Smart Shower

Lucas Gillespie, Andres Huertas, Alex Power, Nicholas Stoll

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — The objective of this project is to simplify the act of taking a shower. Allowing the user of the product to enjoy a temperature-controlled automated shower system without having the dependability of water temperature fluctuation. The solution is to precisely contain and restrain the temperature the user desires via an in-the-shower interface as well as access through a mobile device. This project was chosen because as a team we felt that the act of taking a shower has been mundane for centuries and has a great deal of room for improvement. The Smart Shower team believes that our product can change the way people enjoy showers.**

I. Introduction

Showers can be dated back to the times of the Ancient Greeks and since then the concept has not revolutionized. Some technology has been added to facilitate the use throughout the years, like the introduction of valves, pressurized water, and the adjustment of water temperature. Any other house appliance in a modern, 21st century house, has had the opportunity to be upgraded to relevant technologies except the shower system. Keeping this mundane piece of technology is what users are comfortable with. Perhaps society does not want to introduce a new "solution" for the shower because they have the mindset of *"If it's not broken why fix it*?" But that is where the misunderstanding is prominent. It is not about fixing something that already works but rather improving a legacy piece of technology to better the lives of the users by creating collection of software and hardware technology that will become a new standard feature in everyone's home.

This concept that has been developed by the creators of Smart Shower allows the users to simply simplify their lives. When a person wakes up and goes to sleep, he or she will encounter our product and the result of this interaction will be positive. That is because the product is created for no other reason than to assist the user in this daily routine. Our development team's goal is to allow the user to have precise, automated, and intuitive features that a manual shower system does not offer. This includes exact water temperature control, timers, and water consumption statistics. With the shower alarm and temperature control features, the user is able to set the desired start time for their shower at their desired temperature. A use case for this feature originated from your everyday workers or students that need to wake up early in the morning and have limited time to take their shower. In addition, an important aspect of water consumption is to be integrated throughout the product. A statistical report of previous showers taken with water used during the showers will be displayed to the user.

II. system Components

Due to the complexity of the device, it's easier to break it down into its major sub-components and explain them from there for better clarity. This section provides a semi-technical introduction to each of these components.

A. Microcontroller

The primary functionality of our Smart Shower system is only made possibly by the use of a microcontroller. All physical functionality such as temperature control and water flow measurement will be linked to the microcontroller. The microcontroller will interface with sensors and a servo to provides these features.

For our project, we decided to use the ESP8266; a low-cost microcontroller created by Espressif that has a fully integrated 802.11 wireless stack. The ESP8266 was determined to have all the necessary functionality for our project while also being the most cost-effective making it an easy choice.

B. PCB

The main component of the project that binds all the electrical components together is the mother board. Designed in the electronic design automation (EDA) program KiCAD and manufactured by Osh Park manufacturing, it's a single board construction that manages most of the electronic components for the project. This includes power regulation for the boards three power rails, the microcontroller and all of its supporting circuitry, and binding points for the external interfaces used.

C. LCD Interface

While in the shower our users will use a touchscreen display interface to interact with our system. This interface will allow the user to turn the shower on and off, control the temperature, as well as view helpful information relative to the shower experience and the day. The display unit uses a resistive touch element allowing for input from the user even when wet.

D. Servo

In order to physically control the shower in our system we must have some sort of hardware that can bridge the gap between our microcontroller and the shower valve. To do this, we decided to use a servomechanism. A servomechanism is a device that uses a feedback loop to rotate a shaft to a specific location as determined by a received encoded signal. Using a servomechanism (abbreviated to servo) for our project will allow us to have precise physical control of our shower valve. This is necessary as our temperature control system will need the ability to make precise small adjustments to accurately control the temperature of the water.

E. Sensors

One of the main features of our smart shower is water temperature control. To obtain a temperature that is desired by the user, we must be able to measure current water temperature so the system may compensate by adjusting the shower valve. The feedback provided to the control system by the temperature sensor is necessary for the system to provide this functionality.

