– Independent Work Report Fall 2018 –

Project SafeWater: a Mobile App and Portable Device to Measure Water Potability

Seho Young
Advisor: Dr. Alan Kaplan

Jan. 8, 2019

Abstract

Project SafeWater involves a mobile app that pairs with a Bluetooth Arduino-based device that has several sensors that can analyze water samples and determine whether or not it's safe to drink. The goal of this project is to make the task of analyzing water content and determining its safety more efficient and accessible, especially for users in disaster situations, such as Flint, Michigan or Puerto Rico after Hurricane Maria.

1. Motivation and Goal

Water is life. It is a universally known fact that humans need water to survive, yet the task of finding clean water to drink is, even today, a tough one for many, especially for those without regular access to resources and survivors of natural or artificial disasters.

During the notorious Flint Water Crisis in 2014, the town's decision to switch their water source to the Flint River led to lead contamination in the citizens' drinking water. Between 6,000 and 12,000 children in Flint are estimated to have been exposed to high lead levels in their drinking water, and the percentage of children with elevated blood-lead levels rose from 2.5% in 2013 to as much as 5% in 2015 [9]. The effects of this disaster were not only local but also felt nationwide - the water contamination in Flint was a possible cause of an outbreak of Legionnaires' disease that killed 10 people and affected 77 others in the country [9].

In the aftermaths of Hurricane Maria, which formed in mid September 2017, several citizens of Puerto Rico were left without homes, food, and safe water sources. According to a survey conducted

by the Washington Post and the Kaiser Family Foundation, 2 in 10 said they drank water from a natural source after the hurricane hit, and 53% said they are worried about the quality of water in their homes [7]. The groups most affected by the hurricane were the rural neighborhoods of Puerto Rico (of which there are more than 230), which subsist on their own wells or natural springs [7].

Finding clean water can become difficult in non-disaster situations as well, such as extended hiking/backpacking trips, where one might not have access at all to filtered water. Iodine tablets are a popular method of decontaminating water from a natural source; however, when they run out or become lost, one's health and/or survival becomes dependent on finding clean water.

The goal of this project is to make the task of analyzing water content and determining its safety more efficient and accessible. With a way to quickly diagnose the potability of water samples wherever and whenever, users wouldn't need to worry about the water they are drinking, not only in disasters but in any situation where they need to find their own drinking water. Currently available portable water quality meters are only sold through very niche vendors, and can range from around $600 to $4000.[1] Project SafeWater aims to prototype a device that, if mass-produced, could be widely available at a far lower price point to consumers everywhere.

2. Problem Background and Related Work

There are three projects that have attempted to address this issue. The first is a research project at the University of Illinois, titled "Cellphone-Based Portable Bacteria Pre-Concentrating Microfluidic Sensor and Impedance Sensing System" [2]. The authors of this paper designed a sensor that detects bacteria in water samples using electrical impedance spectroscopy, which is connected to an Arduino which relays this information wirelessly to a smartphone. This project is very similar to Project SafeWater's approach; however, the limitations are that it can only sense bacteria, and will be unable to detect other chemical impurities in the water. In addition, its method of measuring water samples (using a thin syringe to inject water into the sensor) can be impractical for the average user. Figure 1 shows pictures of the device and its basic architecture.

