SMART Water Heater

Mauro Cordoba, Bryan Mitchell, and Vipol Sophonwatthanawichit

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

***Abstract* — After home heating and cooling, water heating is typically the second largest energy expense in the home. Despite buying the new water heater, there are several ways that one can reduce the water heating bills and energy usage. The most common solutions are water conservation and buying the new water heater system. SMART Water Heater offers the alternate solution to the problem by creating a water heater thermostat controller that is easily installed in an existing water heater. SMART Water Heater let the user have a complete control of water heater throughout the day by using Android Devices and Thermostat-like controller.**

***Index Terms* — Android, fluid flow control, heat pump, learning, smart homes, temperature sensor.**

I. Introduction

Energy Conservation is no doubt one of the great important matters to everyone since we heavily rely on energy usage for many daily activities we do. With the increasing number of technologies, more energy usage is required. Thus, many organizations have their researchers working on energy conservation to help us provide the solutions to conserve energy. Because energy supplies are limited, we are suggested to find ways to use energy wisely in order to maintain a good quality of life.

With an ever increasing motivation to conserve energy, much effort has been put forth lately to create smart HVAC systems. According to Duke Energy, water heaters are the second-highest source of energy usage in most homes. This is a largely overlooked area in which a smart solution could increase efficiency, reduce energy costs and carbon footprint, as well as giving the user a more refined control and ultimately comfort. There are several suggested solutions to help us reduce the energy usage for water heater such as buying the new and better water heating system, water conservation, insulate the existing water heater, and so on. SMART Water Heater offers the alternative solution to this problem.

SMART Water Heater focuses on making the usage of water heater more efficient and increasing the awareness of energy usage to the household users. It introduces a digitally controlled thermostat that offers a way to control and monitor water heater system. The users have complete control of the water heater. The system is programmed with the algorithm such that the system recognizes the pattern of energy usage and users' behaviors. The user will be able to control the water heater through several devices, android devices and thermostat controller, that communicate with the water heater tank.

II. System Overview

As shown in figure 1 below, SMART Water Heater system is divided into three main subsystems: the Main Controller, the Native Application and a Mobile Application. These subsystems are necessary for the system to work in the intended way, providing functionality that is not present in other subsystems.

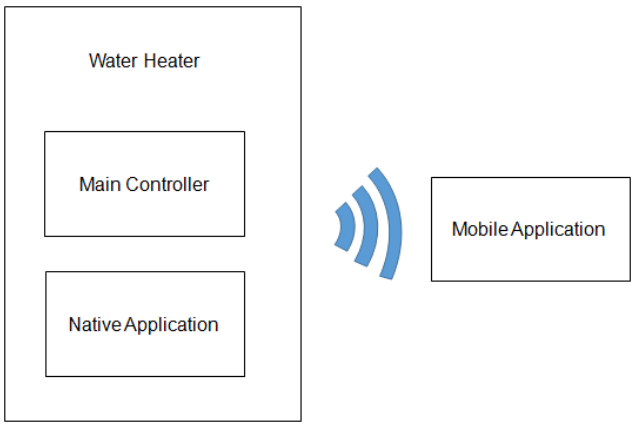


Fig. 1. System overview: the three main components that comprise our water heating system.

Most current electric water heaters normally have two heating elements with two separate thermostats to regulate the temperature of the water. These heating elements operate in either simultaneous or non-simultaneous mode. In simultaneous mode, both heating elements can be on at the same time which is helpful when a high throughput of water is needed. Non-simultaneous mode means that only one heating element is on at a time, usually alternating back and forth to try to keep an even temperature through the entire volume of the tank. On basic water heater models the temperature is set mechanically through a

screw or trim pot on the thermostat itself. Higher end models normally offer a digital interface, however this interface is usually a very basic 7 segment display and a few buttons, not much different than an oven in a kitchen. Lastly, very high end models are starting to offer connectivity options to integrate water heaters into home automation systems, but we have found that none of them offer Wi-Fi natively. Most systems offer a wired Ethernet connection and Wi-Fi adapter modules need to be bought separately. Because of this, we decided to give these appliances a new, more modern face by fitting them with a digital thermostat, an intuitive touch screen interface much like a phone application, and native Wi-Fi connectivity.

