SMART SHOWER

A Prototype for Samsung's Smart Product Line

SAMSUNG ENGINEERING

Team 3: Samsung

Christian Hockenbury
Victor Hui
Joshua Weadick
Joshua Williams

05.05.2022

USC Viterbi EE 459Lx

BS CECS Senior Design Project

Table of Contents

1. Introduction

The Smart Shower is a multidisciplinary engineering project created for EE459Lx, USC's Capstone design course for seniors majoring in Computer Engineering and Computer Science (CECS) Bachelor of Science. For our senior design project, we were tasked with creating and prototyping an innovative real-time "smart" consumer device that provides environmental benefit. Additionally, the product development must follow applicable engineering standards, ethics, and documentation. This report is a detailed view of our final works-like prototype.

1.1 Cross Team Collaboration

EE459Lx seeks to expose its students to a product design experience similar to what an engineer at a company would face. For our course section, we partnered with USC Marshall School of Business marketing students and Otis College of Art and Design design students to form a team for a cross-discipline product design competition. Each team was tasked with creating an environmentally friendly smart consumer device and was composed of four engineering students, two design students, and one marketing student.

To start the design process, all three teams had to develop three concepts. The marketing student then performed market research to finalize which concept would be developed. Our research showed that a smart shower device was most desirable. As part of the class, we selected a brand to develop the product under, and we selected Samsung due to their wide range of household smart devices.

After selecting our specific product, each sub-group in the team began working to produce materials relevant to their field for a new product's development. The marketing student created a marketing plan, the design students created a "looks-like" prototype, and the engineering students created a "works-like" prototype. At the end of the semester, each team gave a pitch-like presentation for their product in a competition for best product. Our product, the Samsung Smart Shower, won first place in the class.

1.2 Problem

For our project, we focused on water conservation as the environmentally friendly aspect of our product. As students in southern California, we have been hearing about the importance of water conservation for years. We wanted to make a product that would be able to provide a way to educate users about how much water they use and encourage them to take steps to convserve water.

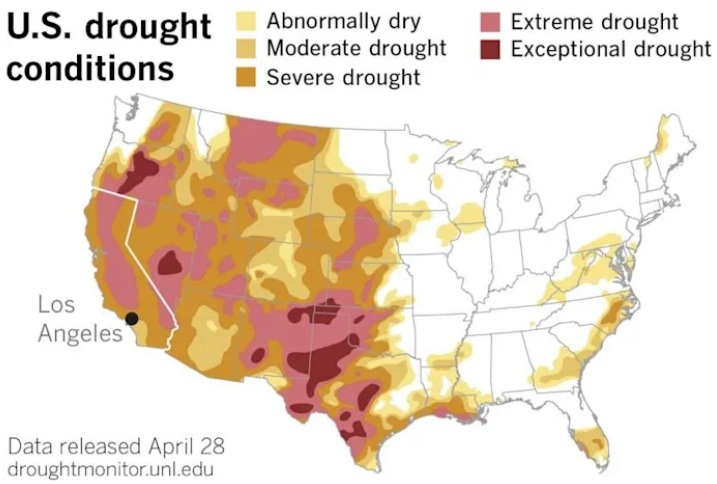


Figure 1: California faces severe and extreme drought [1].

It is no surprise that California and most of the U.S. Southwest faces a serious drought, limiting the amount of water available. With current expectations, the drought will continue unless serious actions are taken to mitigate the effects of climate change by enhancing the current infrastructure surrounding the collection and usage of water [1]. This is our problem: What product can be made that can help mitigate the effects of drought?

As engineers, we decided to attempt to throw modern technology at a centuries old issue. We wanted to focus on an area where a lot of water is used by every household on an everyday basis. While many aspects of water usage are already accompanied by a smart device designed to regulate the use of water efficiently, including smart clothes washers, smart dishwashers, and smart sprinkler systems, there is a severe lack of options related to what goes on in the bathroom, specifically in the shower.

The average shower lasts about 7.8 minutes long. With an average showerhead water flow of about 2.1 gallons/minute, that equates to 16.4 gallons of water per shower [2]. Additionally, roughly half of a kWh of energy is consumed per minute if the shower is using heated water [3]. Therefore, the average shower uses about 4 kWh, so 15 showers is roughly the equivalent of charging the average electric car at 61.5 kWh per full battery charge [4].

We did not realize how much water and energy we used by simply showering, and we believe that most consumers are unaware as well. It can be hard to know how long a shower is taking unless you brought a timer into the shower with you, let alone how much water is being used. It might be possible for a user to check their water meter before a shower, take the shower, and then check again after, but this is a hassle that most people would not think to do, and it might not be accurate due to other sources of simultaneous water consumption in a household.

To address this issue, we have designed a consumer device that hopes educate consumers about how much water they use when showering. Our product aims to show you exactly how much

water you are consuming during your daily showers, along with a cost calculation to shower the user exactly how much a shower costs. In this way, we hope to bring awareness to the damage long showers are doing both environmentally and economically. Additionally, we incorporated quality-of-life-improving features that are sure to upgrade the showering experience of a consumer.

2. Product Prototype Overview

Our prototype is a “works-like” design of the Smart Shower system. While the actual physical design of the product will look very different, as shown in Figure 2, our prototype shows all the core features needed for the Smart Shower system without having to worry about aesthetics. This process allowed our team to rapidly design and test components in order to quickly discover component issues, ideate solutions, and pivot to a successful implementation.

Figure 2: Smart Shower concept design by our team’s Otis College design students, Reanna Brown and Sophia Li.

We have modeled our product on the Samsung line of Smart Devices. Samsung has claimed to be part of an eco-friendly initiative which includes implementing more environmentally friendly packaging [5]. They also take part in reducing their carbon footprint by using renewable energy and have been using 100% renewable energy in their facilities in China and Europe since 2020 [5]. They make an effort to produce eco-conscious products by using recycled materials for parts of their products and creating products using alternative energy [5]. We believe our shower is perfect for Samsung as the shower head can be produced using recycled stainless steel. The smart dial is also able to track usage of water and time spent in the shower to urge users to save water.

In our final works-like prototype, shown in Figure 3, our design features desired temperature control, a temperature notification LED, a shower timer, total shower water usage to the tenth of a gallon, a final cost estimate of the shower just taken, and current bathroom humidity.

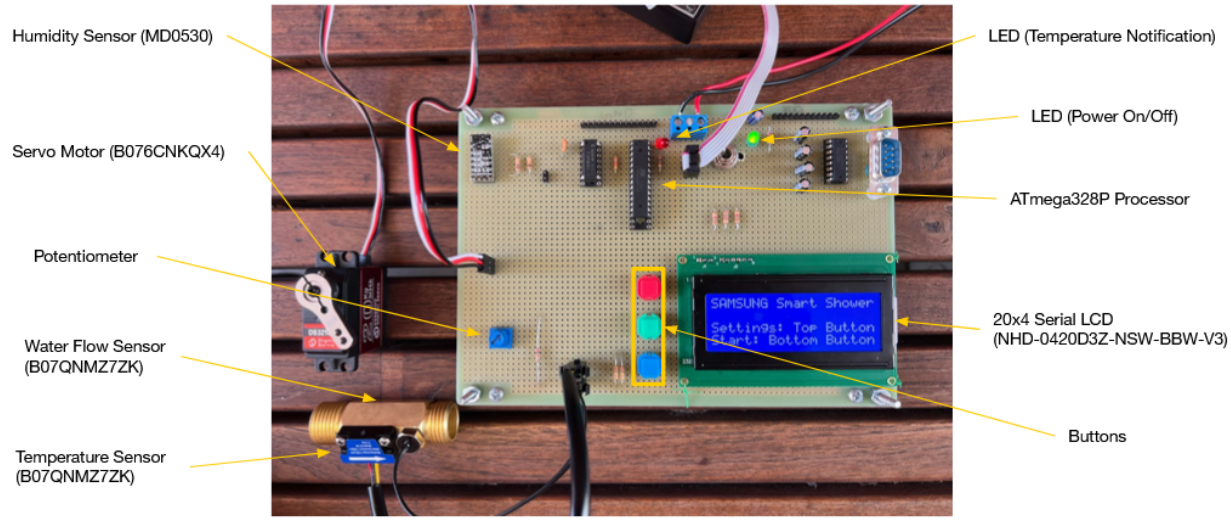


Figure 3: Final "works-like" prototype board.

2.1 Features

2.1.1 Full Temperature Control

The Smart Shower uses a high torque servo, potentiometer, and mixing valve to achieve the user's desired water temperature. The potentiometer provides a variable resistance that correlates the output voltage with a temperature range from 50-120℉, the standard household water temperature range. The temperature value is then converted to a set of electrical pulses that move the servo arm and, consequently, the mixing valve control, to a corresponding location on a 270° range using pulse width modulation (PWM). This allows for precise temperature control to the tenth of a degree.