Another major feature of our smart shower is water flow measurement; This will allow to the user to view statistics on water use and provide necessary information to the control system for limiting user water consumption if desired.

F. Enclosure

Due to the expected environmental conditions that the device as a whole will be in, carful considerations were made to keep the electronics used for the project moisture free. This was done by designing an enclosure with an overlapping seam to reduce water infiltration that can also be back filled with a rubber gasket or liquid sealant like silicone. This same mentality was taken towards the external interfaces when selecting water resistant connectors, as they too would need to be water resistant to inhibit corrosion of the contacts. The connectors were then mounted to the chassis with epoxy resin for water resistance and mechanical rigidity.

G. Android Mobile Device

The Smart Shower mobile application will allow for remote control of the smart shower device. The mobile development portion of the project was programmed for Android devices to be specific. The Target SDK Version of the application is API 25: Android 7.1.1 also known as “Nougat”. Although however it does support a minimum SDK version of 19 being KitKat.

III. SYSTEM CONCEPT

A Use Case Diagram as seen below in Figure 1, will better break down the Smart Shower System.

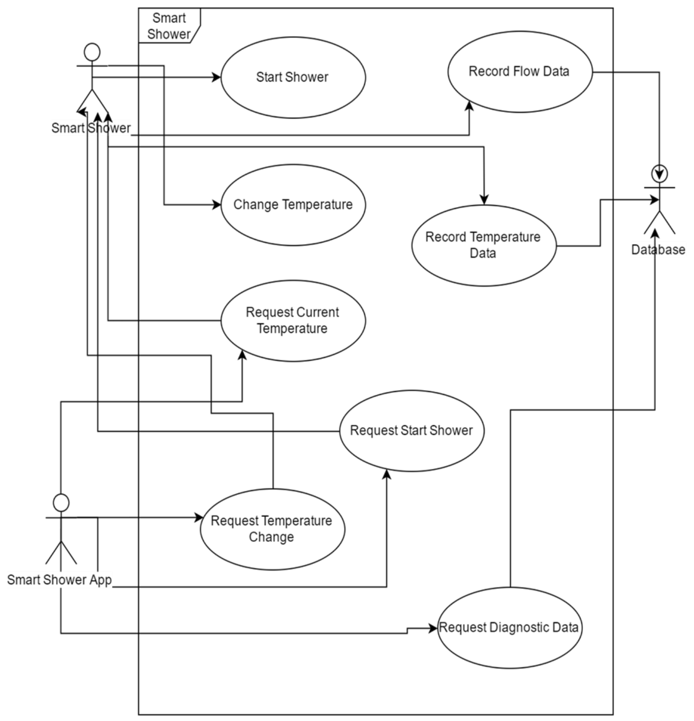


Figure 1: Complete System Use Case Diagram

IV. HARDWARE DETAIL

A. Microcontroller

At less than a dollar per chip, the ESP8266 offers all of the features needed in a microcontroller for our project. Relevant features to our project include: Integrated 802.11 b/g/n with a TCP/IP stack, powerful 32 bit 80Mhz CPU, 512 kb of flash storage, and 16 GPIO allowing UART, SPI, I2S, I2C communication protocols. With these stated features, developing our project on the ESP8266 was not an issue. The plentiful GPIO allow for integration with our display module, servo, sensors and for control of the system.

B. PCB

As outlined in section II, the PCB was designed in KiCAD, being a complete electronic design sweet the designer has the ability to start with a blank circuit design and move directly into laying out a PCB by simply assigning mechanical footprints to the components.