[1] https://www.thomassci.com/

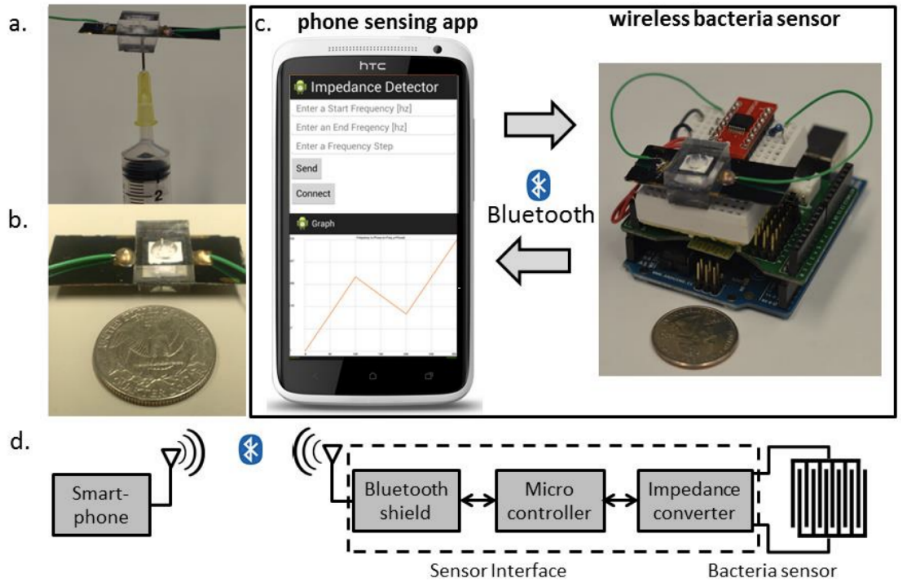

Figure 1: Cellphone-Based Portable Bacteria Pre-Concentrating Microfluidic Sensor and Impedance Sensing System [2]

The second is a product called TestDrop Pro by Lishtot, an Israeli company that builds water quality testing devices [3]. The TestDrop claims to be able to determine the potability of water by just pointing the device at it without any direct contact, using the water's electric fields around its cup. An LED on the device lights blue if the sample is safe to drink, and red if it is not. The TestDrop is able to connect to a smartphone app, which logs each test and its location. However, there have been no studies published that demonstrate the accuracy of the device, and it doesn't provide the user with detailed statistics on the quality and content of the water. In addition, without direct contact to the water sample it fails to measure several crucial parameters of water quality, such as pH, ORP, and TDS, the details of which will be discussed in a later section of this paper.

The last is Riffle, an open source water monitoring project on Public Lab [4]. Part of Public Lab's Open Water Project, the Riffle involves an Arduino-compatible datalogger with several designs for sensing different water quality parameters. Despite providing a cheap and accessible platform for building configurable water sensing devices, the Riffle only logs data to an SD card without any wireless capability, and lacks the versatility of an actual Arduino to read data from several sensors at once.

3. Approach

Project SafeWater addresses all the limitations of the two related projects, by prototyping a cheap, comprehensive device that, when paired with a smartphone app, analyzes water samples based on several different parameters, and makes a conclusive diagnosis on whether or not it is drinkable. The device relays the readings from its sensors through Bluetooth to a smartphone app, which will display the readings and make the diagnosis. Smartphones, especially iPhones, have become extremely widespread, making it the perfect platform for this project. Almost every smartphone today has Bluetooth 4.0 technology, which allows for hassle-free, wireless communication between the device and the app.

In addition, the app will implement a crowd-sourcing feature, where users can upload their results to a map, which will display measurements and diagnostics based on its location. Thus, in an emergency situation, users can easily check the map to see which water sources are safe. A similar example of this feature is the Surfrider Foundation's Blue Water Task Force Project, a mapping of bacteria data collected from water samples around San Juan, Puerto Rico.[2]

The device measures five of the most commonly used parameters for measuring water quality: **Temperature**, **Turbidity**, **pH**, **Oxidation-Reduction Potential (ORP)**, and **Total Dissolved Solids (TDS)**. These parameters were chosen to provide a comprehensive measurement of water quality while keeping it as efficient and low-cost as possible.

- The **temperature** of water is important for measuring water quality because it affects the maximum dissolved oxygen content, the rate of chemical and biological reactions, and the organic life in the water [6]. According to the Guidelines for Canadian Drinking Water Quality, the aesthetic objective for water temperature is 15°C, due to a possibility in increased odor from trace volatiles and micro fungi at temperatures above 16°C [6]. However, since an elevated water temperature is not a direct indicator of unsafe water, SafeWater accepts water with temperatures from 5°C to 35°C inclusive, 15° above and below room temperature.
- **Turbidity** is the cloudiness of a fluid due to a large amount of suspended particles in it, and is

[2] https://www.surfrider.org/blue-water-task-force/chapter/4

therefore the only one of SafeWater's five parameters that can be seen with the naked eye [8]. It is often measured with a nephelometer, which shines a focused light beam onto the fluid and measures how much of the light is scattered by the particles [8]. When using a nephelometer (such does SafeWater), turbidity is measured in Nephelometric Turbidity Units (NTU) [8]. According to the World Health Organization, the turbidity of drinking water should not be more than 5 NTU and should ideally be below 1 NTU. Thus, SafeWater rejects water samples with a turbidity greater than 5 NTU.