2.1.2 Temperature Notification via LED

After the user has set a preferred water temperature, we compare the preferred temperature to the actual water temperature flowing out of the shower. To measure the actual temperature, we use a thermistor as part of a voltage divider. The thermistor resistance changes based on temperature, allowing us to convert the output voltage to a precise temperature measurement. Once the preferred temperature and the temperature measurement are within five-tenths of a degree of each other, a signal is sent to an LED to indicate the water temperatures match. This allows the

user to know right away that the water is warm, preventing them from wasting water while waiting for the water to heat up.

2.1.3 Shower Timer

The shower timer is one of the primary “green” features of the Smart Shower. The timer begins when the user sets their desired temperature, and is shown to the user on the LCD display throughout the shower. The timer can be paused and started again as needed if the shower is interrupted. At the end of the shower, the LCD will display the total shower time until the user returns to the landing page of the Smart Shower.

2.1.4 Water Usage and Shower Cost Estimation

The water usage display is another primary “green” feature of the Smart Shower. As the user is showering, the total water used in gallons appears on the LCD display. Additionally, the user can input the cost of their water per gallon into the Smart Shower settings, allowing for the calculation of a total cost estimate of each shower. At the end of the shower, the total water used in gallons is displayed, along with the total calculated cost of the shower. The cost per gallon is written into EEPROM memory which prevents users from having to re-enter the value at the beginning of the next shower, or in the unlikely event of a loss of power to the house.

2.1.5 Humidity Indication

According to the EPA, mold thrives in environments with humidity levels above 60% [6]. Hot showers can drastically increase bathroom humidity, so the Smart Shower uses a humidity sensor to let users know the current relative humidity of their bathroom as they shower. The humidity level of the bathroom is displayed on the LCD display throughout the shower, allowing users to monitor and reduce high humidity levels as needed so the user can ensure their bathroom has a reduced chance of mold growth.

2.2 Block Diagrams

The three following block diagrams describe the inner workings of the Smart Shower prototype. The state machine controls what is displayed on the LCD, as well as what certain inputs do depending on the state. Our hardware block diagram shows the inputs and outputs of the system from the hardware perspective, and the software block diagram does the same for the software of the project.

2.2.1 Hardware Block Diagram

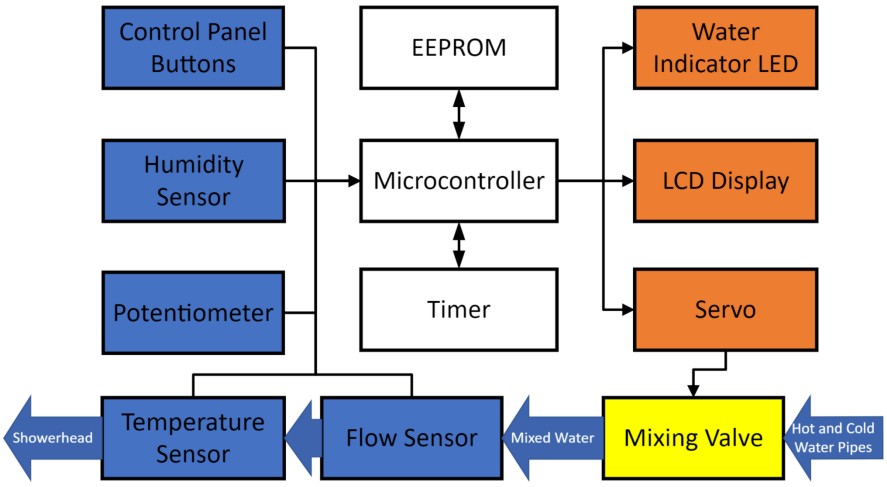


Figure 4: Hardware Block Diagram

The hardware block diagram describes the flow of information through our prototype with inputs on the left in blue, computations in the middle in white, and outputs on the right in orange. The bottom row shows the flow of water through the system and the water's consequent inputs to the system.

2.2.2 Software Block Diagram

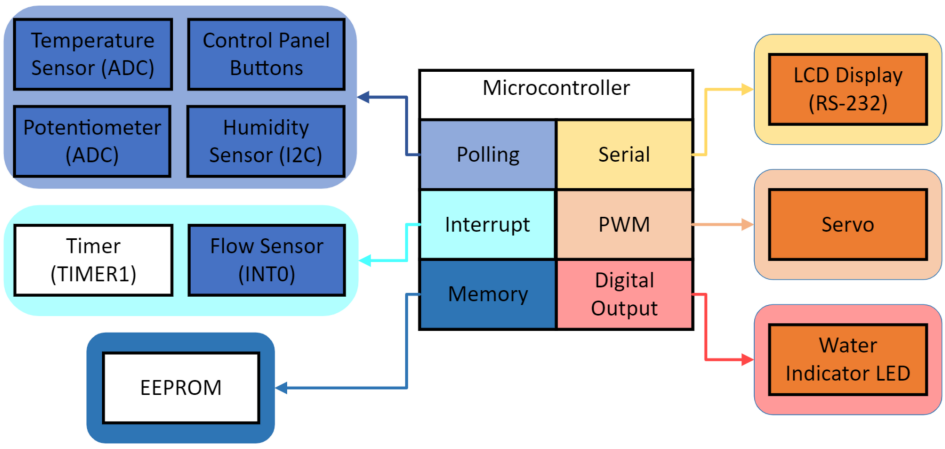


Figure 5: Software Block Diagram

The software block diagram describes the layout of the signals controlled by our microcontroller. On the left are our inputs used to make any needed calculations. On the right are our outputs used to control hardware behavior.

3. Breakdown Of Major Device Components

3.1 Microcontroller

We selected the ATmega328P microcontroller to operate as the brains of our device. Its wide use, cheap cost, and flexibility make it a great choice for project prototypes. It is a very popular microcontroller, highlighted by its selection as the microcontroller in the Arduino Uno. Each member of our team had familiarity with programming Arduinos and flashing an ATmega328P directly. Additionally, the ATmega328P provided all of the performance and internal components necessary for our Smart Shower prototype.

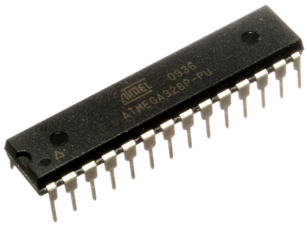


Figure 6: ATmega328P Microcontroller [7]

3.1.1 GPIO Pins

The ATmega328P microcontroller comes in a 28 pin package that provides the ability to communicate with external components. It can be programmed with a simple USB to ISP connected to six of its pins, allowing a Windows, Mac, or Linux computer to easily program the microcontroller with some basic software installations (see “Makefile” in Appendix C) [8]. In our device, we provide an external 7.3728 MHz clock to the microcontroller because it was simple to set up, reliable, fast enough for our device, cheap, and only takes one pin instead of two. When a clock signal is provided externally, the ATmega328P provides 21 general-purpose input/output (GPIO) pins. Six of the GPIO pins can be used as analog inputs. The pinout of the ATmega328P is shown in Figure 7 below.

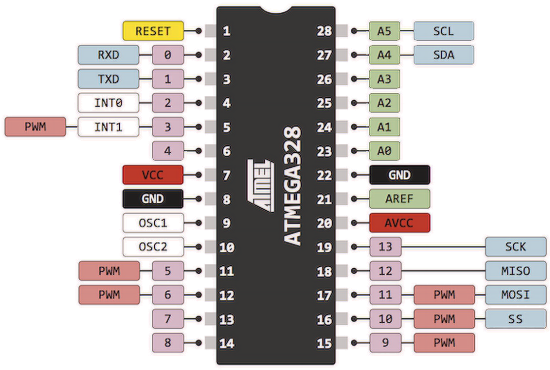


Figure 7: ATmega328P pinout [9]

3.1.2 Communication

The ATmega328P provides the capability to communicate with external devices through the GPIO pins. Multiple serial communication interfaces are provided. Two pins can be used to communicate with the universal synchronous and asynchronous receiver-transmitter (USART) protocol, with one pin being used for transmission (TxD) and the other for receiving (RxD). Two other pins allow communication with the inter-integrated circuit (I^2C) protocol, one pin for the clock (SCL) and one pin for the data (SDA) [10]. The communication setup and process for these two protocols are shown in "LCD" (USART) and "I^2C" in Appendix C. Other pins can also be configured to capture external interrupts or output pulse width modulation (PWM) signals. Additionally, the ATmega328P supports internal and external interrupts with custom interrupt service routines (ISR). Because of this, pins can be configured so that an interrupt is generated when there is a voltage change on the pin.

