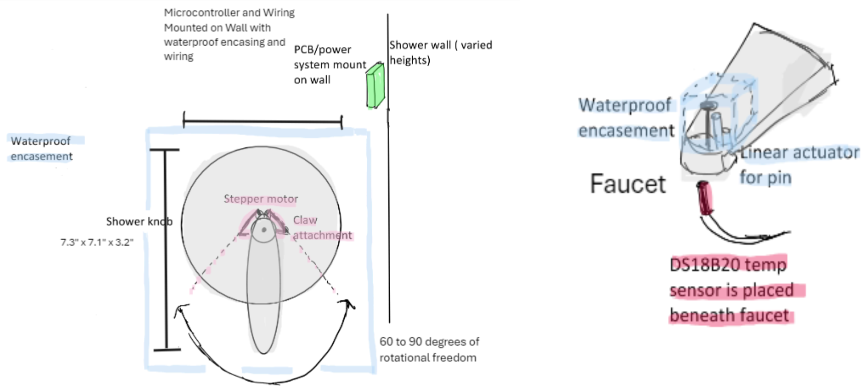
Microcontroller and Wiring Mounted on Wall with waterproof encasing and wiring
PCB/power system mount on wall
Shower wall (varied heights)
Waterproof encasement
Stepper motor
Claw attachment
Shower knob
7.3" x 7.1" x 3.2"
60 to 90 degrees of rotational freedom
Waterproof encasement
Linear actuator for pin
Faucet
DS18B20 temp sensor is placed beneath faucet

Figure 3: Physical Design with measurements of ShowerSync

The physical design we provided focuses on the exact measurements for our physical design and prototype idea so it would be easier to understand. The following headers show the actual setup and structure.

2.4 Physical Implementation

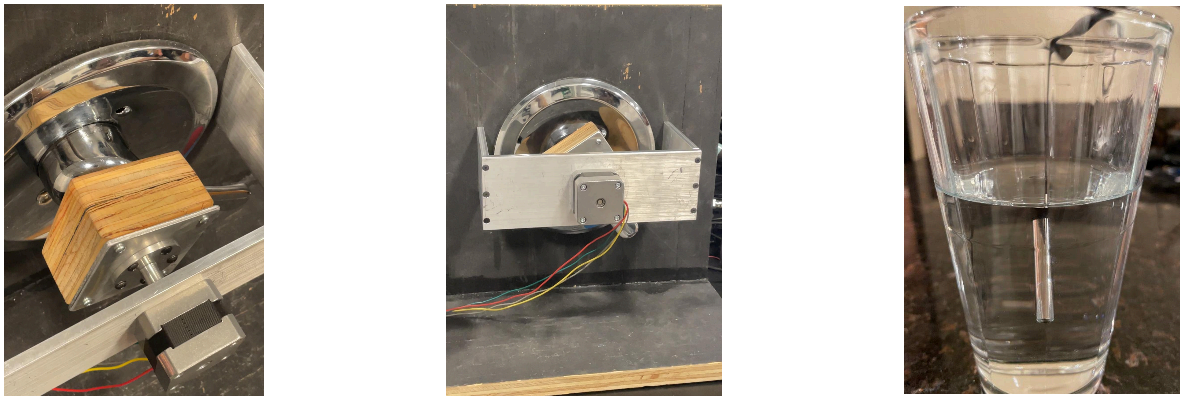
Figure 4: Actual Shower Knob Handle with Water Temperature Sensor within a glass of water

The physical implementation shows what exactly we worked on and what we used for our final demonstration, the water temperature within the glass of water would read the temperature and move the shower knob handle that has the stepper motor mounted properly.

