

# **ShowerSync**

## **ECE 445 Design Document**

Team 21:

Reet Tiwary - rtiwary2

Edward Xiong - exiong2

Keshav Dandu - kdandu2

TA: Nikhil Arora

**March 20th, 2024**

## 1. Introduction

### ■ Problem:

Imagine running late for something. You are not only scrambling to finish last minute work, but you also still need to shower. Now you have to wait on your water getting just to the right temperature, but also waste a ton of water in the process of it all, altogether causing you to just show up late and increasing your water bill.

With this type of situation, there are already products in the market that can adjust temperatures beforehand and have it as a preset. For example, smart showers are nice in that way because users can have a chance to control their desired water temperature, allowing time for the water to reach the desired preference all through a mobile application with Bluetooth included. But not everyone can afford a smart shower as it is expensive and requires a whole shower/bath renovation process. Additionally, how can we come up with the solution of not wasting water and being able to reuse it for alternative purposes while also being cost-effective?

### ■ Solution:

A high-level idea of our solution is to create a convenient and efficient self-adjusting shower knob/pin system with a compartment to store and reuse any water outside of the desired temperature. This would aim towards cost-effectiveness, a user-friendly interaction, a simple installation, and an environmental-friendly approach to a convenient shower process. How we would implement this solution would be broken into system compartments/parts.

The system comprises a temperature sensor, a motorized knob mechanism that can be

easily attached to various shower knobs and faucet pins using a two-piece design, and finally a removable container located near the faucet which stores any water outside of the desired temperature through a tubing system into the container to be reused. When the sensor is able to detect the correct temperature through the faucet, the faucet pin is pulled through the linear actuator system, allowing the accurate water temperature to be dispersed through the shower nozzle and saving the user's time as well as their showering process.

The temperature sensor accurately gauges the water temperature passing through the faucet system, while the motorized knob system ensures that the shower knob is adjusted to match a predefined temperature setting and that the faucet pin is pulled when the desired temperature is reached, indicating to the user that the shower is ready. When the desired temperature has not been reached, the faucet has a tubing system that stores any water outside of the desired temperature into a container which can be removed and reused for other household purposes.

The entire system is controlled by a microcontroller, and to enhance user experience, we plan to incorporate wireless communication using a Bluetooth module. This wireless capability allows users to remotely adjust the shower temperature without the need to physically interact with the knob and pin, so the shower is ready to go.

The self-adjusting shower knob/pin system and the faucet tubing system provide comfort and convenience but also minimizes water wastage by precisely regulating the desired water temperature and saves any water outside of temperature regulation into a usable container for

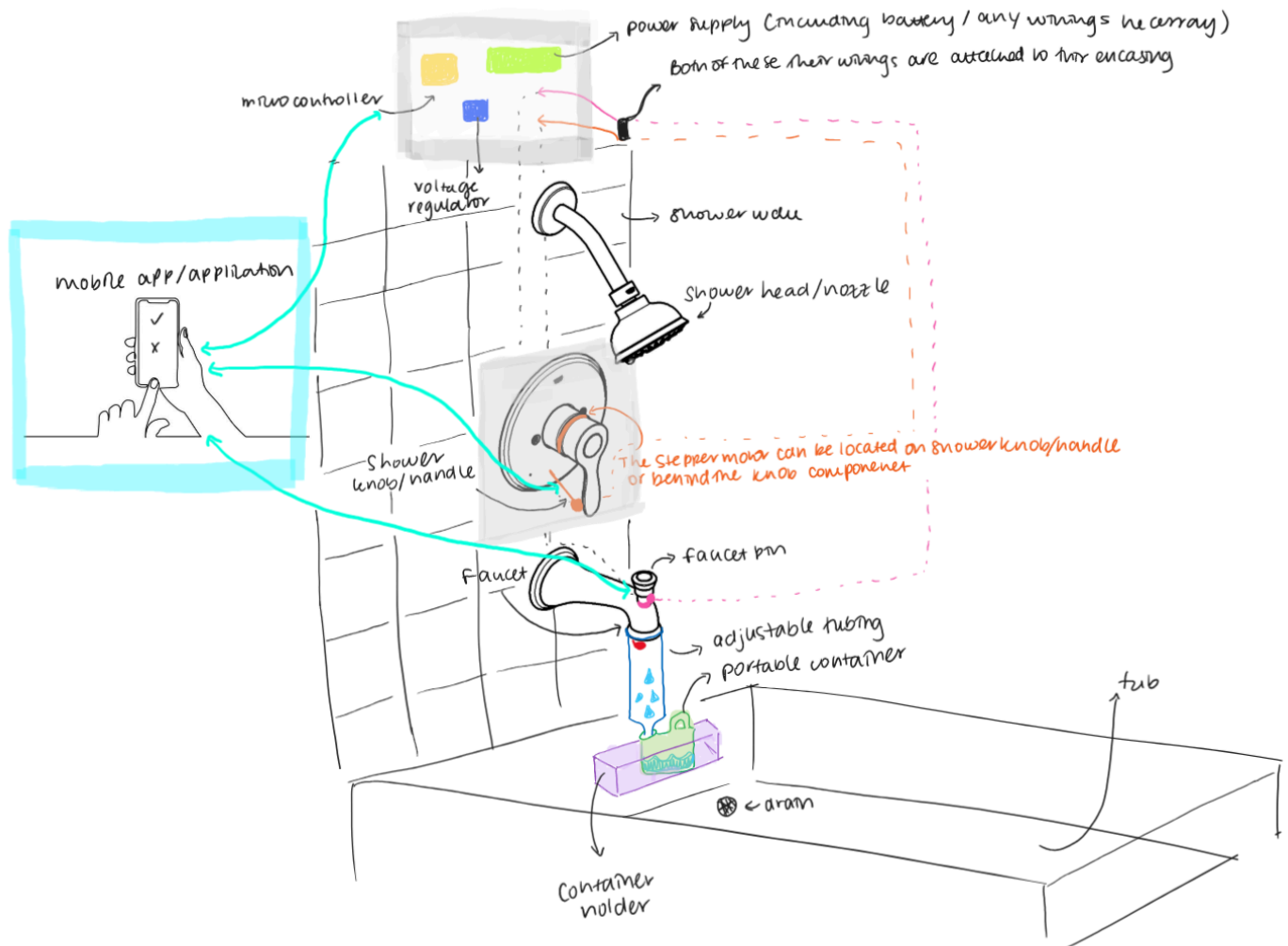
alternative purposes.