3.1.3 Analog-to-Digital Converter

Internally, the ATmega328P contains an analog-to-digital converter (ADC) which allows analog inputs on the analog pins to be converted to digital values. We decided to operate our ADC in 10-bit polling mode (see "ADC" in Appendix C). The analog input voltage is compared to the five volts on pin 20 (AVCC) and the result is converted into a 10-bit integer with a range of 0 to 1023. Because it is in polling mode, the main program will request for the ADC to run for a pin and wait for its result. We opted to use polling because it is fast enough for our program's two conversions per second, multiple pins need to use the ADC, and we wanted to ensure that the value is updated before the display is updated.

3.1.4 Timer

The ATmega328P microcontroller contains three internal timers. One is a 16-bit timer and the remaining two are 8-bit timers. The timers can be used to precisely measure time or to generate the correct timing for PWM signals. For our project, we use Timer 1 (16-bit timer) to measure how much time has elapsed (see "Timer" in Appendix C). After a specified number of clocks, the timer generates an interrupt that sets a flag when one second has elapsed. This allows the sensors to be queried and updated on the display once per second. A counter for the hours, minutes, and seconds is also updated by this timer and displayed. Timer 0 (8-bit timer) is used to generate the PWM output for the servo.

3.1.5 EEPROM

The ATmega328P contains 1K Bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written and has an endurance of at least 100,000 write/erase cycles [11]. The Internal EEPROM works using a combination of registers

that control, erase, read and write depending on the parameters input. EEPROM data is persistent, meaning that the values will not be wiped out if power is lost. Because of this, EEPROM can be used to store and access semi-permanent values that need to be accessed across power cycles.

For our device, the internal EEPROM on the ATmega328P is used to save the custom price of water that is set by the user (see “EEPROM” in Appendix C). To calculate the cost of the water that is used during a shower, there must be a value for how much water costs per gallon. The user can set this value once and it can be written into EEPROM. We selected the EEPROM because it can persist even if power is lost, and the small number of writes to set the cost of the power are well within the lifespan of the memory.

3.1.6 Performance

The ATmega328P has performance that is sufficient for our Smart Shower prototype. It contains an ALU with a multiplier, allowing quick calculations of water usage and cost. The ATmega328P does not contain a floating-point unit (FPU), but it provides support for floating-point operations through other instructions in its instruction set architecture (ISA). These instructions have sufficient speed because our program requires about 10 floating-point calculations per second. In our device, the floating-point operations calculate the humidity, the water temperature, and the temperature setting. The ATmega328P provides all instructions and performance that are needed for the Smart Shower despite the microcontroller’s cheap cost.

3.2 Inputs

3.2.1 DIGITEN Water Flow Sensor

The Smart Shower utilizes a DIGITEN flow sensor to measure the amount of water used by the shower. The sensor is shown below in Figure 8. We selected this sensor because it has an integrated thermistor. It is also made of brass, which is more durable than the other plastic sensors we were seeing, and it is RoHS compliant. It can be attached to two pipes so that water can flow through it.

Figure 8: DIGITEN Flow Sensor and Thermistor [12]

The flow sensor uses the hall effect to measure the amount of water that flows through the pipe. The hall effect is the principle that a voltage difference is produced in the presence of a magnetic field. If a magnet is attached to one end of the turbine, the hall effect can be used to measure when the turbine makes a full rotation. The hall effect circuit can measure that the magnet has induced a voltage difference and create a pulse output. The process is shown below in Figure 9. To interface the sensor with our microcontroller, we connected the sensor output to pin four on our microcontroller. Pin four on the ATmega328P can handle external interrupts, so an interrupt is generated on the falling edge of the pulse generated by the sensor. The microcontroller then counts the number of interrupts that are generated and converts that count into the number of gallons that have been used (see “Flow Sensor” in Appendix C).

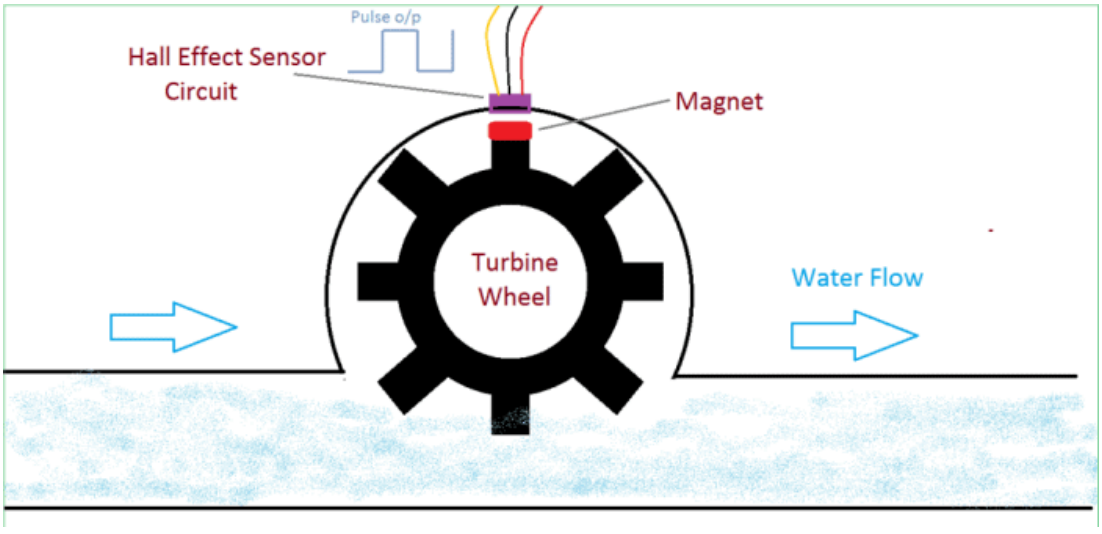


Figure 9: Flow Sensor Hall Effect [13]

Our flow sensor is compatible with a typical household shower. It has an operating flow range of 1 to 25 L/min with an accuracy of ±3%, which is sufficient for the 6.8 L/min that we expect the Smart Shower to operate under. It has an operating voltage range of 5 to 15 volts, so it can be powered by the same 5V power source that powers the microcontroller. Additionally, it can handle a maximum pressure of 1.75 MPa and a temperature range of -20℃ to 100℃, which is well within the range of a typical shower [12].

3.2.2 Temperature Sensor (Thermistor)

An NTC thermistor is used in our device as a temperature sensor to measure the temperature of the water that is flowing through the flow sensor. A thermistor is a resistor where the resistance is dependent on its temperature [14]. The thermistor can be represented by the thermistor beta equation, which is shown below in Equation 1, where $T_{current}$ is the current temperature in K, T_o is the calibration temperature in K, is the beta constant for the thermistor, R_t is the resistance at the current temperature, and R_o is the resistance at the calibration temperature.

$$\frac{1}{T_{current}} = \frac{1}{T_o} + \frac{1}{\beta} \times ln(\frac{R_t}{R_o})$$

Equation 1: Thermistor Beta Equation [14]

The resistance of the thermistor cannot be directly measured by the microcontroller. Instead, the thermistor must be put in series with another resistor to create a voltage divider. The microcontroller can then use the ADC to measure the output of the voltage divider, which can be used to calculate the current resistance of the thermistor. The voltage divider equation that solves for resistance is shown below in Equation 2, where R_t is the resistance of the thermistor, R_S is the resistance of the resistor in series, V_S is the voltage at the source of the voltage divider, and V_O is the output of the voltage divider.

$$R_t = R_s \times (\frac{V_s}{V_o} - 1)$$

Equation 2: Thermistor Voltage Divider Resistance Equation

Therefore, the current temperature can be found by plugging the resistance equation into the beta equation. When the series resistance is equivalent to the thermistor's calibration resistance, which we made sure to do in our product, the terms cancel. The final temperature equation is shown below in Equation 3.

$$T_{current} = 1/(\frac{1}{T_o} + \frac{1}{\beta} \times ln(\frac{V_s}{V_o} - 1)$$

Equation 3: Current Actual Temperature Equation

The thermistor that we use in our device is the thermistor that is integrated into the flow sensor. It can be seen alongside the flow sensor in Figure 8. It is made of RoHS compliant metal, making it a good thermal conductor and responsive to the water's temperature. The thermistor has a beta value of 3950 and a calibration resistance of 50 kΩ at room temperature (25℃) [12]. To simplify the equation as described above, we used two 100 kΩ resistors in parallel as the series resistance in the voltage divider. These parallel resistance have an equivalent resistance of 50 kΩ, which matches the 50 kΩ calibration resistance of the thermistor.