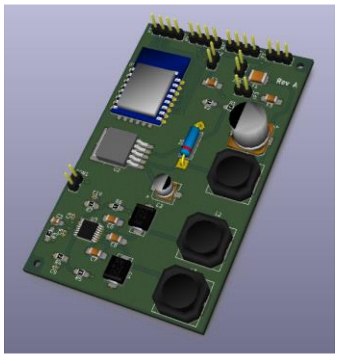


Figure 4: PCB Rendering

Along the way there are a few design considerations to make that could be impotent or just convenient later on. Component choice is an endless struggle, in the case of this project surface mount devices (SMD) were used to keep the PCB area lower by using small components as well as both sides of the board. This choice to a lesser extent results in a lower bill of material (BOM) cost. The other crucial consideration for the project is trace routing. For most of the I/O devices they don't have any special needs as they are low-power low-speed devices that will have no trouble with standard sized traces. However the high torque servo motor also requires sizable amounts of current when it runs. This results in the need for the trace size to be increased because it's basically a flat wire, and the carful use of vias if the trace needs to pass through the board. By deliberate design, the power regulator for the servo was intentionally placed closer to the servo header than the others to reduce it's length and interference to any of the other component's mounting pads.

The final steps were to assemble and test the board. This was done by hand soldering each component to the board which is made easy from the design stage by using over sized pads for the components rather than the standard size used for reflow soldering. Testing is done by starting with a visual overview to spot any solder bridges or abnormal residues that could conduct between pins and then by adding power to the board with the power jumpers removed and verifying that each power rail is within specification.

C. LCD Interface

The Nextion Intelligent Display is the touchscreen display chosen for this project. The Nextion display line offers all of the features we need in our project from a standalone display solution, specifically the NX4832T035_011R Nextion display model. All Nextion (touchscreen) displays (As of 3/29/2017) use resistive touch for touch input. All Nextion displays communicate via serial. All Nextion displays are compatible with their easy to use interface development software.

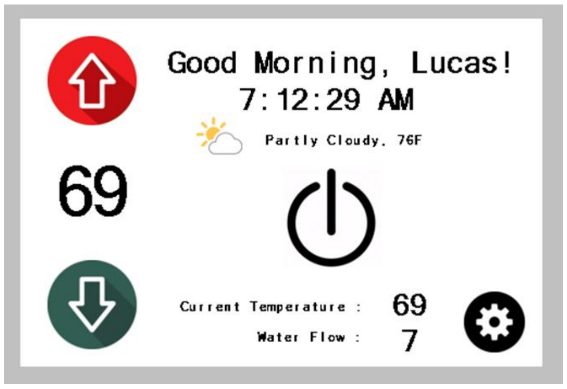


Figure 2: Main Touchscreen Interface

The Nextion design software provides an intuitive interface for designing and testing interfaces. Nextion provides several examples to help speed along development. The software includes a debugger and even has a display emulator so that development may be done even without the physical presence of the display.

Figure 3 shows our main touchscreen interface. This interface allows the user to turn the shower on and off, change the temperature, and view information such as time, current weather, and sensor data. This interface was created using the Nextion UI builder software.

One of the main advantages to this solution is that is allows for fast iteration display interfaces. Instead writing per-pixel changes in code to create a user interface, Nextion display solution uses a graphical interface design software. An interface can be rapidly designed on a desktop computer and then uploaded to the display solution for use. In many cases, no or little code must be written for the display to function on its own with limited functionality such as navigating through interface pages. A complex interface can be created for use with the display solution in a matter of minutes instead of hours or days. This is highly advantageous in our situation with limited time and many display interfaces to implement.

The main microcontroller in the system is only used for certain functionality such as sending sensor data to be shown on the page or reacting to a button press. The display solution can send data to the main microcontroller when certain interface elements are pressed. This can be used to run functions on the main microcontroller or update values. Data can also be sent from the main microcontroller to the display solution for display.

Since this solution requires only a serial connection for operation in a system it puts a much lighter load on the main microcontroller when in use. Only two data lines are required for its full functionality and integration. While interfacing directly with the display or through a display controller may use the entirety of the microcontrollers resources, this solution uses almost none. Also, because the display has its own (usually powerful) microcontroller driving it, it can be very fluid and provide much better(typically) user experience than trying to share resources with the main microcontroller.