- The **pH** of a fluid measures how acidic or alkaline it is, based on properties of the hydrogen atom [5]. Lower pH values indicate acidity (e.g. sulfuric acid, orange juice, vinegar), and higher pH values indicate alkalinity (e.g. baking soda, ammonia, bleach) [5]. Pure water has a pH of 7.0, but water sources tend to be slightly acidic due to natural contaminants in the water [5]. In the United States, the EPA's Secondary Maximum Contaminant Level for pH is between 6.5 and 8.5; thus, SafeWater rejects water samples outside of this range [5].
- **ORP** measures the potential of a water sample to break down contaminants, depending on the amount of dissolved oxygen in the water, and is measured in millivolts (mV) [1]. Generally, higher absolute value ORP values indicate a decreased likelihood of harmful bacteria and dead or decaying material in the water [1]. Because the ORP value of water can range from -2000 to +2000, depending on pH, SafeWater does not depend on ORP to make its diagnosis, but is a very helpful measure of water quality for the user to see. Figure 2 displays a chart of approximate ORP values for some common liquids.
- **TDS** is the total concentration of substances dissolved in water, such as inorganic salts and some organic matter, and is measured in parts per million (ppm) [5]. While relatively small values of TDS are not harmful at all (in fact sometimes desirable due to its "mineral" taste), large values of TDS indicate an increased possibility of harmful contaminants (e.g. iron, sulfate, and arsenic) to be present in the water [5]. In the United States, the Secondary Maximum Contaminant Level for TDS is 500 ppm [5]. Thus, SafeWater rejects water samples with a TDS greater than 500 ppm. Figure 3 displays a chart of approximate TDS value ranges for different types of water.

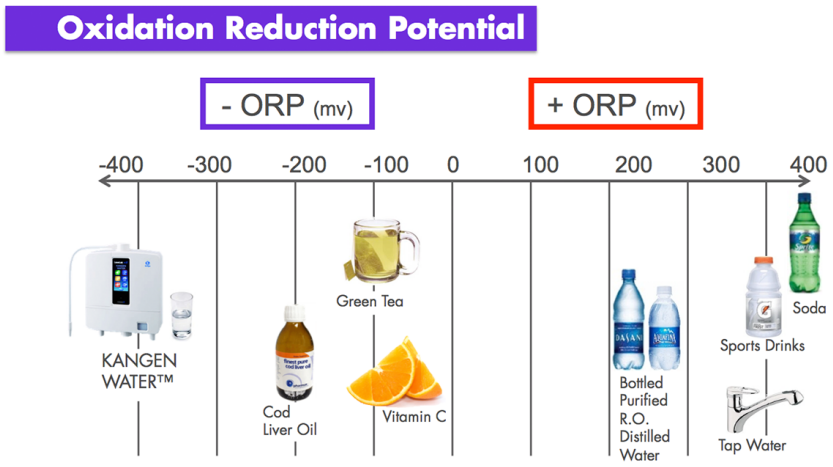

Figure 2: ORP Chart
http://kangenwaterhq.com/what-is-kangen-water/oxidation-reduction-potential/

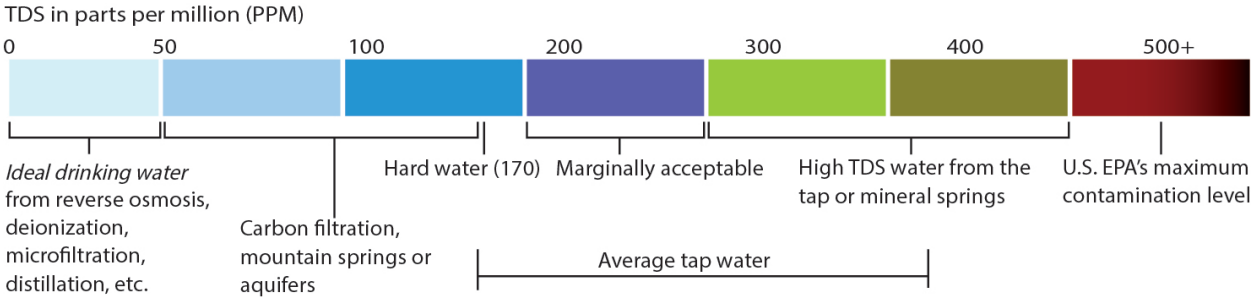

Figure 3: TDS Chart
http://atompurifiers.com/faq.html

4. Implementation

SafeWater is comprised of two parts: hardware and software, which communicate through Bluetooth. The architecture is shown in Figure 4.