2.5 PCB Design

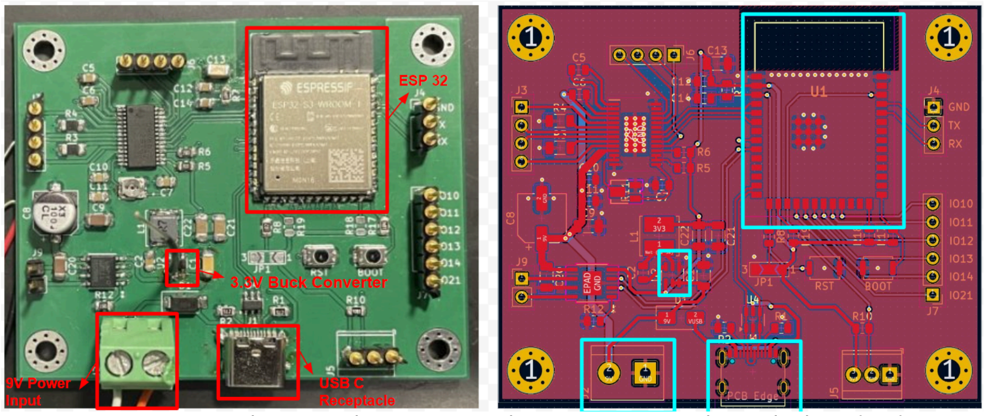

Figure 5: Actual PCB with Design Implementation on the right hand side

This was our PCB design as we pointed out the crucial components where the power was to supply most of our subsystems. We used datasheets to figure out which components at which values would be necessary to use along with cross checking our work with TAs when we were confused.

2.6 Web Application

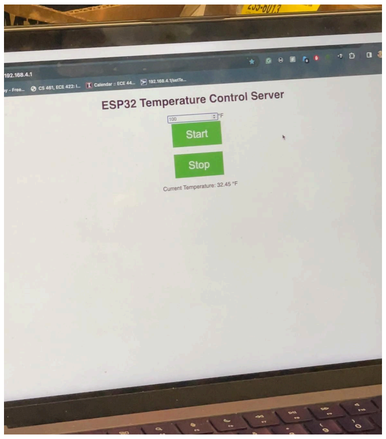
ESP32 Temperature Control Server
Start
Stop
Current Temperature: 32.45 °F

Figure 6: Actual Web Application

Our web-based application displays the current water temperature reading (ex: 100 degrees Fahrenheit) and allows users to input their desired temperature. Users can start the shower knob system, and once the desired temperature is reached, they can request a stop. This action triggers the faucet-pin subsystem, but in our case, we allow the shower knob handle to return to its resting state as compensation.

2.7 Subsystem Overview

Below we highlight the separate five subsystems that made up ShowerSync.

2.7.1 Shower Knob Subsystem

The Shower Knob Subsystem, housing the DS18B20 temperature sensor and NEMA 17 stepper motor, connects to the ESP32 microcontroller for receiving preset user information. The stepper motor is powered by a 9V battery pack, while the temperature sensor receives regulated 3.3V from a buck converter. It adjusts the temperature based on user input and signals the microcontroller once the desired temperature is reached. The web app notifies the user, prompting the faucet-pin subsystem to initiate the shower process. The water-saving subsystem stores excess water not meeting user preferences for external use.

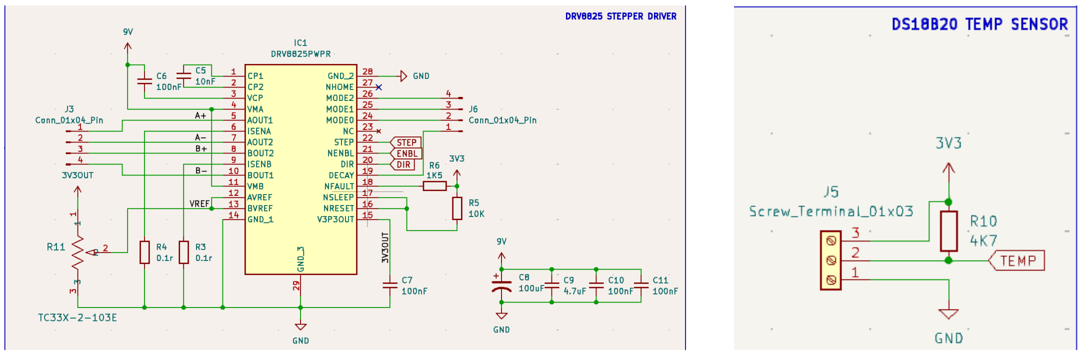

Figure 7: Schematic of DRV8825 Stepper Motor Driver and DS18B20 Temperature Sensor

Requirements

For the Shower Knob subsystem we had two main requirements:

1. The stepper motor should be able to react to the temperature sensor and move the handle accordingly within a range of 60-90 degrees.
2. The temperature sensor could accurately read the temperature with an error of 4 degrees

Design Choices

In our Shower Knob Subsystem design, we selected a NEMA 17 stepper motor for precise step control, enabling a variety of handle adjustments to regulate speeds based on temperature. Despite its nominal 12V requirement, we operated the motor at 9V to maintain a suitable voltage difference between the ESP and motor, ensuring ample power to move the shower handle. The DS18B20 temperature sensor was chosen for its waterproof features, crucial for accurately measuring water temperatures.