D. Servo

One of the main reasons we chose to use a servo is that it is a self-contained unit that can (usually) be controlled by a relatively simple PWM signal. At first, we considered using a motor to control our valve, but this involves a control circuit to be built for the motor unit. These motor control circuits can be very difficult to design, let alone implement in a project. Building a motor control circuit suitable for our project would be very time consuming and could potentially be a very costly endeavor. With this in mind, we opted for a self contained servo commonly used by remote control (RC) vehicles. These servos come in a large variety and are low cost and commonly available.

There are two major types of servos in use today (with remote control hobbyists and in robotics): the continuous rotation servo and the fixed rotation servo. The continuous rotation servo acts much like a motor with a controller would. While it is a self-contained unit, it used the PWM signal it receives to move the servo in a certain direction at a specific rate. Much like a motor, using this type of servo in our project would require a timer based control of the shower valve (turn angle = rotation speed * time). This is undesirable for our project as it adds another element into our project which could reduce accuracy. It is also not possible to track the current location of the servo, only direction and rotation speed are known. The next major type of servo, known as fixed rotation, uses its PWM input signal to rotate to a fixed and know location. The PWM signal it receives directly maps to a predetermined angle on the servo. This is very desirable for use with our project as we will be able to directly send a signal to the servo that will correspond to a valve location. Not only does this remove the element of time, but this allows us to always know the location of the servo and valve based only on the servo PWM input.

For our project, we require a powerful servo to turn the shower valve. The servo will interface with a small metal spline on the shower valve that requires a high torque to turn. Fortunately, with the rise of cheap electronics manufacturing in China, high torque servos are available at a relatively low cost. Of the high torque servos available, the most desirable for longevity and strength are those that consist of metal gears. Metal geared servos can withstand much higher stall torques and can last for a much longer time than similar plastic geared servos.

After comparing many servos that could potentially be used for our project, we decided to use the DS3218 high torque metal gear servo as it met our needs the best.

E. Sensors

Temperature accuracy is arguably the most important constraint to consider when choosing a sensor type for our project. Providing inaccurate data to the control system may cause some serious issues (harm to user such as scalding). Thermistors, thermocouples, and digital temperature sensors all have the ability to provide accurate and precise temperature readings. For our project, the temperature sensor is located several feet away from the microcontroller. Using a thermistor or thermocouple in the situation may be troublesome as propagation of the analog signal through several feet of wire introduces parasitic resistance, capacitance, and other factors that may significantly reduce the accuracy of the temperature sensing. A solution to this is to run an external analog to digital converter from the microcontroller to a close proximity of the sensor. On the other hand, digital temperature sensors do not have this issue since they are digital and not analog. Wire may be simply run to the digital sensor and an accurate reading of temperature will still be received. While analog temperature sensors (and other analog sensors) can be very accurate there are many more factors that determine this when compared to digital sensors. The overall accuracy of an analog sensor relies on both the sensor itself and the analog to digital converter plus factors such as wire length. These additional factors make accuracy calculation much more complicated when compared to a digital sensor. Digital sensors usually have a guaranteed accuracy which is calibrated at time of manufacture. This calibration provides an accuracy statement that can be used directly for implementing requirements.

We decided to use the DS18B20 digital temperature sensor. This sensor was chosen for its cost effectiveness, ease of interfacing, and most importantly high accuracy.

Hall effect water flow sensors provide a variable frequency pulse output that corresponds with the rate of water flow they measure. When given water flow, a turbine inside of the sensor rotates. This turbine has a magnet attached to the shaft of the turbine. As the turbine rotates the attached magnet generates a changing magnetic field. For either every rotation or specified amount stored charge a pulse output of a fixed voltage is generated. As the rotation speed increases (by the increased rate of water flow) the frequency of the pulse output follows. This can be easily measured by a microcontroller by counting the number of pulses it receives from the sensor over a fixed period of time. These pulses/time can then be mapped or translated to water flow rate using a conversion specified by the manufacturer of the sensor.