A. Main Controller

The main controller is hardware portion of the system. It is divided into two separate boards, the Crystalfontz CFA920-TS System-On-Module with a built in touchscreen, and a daughter board containing our power supply as well as a hardware interface for the heating elements and breakout connections for sensors.

The Crystalfontz board is an embedded computer module capable of supporting the user interface software, thermostat and control logic as well as having networking capabilities simultaneously. It is shipped with a running general purpose Linux-based operating system, but we need to customize it to fit our needs.

B. Native Application

The native application is responsible for the function of all major aspects of operation. Our firmware is made up of different layers of software, starting with the operating system, drivers, software services and finally the graphical user application. We view this as a single functional unit because it all lives in the water heater controller and it allows for full functionality independent of any other software. The goal of the native application is provide a greater deal of control over the appliance than existing controllers in the market.

C. Mobile Application

The mobile application's main focus is to allow for full control of the water heater's functionality remotely. Although it is not necessary for the normal operation of the heating system, since everything can be set or checked natively on the water heater, we felt that the convenience of being able to access the appliance from anywhere would truly make it a modern device. Users are increasingly getting accustomed not only to control many aspects of their lives through smart devices, but also expect more and more to be provided with metrics to help incentive better habits.

III. Hardware Detail

This section provides a summarized technical report of overall and each individual component used in the SMART Water Heater hardware design. The following figure shows the block diagram for the hardware design.

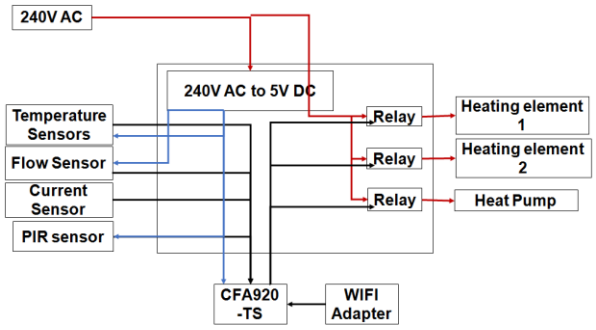


Fig. 2. Hardware Block Diagram showing connections of expansion board and controller board

The overview of our design is shown above. Due to constraints, instead of using 240 VAC, we will be using 120VAC. The AC voltage will come in to the expansion board and be rectified and stepped down into a usable 5V DC voltage.

The 5V DC is supplied to the main control board, as well as the individual sensors. The CFA board then provides data back to the sensors through the expansion board. Also, since the main control board's output pins cannot supply the needed current to trigger the solid state relays, a sub-circuit utilizing transistors is included on the expansion board. This sub-circuit allows the main control board to control the relays without causing irreparable damage to itself.

A. Main Board

The main control board for our design is the Crystalfontz CFA920-TS. It is a System-On-Module, with full Linux operating system, mounted on to a small motherboard. The motherboard itself is only a few millimeters larger than the included 480x800 color touch screen display. The CFA920-TS board houses multiple input/output pins, a 10/100 Ethernet port, 256 megabytes of DDR2 RAM, standard USB port, and a micro-SD card slot that can support micro-SD cards up to 64 gigabytes in capacity. The USB and Ethernet port allow ease of communication and the large support of the SD card slot gives the CFA board a sizable storage area. The Linux operating system allows for a familiar development environment, and the processing power of the Freescale i.MX287 processor will allow the main control board to

handle the analysis and cataloging of the data it gleans from the attached sensor

B. Expansion Board

The expansion board will contain the main board's power supply, as well as input and output headers to interface with the various sensors that will be used. The sensors will receive power from the expansion board, and then pass any data they have obtained through the expansion board and in to the main control board.

This expansion board will also house the supplementary circuits to control the solid state relays that will be used to activate and deactivate the heating elements and heat-pump subsystems.

Included on the expansion board is multiple sets of headers. Since our main control board cannot provide adequate power via its output pins to the sensors, the supply voltages must come from the expansion board. In order to keep the amount of clutter and wires to a minimum, headers were added to the expansion board so the sensors can be attached in a single point of contact. The expansion board will be connected to the main control board via a small ribbon cable, hopefully keeping the innards of the device tide.