In our device, the temperature of the water is measured by starting an ADC on the pin of the output of the thermistor voltage divider. The program waits for the result and then calculates the current temperature with it. The temperature is updated once per second. The temperature equation gives the value in Kelvin, so it is converted to Fahrenheit (see "Thermistor" in Appendix C).

3.2.3 Modern Device Humidity Sensor

The Smart Shower utilizes a Modern Device humidity and temperature sensor to measure the humidity in the shower. The sensor is shown below in Figure 10. We selected this sensor because it is very small, operates using I^2C, and works in a very moist and humid environment. The sensor has an operating temperature range of -40℃ to 125℃ and humidity range of 0-100% [15]. Therefore, the sensor will operate successfully in the shower environment.

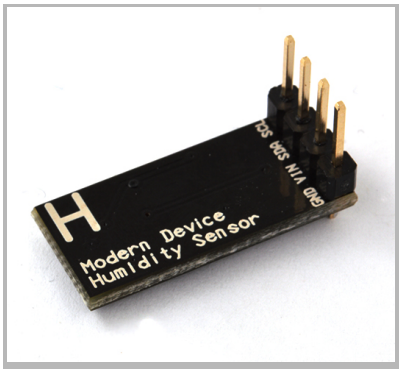


Figure 10: SHT21 Humidity and Temp Sensor [15]

The microcontroller communicates with the sensor using the I^2C protocol, allowing it to be connected with only two data pins. The two pins operate as a shared bus between devices, where the microcontroller can communicate with individual devices that are connected to the pins. Both pins are connected to 10 kΩ pull-up resistors so that the bus is considered "open drain," leaving the line high when no device is controlling it [10]. The I^2C communication always begins with the microcontroller sending a start signal to the device, followed by a byte that represents the address on the bus line to which the signal is to be received. The address byte is followed by an ACK from the sensor acknowledging the message received. Then the controller sends another byte containing the command to trigger measurement of the relative humidity. Table 1 shows the different commands that can be sent to the humidity sensor.

Command	Comment	Code
Trigger T measurement	hold master	1110'0011
Trigger RH measurement	hold master	1110'0101
Trigger T measurement	no hold master	1111'0011
Trigger RH measurement	no hold master	1111'0101
Write user register		1110'0110
Read user register		1110'0111
Soft reset		1111'1110

Table 1: Humidity Sensor Commands [16]

For our implementation, we selected the no hold master command ('1111'0101) to read the humidity as it does not complicate the communication of any additional I^2C components that may be added to the bus in the future. After sending a command signal, the sensor will respond with an ACK acknowledging that it has received the command. After receiving the ACK, the microcontroller waits 30 ms before polling the sensor with a read command ('1000'0001). When the sensor has completed the measurement, the sensor responds by sending another ACK and follows with three bytes of data. The first two bytes are the measurement while the last byte is the checksum. The calculation of the relative humidity R_H is then computed via Equation 4, where S_{RH} are the two bytes combined as one 16-bit integer (see "Humidity" in Appendix C).

$$R_H = 125 \times \frac{S_{RH}}{2^{16}} - 6$$

Equation 4: Relative Humidity Equation [16]

3.2.4 Buttons

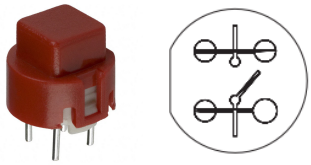


Figure 11: Digikey Button [17]

For simple navigation through our user interface, we chose a set of three buttons purchased from Digikey in three different colors (red, green, and blue). Although any button would provide the same functionality, we chose these sets specifically due to their RoHS compliance, as well as having an operating temperature range of -20℃ to 85°C, which comfortably sits within the temperature range of a shower. The Digikey 401-196X also has a level 1 classification for Moisture Sensitivity Level [17]. This means it is entirely unaffected by high moisture levels and is rated to last an unlimited amount of time under conditions of 30 Celsius and 85% Relative Humidity [18], making it perfect for a showeroom environment. On the microcontroller, we enabled digital pins 3, 4, and 6 as input pins and connected them to the red, green and blue buttons respectively.

For our project, we implemented software debouncing by employing a simple state machine with a small delay of 5 milliseconds. Because the buttons are connected to the microcontroller with external pull up resistors, the input pin on the microcontroller detects a logical zero when a button is pressed. The debouncing state machine checks if the input pin is reading a logical zero, and, if so, it recognizes that the button has been pressed. It then updates the button's state to "pressed" and executes a delay of five milliseconds. After the delay, the function will return, allowing the program to resume normal execution. As the button is held down, the state continues to be in the pressed state. The state is only updated once the button is released. After detecting that the button is released, another delay is executed to account for the release debouncing. Operating the buttons in this way allows for the program to recognize only one press per button press without freezing operation if a button is held down (see "Buttons" in Appendix C).

3.2.5 Potentiometer

A potentiometer is used in our device to capture the user's setting for water temperature. We connected the potentiometer so that the temperature increases as the knob is turned clockwise. An image of the potentiometer that we selected is shown below in Figure 12. We selected a potentiometer because it operates similarly to a dial, it cannot freely rotate, and it can reuse the same ADC code as the thermistor.

Figure 12: Potentiometer [19]

Like the thermistor, the resistance of the potentiometer can be converted into a temperature value. The resistance of the potentiometer is found through another voltage divider, similar to the one in Equation 2. The potentiometer voltage divider formula is shown below in Equation 5, where R_P is the resistance of the potentiometer and, for our design, R_S is a 20 kΩ resistor in series.

$$R_P = R_s \times \left(\frac{V_s}{V_o} - 1\right)$$

Equation 5: Thermistor Voltage Divider Resistance Equation

With the potentiometer's current resistance, the temperature setting can be calculated using Equation 6, where T_{max} is the maximum temperature, T_{min} is the minimum temperature, and R_{max} is the maximum resistance of the potentiometer.

$$T_{setting} = (T_{max} - T_{min}) \times \frac{R_P}{R_{max}} + T_{min}$$

Equation 6: Current Temperature Setting Equation

We selected a potentiometer with a resistance range of 10 Ω to 10 kΩ. The small minimum resistance makes it negligible compared to the maximum resistance, so it can be excluded in the temperature equation. The user can set the water anywhere from 50-120℉. Once per second, the potentiometer polls to start an ADC on its pin, waits for the result, and calculates the temperature with the formula (see "Potentiometer" in Appendix C).

3.3 Outputs

3.3.1 Newhaven Display 20x4 Serial LCD

The LCD is perhaps the most important output of the project, as it is used to convey shower information directly to the user. The main function of the LCD is to display the User Interface, which we will explore further in section 4.1.

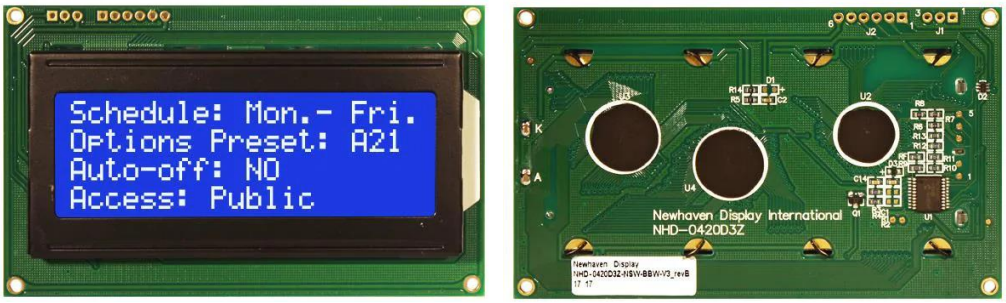


Figure 13: Newhaven Display 20 x 4 LCD [20]

We chose the Newhaven Display 0420D3Z-NSW-BBW-V3 for its outstanding set of specifications. The Newhaven Display has a display format of 20 x 4 which allows us to show more information at a given time compared to traditional 20 x 2 displays. The display also supports three different types of interface I2C, RS-232, and SPI, for better flexibility [20]. This was especially important to us as we quickly realized during our planning phase that we could easily run out of output pins on the ATmega328p, and thus using I2C or SPI would be difficult given the amount of features we planned to support. Therefore, it was a relief to find that the Newhaven Display accepts serial input as an interface, costing us only a PD1 pin on the microcontroller for serial data transmission.

Since our operating environment is inside a bathroom, it was important that the display was RoHS compliant, as well as has an operating temperature between -20℃ and 70℃ [20]. The display works by sending a serial stream of a 16 bit prefix, a command, and followed by the parameter which could range from 0 to 9 bytes depending on the functionality (see “LCD” in Appendix C). Average execution time for the display is 100 microseconds which allows for very quick transitions.