In order to provide our water measurement capabilities, we used the YF-S201C hall effect water flow sensor as it is inexpensive, easy to interface, and accurate enough for our needs.

F. Enclosure

Before the enclosure can be made water resistant it must be designed to accommodate the projects components. The main driving force behind the final size of the enclosure was the size of the LCD and the PCB. Conveniently the LCD is an off the shelf unit that is designed to be mountable and includes mechanical drawings to do so. On the flip side the PCB was purpose built and was designed to be as small as possible with the given components, providing two mounting holes in the unused corners for its eventual mounting. A trap for the unsuspecting designer at this point is how deep to make the enclosure. Both PCBs are plagued with irregular shaped components the jut out from the surface just waiting to be a head ache down the road. The easiest way is to take note of the tallest component of each and add a tolerance margin if desired. Other minor considerations include mechanical fasteners to hold the device together and or to a surface. This is done to hold the face plate to the bulk of the enclosure with long screws and screw holes and for the PCB with shorter mounting screws to secure it. Mounting holes were omitted for mounting the assembly to a wall because the group opted to for a semi-permanent method like self adhesive tape or hook and loop strips to reduce the work required for an end user as well as reduce to impact of the device if it were removed.

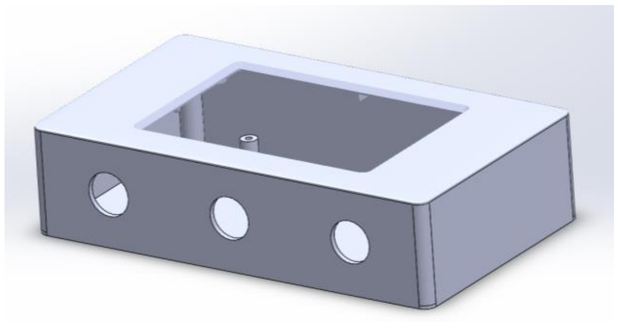


Figure 4: Enclosure 3D render

V. Embedded Software Detail

The primary functionality of our project will come from an ESP8266 microcontroller. For our project, this microcontroller is programmed with Embedded C through the Arduino IDE. An ESP8266 specific Arduino library is used for many of the basic functions on our microcontroller and some other more complex functionality such as wireless communication.

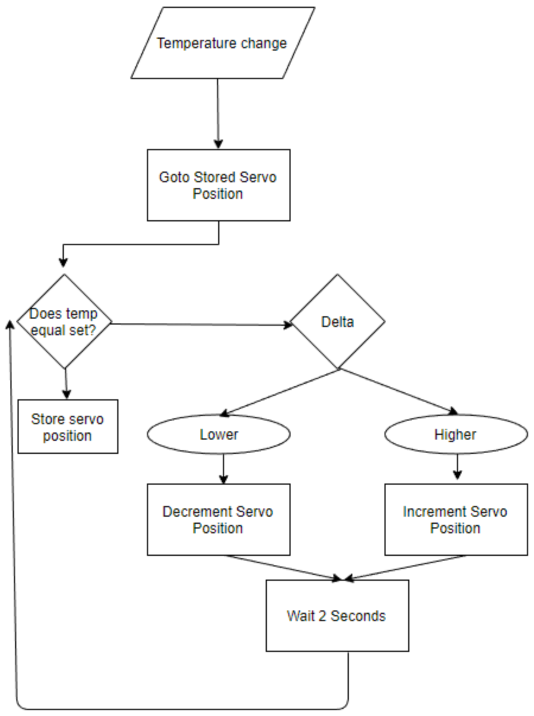


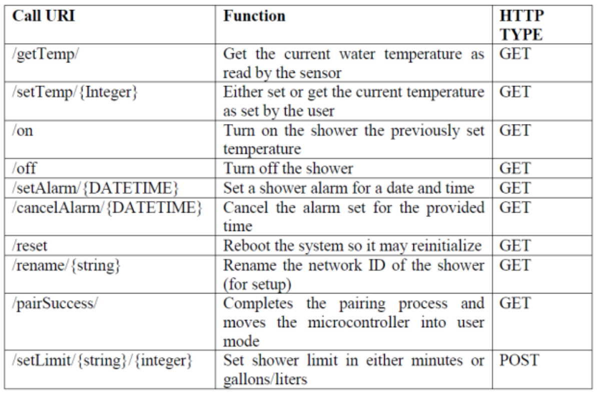
Figure 5: Simplified Temperature Control Algorithm