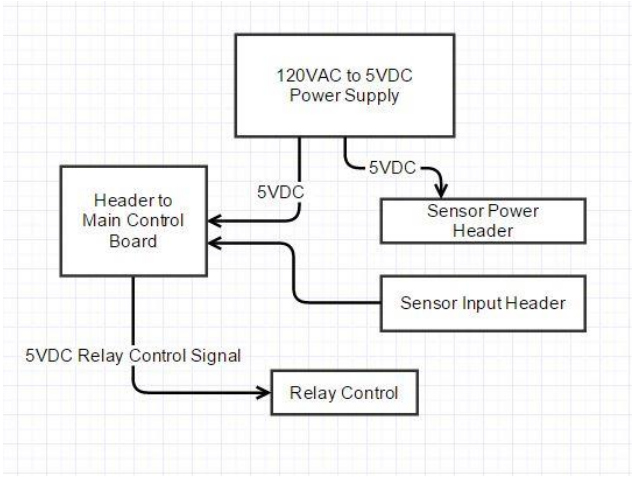


Fig. 3. Expansion Board Block Diagram

The power conversion portion of the expansion board is utilizing a Texas Instruments UCC28710 flyback power supply controller. The UCC28710 provides constant-current and constant-voltage output regulation and allows the power supply to be easily tailored to different voltages and currents, should the need arise during any sensor expansion.

C. Temperature Sensor

A common upper temperature limit for most household water heaters is between 48 °C and 54 °C. A lower temperature allows for a slower buildup of minerals and corrosion, but will also pose a possible problem of bacterial growth. Higher temperatures eliminate the problem of bacterial growth, but increase the annual cost of energy usage. Temperatures over 60°C increase the chances of scalding injuries. The temperature system will continually update the control module with the current water temperature. In conjunction with a PID controller, the temperature of the water can be maintained at a preset temperature more efficiently than the standard analog controls on most water heaters. The temperature sensor that will be utilized is the DS18B20 digital temperature sensor. It has a single wire data interface and a unique embedded identifier that would allow for multiple units to be on a single wire. As it is a digital sensor, the resolution is user-configurable to increments of 0.5 °C to 0.0625°C.

TABLE I
DS18B20 TEMPERATURE SENSOR SPECIFICATIONS

Supply Voltage	3.0 VDC to 5.5 VDC
Temperature Range	-55°C to +125°C
Accuracy	±0.5°C from -10°C to +85°C

The DS18B20 that will be used in this design is waterproofed with a PVC jacket good to over 100 °C. This is well over our safe operating range of 50° C (around 120 °F).

With a waterproof temperature sensor, accurate temperature readings can be made from multiple points along the height of the tank. With the current temperature from the upper and lower regions of the tank available at any time, the main control board will be able to activate elements individually if one portion of the tank will require more heating than the other.

D. Flow Sensor

The Flow Sensor subsystem will allow the control module to obtain accurate information on the amount of water used. This data will allow the learning algorithm to determine a general schedule of water usage. After a learning period, this algorithm will be able to predict when the user will need hot water and activate the heat controls. The two main types of flow sensors are in-line and non-contact. In-line flow sensors are designed to be added to existing plumbing, allowing a sensor to be placed directly in the flow of the measured liquid. These sensors are inexpensive and reliable, but

require modification to plumbing pipes if they are installed in an already existing setting.

Non-contact flow sensors use various frequencies of electromagnetic radiation to measure flow rate. Major downsides to non-contact flow meters is that they are expensive and do not work for clear or non-magnetic liquids. The non-contact flow meters require magnetic properties in the liquid, or require the liquid to have particulates that can be measured to determine flow speed of the liquid carrying them. Since this component is for a household water heater and the water must be potable, the non-contact flow meters will not work.

TABLE II
YF-S201FLOW SENSOR SPECIFICATIONS

Supply Voltage	+5V to +18V
Current Draw	15mA at +5V
Working Temp Range	-25°C to +80°C
Working Pressure Max	2.0 MPa
Mechanical	½" NPS pipe connection

The flow sensor that will be utilized in this hardware revision will be a plastic-housed, in-line pinwheel style flow meter. As liquid flows through the meter, the pinwheel spins. Attached to the pinwheel is a very small magnet. The rotation of the pinwheel and magnet combo is measured by a hall effect magnetic sensor. The use of the magnet and hall effect method allows the separation of liquids from electronics. Shown above is the specifications of the flow sensor that for the project. Many Hall Effect flow sensors have the same basic design, and currently the flow sensor of choice is an YF-S201.