Finally additional features, such as a user-friendly mobile app interface and real-time temperature feedback, can be integrated for an enhanced and customizable showering experience while also being cost-effective.

## ■ Visual Aid:

Rough Sketch of the visual aid for ShowerSync which uses a temperature sensor located underneath the faucet to preset the user's desired water temperature signaling the shower knob to find the right temperature and finally signaling the faucet pin to allow water through the nozzle. Any water outside of the desired temperature will be kept in a portable container.

Detailed Description Provided Below:



- 1) The faucet as shown in the visual aid shows how the water temperature sensor will be located underneath the faucet opening as it needs to detect the temperature of the water for verification that the desired temperature has been acquired.
- 2) The faucet opening also has a tubing system set up to initiate the water-saving aspect for this project to prevent any loss of water. Beneath the faucet we will have a tubing system that will not expose the opening of the faucet to the rest of the tub. The tube is adjustable and it will allow water to flow into the portable container (color coded - green) which is beneath the faucet held by a container holder (color coded - purple) which can be easily mounted.
- 3) The faucet-pin will use a linear actuator which will be pulled up once the user confirms that they are ready to begin the shower process. Otherwise, we will need to determine how long the linear actuator can wait to lift up. When the user has completed their showering process, the linear actuator will allow the faucet-pin to drop down, once the knob/handle is back to rest.
- 4) The shower knob/handle communicates with the temperature sensor in that it should be able to find the temperature desired. We will utilize a stepper motor that moves the knob/handle where the desired temperature may be. It may be located behind the knob/handle for better installation. To waterproof the shower knob/handle, we will encase it since the user does not have to manually touch the knob/handle at all. To encase it we will either use a waterproof clear encasing or whichever may be suitable.
- 5) The power supply, voltage regulator, microcontroller and any of the wirings for the stepper motor as well as the linear actuator will use waterproof wiring to

prevent any water damage, all of these major components will be mounted on a bathroom wall to make sure that there is no water reaching it and the encasing will help prevent any humidity exposure and more.

- 6) The mobile application communicates with the faucet to set and find the user's desired temperature, then it will communicate again with the entire shower system when the knob/handle locates the temperature (angle wise), notifying to the user that desired temperature has been reached and whether they want to proceed with showering or not. This will communicate with the faucet-pin to either be lifted to allow the water to pass through the shower head and the showering process begins or to wait until the user is ready. Finally, the mobile application will send a signal to the shower knob/handle to return to rest and return the faucet pin back to rest as well.