4.1. Hardware

The core of the device is a Bluno, a variant of the popular Arduino Uno with integrated Bluetooth 4.0 Low Energy (BLE).[3] As an Arduino-programmable development board, one can easily upload C/C++ code to run on the device through the Arduino IDE, and can read/transmit values from its many digital and analog input pins. An additional IO Expansion Shield was put on top of the board,

[3] https://www.dfrobot.com/product-1044.html

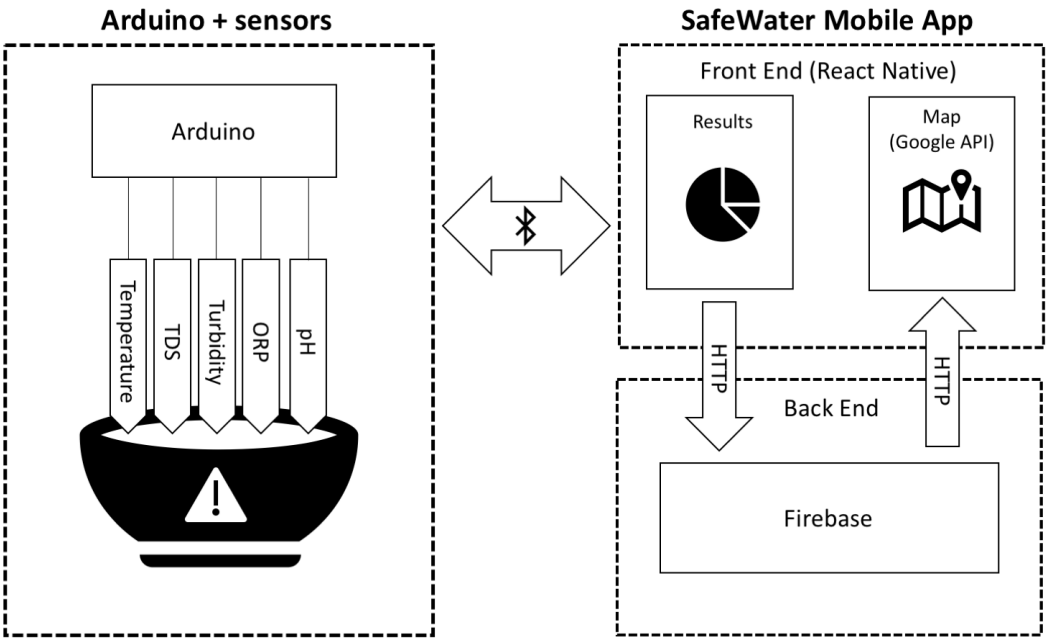

Figure 4: Architecture

which increases the amount of input terminals and eliminates the need for soldering. The Bluno is connected to one digital (temperature) and four analog sensors from DFRobot, an open source hardware provider.[4] Because the analog sensors can interfere with each other's signals, three out of the four analog sensors were connected to an analog signal isolator, which was then connected to the Bluno. Raw sensor values were read every 40 milliseconds, and each sensor requires a specific way to convert its raw input data to a readable value for transmission to the mobile app.

Bluetooth transmission was done through the Bluno's 115200 baud Serial port, sent in data packets of five four-byte arrays, each representing a float value of the sensor reading. Since there was no easy way to convert floats to byte arrays, each value was represented as a union, where setting the union's float value would overwrite its byte array with the same bytes, like so:

```
typedef union {
  float f;
  byte b[4];
} binaryFloat;
```

[4] https://www.dfrobot.com

4.1.1. Temperature

DFRobot's DS18B20 Waterproof Digital Temperature Sensor was used for this project. This 3.0-5.5V sensor can handle up to 125°C, and provides 9 to 12-bit (configurable) readings over a OneWire interface, for which the code was provided. In addition, this project borrowed heavily from example code posted on its product wiki page.[5]

Reading the temperature value from the sensor was simple: bits from the OneWire interface were fed into an array of length 12, which would then be converted into a float using Two's Complement.

4.1.2. Turbidity

DFRobot's Turbidity Sensor was used for this project. It operates at 5V at temperatures from 5°C to 90°C, and outputs voltages from 0-4.5V.