E. Current Sensor

To monitor current usage, an ECS1030 series split core transformer will be used. The split-core form factor allows the placement of the sensor around a wire instead of having to be installed in-line. The ECS1030 is rated for 1 to 60 amperes with a 2% accuracy. The current usage information will be processed by the main control board and used to determine the amount of energy used. The current ratio of 30A to 15mA makes taking measurements relatively easily. Multiplying the reported current by a factor of two thousand will provide the device with the current flow through the element supply wire.

TABLE III
CURRENT SENSOR SPECIFICATIONS

Primary Current Rating	30A nom., 60A max
Current Ratio	30A / 15mA
Accuracy	2%
Operating Temp Range	- 45°C to 65°C

F. Motion Sensor

For ease-of-use, a PIR sensor will be used to allow the device to sense the approach of the end-user. This design utilizes a passive infra-red sensor manufactured by Parallax.

Motion can be detected by checking for a logic high signal on the single input/output pin. This sensor is powered by a DC voltage from +3.3V to 5V and draws less than 100 micro-amps when in use. We will use this sensor to dim the screen when no presence is sensed around the water heater. It would make very little sense to keep the screen on at all times if the intention of the device is to in fact conserve energy.

TABLE IV
MOTION SENSOR SPECIFICATIONS

Supply Voltage	+3.3VDC to+5VDC
Range	0 to 20 ft
Vision Angle	120°

G. Relay

The relay circuit will allow the device to control the high current without harming the delicate electronics of the control board. In an actual water heater, the voltage supplied is 240VAC. Due to location constraints and testing safety our preliminary design will utilize 120VAC. For this design, an ANV solid state relay of the SSR series will be employed. The SSR series will allow for a 5VDC control signal and can handle a 120VAC / 20A load. This type of solid state relay uses an LED to signal a light sensitive MOSFET that switches the load.

Solid state relays offer multiple benefits over physical electromechanical relays. The solid state relays have a much smaller profile and increased lifetime. SSRs also eliminate the voltage spikes when electromechanical relays are switched on and off. Solid state relays also employ no moving parts, therefore eliminating the possibility of sparking when switched on and off. Due to the closeness of the relay to static sensitive components, this is a very important feature.

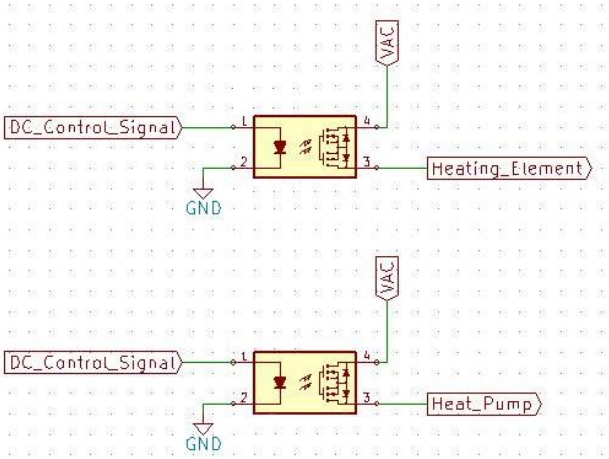


Fig. 4. Relay circuit used to toggle heating elements

The particular relay in use in this prototype is the SSR-25DA. As the model number suggests, it is capable of sinking up to 25 amperes when using the included heat sink. The output voltage range is 24 to 380VAC and can be activated by a DC input as low as 3.3 VDC, which is ideal for use with the output pins of our Crystalfontz control board.

TABLE V
SOLID STATE RELAY SPECIFICATIONS

DC Control Voltage	+4 VDC to +32 VDC
Trigger Current	7.5mA @ 12VDC
AC Operating Voltage	+24 VAC to +380 VAC
Leakage Current	<3.0 mA
Load Current	25A (with heatsink attached

H. Wi-Fi Adapter

For the wireless connectivity, the design uses a simple, miniature USB Wi-Fi module. Inexpensive and easy to use (plug and play), it is very similar to the devices used to connect personal computers to wireless networks.

The physical size is small enough that it does not add any extra dimensions to the main control board when it is plugged in. It supports all of the various wireless and communications standards current in use.

The supported frequency band is the 2.4GHZ ISM Band, and it can be used as a communicator to a wireless hub or part of an ad-hoc network.

It is important to note that while the operating system supports a large number of Wi-Fi adapters, ours will be enclosed and it will not be removable by the end user to avoid compatibility issues.