#### ■ High-Level Requirements List:

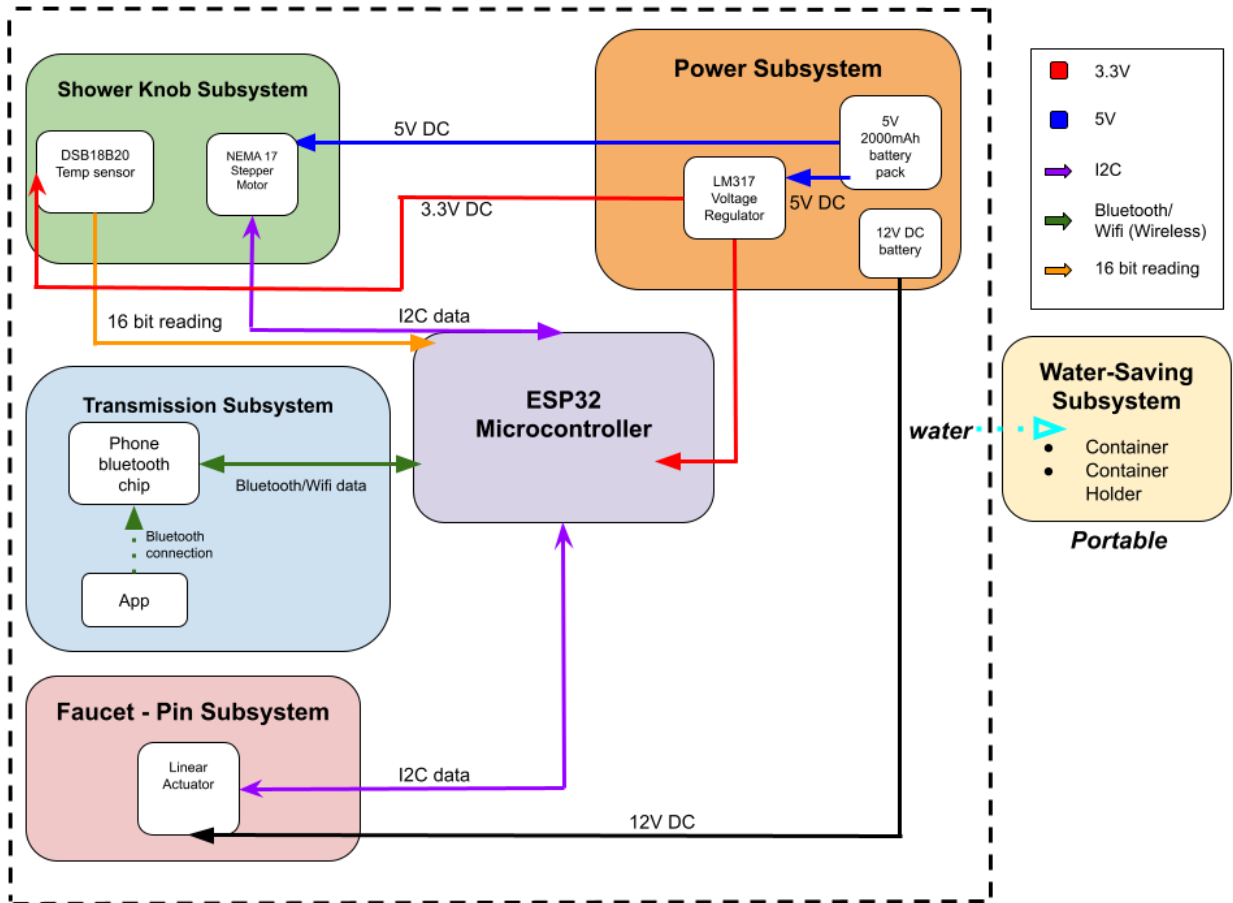
1. When activated by the user, the ShowerSync system should automatically turn on the faucet and redirect the water to a separate storage until the matching user temperature is reached (i.e. 80 degrees). The temperature should be **accurate to an error of 4 degrees\***  
\*We aim to decrease this error number but as of right now we need to test the temperature sensor to be able to make a better judgment of it.
2. Once the shower knob/handle has acquired the desired temperature, the shower system should be able to notify the user to begin the shower process by initiating the faucet-pin subsystem. The time between when the desired temperature is reached and when the pin

is pulled should be **no more than 25 seconds**.

3. The device should be able to react to remotely-issued commands. The entire device should be able to respond and **begin the entire process within 10 seconds** of the remotely-issued command.