3.3.2 Servo Motor

The servo in our prototype is used to control the mix of hot and cold water flowing out of the shower. As the user selects their desired temperature using the potentiometer, the ATmega328P sends square waves to the servo. Depending on the percentage of time the pulse is high instead of low, the servo moves to the position necessary to create that corresponding temperature (see “Servo” in Appendix C).

Figure 14: ANNIMOS Servo Motor [21]

We chose a 20kg servo for its high torque capability. The servo will be moving metal against flowing water, so it needs to be strong enough to push against the force of the flowing water on top of the friction of the mixing valve itself. The servo is connected to pin 12 of the ATmega328P, which is one of six pins on the microcontroller capable of PWM [11].

3.3.3 Power and Warmed Water LED

Figure 15: LED (green and red) [22]

We placed a red LED to indicate when the desired water temperature has been reached, and a green LED to show if the smart shower is powered on. We used standard issued LEDs that were found in our lab supplies. The green LED is connected directly to the +5 V line, turning it on as soon as the device is powered on properly. The red LED is connected to PC0 of the microcontroller to enable software control. When the temperature setting is within half a degree Fahrenheit of the current water temperature, the red LED is turned on. Both LEDs are connected in series to a 220 Ω resistor to limit the current through the LED to prevent excess current that can damage the diode in the LED.

4. User Manual

Our device has a simple operation characterized by a simple user interface and four main states. The user interface consists of an LCD and three buttons, which displays four main pages that are implemented by four states.

4.1 User Interface

Our device provides a User Interface via the LCD display to showcase shower statistics. Throughout the operation of the smart shower, our LCD will show four separate landing pages, each showcasing user instructions and in shower data. These four pages can be seen below in Figure 16. The buttons, located to the right of the LCD and labeled B1, B2, and B3, indicate which screen is transitioned to after a press.

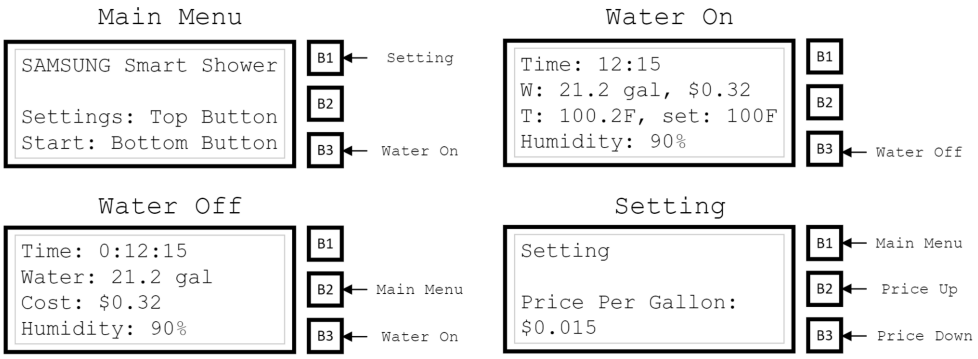


Figure 16: The LCD and buttons of each page of the User Interface.

When our device is first switched on, the LCD will show the “Main Menu” landing page, which shows our product name, as well as instructions on how to navigate through the different pages. If the top button is pressed, the user will be provided with a settings menu where they can set the price per gallon of water. When the water is turned on, the LCD will display the “Water On” landing page which shows time elapsed, water usage, cost, water temperature, desired temperature, and humidity. In this state, the user will also be able to control the desired water temperature by turning the potentiometer clockwise/anticlockwise. When the shower is paused, the LCD will display the “Water Off” landing page which shows the same statistics except for the temperature statistics.

The only other component that the user can interact with is the potentiometer. The potentiometer is used to determine the temperature setting of the water. As the user rotates the potentiometer clockwise, the temperature increases. Conversely, the temperature decreases when the user rotates the potentiometer counterclockwise. The potentiometer value is converted into a temperature setting, which changes the position of the servo and, consequently, the amount of hot and cold water that is mixed by the mixing valve.

4.2 Device Operation

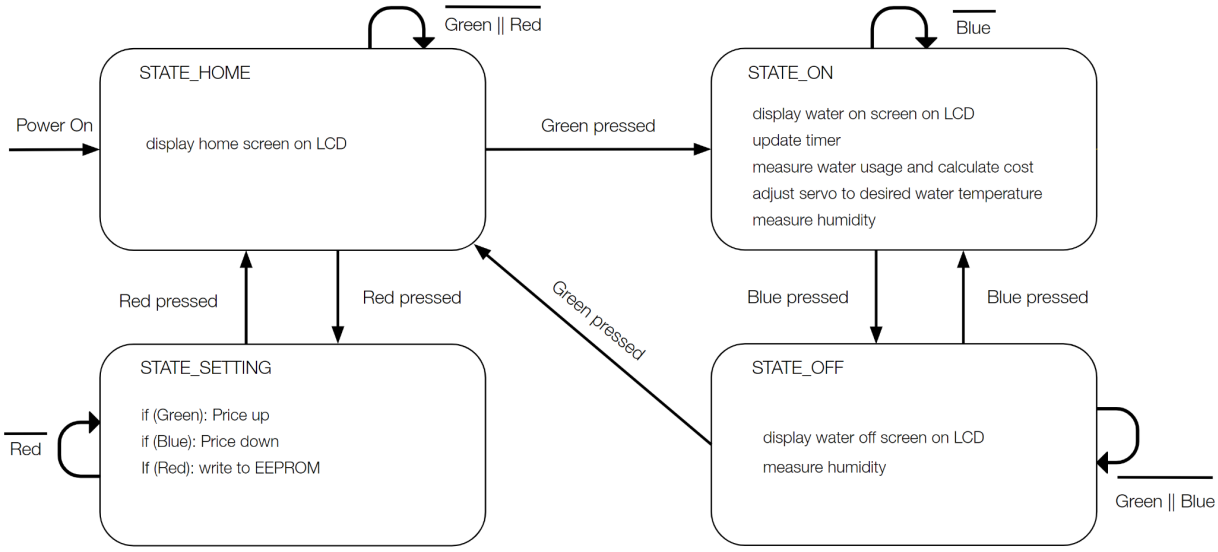


Figure 17: Device Operation State Machine

Our state machine is a straightforward implementation of the UI that is shown in Figure 17. STATE_HOME acts as a landing page for the user to see when they wake the device. STATE_ON turns on the shower and begins all of the features that are implemented by our Smart Shower device. Also during this state, the potentiometer value that is set by the user is polled and sent to the servo. STATE_OFF turns off the shower and displays the final shower cost. STATE_SETTING allows the user to adjust the cost of their water per gallon. The state transitions are controlled by button presses which allow for easy navigation through the user interface. The red button represents B1, the blue button represents B2, and the green button represents B3, respectively, in Figure 3.

5. Engineering Standards

As part of the design requirements from EE459Lx, the Smart Shower prototype needed to be created in accordance with engineering standards found in industry. Our prototype is RoHS compliant, and our code was written in accordance with Barr Embedded C Coding standards. Should we develop our prototype further, we also researched a set of engineering standards we would seek to comply with.

5.1 Restriction of Hazardous Substances

The Restriction of Hazardous Substances standards, or RoHS for short, was a set of rules introduced by the European Union that aims to prevent risks posed to human health and our environment by managing the release of heavy metals and chemicals through electronic and electrical waste [23]. The standard was adopted by the European Union in February 2003, but many states in the United States such as California, New York, New Jersey, Illinois have also adopted a similar version of the RoHS regulations [24]. The RoHS standard restricts the use of ten hazardous materials found in electrical and electronic products, namely: Lead, Cadmium, Mercury, Hexavalent Chromium, Polybrominated Biphenyls, Polybrominated Diphenyl Ethers, Bis(2-Ethylhexyl) Phthalate, Benzyl Butyl Phthalate, Dibutyl Phthalate, Diisobutyl Phthalate [23].

During the planning phase of our product, we made sure that our components met the RoHS standard so that it can safely be used in a shower environment, as any of the hazardous materials mentioned above could potentially harm the user. For example, lead exposure can lead to serious consequences for brain health, causing irreparable brain damage.

5.2 Barr Embedded C Coding Standards

The Barr Embedded C Coding Standard (BECCS) is a set of rules published and updated by The Barr Group, an independent group of software developers, to ensure code written in Embedded C is easy to read, portable, and has minimal defects [25]. Our team implemented the following standards from the guide:

- BECCS 2.1: Line widths shall not exceed 80 characters.
- BECCS 4.1a: All module names shall consist entirely of lowercase letters, numbers, and underscores. No spaces shall appear within the module's header and source file names.
- BECCS 4.1b: The header file shall identify only the procedures, constants, and data types (via prototypes or macros, #define, and typedefs, respectively) about which it is strictly necessary for other modules to be informed.
- BECCS 6.1g: Underscores shall be used to separate words in procedure names.
- BECCS 6.1h: Each procedure's name shall be descriptive of its purpose.