III. Software Detail

This section provides a summarized technical report of overall and each individual component used in the SMART Water Heater software design. The following figure shows the block diagram for the software design.

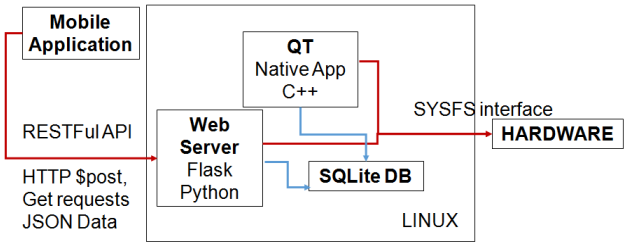


Fig. 5. Software Block Diagram

The software portion are distributed into two different platforms. The controller software or the native application will be responsible for bringing up the system and interfacing with the physical components as well as making decisions from the gathered data. On the other hand, the mobile application will be a portable user interface that will act as a remote control for the system.

Following sections explain subcomponents and their details in software design.

A. Operating System

The operating system is Linux-based and custom made for our board to better leverage the hardware. The operating system was made using Yocto, an open source project that provides with tools and methods to create custom Linux-based systems for embedded devices. These operating systems are usually meant for a single, specific task, as in our case.

With Yocto we are able to select the architecture and define a hardware profile that defines all the peripherals for the board that we are using, the CFA920-TS. Then we can select which packages we need to include into our image. This allows us to only include the bare minimum needed to perform our tasks. In our case, we are only including the necessary hardware drivers, graphic libraries and web frameworks to run our controller and provide connectivity. Once the image is created we can copy it onto an SD card and boot directly from it.

We decided to go with a full-fledged operating system because we needed to manage several asynchronous tasks and also because it would allow us to use drivers and libraries already developed for Linux. Using Yocto is also advantageous because it makes the design portable. We would be easily be able to transfer our prototype to a

completely different architecture by simply modifying the hardware profile and creating a new image.

Since we're trying to keep the size of our operating image small and the user will not have access to the lower level of the software anyway, our custom image needs to be developed on a separate system, tested and then transferred over. Our final image will not contain any sorts of package manager. Everything needed to run our programs needs to be shipped out in the image from the get go. To aid with this process, we are able to compile Yocto images and run them in emulators on our development machines before compiling them for the final hardware. This way we can speed up the development and testing process as it eliminates having to copy entire new images to the target device.

B. Backend

The backend of the system is built around a Flask application. Flask is a python web framework that allows us to create and run a webserver directly on our hardware. By creating models representative of the data that we need to manage, Flask creates a corresponding SQLite database. This database is then accessible throughout the system and as such, we can work with its contents either through the front end application or remotely through the webserver.

Since Flask is based on python, we can include standalone python scripts and access its methods from our application. Using this methodology, we created a python script that gets periodically called to check the state of the water heater (water temperature, current, element state, etc) and use this data to populate the database.

To provide a web interface for the android applications, we have implemented a Representational State Transfer Application Programmers Interface (REST API). REST APIs work through HTTP request methods to communicate back and forth with servers. In our case, we have defined certain URLs, such as /get_temperature, that expect JavaScript Object Notation (JSON) formatted data or send JSON formatted data. If the data received is validated, it is appropriately processed and validated, and if needed, a response is sent back to the originating device. By doing this, the android device can request the server for information like the current temperature, it can request that the server turn on or off the elements, etc.

We will be collecting water temperature related data periodically in intervals of five minutes. Water has a very high specific heat, meaning that it takes a lot of energy to raise its temperature, so a sampling rate faster than 5 minutes would probably not benefit our system, but also add a lot of overhead. In order to predict what times of the day the user will need hot water we need to take a closer look at the flow meter. While hot water exits the tank, cold water rushes in and we can measure this. Assuming the user has a routine, the usage times will be very similarly for most days, allowing us to set a baseline of usage.

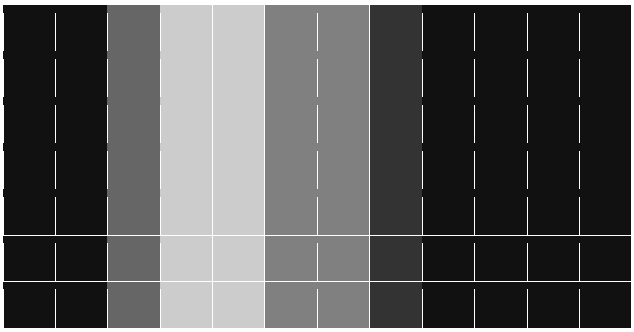


Fig. 6. 1 hour interval over 7 days. Lighter colors indicate that water is on demand for that period of time. Darker colors mean that no water entered or exited the tank.