## 2. Design

### ■ Block Diagram:



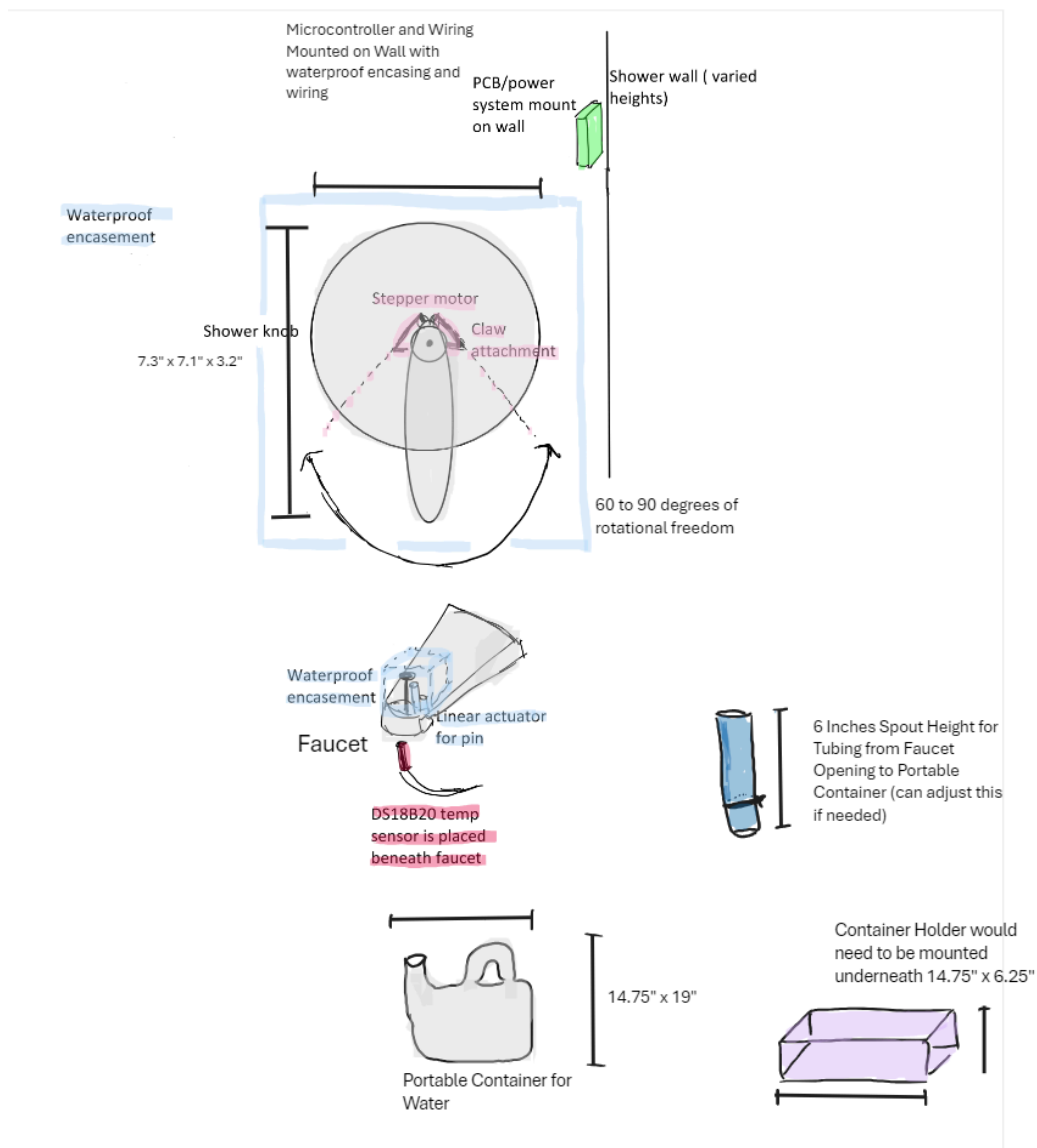


The power subsystem will provide power to the entire showering mechanisms we plan to integrate within. This will supply the Microcontroller, Shower Knob Subsystem, Transmission Subsystem, and Faucet - Pin Subsystem. The Transmission Subsystem contains the mobile application with the phone bluetooth chip and takes in the user's desired temperature request. This will communicate with the Microcontroller leading to communication with the Shower Knob Subsystem and the Faucet - Pin Subsystem. The Shower Knob Subsystem includes the temperature sensor which will sense the temperature of the water and use the stepper motor to angle it to the desired temperature. If the desired temperature has not been reached, any water passing through the faucet will be processed through the Water-Saving Subsystem which contains external water in a portable container held by a mounted container holder. Once the desired temperature has been reached, this will notify the Microcontroller which will notify the Transmission Subsystem that the shower temperature has been acquired and whether the user wants to begin the showering process which will notify the Microcontroller to communicate with the Faucet - Pin Subsystem. The Faucet - Pin Subsystem will use the linear actuator to start the entire showering process, allowing water to flow through the nozzle.

## ■ Physical Design:



For demo purposes, the temperature sensor will be placed in cups of water



Each Subsystem is broken down component wise with the sensors affiliated with it and measurements for components we assume would work. This would allow us to test and demonstrate the functionality of each subsystem component individually. For example, we could use different cups of water at different temperatures and show how when the sensor sees different temperatures the motor for the handle reacts accordingly. We could also use this setup at the desired temperature from the app to show how the user is notified and how the faucet pin subsystem works.

## ■ **Subsystem Overview:**

### **1) Shower Knob Subsystem:**

The Shower Knob Subsystem contains the temperature sensor and a stepper motor for the shower knob/handle. How it connects to other subsystems is that the ESP32 microcontroller will receive preset information from the user and send it toward the shower knob subsystem. The shower knob subsystem will take in the data and set the temperature accordingly with the sensor and use the motorized mechanism to find the correct temperature. Once the desired temperature has been achieved, the microcontroller communicates the fact that the desired temperature has been received and the mobile application notifies the user about this. Once the user is prompted to begin their shower, this will initiate the faucet-pin subsystem which will allow the shower process through the nozzle/shower head. Any water that is outside of the desired temperature will automatically go into the water-saving subsystem which just stores water outside of user preference for external use.

With the components, the DS18B20 temperature sensor (5V) is a sensor that is specifically designed to be used for reading water temperature. We chose this sensor due to it being designed specifically with similar applications in mind and since it is cost-effective. The N17 stepper motor (6V) is the motor we plan to use to rotate the shower knob. It would be configured to do a pushing or pulling motion based on the temperature reading. Due to our system being in a shower and likely having water get all over it, we have begun looking into ways to waterproof the system beginning with the electrical components. We found waterproof wiring that we would likely use between the sensor to the PCB and the PCB to the motor. We also have to consider water getting onto pieces such as the motor so we are also looking into the possibility of a 3D printing material that would be waterproof so that we could create perfect structures to encase the motor and other delicate components in. If this is not feasible the next option we would consider is waterproof encasing that are premade and are made of a material similar to silicon.

## **2) Faucet-Pin Subsystem:**

The Faucet-Pin Subsystem connects to other subsystems in that once the the ESP32 microcontroller receives the sensing from the temperature sensor that the user's desired temperature has been reached, this will send the data to the transmission subsystem which will prompt the user if they are ready to start their shower process. User will either select "ready" which will send the data to the microcontroller which will send the data to the faucet-pin subsystem where the faucet pin will utilize a linear actuator mechanism to lift the pin up, allowing the water to go through the shower head/nozzle. If

the user selects “no” then the faucet-pin subsystem will not be initiated. When the user is finished with their shower, they can select “done” which will allow the microcontroller to send the data to the shower knob subsystem to go back to resting position and once the shower knob subsystem goes back to rest, then the microcontroller will signal the faucet-pin subsystem that the shower-knob has been returned to its original state and allow the linear actuator to drop the faucet pin down to rest as well, switching off/ closing off any water flow.