2.7.2 Power Subsystem

The power subsystem is crucial for the project's functionality, enabling communication between subsystems. Initially, we planned to use an LM317 Voltage Regulator with a 9V (2500 mAh) battery pack, but opted for a buck converter to regulate voltage directly from 9V to 3.3V. Encasing the power supply to protect it from water damage is a safety measure.

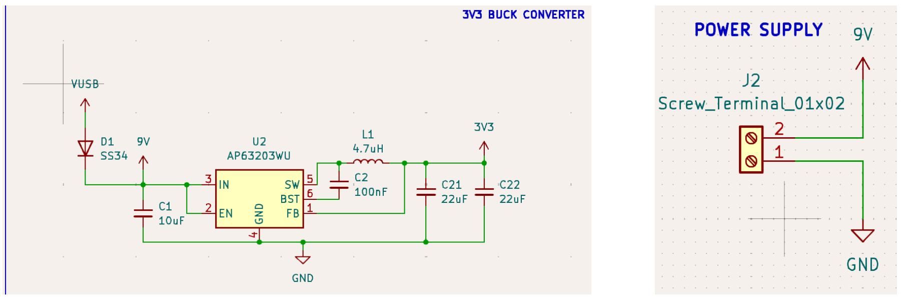

Figure 8: Schematic of Buck Converter and Power Supply Converter

Requirements

For the Power Subsystem, the requirements were very straightforward. Make sure that each component receives the necessary voltage.

Design Choice

We decided to use a 2500mAh battery pack to ensure a longer battery life for the design. Below are the calculations based on a 10-minute shower estimate. To account for potential losses to heat and voltage variation, we assume that only 70% of the 550mAh is available to use.

$$0.7 \times 550mAh = 385mAh$$

Our total power consumption needed from all our parts can be calculated as follows:

$$150mA + 100mA + 1mA = 251mA$$

Finally, we can calculate the total time our system can run:

$$0.7 \times 2500mAh = 1750mAh$$

$$1750mAh/251mA = 6.97\ h$$

Assuming 10-minute showers, this comes out to be around **41 showers** which we believed to be sufficient.

Additionally, Our initial design consisted of using two LM317 regulators and regulating from 9V to 5V to 3.3V. This introduced an unnecessary step to 5V since none of our subsystems required 5V to operate. However, because of the voltage regulation calculations supplied below, we needed the extra step of regulating to 5V to ensure that the circuit does not overheat.

$$T = i\,out * (V\,in - V\,out) * (\Theta\,ja) + T\,a$$

Equation 1: General equation for calculating regulator voltage

$$T = 332mA * (9 - 3.3) * 100 + 40 = 229.24C$$

Equation 2: Regulating from 9V to 3.3V. Resulting temperature is much too high

$$T = 332mA * (5 - 3.3) * 100 + 40 = 96.44C$$

Equation 3: Regulating from 5V to 3.3V.

Design Changes

As mentioned previously, we needed to regulate twice to ensure our circuit would not overheat. However, this introduced an extraneous 5V voltage that we would never use. As such, we decided to switch to using a buck converter design that would allow us to regulate 9V to 3.3V without introducing an extra 5V.

2.7.3 Faucet-Pin Subsystem

The Faucet-Pin Subsystem activates when the ESP32 microcontroller detects the user's desired temperature through the temperature sensor. The transmission subsystem prompts the user to start the shower process, and upon confirmation, the faucet-pin lifts via a linear actuator, allowing water flow through the showerhead. If the user opts out, the faucet-pin subsystem remains inactive. When the user finishes, selecting "done" returns the shower knob subsystem to its resting position. The microcontroller signals the faucet-pin subsystem to close off water flow by lowering the faucet-pin.

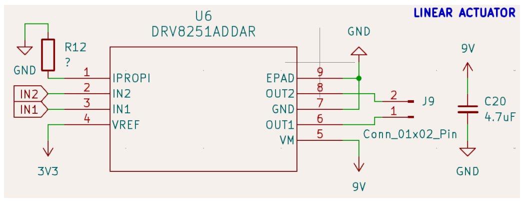

Figure 9: Schematic of Linear Actuator Driver

Requirements

The faucet-pin subsystem should be able to accurately react to inputs from the user specifying whether to start or stop the shower.