5.3 Other Standards for Actual Product

5.3.1 WaterSense Showerhead Standard

A major standard that our future product would strive for would be the WaterSense label. WaterSense is a voluntary partnership program sponsored by the U.S. Environmental Protection Agency (EPA) [26]. The label is given for water-efficient products. Standard showerheads use 2.5 gallons of water per minute (gpm). However, water-saving showerheads that earn the WaterSense label use no more than 2.0 gallons per minute [26]. If our product was implemented, the smart shower's showerhead would be below 2.0 gallons per minute to earn this label.

Products bearing the WaterSense label:

- Perform as well or better than their less efficient counterparts [26].
- Are 20 percent more water efficient than average products in that category [26].
- Realize water savings on a national level [26].
- Provide measurable water savings results [26].
- Achieve water efficiency through several technology options [26].

5.3.2 IEEE 1851-2012 Test Software

The IEEE 1851-2012 standard provides the criteria and framework for the design of integrated test software for sensor-based household appliances [27]. Our future product would abide by this standard by ensuring an integrated test software is defined in the codebase.

5.3.3 IEC 60529 IP Ratings

The IEC 60529 Standard was created by the International Electrotechnical Commission and details the classification of degrees of protection provided by enclosures for electrical equipment with rated voltage not exceeding 72.5 kV, which are known as IP Ratings [28]. Our product would be operating at a voltage less than 72,5 kV and, because of its location in a shower, would be at potential risk of water damage if not properly enclosed. Therefore we would strive to protect the equipment inside the enclosure against harmful effects due to the ingress of water. We would strive for an IP rating of 68, which specifies that the device is dust-tight and protected against continuous immersion in water [28].

6. Future Improvements

Our Smart Shower prototype served to act as a “works-like” prototype to perform as a proof-of-concept for the smart device. If this product were to continue in development, more steps would be required to produce future iterations of prototypes before creating a manufacturable product.

6.1 Next Prototype Iteration Testing

Due to limits in time and the scope of the class, we could only design a basic “works-like” prototype that was not installed in an actual shower. Consequently, the next iteration of a prototype for the product would be a Smart Shower that could be installed and tested in a real shower environment..

6.1.1 Real Shower Piping

While we were able to test that the flow meter worked with water going through by pouring water in through a funnel, the next steps would be to use real piping that would feed water at higher pressure through the meter. In this situation we would be able to verify that the flow meter was durable enough to withstand the higher pressure as well as assess how difficult it would be to modify the edges to allow it to fit to pre-installed piping in a wall.

6.1.2 Connecting the Servo to a Mixing Valve

Our current prototype simply moves the servo around based on the setting, but the servo is not connected to anything. The next step would be to design an attachment that connects the servo to a mixing valve. This would introduce mechanical challenges and likely require modification of the program code.

6.1.3 Mixing Valve Water Temperature Algorithm

Our current temperature setting and servo algorithm simply moves the servo to a set percentage based on the location of the potentiometer. A working shower prototype would involve a redesign of the servo algorithm to incorporate both the current temperature setting and the current water temperature, using the data in real-time to adjust the mixing valve to the correct location. With the mixing valve connected the servo and the system properly connected to real shower pipes containing hot and main water feeds, the parameters necessary to control the water temperature could be calibrated to achieve the correct the mixture of hot and cold water to match the temperature specified by the user. To save water after the water is heated up to the user’s initial temperature setting, it would also be possible to add another state in the state machine where the water is turned off and the device enters a waiting state with the water ready.

6.2 More Features

In both implementing and testing our device, we saw the possibility of additional features being incorporated into the product with varying levels of additional components.

6.2.1 Timed Shower

The timer could be modified so that the user can set a predetermined amount of time for the shower to run. This could be used for the product to align with standards set during periods of drought that limit the amount of showering and watering. If a user is unaware of the amount of gallons used and the impact on cost, they could be supplied with a simple time metric to abide by. The user could then set a reminder that would alert them when the time is up or have the shower automatically stop after the set amount of time.

6.2.2 Phone App For Better Water Usage Tracking

Adding bluetooth or network connectivity to the system would enable the product to get real time data and allow for a better graphical representation of the data. A phone app could be created to connect to the Smart Shower product in order to gather and store water usage data. Additionally, the creation of a phone app could allow the user to start the shower by simply setting the shower settings on the app.

6.2.3 Additional Statistics

The cost of the energy used to heat up the water is currently not incorporated into our shower cost calculation. Additional price metrics could be included that would be able to calculate how much energy is used during a shower and how much that energy costs, providing a more accurate estimate of the cost of a shower and, ideally, contributing to water and cost saving habits.

6.2.4 Proper Integration of Smart Shower with a Smart Water Heater

As was suggested to us during our presentation phase of the class, the system could be modified to communicate with any pre-existing smart infrastructure that controls the production of hot water in a household. If our product is paired with a smart water heater, the cold water wasted while waiting for the shower to heat up will no longer be an issue because the water will begin dispensing water at the user’s set temperature immediately.

7. Conclusion and Acknowledgements

Bringing the Smart Shower prototype from an idea to a physical realization was a new journey for our team and we are extremely grateful for the opportunity to create an idea that was wholly our own. We'd like to thank Dr. Allen Weber for his aid and guidance throughout EE459Lx, Therese Wilbur for partnering her marketing organization with the EE459Lx class, Lance Winkel for partnering his design course at Otis with our class, Reanna Brown and Sophia Li for their beautiful ideas and renderings for the final design of the Smart Shower, and Alexis Lee for her work conducting our marketing research.

References

[1] P. Duginski, A. Wigglesworth. "With water running out, California faces grim summer of dangerous heat, extreme drought." Los Angeles Times. https://news.yahoo.com/water-running-california-sees-no-120058249.html (Accessed May 09, 2022).

[2] "Showers: Showering to Savings." HomeWaterWorks. https://home-water-works.org/indoor-use/showers (accessed May 09, 2022).

[3] "How many watts does shower use?." Find Any Answer. https://findanyanswer.com/how-many-watts-does-shower-use (accessed May 09, 2022).

[4] "Useable battery capacity of full electric vehicles." Electric Vehicle Database. https://ev-database.org/cheatsheet/useable-battery-capacity-electric-car (accessed May 09, 2022)

[5] "Samsung Electronics Sustainability Report 2021." Samsung. https://image-us.samsung.com/SamsungUS/home/pdf/Samsung-Electronics-Sustainability-Report-2021.pdf (accessed May 09, 2022).

[6] "Mold Course Chapter 2: Why and Where Mold Grows." EPA. https://www.epa.gov/mold/mold-course-chapter-2 (accessed May 08, 2022).

[7] "ATmega328." Wikipedia. https://en.wikipedia.org/wiki/ATmega328 (accessed May 05, 2022).

[8] Allan G. Weber. *The Microchip ATmega328P Microcontroller,* (2021). Accessed May 09, 2022. [Online]. https://ece-classes.usc.edu/ee459/documents/AT328_Handout.pdf

[9] R. Brown. “ATmega328P Microcontroller Pinout & Features.” NerdyTechy. https://nerdytechy.com/atmega328p-pinout/ (accessed May 05, 2022).

[10] Allan G. Weber. *Using the I2C Interface on the ATmega328P and MC908JL16,* (2017). Accessed May 09, 2022. [Online]. https://ece-classes.usc.edu/ee459/library/documents/I2C.pdf

[11] *8-bit AVR Microcontrollers ATmega328/P,* (2016). Accessed May 09, 2022. [Online]. https://ece-classes.usc.edu/ee459/library/documents/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf

[12] “DIGITEN G1/2"Male Thread Water Flow Hall Sensor Switch Flowmeter Counter with Temperature Sensor 1-25L/min.” Digiten. https://www.digiten.shop/products/digiten-g1-2male-thread-water-flow-hall-sensor-switch-flowmeter-counter-with-temperature-sensor-1-25l-min (accessed May 05, 2022).

[13] “Arduino Water Flow Sensor to Measure Flow Rate & Volume.” How2Electronics. https://how2electronics.com/arduino-water-flow-sensor-measure-flow-rate-volume/ (accessed May 05, 2022).

[14] P. Kane. “Thermistors/Temperature Measurement with NTC Thermistors.” Jameco. https://www.jameco.com/Jameco/workshop/TechTip/temperature-measurement-ntc-thermistors.html (accessed May 03, 2022).

[15] “SHT21 Humidity and Temp Sensor.” Modern Device. https://moderndevice.com/products/sht21-humidity-and-temp-sensor (accessed May 05, 2022).