The faucet-pin subsystem would use a servo-based setup similar to the shower knob subsystem which would pull the pin that would switch the water from coming out of the faucet to coming out of the shower head. We would likely use the N17 stepper motor for simplicity instead of trying to figure out how to use a second type of motor. We will also use the waterproofing solutions from the shower knob system to find ways to protect the electronics in this subsystem.

### **3) Water-Saving Subsystem:**

The Water-Saving Subsystem contains a water tubing configuration that will be attached to the faucet and have the other end connected to the portable container which stores any water outside of the desired temperature. The container is held by a container holder which will be mounted underneath the faucet on the side of the tub to allow the portable container to be held. The portable container will be removable by the user to use any water stored in it for external use.

How the water-saving subsystem works with the other subsystems is that while it does not need to use any electronic device in its configuration, once the shower knob

subsystem receives the desired temperature the user has requested (from transmission subsystem to microcontroller to the shower knob subsystem), then while the shower knob is adjusting itself to reach the user's desired temperature, the faucet is basically opening allowing any water that is not reaching the desired temperature to be wasted. Rather than being wasted with the tubing system in place, any water outside of the desired preferences will automatically store into the portable container and altogether save any water to be wasted externally into the tub drain. The portable container will be big enough to store enough water outside of desired temperatures and the container holder should be able to hold it up based on how much water is being carried over. This subsystem interacts with all of the subsystems altogether since it is storing the water and at the end the user can use the portable container for other external uses.

With the components, the faucet tube is adaptable and can be adjusted from our side to keep the water flowing into the water container/storage which can be held on the bathroom shower wall beneath the faucet in a straight manner allowing the flow of the water to enter into the container. The mounting strips mount the container holder where we can place the water container in and users can move the container in and out of the holder easily and use it for other household purposes.

#### **4) Transmission Subsystem:**

The Transmission Subsystem contains a mobile application where the user can set a temperature request, start the shower process and end the shower process with ease.

How it connects to other subsystems is that it will signal to the ESP32 microcontroller to deliver the preset information. ESP32 will receive preset information

from the user and send it toward the shower knob subsystem. The shower knob subsystem will take in the data and set the temperature accordingly with the sensor and use the motorized mechanism to find the correct temperature. Once the desired temperature has been achieved, the microcontroller communicates the fact that the desired temperature has been received and the mobile application notifies the user about this. Once the user is prompted to begin their shower, this will initiate the faucet-pin subsystem which will allow the shower process through the nozzle/shower head. Any water that is outside of the desired temperature will automatically go into the water-saving subsystem which just stores water outside of user preference for external use. After the shower process is complete, the transmission subsystem will take in the user's "done" indicating that he/she is finished with the shower which will allow the microcontroller to indicate to the shower-knob system to return the knob/handle back to rest, which will allow the microcontroller to indicate to the faucet-pin subsystem to drop the faucet pin down to rest, switching/closing off all water and completing the shower as a whole.

With the components, the HC-05 transceiver is a very cost-effective receiver that would be compatible with our board. We currently are planning on using Bluetooth connectivity in our app to connect with our system or doing it over a WIFI network. In either case, the transceiver would be able to do the job. The ST7735R is a component in the case that we decide to pivot from the original idea of using a phone application and instead opt for a system where we would implement buttons or connect some kind of remote to control the temperature. It is unlikely that we choose to go in that direction but we have gone ahead and found this component as a starting point just in case.

## **5) Microcontroller Subsystem:**

The Microcontroller Subsystem is perhaps the second most crucial part of the project aside from the power subsystem. How the ESP32 microcontroller connects to other subsystems is that it will receive preset information from the user which is from the transmission subsystem and send the data toward the shower knob subsystem. The shower knob subsystem will take in the data and set the temperature accordingly with the sensor and use the motorized mechanism to find the correct temperature. Once the desired temperature has been achieved, the microcontroller communicates the fact that the desired temperature has been received and the mobile application notifies the user about this. Once the user is prompted to begin their shower, this will initiate the faucet-pin subsystem which will allow the shower process through the nozzle/shower head. Any water that is outside of the desired temperature will automatically go into the water-saving subsystem which just stores water outside of user preference for external use. After the shower process is complete, the transmission subsystem will take in the user's "done" indicating that he/she is finished with the shower which will allow the microcontroller subsystem to indicate to the shower-knob system to return the knob/handle back to rest, which will allow the microcontroller subsystem to indicate to the faucet-pin subsystem to drop the faucet pin down to rest, switching/closing off all water and completing the shower as a whole.

Component-wise, the ESP32 Devkit V1 board is very powerful with a dual-core processor and Bluetooth and WIFI connectivity which is specifically useful to make our transmission subsystem. It is also compatible with Arduino DOIT which could simplify work down the line. It is also very compact and has a very low power consumption while



having a wide range of pins which would be useful in our case since we would want to make it as small as possible since it would make waterproofing or making a waterproof encasing easier.

## **6) Power Subsystem:**

The power subsystem is an important part of this entire project as it supplies everything to work the way it may need to (other subsystems will not be able to communicate with each other without this). The power subsystem will contain a LM317 Voltage Regulator and a 5V (2500 mAh) battery pack. We do know that a battery can run out fast, however with a plug-in there would be more bigger concerns. We will encase the power supply to be safe from water, details regarding this are further discussed in this design document.