Design Choice

We decided to use a linear actuator because there was one degree of motion: up/down. Since there was no need for a rotating motor for this part, we decided to implement the faucet-pin subsystem with the linear actuator.

2.7.4 Transmission Subsystem

The Transmission Subsystem enables user interaction via a web server for temperature setting, shower initiation, and completion. It relays preset data to the shower knob subsystem, which adjusts temperature and alerts the microcontroller when the desired level is reached. The web app prompts shower start, activates the faucet-pin subsystem for water flow, and directs excess water to the water-saving subsystem. Upon completion, it receives a "done" signal, prompting the ESP32 to reset the shower knob and stop water flow through the faucet-pin subsystem.

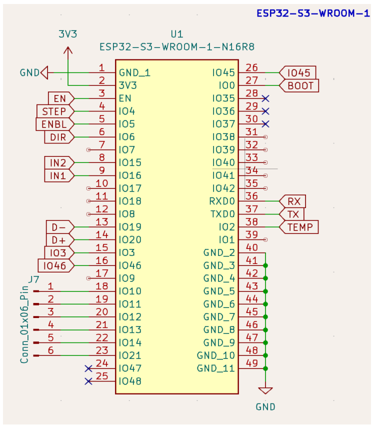

Figure 10: Schematic of ESP32 connections

Requirements

For the transmission subsystem, we needed to ensure that the web server from the ESP-32 access point could be connected from the user's device. Additionally, it must be able to receive inputs from the user and accurately relay them to the rest of the subsystems, namely the shower knob subsystem and the faucet-pin subsystem.

Design Choices

We decided initially to create a phone app and use the Bluetooth module on the ESP to connect the two components. However, during testing, we came across many connection issues that we needed to resolve. More details about this are mentioned in the Design Changes section.

Design Changes

When we initially used the Bluetooth module on the ESP, the connection was very spotty and not consistent. As such, we decided to switch to a web server that is connected to the WiFi module on the ESP-32. This would create an access point that any user device could connect to. This was much more consistent and worked well with our overall design.

2.7.5 Water-Saving Subsystem

The Water-Saving subsystem was designed as a prototype featuring an adjustable tubing system connected to a portable container positioned below the shower faucet. Excess water, outside the user's temperature request, is collected for reuse. However, implementation was not feasible due to the lack of a suitable faucet and knob system, otherwise testing would need to involve a real-life shower.

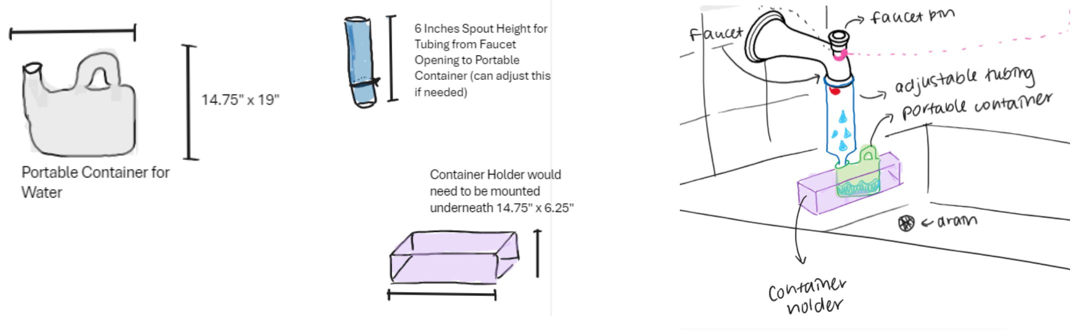
14.75" x 19"
Portable Container for Water
6 Inches Spout Height for Tubing from Faucet Opening to Portable Container (can adjust this if needed)
Container Holder would need to be mounted underneath 14.75" x 6.25"
Faucet
faucet pin
adjustable tubing
portable container
drain
container holder

Figure 11: Physical Design and Visual of Water-Saving subsystem components

Requirements

For the water-saving subsystem the important prototype requirements we had that we would implement into the future would be having a 6 inch adjustable tubing adhesive which can be found on Amazon, a portable container (14.75" x 19") which would be able to handle weight of how much water is caught and user-friendly since it can be held by the user. The container holder (14.75" x 6.25") would be mounted using waterproof command strips and be able to handle the weight and the size of the portable container.

Design Choices

We opted for cost-effective components available on Amazon and explored DIY construction to withstand water pressure and weight. As a non-electronic prototype, we focused on the microcontroller and power, omitting detailed functionality.