Our initial temperature control is provided by moving the valve by a small amount in the desired temperature delta per degree via the servo, checking the temperature after a set amount of time, and then either repeating the process or halting after the temperature has met the set amount by the user.

Upon the first successful setting of a user desired temperature, the system will store the servo position and associate it with that temperature for future use. When the user desires to obtain a previously set temperature, the servo first moves the valve to the stored location associated with that temperature. The temperature is then read and compared to the set temperature. If it is not at the correct temperature, the same process mentioned above will occur until the set temperature is reached.

Another primary feature set for our smart shower is its wireless communication capability. Our system will communicate with a mobile device and send data directly to a database. For this to function, a communications stack must be established on the microcontroller. A well-established wireless communication library exists for the ESP8266 which allows us to quickly iterate across wireless protocols.

We have also integrated our Smart Shower system with Amazon Alexa using our web API. Alexa communicates with the ESP web server to direct the smart shower to perform actions such as turning the shower on and off or changing the temperature.



Call URI	Function	HTTP TYPE
/getTemp/	Get the current water temperature as read by the sensor	GET
/setTemp/{Integer}	Either set or get the current temperature as set by the user	GET
/on	Turn on the shower the previously set temperature	GET
/off	Turn off the shower	GET
/setAlarm/{DATETIME}	Set a shower alarm for a date and time	GET
/cancelAlarm/{DATETIME}	Cancel the alarm set for the provided time	GET
/reset	Reboot the system so it may reinitialize	GET
/rename/{string}	Rename the network ID of the shower (for setup)	GET
/pairSuccess/	Completes the pairing process and moves the microcontroller into user mode	GET
/setLimit/{string}/{integer}	Set shower limit in either minutes or gallons/liters	POST

Table 1: Non-exhaustive table of ESP API

VI. Mobile Software Detail

The Mobile Software portion of the project consists of two parts: front end and backend. The main objective of the mobile application user interface is to be rather simple. Simplicity is going to be a key aspect in our design because the users that will be utilizing the application will be average non-technical families. By allowing the application to be simple it will cause less complications and after all the goal of the Smart Shower is to make peoples life's easier not more complicated.

A. User Interface

The application will be designed in an Android platform using Android Studio, with several fragments that comprised will make up the front end of the application. The application will first open when clicked on and an introductory splash screen will appear. Following the splash screen is the Synchronization page, this is where the application will connect with the microcontroller. After a

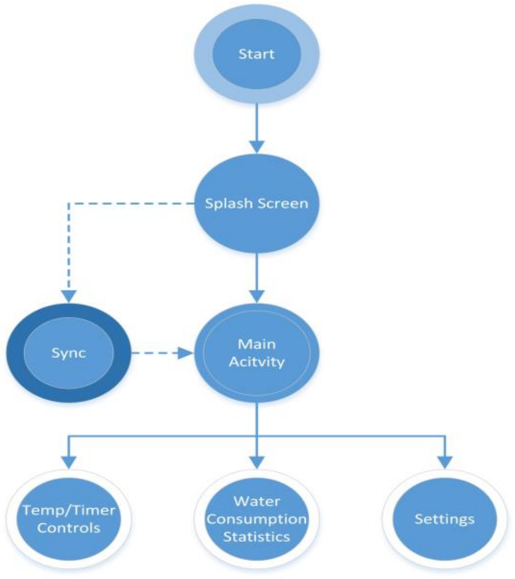


Figure 6: Mobile Software Flow Model

successful pair, the application will then open the Home Screen activity which is where the shower controls will be presented to the user. In addition to the Home Screen fragment, there will be a Settings fragment, Water Consumption Fragment, and a disconnect option in the menu bar. Seen above in Figure 6, is the software flow of the mobile application.