There are two main reasons why we decided to use a battery pack for our power subsystem. Using a battery pack instead of a wall outlet for powering the self-adjusting shower knob system enhances both safety and convenience. Directly connecting electrical components to a wall outlet in a wet bathroom environment poses risks of electric shock. Battery power eliminates this hazard by removing the need for exposed wiring near water sources. We can choose the placement of the battery pack and easily extend waterproofing wires to areas that require power near a water source. Additionally, battery operation offers flexibility in system placement without the constraints of traditional power sources, simplifying installation and enhancing user experience. By prioritizing safety and convenience, battery power ensures a reliable and user-friendly solution for the self-adjusting shower knob system.

■ **Subsystem Requirements:**

REQUIREMENTS:	VERIFICATION:
<p><b>Shower Knob Subsystem:</b></p> <ul style="list-style-type: none"> <li>● Must be able to move the shower knob as necessary efficiently within the range of motion (about 60-90 degrees). The motor should be able to turn the knob in both directions to turn the water on and off.</li> <li>● Should be able to communicate efficiently with the microcontroller relaying the temperature. Microcontrollers should be able to efficiently control the motor based on readings and create the intended motion.</li> <li>● Should be able to give readings of the temperature within three degrees of the actual temperature.</li> </ul>	<ul style="list-style-type: none"> <li>● We will code and test the motion of the motor in its setup to see if it can rotate the handle of the shower. We will do this by using a simplified motor driver to see motion and find the amount to turn it precisely.</li> <li>● We will first make sure that the microcontroller is able to communicate with the motor by putting a multimeter and taking current readings as they are inputted into the pin connector to make sure that it is communicating as intended to the motor. From there, we will use our algorithm for turning the motor and measure that the amount the handle is rotated matches precisely with our intended amount using a protractor or other device to measure the angle.</li> <li>● To ensure certain motion within three degrees of the temperature we will get water at a set temperature that is kept as constant as we can. Then we will try different inputs ranging from inside and outside this three-degree range and ensure that the handle is moving exactly in the way we want it to. We will test this with multiple cups of water at different temperatures.</li> </ul>

<p><b>Faucet-Pin Subsystem:</b></p> <ul style="list-style-type: none"> <li>• Must be able to efficiently switch between the shower head and the faucet at the bottom based on the microcontroller. When the user is ready to shower it goes to the shower head and when the user is finished with their shower it goes back to the faucet.</li> <li>• Must be able to work on multiple instances back-to-back without manual intervention being required. Upon use for one shower, it should not mess with the setup or require a person to go and adjust the motor to function a second time.</li> </ul>	<ul style="list-style-type: none"> <li>• To ensure that the system works as intended we will first set up a start button on the app and when clicked we will see if the actuator can raise and lower the faucet pin. We will then set up a timer and have it go for about a minute and just watch whether or not the pin is raised or lowered by the actuator.</li> <li>• Since pins naturally fall allowing it to fall could cause issues with the actuator coming down and messing it up over time. So we will check for multiple instances such as ten times, that the linear actuator is mainly raising and lowering the pin, not allowing it to just fall. We will also ensure it does not require human intervention between uses of the showersync.</li> </ul>
<p><b>Water Saving Subsystem:</b></p> <ul style="list-style-type: none"> <li>• Must efficiently hold excess water as the water is heating up and be able to store all the water.</li> <li>• Should not require any voltage and include preliminary precautions to avoid safety risks.</li> </ul>	<ul style="list-style-type: none"> <li>• We will place the system underneath the faucet, wait till the water is warm and check visually to see if the system can hold all of the water that would originally be wasted. We will then also make sure that it is feasible to move and remove from the shower by physically taking out the system with the water still in it.</li> <li>• We will visually make sure that there are no electronic components that are within a foot of the system.</li> </ul>

**Transmission Subsystem:**

- Must accurately receive information from the User via an application and send it to the microcontroller. Must be able to see clear interaction of data between microcontroller and application where they can see information from each other.
  - Must allow the user to control certain parameters such as temperature and notify the user when the water is ready.
- We will first set up the website and ensure that there are proper connections through a simple program to ensure that it outputs to the website. We will then set up a simple way to ensure that there is clear input and output by entering values in the way that we intend for processes such as setting the temperature. We will then also set up buttons and ensure that there are real-time results and we will see if it integrates better with polling or interrupt-based approaches.
  - We will find out how to send notifications or a make-shift alert to the website that will either make a noise or visually alert the user. We will first test this by being able to do it with a set timer for a time such as one minute, then once we have the shower knob subsystem working we will use its interactions with the temperature sensor as an interrupt which causes the website to notify the user.