We decided to implement the learning algorithm by representing each day as greyscale image or array as show in figure 6, with each pixel taking a value from zero to 8 corresponding to a 5 minute interval. Starting out with each pixel at the maximum value, the water heater assumes that hot water is needed at all times. Each subsequent day if for any given time period hot water is not needed (as measured by no water flowing into the tank) we decrease the value of the corresponding pixel. If the value of each five-minute pixel falls below a value of 4 then we understand this to mean that no hot water will be needed from now on at this specific time. If the user were to change his habits and start using hot water a different times, it would only take 4 days for the water heater to learn the new times.

By looking at the state of the heating element and temperature at each time interval we can calculate how long it takes to heat up water for any given starting temperature. Using this information together with the expected demand times we can predict how long before water is needed we need to start the elements to make it seamless for the end user.

C. Frontend

Frontend portion deals with two main applications: Native application running on the Crystalfontz board and the Mobile Application on a remote android device. As mentioned, the native application acts as a thermostat for the water heater while the mobile application is able to read values and set values to change settings for the running water heater.

For the native application for the touch screen interface, the application is developed in Qt 5.4. The main reason

behind choosing Qt is that it integrates well with the chosen backend software. Qt is application framework for UI development. It is crossed platform and can output to our frame buffer without a window manager. It can also interface with SQLite database created in flask as well.

Although we are running Linux in our board, since we only need to run one application and nothing else, we prefer not to use a window manager as it would unnecessarily use up valuable resources. For this reason, Qt makes a great choice since it can run directly on the frame buffer. After the board is done booting up, a script is called to start the Qt application immediately.

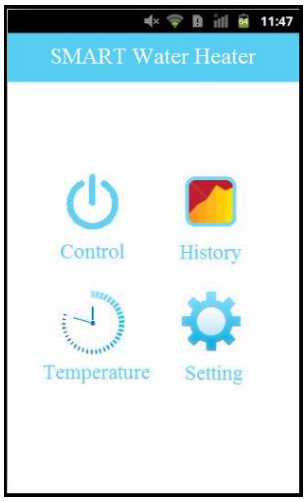


Fig. 7. Main Screen of the android application

When considering the platform of mobile application, the goal is to have the application available to as many users, that don't have the modern version of water heater, as possible. With that in mind, android API with android 4.0.3 or Ice Cream Sandwich is chosen. Lower API levels target more devices, but it have fewer features available. By targeting API 15 and later, the application will run on approximately 90.4% of the devices that are active on Google Play Store.

The application contains 4 main interfaces: Control, History, Temperature, and Setting. In the Control interface, the user will be able to remotely control the water heater tank. The user is able turn on and off the water heater as well as set the temperature to the desired value. History interface deals with user's past history usage. It contains both energy usage as well as the change in temperature of the water heater within time period. Next is Temperature interface. This interface contains the statistic of current state of water heater such as current temperature, current energy usage and time stamp. Lastly, Setting interface deals with the setting of the application and the device such as Wi-Fi setting for example.

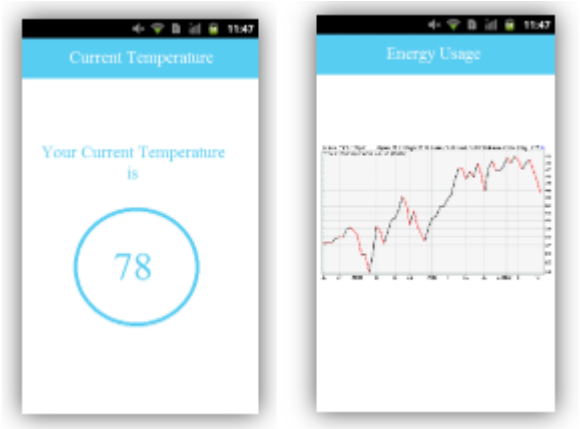


Fig. 9. Temperature Screen and History Screen

As explained before, the Android app communicates with the webserver through HTTP sockets. Sending appropriately formatted HTTP requests to addresses we've defined triggers actions in the server that can directly control the heater's functionality.