On the Bluno, analog readings range from 0-1023; therefore, to convert it to a voltage, all raw readings must be multiplied by $5/1024$ to get a value in volts. Once the voltage is calculated, it can be converted to an NTU value using this equation:

$$y = \begin{cases} \frac{1}{100}(-1120.4x^2 + 5742.3x - 4352.9) & \text{if } x \geq 2.5 \\ 30 & \text{otherwise} \end{cases}$$

where x is the voltage and y is turbidity in NTU.[6]

4.1.3. pH

DFRobot's Analog pH Sensor/Meter Kit V2 was used for this project. It operates at 3.3-5.5V at temperatures from 5-60°C, and includes standard buffer solutions with pHs of 4.0 and 7.0, for initial calibrations.

Like the temperature sensor, the pH sensor provides its own interface for converting its voltage to a pH value with temperature compensation, using neutral voltage and acid voltage values derived from the sensor's calibrations.[7]

4.1.4. ORP

[5] https://www.dfrobot.com/wiki/index.php/Waterproof_DS18B20_Digital_Temperature_Sensor_(SKU:DFR0198)
[6] https://www.dfrobot.com/wiki/index.php/Turbidity_sensor_SKU:_SEN0189
[7] https://github.com/DFRobot/DFRobot_PH

DFRobot's Analog ORP Sensor Meter was used for this project. It operates at 5.0V at temperatures from 5-70°C, and features a calibration button that allows one to define an offset based on its neutral reading. This project borrowed heavily from example code posted on its product wiki page.[8]

Unlike the other analog sensors, the ORP sensor is sometimes shaky and unstable. Thus, to provide a good reading, it was necessary to create a buffer array, fill the array with values every 40 milliseconds, and update the ORP value with the average of that array, as per the following code:

```
while (true) {
  orpArray[orpArrayIndex++]=analogRead(orpPin);
  if (orpArrayIndex==ArrayLenth) // update ORP value
  {
      orpValue.f=((30*(double)VOLTAGE*1000)-(75*averageArray(orpArray,
    ArrayLenth)*VOLTAGE*1000/1024))/75-OFFSET;    //convert the analog value to
     ORP according to the circuit
      orpArrayIndex=0;
  }
}
```

A buffer array of length 20 was used.

4.1.5. TDS

DFRobot's Analog TDS Sensor/Meter was used for this project. It operates at 3.3-5.5V, outputs 0-2.3V, and its probe consists of two needles that measure the water's conductivity, which is directly to its TDS value. This project borrowed heavily from example code posted on its product wiki page.[9]

Similarly to the ORP sensor, a buffer array was necessary for reading the TDS sensor voltages; however, the median of the array was used instead of the average. Then, the water temperature was used to create a compensated voltage, which was converted to a TDS value in ppm using this equation:

[8] https://www.dfrobot.com/wiki/index.php/Analog_ORP_Meter(SKU:SEN0165)
[9] https://www.dfrobot.com/wiki/index.php/Gravity:_Analog_TDS_Sensor_/_Meter_For_Arduino_SKU:_SEN0244

$$y = \frac{1}{2}(133.42x^3 - 255.86x^2 + 857.39x),$$

where x is the temperature-compensated voltage and y is TDS in ppm.[10]

Figure 5 shows the Bluno and all of its connected sensors:

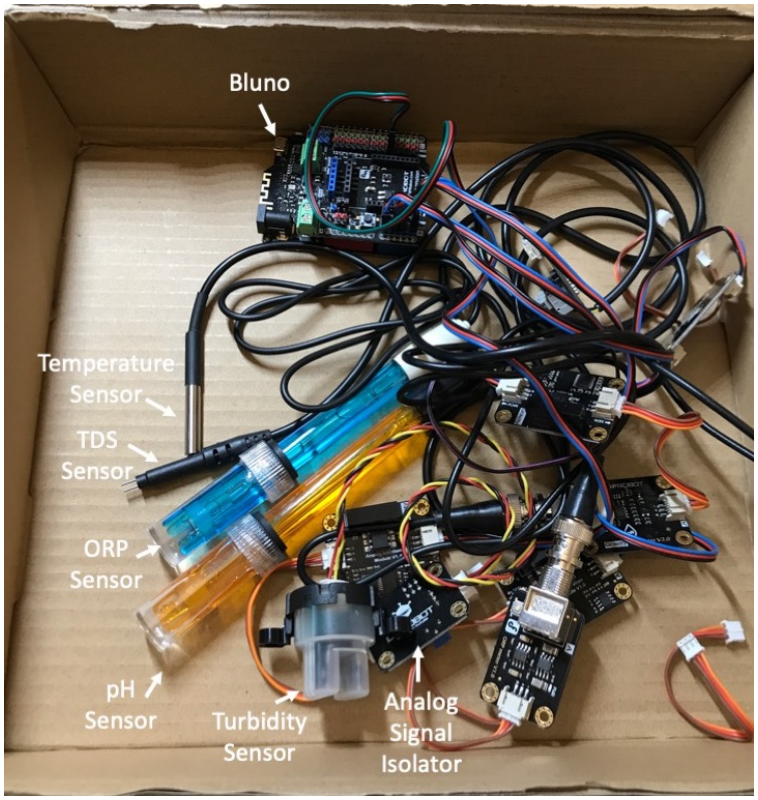

Figure 5: Hardware

4.2. Software

The mobile app is built on React Native for the frontend, because of its ease of development, ability to deploy to both Android and iOS, and variety of third-party packages available on npm. React Native, owned by Facebook, is a framework for building native mobile apps using Javascript and React, which are used for web apps. The typical MVC structure for developing user interfaces are bundled into components, which render using a unique syntax called JSX, a sort of amalgamation

[10] https://www.dfrobot.com/wiki/index.php/Gravity:_Analog_TDS_Sensor_/_Meter_For_Arduino_SKU:_SEN0244

of HTML and Javascript. React Native also features Hot Reloading, which reloads your app simulator instantly during development while retaining the application state, eliminating the need for recompilation. The work flow of the SafeWater app is shown in Figure 6.

In addition, Google Firebase is used as a backend, because of its ease of use, powerful complimentary service, and integrated infrastructure management. Firebase features simple authentication definitions, which manage who can read/write data to the database, and a realtime-updating self-scaling database, that can be viewed and managed on a web console.

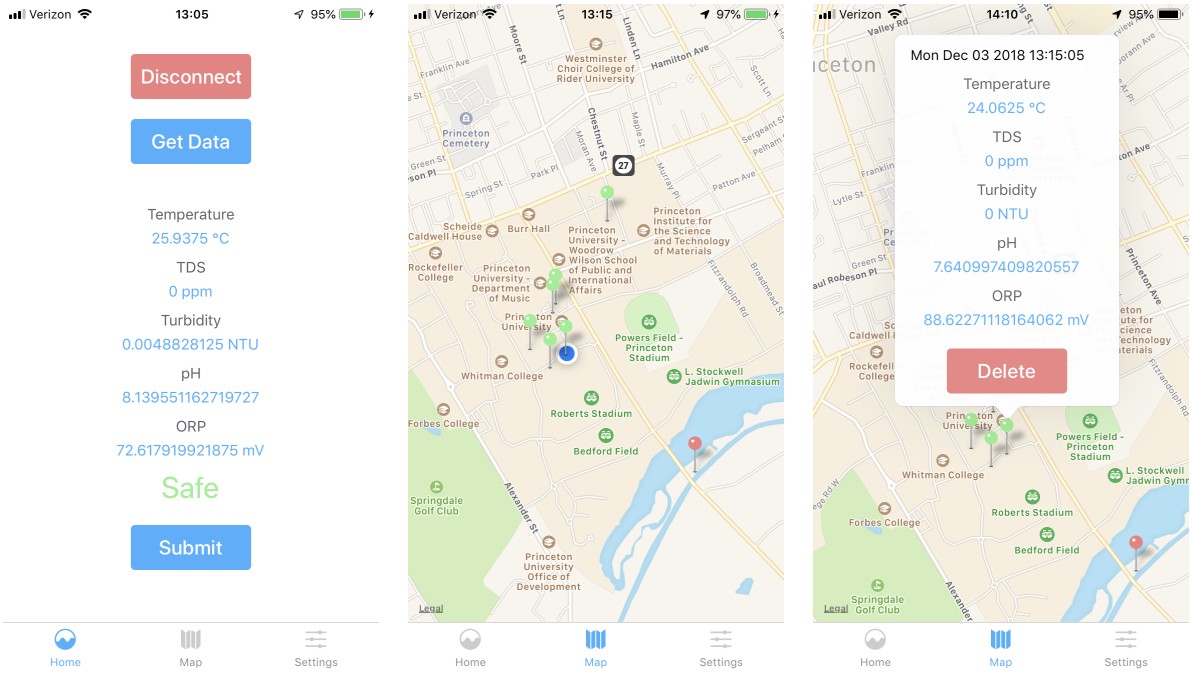

Figure 6: Work Flow

4.2.1. Frontend

The SafeWater app uses several third-party packages, including **react-native-ble-manager**, **react-native-blur**, **react-native-button**, **react-native-maps**, **react-native-spinkit**, **react-native-vector-icons**, and **react-navigation** (listed in app/SafeWater/package.json).