Design Changes

The design changes we would implement into the future would perhaps be not only physically implementing this design idea, but also having a waterproof sensor on the portable container to the limit of water it can hold, indicating to the user that the portable container has held a certain amount of water and can be ready to use. In terms of actual component design changes, we would have to physically implement and test our prototype to see what could be a potential issue and create a solution from it.

3. Design Verification

In this section we will cover our procedure for verifying the results of our design. In the Appendix we have included the entire Requirement and Verification table for our project. The results are described below for each requirement.

3.1 Shower Knob Subsystem

This subsystem controlled the rotation of the shower knob along with the readings from the temperature sensor. They then sent this to the microcontroller which would instruct it to turn based on the readings and analysis.

3.1.1 Requirements

1) For the demonstration we had the motor rotate the knob more quickly doing multiple steps at once to reach the upper limit quickly. Prior to that during testing we went all the way down to 1/32 steps and utilized this to get fine adjustments from the motor. For this testing we wrote a simple program utilizing the motor driver and the motor which had it step by 1/32 steps with delays in between so we could effectively measure.
2) For this it inherently worked when we tested the motor since this encompassed communication with the microcontroller. This was primarily a requirement to make sure that the motor would move or stop when we wanted it to based on the code we wrote to the microcontroller. Since the motor matched the behavior we wanted we were able to meet this requirement.
3) We initiated testing by programming the temperature sensor to transmit readings to the microcontroller and output them to the serial terminal with minimal delay. Under similar conditions as another thermometer, we compared the readings for any discrepancies. While occasional variations of a few degrees occurred, they never exceeded 4 degrees.

3.2 Transmission Subsystem

This subsystem was primarily the application, and encompasses all user interaction with the actual program. It handles all input and output. This utilized the onboard wifi capabilities of the ESP32 and played a large part in the interaction of different subsystems. This also encompassed all the interactions between components such as the site and the actuator, and the motor and the temperature sensor.

3.2.1 Requirements

1) We rigorously tested input and output functionality to ensure effective communication between the site and the microcontroller. Starting with basic LED control for instant data transfer, we progressed to string inputs, printing to the terminal. Continuous updates were added, comparing real-time information between the app, terminal, and site to verify accuracy and minimize latency.
2) We implemented string input for temperature with error handling, start/stop buttons, and real-time temperature display on the website. Rigorous testing and error handling were added for seamless interaction. We simplified by automating the process instead of using a timer, but provided user control options on the site.
3) Lastly, we tested the interactions between each component. We tested the shower knob by implementing a loop with a temperature sensor that reported readings to the terminal. Next, we

combined a motor driver with the sensor data using a complex algorithm that adjusted for water temperature and motor limits. The transmission subsystem was integrated by adding rotation controls to the website, with ongoing adjustments based on output data. The faucet-pin subsystem was not implemented, but a control button was planned for the website.

3.4 Power Subsystem

This subsystem was used to supply power to each individual component across each subsystem. It encompassed getting the voltage, regulating it down and providing consistent voltage for each item.

3.4.1 Requirements

1) We utilized a multimeter and had it read for extended amounts of time from a direct 12v source on our initial breadboard. This was used to make sure that each component could handle the voltages for extended periods of time.
2) While inspecting the PCB, we checked connections to and from the buck converter, ESP, temperature sensor, motor, and linear actuator. Issues arose with the expected 3.3V reading from the buck converter to the ESP, revealing why the PCB was not functioning correctly.

3.5 Faucet-Pin Subsystem

This subsystem handles the linear actuator and would in theory raise and lower the pin that switches the shower from a shower to the faucet underneath. This was to eliminate the need for the user to interact with the rest of the system.

3.5.1 Requirements

1) Testing the unimplemented subsystem was not possible due to our PCB not working, but we aimed for minimal leaks, high capacity, and portability. We planned visual inspections and component mobility checks, keeping electronics safely away.

3.6 Water-Saving Subsystem

This was a prototype subsystem which we were not planning to make, but was a concept we would utilize for further iterations or development of the project.

3.6.1 Requirements

1) Since we did not implement the subsystem, testing was not possible. However, we aimed for minimal leakages, high capacity, and portability. Testing methods included visual inspections and component mobility checks. Electronics were kept at a safe distance to prevent interactions.