<p><b>Power Subsystem:</b></p> <ul style="list-style-type: none"> <li>• Must be able to consistently maintain and regulate voltage levels for each of the subsystems.</li> <li>• Supplies 3.3 V to Shower Knob Subsystem, Transmission Subsystem, Micro-Controller Subsystem and Faucet-Pin Subsystem.</li> <li>• Supplies 6 V to Shower Knob Subsystem.</li> </ul>	<ul style="list-style-type: none"> <li>• 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.</li> <li>• We will use a multimeter to ensure that when the voltage is regulated down it is at the voltage we want it to be at the measure that it is going to the intended devices. We will also run this for extended periods of time.</li> </ul>
<p><b>Microcontroller Subsystem:</b></p> <ul style="list-style-type: none"> <li>• Must be able to efficiently communicate with the other subsystems. Relaying data to and from every subsystem as intended.</li> <li>• Specifically, it should be able to do the following: <ul style="list-style-type: none"> <li>○ 1. Receive readings from the temperature sensor and instruct the shower knob subsystem's motor.</li> <li>○ 2. Be able to send information to and receive information from the application and be able to see them correctly on both sides.</li> <li>○ 3. Be able to instruct the motor in the faucet-pin subsystem based on inputs from the transmission subsystem.</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• We will measure this individually based on the subsystem. <ul style="list-style-type: none"> <li>a. For the temperature sensor, we will create a simple driver that is able to output to either the terminal or the site the temperature sensor's reading.</li> <li>b. For the motor, we will connect a multimeter to the motor's inputs since certain voltages create the rotation of the motor. We will then write a simple motor driver that allows us to test both directions and then watch the multimeter readings to ensure that it is sending the expected voltage.</li> <li>c. For the linear actuator in the faucet pin subsystem, we will use a procedure similar to testing with the motor where we</li> </ul> </li> </ul>

	<p>make a simple driver. In this case, it will be us connecting a multimeter to where the actuator connects and using a simple system such as 10 seconds on 10 seconds off or a button we will see whether there is a high or low voltage.</p>
--	--

## **Tolerance Analysis:**

### **1) Power Analysis**

A factor that we have to consider is how much power our system needs to be able to operate. If we do not choose a correctly-rated battery/power system, then we are unable to power all the parts successfully. Initially, we were thinking of using a typical six-cylinder 5V battery that would provide us around 550 mAh. Additionally, we would regulate the voltage to match the necessary voltages we would need for our parts. Unfortunately, many of our parts did not provide a clear measurement of the power consumption, so we had to create an educated guess on what the power consumption would be based on similar parts.

The N17 stepper motor we plan on using did provide a power consumption of 120mA at a nominal voltage of 12V. Since we are not using a 12V power source, we estimate that the motor would take about 150mA. The ESP32 microcontroller we plan on using does not provide a power rating but it is mentioned in the description that it is an ultra-low power consumption board. As such, we give a generous estimate of around 200 mA for the microcontroller. Since we plan on using an app to connect with our Bluetooth/wifi transceiver, we will not need to power a remote for our project. The only remaining part that we have to power is the DS18B20 temperature sensor, but sensors like these consume little power, so we can assume it takes 1mA.

With all the parts accounted for, we can calculate the necessary power supply we need. First, let's assume we use a 550mAh battery. 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:

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

To calculate how many hours of use we get, we divide the two numbers:

$$385mAh/251mA = 1.53 h$$

This value of 1.53 hours is fairly low considering that we want the battery to last for numerous showers before having to replace it. As such, we decided to look into batteries with a higher power rating. We eventually found a 5V battery that could provide 2500mAh to our system. By doing the same calculations again, we can find the number of hours our device lasts again.

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

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

This estimate of 6.97 hours is much higher than our initial 1.53 hour estimate. Assuming the average user takes 10 minute showers, we reach a usage of around 41 showers. Although this number is not as ideal as we hoped for, it is still a reasonable amount of showers before the battery needs to be recharged.

## 2) Voltage Regulation

To regulate the different voltages we need for our project, we plan on using a LM317 voltage regulator. As such, we need to analyze the temperature it operates on to ensure that we don't have any issues with the regulator. To start we know that the max temperature the LM317 can operate regularly at is 150 C. To calculate the max temperature we may encounter, we use the following equation.

$$T = i_{out} * (V_{in} - V_{out}) * (\theta_{ja}) + T_a$$

Next, we need to find all the unknowns in this equation. We know our  $V_{in}$  is 5V and the lowest  $V_{out}$  we have is 3.3V, so we can substitute those numbers immediately. From the datasheet, we



also know that  $\theta_{ja}$  is 100C/W and  $T_a$  is 40 C. To calculate  $i_{out}$ , we can assume that  $i_{in} = i_{out}$  and calculate the current within our system. From the ESP-32 WROOM1 datasheet, we see that there is a current draw of 165 mA on average and 211 mA in the worst case. For our purpose, we assume the worst case. Our DSB18B20 temperature sensor only has a current draw of 1mA. Finally, our motor has a current draw of 120 mA. Adding these up, we get a current of around 332 mA. Now, we can plug into the equation.

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

As we can see, this value is much lower than the 150 C threshold provided by the datasheet, so we know it is safe for us to apply the voltage regulator to our design.

### 3. Cost and Schedule

#### ■ Cost Analysis:

Quantity	Manufacturer	Part #	Description	Cost
1	Espressif	ESP-32 WROOM	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	USLI CCX	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	N/A	Resistor, capacitor, etc.	\$5

#### Labor Costs

For labor costs, we estimate around \$50/hr for ECE graduates. We plan on working for around 12 hours per week for 9 weeks. Accounting for overhead costs, we can estimate a salary of around  $50 \times 9 \times 12 \times 2.5$  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.