- **react-native-ble-manager**[11] provides methods to connect to, disconnect from, retrieve services from, start notifications on, and write to a Bluetooth peripheral. Every Bluetooth device has a unique UUID, service UUID, and a characteristic UUID - the Bluno only has one of each, and

[11] https://github.com/innoveit/react-native-ble-manager

they are defined in app/SafeWater/constants/Globals.js. Because this library contains native code (written in Objective-C), it was necessary to link this library to the iOS project file, unfortunately making the app iOS-only.

- **react-native-blur**[12] provides components for blur and vibrancy effects. SafeWater uses it as an overlay under the loading spinner (while the app is connecting to the device).
- **react-native-button**[13] provides a component for a simple, customizable button. SafeWater uses it for every button on the interface.
- **react-native-maps**[14] is a powerful library for customizable map components. Including Apple Maps in the SafeWater app was as simple as:

```
<MapView
   region={this.state.region}
   style={styles.map}
   showsUserLocation={true}
   onRegionChange={this.onRegionChange}
>
</MapView>
```

  This library also includes marker components that can be rendered on the map, given a latitude and longitude, and custom callouts for those markers. Similarly to react-native-ble-manager, react-native-maps required linking to the iOS project.
- **react-native-spinkit**[15] is a collection of animated loading indicators. SafeWater displays it while the app is connecting to the Bluno.
- **react-native-vector-icons**[16] is a collection of customizable icons for React Native. It includes such popular open-source icon bundles as FontAwesome, Ionicons, MaterialIcons, and Octicons. SafeWater uses icons from Ionicons in its navigation bar on the bottom of the screen.
- **react-navigation**[17] provides routing and navigation between different screens in React Native

[12] https://github.com/react-native-community/react-native-blur
[13] https://github.com/ide/react-native-button
[14] https://github.com/react-native-community/react-native-maps
[15] https://github.com/maxs15/react-native-spinkit
[16] https://github.com/oblador/react-native-vector-icons
[17] https://github.com/react-navigation/react-navigation

apps. SafeWater's main navigation components are defined in app/SafeWater/navigation, while the screens between which it navigates are defined in app/SafeWater/screens.

4.2.2. Backend

Accessing the Firebase database is done through the Firebase Node.js package.[18] In SafeWater, this package is interfaced in app/SafeWater/constants/Database.js. This package not only provides methods for all four CRUD functions of persistent storage - create, read, update, and delete - but also provides an asynchronous function that, when called, subscribes to updates and calls any callback function every time the database is updated. In the SafeWater app, this is used to update the map in realtime as new entries are being created, like so:

```
Database.database.ref('data').on("value", (snapshot) => {
      markersTemp = [];
      snapshot.forEach(function(d) {
        markersTemp.push({
          time: d.key,
          diagnostic: d.child('diagnostic').val(),
          latlng: {latitude: d.child('lat').val(), longitude: d.child('long').
    val()},
          sensorData: d.child('sensorData').val()
        });
      });
      this.setState({markers: markersTemp});
});
```

ref('data') provides access to the main data tree in the database, and on() is the asynchronous function. Every component in React Native has a internal list of state s, which can be modified using the component's setState() function. Every time setState() is called, the component re-renders itself, allowing the map to update its markers in realtime.

The user's latitude and longitude were determined through an HTML Geolocation object and its getCurrentPosition() function, called like so:

[18] https://github.com/firebase/firebase-js-sdk

```
navigator.geolocation.getCurrentPosition(
      (position) => {
        this.setState({
          region: {
            latitude: position.coords.latitude,
            longitude: position.coords.longitude,
            latitudeDelta: 0.0922,
            longitudeDelta: 0.0421,
          }
        });
      },
      (error) => {
        Alert.alert(error);
      },
      { enableHighAccuracy: true, timeout: 5000, maximumAge: 5000 },
);
```

The above code is written in the MapScreen (app/SafeWater/screens/MapScreen.js) component's componentDidMount() function, so that it would be called as soon as the screen mounts in the app.