4. Cost and Schedule

Below are the calculated costs of ShowerSync

4.1 Component Parts

Quantity:	Manufacturer:	Part #:	Description:	Cost:
1	Espressif	ESP-32 WROOM S3	Microcontroller	$10
1	Stepperonline	Nema-17 Stepper Motor	Stepper Motor	$13
1	Analog Devices	DS18B20	Temperature sensor	$8
1	Reliance	N/A	Portable Container	$20
1	EZ-Flo	N/A	Tubing	$4
1	USLICCX	N/A	Linear actuator	$30
TBD	Home Depot	N/A	Waterproof wiring	$15
1	Farmplast	N/A	Milk crate (waterproofing)	$10
N/A	Miscellaneous components (including drivers)	N/A	Resistor, capacitor, etc.	$5

4.2 Labor

For labor costs, we estimate around $50/hr for ECE graduates. We worked for around 12 hours per week for 9 weeks. Accounting for overhead costs, we can estimate a salary of around 50 x 9 x 12 x 2.5 which comes out to be $13,500. Since we have 3 members, this total would be $40,500. For the machine shop, we estimate it will take 20 hours to complete the mounting. At around $25/hr, this comes out to be $500. Summing up everything from the table and the labor costs, we get our total sum to be $41,115.

4.3 Schedule

Week:	Task:	Team Member:
Feb 19th - Feb 25th	Order Parts to test with	All
	Finish Proposal Rewrite, Finish Design Document, Meet to go over Design Document details and plan the design for first PCB order	All
	Test Temperature sensor	All
	Check the status of the remaining parts that are supposed to arrive 2/22, figure out what else we need to order	All
	Talk with Machine shop again	All
Feb 26th - March 3rd	Design Review 11:00am	All

	TA Meeting, PCB review	All
	Finalize the Design for first PCB order - split the PCB by systems	All - schematic designs split by each member
	Talk with Machine Shop: Shower Knob subsystem with claw stepper motor	All
March 4th - March 10th	Order the PCB, TA Meeting	All
	Teamwork Evaluation	Individual
	Order anything with feedback from Design Review, Talk with shop if needed	All - split into to components and parts for each person to order
	Start planning on web/mobile application	All - Keshav started implementation
March 11th - March 17th	Spring Break	
March 18th - March 24th	TA Meeting	All
	Solder the PCB and test the functionality, work on being able to see the issues within the PCB, redesign and reorder if needed	All/Each team member go in individually, if not available for a time block
	Work on web application	All/Each team member can discuss where they can contribute here
	See if the shower knob subsystem can be working (motor can move angles)	All
March 25th - March 31st	TA Meeting	All
	Solder the PCB, test functionality, work on any issues with PCB (Redesign/Reorder if needed)	All/Each team member go in individually, if not available for a time block
	Work on web application	All/Each team member
	See with the faucet-pin subsystem how it can be mounted/paired up	All
April 1st - April 7th	TA Meeting	All
	Solder the PCB and test the functionality	All/Each team member go in individually, if not available for a time block
	Work on any issues with PCB (Redesign/Reorder if needed)	All/Each team member go in individually, if not available for a time block
	Finalize web application, See if the web application can work with the the subsystems individually	All
April 8th - April 14th	TA Meeting	All

	Finalize any issues with web application and code	All
	Prepare for Demo	All
April 15th - April 21st	Mock Demo with TA, Practice Demo and Presentation w/o TA	All
	Team Contract Fulfillment, Start Final Paper Documentation	All/Individual
April 22nd - April 28th	Final Demo with Instructor and TAs, Mock Presentation with Comm and ECE TAs	All
	Work on Final Paper Documentation	All
April 29th - May 2nd	Final Presentation and Submit Final Report	All
	Lab Checkout and Turn in Lab Notebooks	All

5. Conclusion

The conclusion of our report focuses on the accomplishments, uncertainties, future work/alternatives, and finally ethical considerations.

5.1 Accomplishments

Our project successes focused on the aspect that our shower knob subsystem could move in rotation with 60-90 degrees angle and react correctly to the updated temperature information. Our transmission subsystem was able to transfer data and update information from the user through the web application to and from the microcontroller and shower knob subsystem. Additionally, our microcontroller was able to communicate throughout the subsystems and handle most of the logic/calculations. Finally the water temperature sensor was able to accurately read the temperature to 4 degrees. The final functionality included a working shower knob communicating with the user requests.