[16] *Datasheet SHT21 Humidity and Temperature Sensor IC,* Edition 4 (2014). Accessed: May 09, 2022. [Online]. https://sensirion.com/media/documents/120BBE4C/61642236/Sensirion_Humidity_Sensors_SHT21_Datasheet.pdf

[17] “D6C40 F1 LFS.” Digikey. https://www.digikey.com/en/products/detail/c-k/D6C40-F1-LFS/1466323 (accessed May 05, 2022).

[18] “What is the 'floor life' for the different moisture sensitivity levels?.” Surface Mount Process. https://www.surfacemountprocess.com/moisture-sensitivity-level.html (accessed May 05, 2022).

[19] “Trimpot 10K Ohm with Knob.” Sparkfun. https://www.sparkfun.com/products/9806 (accessed May 05, 2022).

[20] “Newhaven Display NHD-0420D3Z-NSW-BBW-V3.” Mouser Electronics. https://www.mouser.com/ProductDetail/Newhaven-Display/NHD-0420D3Z-NSW-BBW-V3?qs=qFF%252BBcFBHbaY5gMvZJqjGw%3D%3D (accessed May 06, 2022).

[21] “ANNIMOS 20KG Digital Servo High Torque Full Metal Gear Waterproof For RC Model DIY (Ds3218mg Datasheet).” Pib Deals. https://pibpac.org/product/annimos-20kg-digital-servo-high-torque-full-metal-gear-waterproof-for-rc-model-diy-ds3218mg-datasheet/ (accessed May 08, 2022).

[22] “uxcell 16pcs 2 Colors x 8pcs 5mm Red Green LED Diode Lights Colored Lens Diffused Round 20mA Lighting Bulb Lamp Electronic Components Light Emitting Diodes.” Amazon. https://www.amazon.com/uxcell-Diffused-Lighting-Electronic-Components/dp/B07G2Z3SG1 (accessed May 05, 2022).

[23] “Welcome to RoHS Guide.” RoHS Guide. https://www.rohsguide.com/ (accessed May 05, 2022).

[24] V. Cheng. “RoHS Requirements in the United States: An Overview.” ComplianceGate. https://www.compliancegate.com/rohs-requirements-united-states/ (accessed May 05, 2022).

[25] Michael Barr. *Embedded C Coding Standard*, Edition: Barr-C: 2018. Accessed: May 07, 2022. [Online]. Available: https://barrgroup.com/sites/default/files/barr_c_coding_standard_2018.pdf

[26] “About WaterSense.” United States Environmental Protection Agency. https://www.epa.gov/watersense/about-watersense (accessed May 09, 2022).

[27] “1851-2012 - IEEE Standard for Design Criteria of Integrated Sensor-Based Test Applications for Household Appliances.” IEEE Xplore. https://ieeexplore.ieee.org/document/6265331 (accessed May 09, 2022).

[28] “IP Ratings.” International Electrotechnical Commission. https://www.iec.ch/ip-ratings (accessed May 09, 2022).

[29] “7.3728 MHz SMD/SMT Crystals”. Mouser Electronics. https://www.mouser.com/c/passive-components/frequency-control-timing-devices/crystals/?frequency=7.3728%20MHz&termination%20style=SMD%2FSMT (accessed May 05, 2022).

Appendix A: Schematic

Our prototype design utilizes 14 out of the 21 available GPIO pins on the ATmega328P. If more features were to be added in the future, I^2C components or a more capable microcontroller would need to be selected.

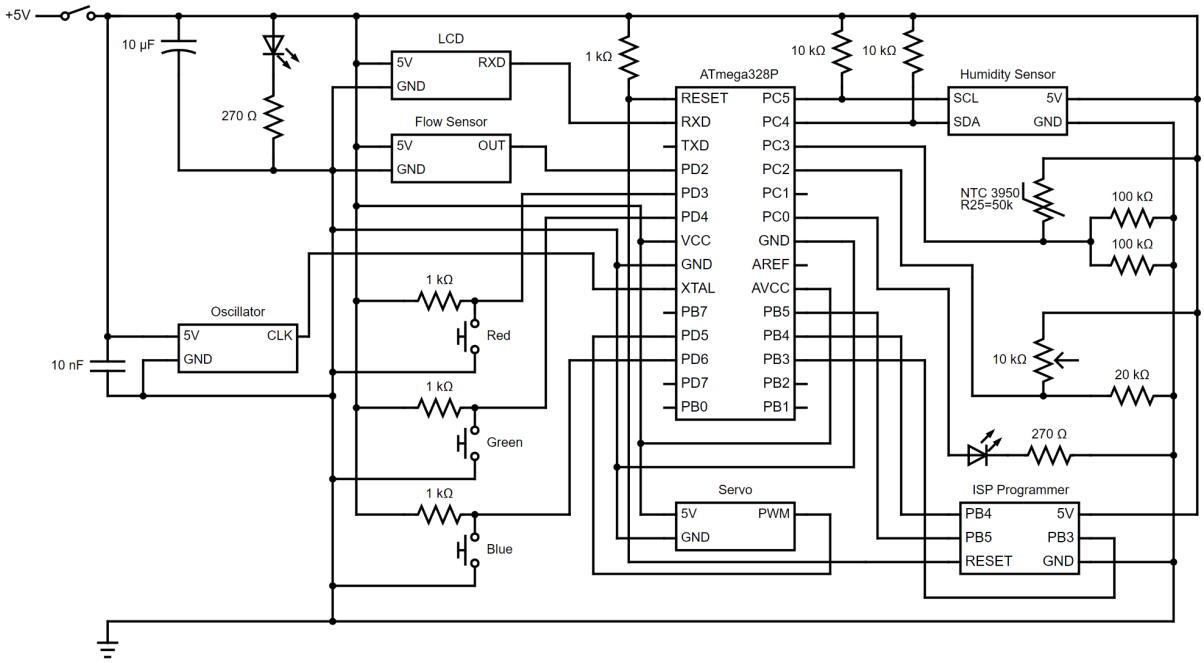


Figure A1: Schematic of Smart Shower circuit and connections.

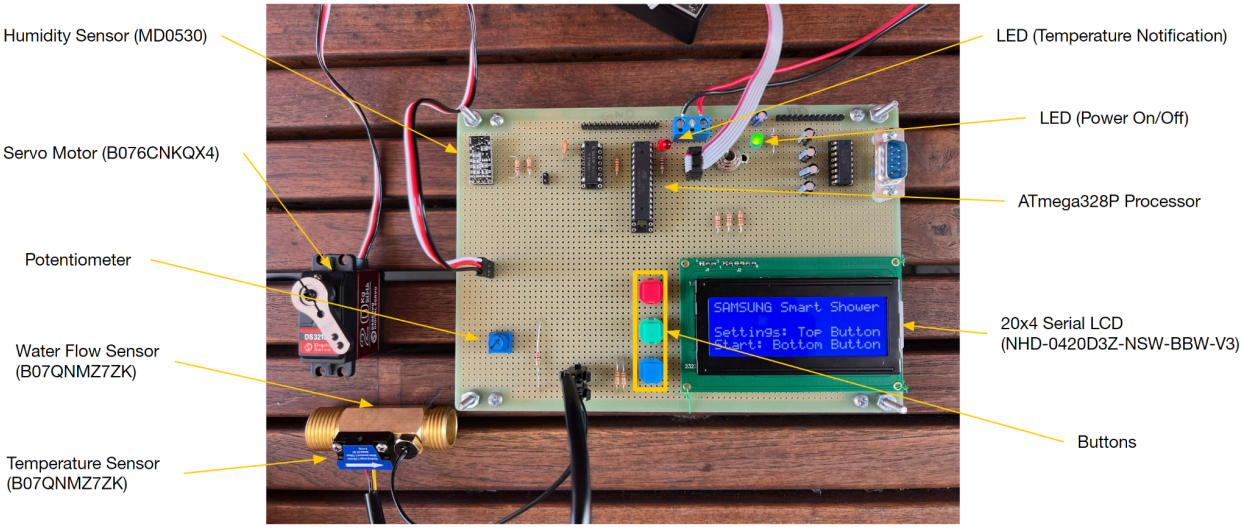


Figure A2: Component side of final prototype board.

While the components were mounted on the component side of the board, the soldering was done on the wiring side of the board. Additionally, the connective wiring for the board was done using wire wrapping.