Having the mobile application allows the user to control their shower remotely. But the difference between the mobile application and the Smart Shower user interface inside of the physical bathroom is that the mobile application permits the user access to water consumption data. This is a very important feature that our project has because even though there are two competitors in the market with similar products, neither of them have the water consumption feature. In addition to the highly anticipated water consumption statistics feature it will also incorporate a Timer/Alarm feature. This feature allows the user to set a static shower time to force themselves to expedite their showers. On expiration of the timer, it will trigger the shower to shut off.

The mobile application will have direct connectivity to

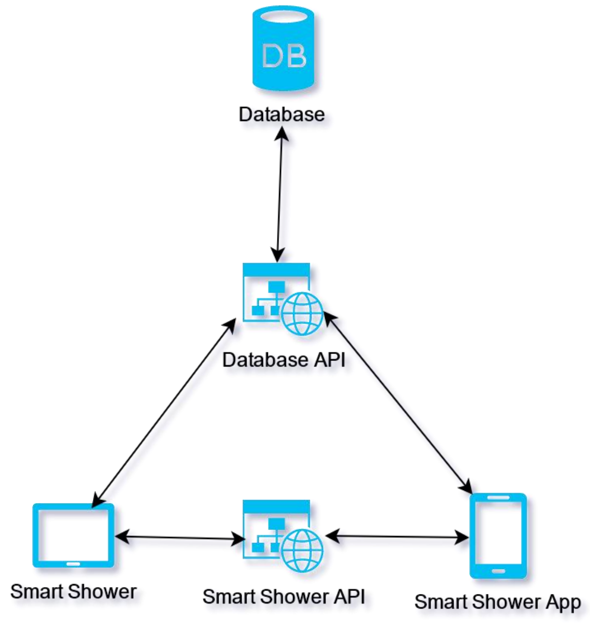


Figure 7: Communications Overview

the backend database which will hold the shower data as well as the water consumption data. In addition it will have direct communication with the Smart Shower (microcontroller) to control the temperature and on/off water valves.

B. Database

The database allows us to access data from any location at any time, and record information in a place that is independent of the smart showers and smart shower applications. This allows us to have effectively infinite storage – our data storage will not be stifled by a lack of storage on the phone or smart shower microcontroller. If more storage is needed, simply adding storage to our remote database will allow us to store more information.

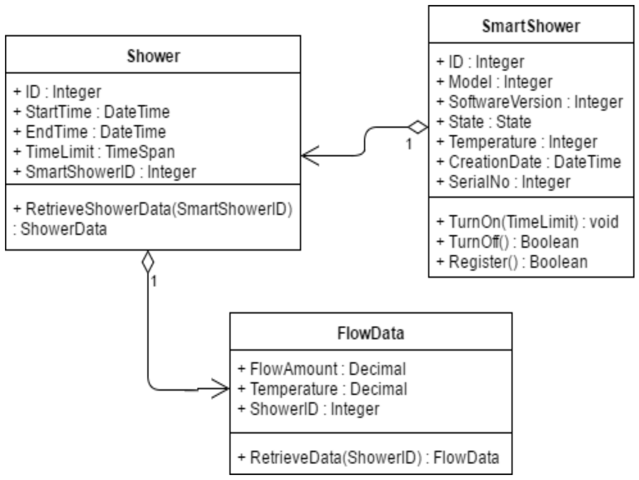


Figure 8: Database Design

The database system we have chosen to use is Microsoft SQL Server 2014. This is a relational database which uses 3 tables – a Smart Shower table, to keep track of each smart shower device, a Shower table, which keeps track of each shower, and a FlowData table, which stores incremental flow and temperature information for each shower, which is used in the Android app's statistical analysis.