5.2 Uncertainties

We encountered significant issues with our buck converter and PCB design for power regulation, likely due to the component's incapacity to handle high voltages, leading to burnout. Further research or circuit redesign may be necessary to address this issue, possibly requiring additional pathways for debugging from the ESP microcontroller. Soldering or pre-existing damage could also be contributing factors, necessitating a thorough PCB analysis for future functionality. The uncertainty surrounds the implementation of the faucet-pin and water-saving subsystems. While the water-saving subsystem presents minimal challenges due to its lack of electrical components, integrating the faucet-pin subsystem entails substantial integration and programming work to synchronize with the project's overall functionality.

5.3 Ethical considerations

The primary IEEE ethical dilemma with ShowerSync revolves around ensuring safety in the presence of water. Our safety measures focus on preventing electrocution, especially since we utilize a waterproof temperature sensor directly interacting with water. It's essential that all components, especially motorized ones, are waterproof to mitigate risks effectively.

Regarding ShowerSync's power source, we are prioritizing safety by ensuring no batteries are near water to avoid risks. Components will be housed in waterproof encasings mounted above the shower head, minimizing water contact. Waterproof wiring will support motor-based systems and connections, ensuring system integrity despite water exposure and humidity.[3].

Our solution approach supports Section 1.1 of the IEEE Code of Ethics which states “to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment” [2]. Safety is crucial in ShowerSync to protect both users and components during water interaction. Emphasizing safety enables us to design the device with user protection in mind while achieving our goals effectively.

By encasing the battery, controller, and motor system in waterproof materials and utilizing waterproof wiring, we prevent water disruption. Placing the waterproof temperature sensor inside the faucet ensures water resistance. An impermeable encasing material safeguards against pressure and impact. Providing clear user instructions is crucial for ShowerSync to prevent short circuits and hazardous situations. To maintain battery health and prevent overcharging, we use a rechargeable 9V DC battery pack that can be removed for recharging.

We will create a safety manual for ShowerSync, covering operation, emergencies, and troubleshooting to ensure user and developer safety. This supports the ACM Code of Ethics Section 2.7 because we are "foster[ing] public awareness and understanding of computing, related technologies, and their consequences" [1]. With this, ShowerSync will highlight the importance of the manual emphasizing waterproofing and provide guidance for handling device issues.

Lastly, the IEEE's Code of Ethics and ACM's Code of Ethics both explain the duties that engineers must adhere to in their professional careers. Section 2.2 of IEEE states to "support colleagues and co-workers[,] to strive to ensure the code is upheld, and to not retaliate against individuals reporting a violation" [2]. In ShowerSync, prioritizing user safety and welfare is a necessity. With reference to ACM's Code of Ethics Section 3.7, "[we need to] recognize and take special care of systems that become integrated into the infrastructure of society" [1], implying the importance of engineers' honesty and ethical conduct, we uphold ShowerSync's standards and safety concerns.

5.4 Future work/Alternatives

In future iterations, our focus will be on three key areas: improving the PCB design and testing process, implementing initial versions of the faucet-pin and water-saving subsystems, and fine-tuning details like handle rotation and website functionality. Redesigning the power subsystem and adopting more effective soldering testing practices will be crucial for achieving a working PCB since earlier we faced issues with the buck converter due to limited time and insufficient components for fixes.

We aim to develop initial versions of the unimplemented subsystems. The water-saving subsystem, comprising tubing and a portable container, lacks electrical components and is relatively easy to construct and test. The faucet-pin subsystem, involving a few electrical components, was not tested due to PCB issues. With more time, we plan to test it using a simple version on a breadboard and integrate it with other subsystems, contributing to the project's completion.

Lastly, we plan to improve the precision and smoothness of the handle's turning, possibly by using a higher torque motor. Additionally, we intend to rework the website code to enable real-time updating of the displayed water temperature.

References

[1] “ACM Code of Ethics.” *Code of Ethics*, www.acm.org/code-of-ethics (*ACM Code of Ethics and Professional Conduct*, n.d.). Accessed 21 Feb. 2024.

[2] “IEEE Code of Ethics.” *IEEE*, www.ieee.org/about/corporate/governance/p7-8.html. Accessed 20 Feb. 2024.

[3] “Illinois Wiki.” Services, Engineering IT Shared. *ECE 445 - Senior Design Laboratory*, courses.engr.illinois.edu/ece445/wiki/#/. Accessed 21 Feb. 2024.