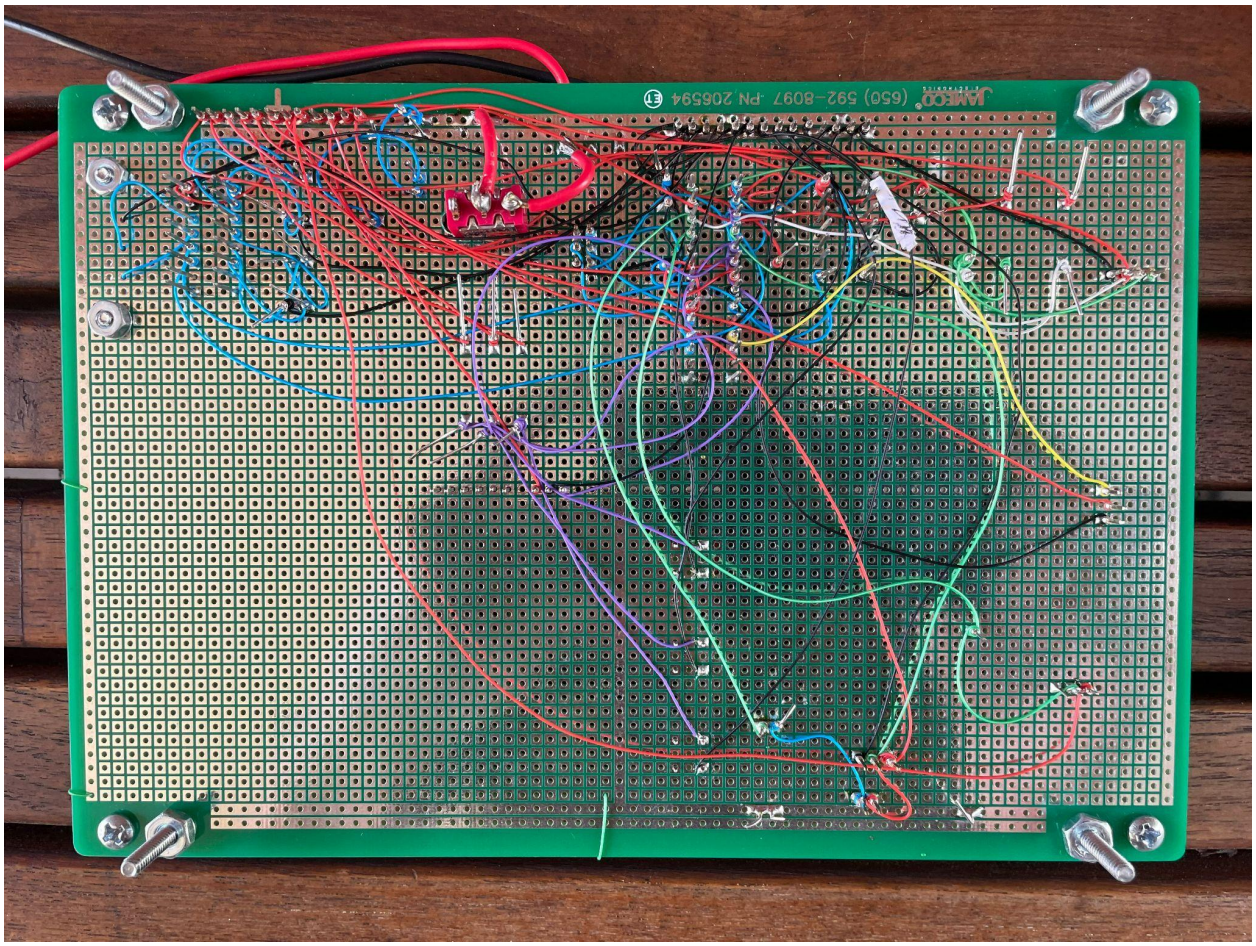


Figure A3: Wire wrapped side of final prototype board.

Appendix B: Parts And Costs

Prototype Parts and Costs

- DIGITEN G1/2” Water Flow Hall Sensor with Temperature Sensor = $16 [12]
- ANNIMOS servo motor = $16 [21]
- Modern Device SHT21 Humidity Sensor = $12 [15]
- Newhaven Display NHD-0420D3Z-NSW-BBW-V3 20x4 LCD = $30 [20]
- Buttons = $1 * 3 = $3 [17]
- LED = $0.10 * 2 = $0.20
- Sparkfun potentiometer = $1 [19]
- ATmega328P microcontroller = $3
- Clock = $0.35 [29]
- Miscellaneous (capacitors, resistors, pin headers, etc.) = $3

Total Prototype Cost: ~$85

Estimated Product Cost

Additional Costs in real device:

- Wall adapter power supply
- Better LCD
- Showerhead
- Mixing Valve
- Piping
- Other business costs (marketing, packaging, distribution, etc.)

Final Product Cost to Manufacture (COGS): ~$200 plus installation

Appendix C: Code

We implemented our code one input or output at a time, with each having its own directory within the “sensors” subdirectory and its own “main” file to test it. After verifying all the software of individual components worked, we began integrating all of them into one “main” file. This “main” file is located in the root directory and is flashed onto the ATmega328P using a Makefile. Our code repository directory is visualized below in Figure C.

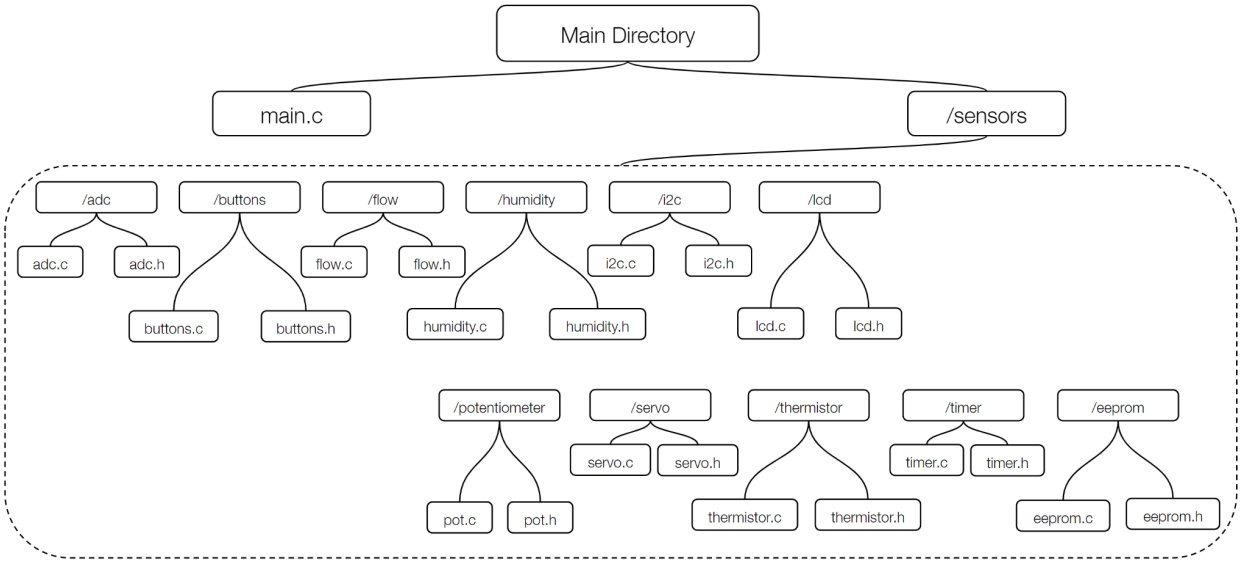


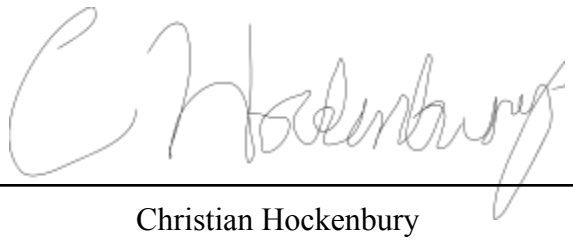
Figure C: Diagram of our software’s directory structure.

Our code base is too large to be included directly in this report. Please contact any of the authors for a copy of the code or visit the following Github repository:

https://github.com/Victorhu1/SmartShower

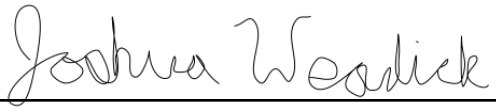
Appendix D: Work Distribution

	Hockenbury	Hui	Weadick	Williams
Labs	25	25	25	25
DDR	25	25	25	25
Hardware Design	5	45	10	40
Software Design	10	20	10	60
Cross-Discipline Presentation	5	10	5	80
Project Report (Oral)	5	65	5	25
Project Report (Written)	20	20	30	30

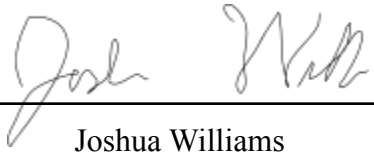


Christian Hockenbury — 5/9/2022

Victor Hui — 5/9/2022



Joshua Weadick — 5/9/2022



Joshua Williams — 5/9/22