D. Functionalities and User Interface

The user interface should be easy to use and intuitive. On both platforms the user will have an almost identical set of actions. Users will be able to set the desired water temperature, as well as view the current water temperature and the state of the elements whether they are currently active or not. Users will have the ability to set different modes such as normal, power-saving, high-demand or turn off the heaters completely if desired.

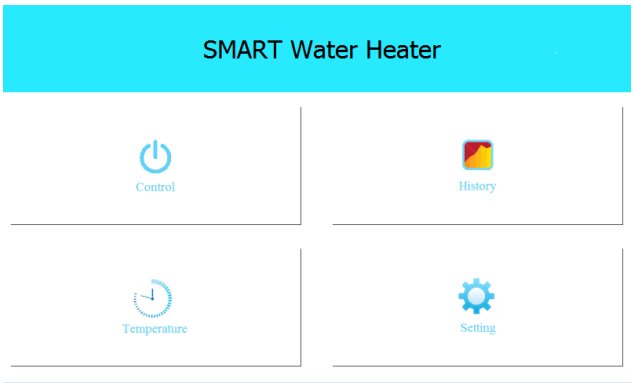


Fig. 9. Touch Screen Interface

On normal mode the water heater will function as a normal water heater in non-simultaneous mode. It will attempt to hold a steady water temperature at all times. Power-savings mode will take into consideration usage statistics and keep hot water ready for the times it predicts usage, and will otherwise hold the water at a much lower temperature in order to save power. High-demand mode will allow the thermostat to go into simultaneous mode if too much hot water is being supplied too quickly in an attempt to keep up with demand. Lastly, the user will have the option to turn off the heating elements completely and thus allowing the water temperature to fall down to room temperature. This mode will be especially useful when going away for extended periods of time.

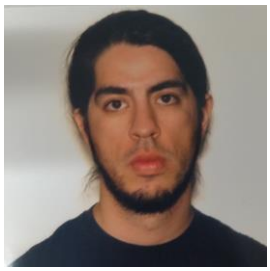
IV. Conclusion

The design goal was to create a known appliance that works in exactly the same way users are used to while at the same time modernizing it and giving the user more control in terms of both function and conform. By following the aforementioned design guidelines we believe we have succeeded. The functionality should be seamless from a user perspective and the new ways of interfacing should hopefully feel intuitive enough that he user will not need a manual to fully grasp the everyday functionality.

Acknowledgments

The authors wish to acknowledge the assistance and support of Dr. Samuel Richie as well as Mr. Arup Guha, Dr. Sarah Angell, and Dr. Ali Orooji for being part of our review committee as well as Dr. Yier Jin for giving us the initial inspiration to choose a topic.

Biography



Mauro Cordoba is 26 years old and is currently a student of Electrical Engineering as well as working in Circuitronics corp. He plans to continue his career in Engineering while also trying to find time to pursue musical interests.

Bryan Mitchell is 39 years old Electrical Engineering student. Bryan plans on graduating in Spring 2015. With an interest in troubleshooting and stress testing, Bryan hopes to attain a career in testing and QA.

Vipol Sophonwatthanawichit is 22 years old Computer Engineering student. He plans to pursue his career in software engineering at Harris corp. after graduating.

References

[1] "Electrical Thermostats," March 2015. Web. < http://www.rheem.com/docs/FetchDocument.aspx?ID=c468a4e6-d7d6-448d-9ec5-5615734c2fa3>

[2] "Yocto Project," March 2015. Web < https://www.yoctoproject.org>

[3] "JSON," March 2015. Web. < http://json.org>

[4] "CFA920-TS," March 2015. Web < https://www.crystalfontz.com/product/CFA920TS#datasheets>

[5] "RESTful Web Services," March 2015. Web. < http://books.google.com/books?id=XUaErakHsoAC>

[6] "i.MX287," March 2015. Web. < http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=i.MX287>

[7] "YF-S201," March 2015. Web. < http://www.dx.com/p/yf-s201-hall-effect-water-flow-counter-sensor-black-217625#.VSfyofn4-So>

[8] "DS18B20," March 2015. Web < http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf >

[9] "UCC28710," March 2015. Web. < http://www.ti.com/lit/ds/symlink/ucc28710.pdf>