Appendix A Requirement and Verification Table

Subsystems:	Requirements:	Verifications:
Shower Knob Subsystem	1) The stepper motor must efficiently move the shower knob within the range of motion (about 60-90 degrees), as the knob moves it enables it to turn the water on and off in both directions.	We will code and test the motion of the motor setup to see if it can precisely rotate the shower handle, we will do this by using a simplified motor driver for motion observation and calibration.
	2) The motor should efficiently communicate with the microcontroller which relays the temperature information and reading. The microcontroller should then control the motor based on readings to create the desired motion effectively.	We will verify that the microcontroller is able to communicate with the motor by using a multimeter to take current readings as they are inputted into the pin connector. Then, we will use our algorithm to ensure that the motor's rotation angle aligns with how much the handle is intended to be rotated, measuring with a protractor or similar device.
	3) The entire subsystem should be able to give readings of the temperature within four degrees of the actual temperature.	To ensure precise motion within a four-degree temperature range, we will maintain constant water temperature and test various inputs within the four-degree range to ensure that the handle is moving accurately. Testing will involve multiple cups of water at different temperatures.
Subsystems:	**Requirements:**	**Verifications:**
Transmission Subsystem	1) It must accurately relay information from the user via an application to the microcontroller. It also must be able to see transparent data interaction between the microcontroller and application, allowing each to access information from the other.	We will initially set up a website connectivity and verify the output through a simple program. We will ensure that there is a clear input and output alignment, where we test processes such as temperature setting. Additionally, we will implement buttons for real-time results and see if it integrates better with polling or interrupt-based approaches.
	2) It must allow the user to control certain parameters such as temperature and notify the user when the water is ready temperature wise.	We will explore methods to send notifications/alerts to the website, either through sound or visual cues. Initially, we will test with a timer, then integrate it with the shower knob subsystem and temperature sensor to trigger the web notifications upon specific events.

Subsystems:	Requirements:	Verifications:
Microcontroller Subsystem	1) Efficient communication with subsystems is crucial, ensuring data exchange as intended. Specifically: i. Receive temperature sensor readings and control the shower knob motor. ii. Facilitate bidirectional communication with the application, ensuring accurate data display. iii. Instruct the faucet-pin motor based on inputs from the transmission subsystem.	We will measure this individually based on the subsystem. a. We will create a basic driver for the temperature sensor to output readings to either the terminal or the website. b. To test the motor, we'll connect a multimeter to its inputs to monitor the consistent voltage levels necessary. Then, we'll develop a simple motor driver to test rotation in both directions and verify voltage reading outputs. c. Similarly, for the linear actuator in the faucet-pin subsystem, we'll develop a simple driver. Using a multimeter, we'll monitor voltage at the actuator connection while implementing simple control methods like timed intervals (ex: 10 seconds on/off each) or button presses to observe voltage changes.
Power Subsystem	1) It must be able to consistently maintain and regulate voltage levels for each of the subsystems.	We will monitor the current and voltage by using a multimeter. We will use it on each individual subsystem and test it for an extended period of time to ensure there is no fluctuation.
	2) Supplies 3.3 V to temperature sensor, Transmission Subsystem, Microcontroller Subsystem, Faucet-Pin Subsystem, and supplies 9 V to Shower Knob Subsystem (specifically stepper motor).	We will use a multimeter to verify regulated voltage matches the intended level and reaches the intended devices. Testing will be conducted over extended durations.
Subsystems:	**Requirements:**	**Verifications:**
Faucet-Pin Subsystem	1) It must be able to efficiently switch between shower head and faucet based on microcontroller commands. It toggles to the shower head when in use and returns to the faucet afterward once the user completes showering process.	To verify system functionality, we will start by implementing a start button on the app to test actuator control of the faucet-pin. Then, we'll activate a one-minute timer to observe actuator behavior in raising and lowering the pin.
	2) The system should operate on for consecutive uses without manual intervention. After one shower, it should retain its setup and function without requiring adjustment for	We will assess the linear actuator's performance over instances (ex: 10 instances) to ensure consistent raising and lowering of the pin, preventing potential issues from the natural

	the next use.	falling mechanism of faucet-pins. Additionally, we will confirm the system operates without human intervention between showers.
Water-Saving Subsystem	1) The subsystem must efficiently hold excess water as the water is reaching the requested temperature and be able to store the excess water.	We will position the system under the faucet, wait for warm water, and visually confirm its capacity to contain the water that would otherwise be wasted. Additionally, we will assess its portability and ease of removal by physically extracting it with water still inside.
	2) The subsystem should not require any voltage and include preliminary precautions to avoid safety risks.	We will visually make sure that there are no electronic components that are surrounding the system.