■ **Schedule:**

<b>Week:</b>	<b>Task and Team Member:</b>	
Feb 19th - Feb 25th	Order Parts to test with	All, Edward Ordered
	Finish Proposal Rewrite	All
	Finish Design Document	All
	Test the Temperature Sensor	All
	Check the measurements of remaining parts that are supposed to arrive 2/22	Eddie/All
	Meet to go over Design Document Details and planning the design for first PCB order	All
	Talk with Machine Shop again once all major components have arrived	All
Feb 26th - March 3rd	Design Review 11:00AM	All
	TA Meeting	All
	Finalize the Design for the first PCB Order	All
	PCB Review	All
	Talk with Machine Shop: Shower Knob Subsystem with Claw stepper motor	All

March 4th - March 10th		
	Order the PCB	All
	TA Meeting	All
	Teamwork Evaluation	All/Individual
	Order anything with feedback from Design Review aka for mounting	All Decide/Eddie
	Talk with Machine Shop if needed	All
	Start planning on mobile application	All
March 11th - March 17th	Spring Break	
March 18th - March 24th		
	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 Being able to see the issues within the PCB	All/Each Team Member go in individually, if not available for a time block
	Redesign and Reorder PCB if needed	All/Each Team Member go in individually, if not available for a time block
	Work on Mobile 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 and test the functionality	All/Each Team Member go in individually, if not available for a time block
	Work on any issues with PCB	All/Each Team Member go in individually, if not available for a time block
	Redesign and Reorder PCB if needed	All/Each Team Member go in individually, if not available for a time block
	Work on Mobile 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	All/Each Team Member go in individually, if not available for a time block
	Redesign and Reorder PCB if needed	All/Each Team Member go in individually, if not available for a time block
	Finalize Mobile Application	All/ Each Team Member
	See if Mobile Application can work with the subsystems individually	All

April 8th - April 14th	TA Meeting	All
	Finalize any issues with mobile application and code	All
	Finish any last touches to mobile application such as commenting	All
	Prepare for Demo	All
April 15th - April 21st	Mock Demo with TA	All
	Practice Demo and Potential Presentation w/o TA	All
	Team Contract Fulfillment	All
	Begin Final Paper Documentation	All
April 22nd - April 28th	Final Demo with Instructor and TAs	All
	Mock Presentation with Comm and ECE TAs	All
	Work on Final Paper Documentation	All
April 29th - May 2nd	Final Presentation	All
	Submit Final Paper	All
	Turn in anything that was checked out	All
	Turn in Lab Notebooks	All

#### **4. Discussion of Ethics and Safety**

The main IEEE ethical dilemma we are facing with ShowerSync is the safety aspect of water with all of the components and materials we will be utilizing. A lot of our safety features regarding ShowerSync will have the involvement of water and how to prevent the aspect of electrocution. As we are using a waterproof-based temperature sensor, which will interact directly with the water, we also need to make sure that anything that we are using to mount any of the motorized-based components with the various subsystems is also waterproof.

Additionally, we need to take into consideration the power source of ShowerSync and how that can be affected by any interaction of the water if it so happens. We will not have a battery near any water because that will be dangerous and will pose a risk factor. However, we plan on keeping any of the components out of reach from the water by mounting them in an encasing near the wall above the shower head tub configuration and allow water proof based wirings to help serve as the motor based systems and anything we may need to connect our divide too. By encasing the power source, using waterproof-based encasing and waterproof wirings whilst also being mounted above the shower head where it would be attached to the wall of the building, there can be a guarantee that water will not disrupt the system and neither will 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]. It is important to have safety considered in ShowerSync because we need to protect not only the user, but also the components and materials we will be utilizing to create the interaction with water as safely as

possible. This will also allow us to focus on the device's design from the perspective of how we can achieve our goals whilst also remaining safe and use materials that will allow the user to be safe while delivering its purpose.

By mounting the waterproof encasing/box which will contain the batter and controller and use waterproof wiring and waterproof wiring for the motor system this will allow any prevention of water disrupting the device. The waterproof temperature sensor will be placed inside of the faucet, since it is waterproof and sensing just the water temperature, we believe that it should be fine if the enclosing of it is also waterproof. We would need an encasing or box based material which will be impermeable to water and can withstand any pressure or impact that it may receive if it does.

Moreover, it will be critical for ShowerSync to provide instructions to users on how to use the device properly so that there are no issues with any short circuiting, and to prevent bigger issues like hazardous fires starting. With our battery usage since we are using any 5V DC based battery we want to make sure that we can find a way to maintain the health of the battery so it does not overcharge, our battery pack would be rechargeable where it can be taken out to recharge it.

While we still have much to research on what exactly we would want to make sure we are also being safe with water in the picture, we would include a safety manual on how to operate the system in this case, what to do in cases of emergencies and finally, what to do in terms of malfunctioning. This will not only help the users who are utilizing this product, but also the developers working on this product. 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 keeping the features waterproof and dry so materials are not malfunctioning either. If there is damage to the wirings, there will be information on how to handle the device and who to contact if there are any 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]. With ShowerSync being a project where a device is being created to help user's with an end goal, it is important that the users' safety and welfare come first. 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 to us that it is also important that we as engineers are truthful in our job and do not engage in any areas of unethical based actions as well as conflicts, by doing this we will uphold the value and standards of engineers and ShowerSync's standards as well as safety concerns.

## 5. Citations:

[1] “ACM Code of Ethics.” *Code of Ethics*, [www.acm.org/code-of-ethics](http://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](http://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/#/](http://courses.engr.illinois.edu/ece445/wiki/#/). Accessed 21 Feb. 2024.