C. Database API

The database will also be accessed through an API. This API allows us to hide the back-end data from the client, allowing us to change the way the client or the server does computation or the schema of the database without having to change the way that they communicate. Each end will have to hold up its part of the bargain – the client will provide the parameters of the data it needs, and form the request, and the database will fill the request. The API does not care how either of these things get done, and neither do either of the endpoints. This API will be used both in the microcontroller's software to send shower data to the database, and in the mobile application to retrieve shower data from the database.

The database API is a RESTful API written in ASP.NET MVC. HTTP calls will retrieve, update, and create data in the database. This RESTful design allows us to adhere to a client-server architecture which is easily accessed by many clients without any changes to the server. We access data by using the smart shower serial numbers, allowing both the smart shower app and the smart shower device itself to access the data based on your own smart shower device info.

VII. Conclusion

The smart shower project consists of a few different parts – The smart shower microcontroller and touchscreen display, the smart shower mobile phone app, and the smart shower power system and servo control. These parts, while different, were not independent in the slightest. The success of the smart shower product entails the successful integration of all of these elements, with a unified design objective.

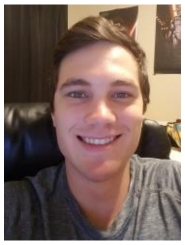
The smart shower microcontroller setup and programming is responsible for the majority of the operation of the smart shower device. Through the microcontroller, the shower is actually manipulated using the servo. The microcontroller we selected is inexpensive and has all of the connectivity and computation power we needed. We are able to connect this microcontroller to the wireless network, and even use it as an access point itself, which will greatly assist in the connectivity of the smart shower. This connectivity will assist in connecting to the phone, and to the database.

The Nextion touchscreen display, meant to be used within the shower, is an essential component for the smart shower product. It allows the user to interact with their shower in a relatively traditional way: from within the shower. Touchscreens are intuitive to use and understand, unlike buttons. The issues associated with using Nextion display instead of simple button interface with LED's, such as waterproofing, programming, and ensuring touch capabilities while damp, were greatly outweighed by the benefits of quality, ease of use, and ability to display much more detailed information.

The smart shower mobile application was another essential element of the smart shower product. The mobile application is what allows the user to use all of the advanced features of the smart shower. These features are what allow the product to go from an electronic shower replacement device, to a luxury user product. It allows users the new ability to make a shower alarm, to view their temperature and flow data for each shower, to set shower time limits, and to connect and manage multiple showers. All of these features utilize the internet, which allows the user to access these setting from absolutely anywhere with an internet connection.

With all of these elements working together, the smart shower offers a suite of features to smart shower users. Almost every shower can be converted to a smart shower and implement smart technology in yet another area of the home. Not only is it a luxury item used to increase shower usability, but it also encourages environmentally friendly shower use, allowing the user to regulate the shower's time limit and flow and be aware of the temperature and energy being used during their shower. All of this packaged into an inexpensive, simple, modular, and easy to use device

VII. Smart Shower Team



Lucas Gillespie is a 23-year old graduating Computer Engineering student who is taking a job with Harris Corporation in the Electronic Systems segment located in Melbourne, Florida. His primary focus at Harris will be vulnerability research and reverse engineering.

Andres Huertas is a 22- year Computer Engineer Major at the University of Central Florida. He has accepted an offer from Visa Inc. in Austin, TX as a Cloud Software Engineer. He will also commence his graduate education at Johns Hopkins in the Fall.

Nicholas Stoll a 22-year-old graduating Electrical Engineering student. Nicholas hopes to pursue a career as a research engineer.

Alex Power is a 21-year-old Computer Engineering major at UCF. He has accepted a role with FIS in Orlando as a Software Engineer. Alex intends to continue his education in the future and pursue new skills in firmware engineering, digital design, and embedded software development.

Acknowledgement

The authors wish to acknowledge and thank Dr. Lei Wei and the professors that make up the Electrical and Computer Engineering department at the University of Central Florida.