Figure 7 shows how the data is structured in the database. Each entry in the database is keyed by the Unix timestamp at which it was created. The entry has five children: a diagnostic of either "Safe" or "Not Safe", its latitude and longitude values, and its array of sensor data.

5. Evaluation

Project SafeWater's evaluation framework consisted of two parts: **lab work** and **field work**.

5.1. Lab Work

Lab work involved testing the device on various different sources of water and other liquids in a controlled environment, to check if the sensors were working and giving appropriate results. The liquids were poured into an 18oz Solo cup, into which the sensors were then put. Room temperature was approximately 20°C.

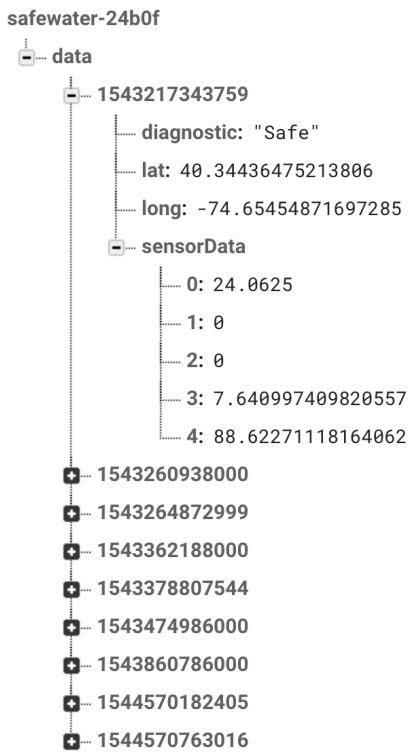

Figure 7: Firebase data tree

Data:

	Temp.	TDS	Turb.	pH	ORP
Tap water	19.06	186.59	3.00	7.50	227.24
Tap water (warm)	29.31	158.52	7.86	7.77	204.45
Apple Cider Vinegar	21.00	1664.75	19.45	4.98	307.81
Filtered water	16.56	150.86	4.18	6.72	194.42
Makgeolli	5.63	274.70	30.00	5.18	300.48
Listerine	21.00	362.70	-3.65	5.30	331.68

The temperature, TDS, and pH sensors worked very well, giving expected approximate values. The ORP sensor only output a very small range of values, 200-300, and due to no prior knowledge of expected ORP values, its accuracy was unable to be determined. The turbidity sensor output mostly expected values; however there were two outliers: Tap water (warm) and Listerine. For the former, an increase in temperature added 4.86 NTUs to its previously recorded value of 3.00, which brought it outside of safe water range. For the latter, the sensor output a negative value, which was completely unexpected and erroneous. These errors may be due to inherent inaccuracies in the

sensor, or a flaw in the equation stated in section 4.1.2.

5.2. Field Work

Field work involved walking around campus and collecting data points at various locations, even some on the Princeton Towpath along Lake Carnegie. Evaluation metrics included user experience of both the app and the device (feedback from friends), and software bugs. Data points are visualized in Figure 6.

Pros:

- UI is simple and intuitive
- Map works in realtime and color of pins clearly show water safety
- Geolocation is very accurate, can distinguish between different rooms of a building

Cons:

- Occasional bugs when reading data (null pointer, array index out-of-bounds)
- Device is difficult to carry around and sensor wires get tangled up - form factor could be improved
- Could add more features to the app

6. Conclusion

Project SafeWater prototypes a device that is **cheap**, **mobile**, and **open-source**. If mass-produced, its low price point would make it accessible to people around the world who may not have access to many resources. Its iOS app and hardware design make it mobile, so users don't have to carry filtration systems, batteries, and/or purification chemicals, all of which are limited by resources. Most importantly, it was built from only open-source components and will remain to be open-source, so like the Riffle, any developer can build upon it and improve it.

Figure 8 compares SafeWater with two water quality multiparameter meters on the market, both sold by Thomas Scientific.[19] The product on the left is a high-end industry meter, while the one in the middle is a lower-end consumer meter. SafeWater has almost all of the capabilities of the

[19] https://www.thomassci.com/scientific-supplies/Water-Quality-Meter

consumer meter, albeit with lower quality sensor probes, but with more than half the price, reduced power consumption, and a more accessible data transfer method.

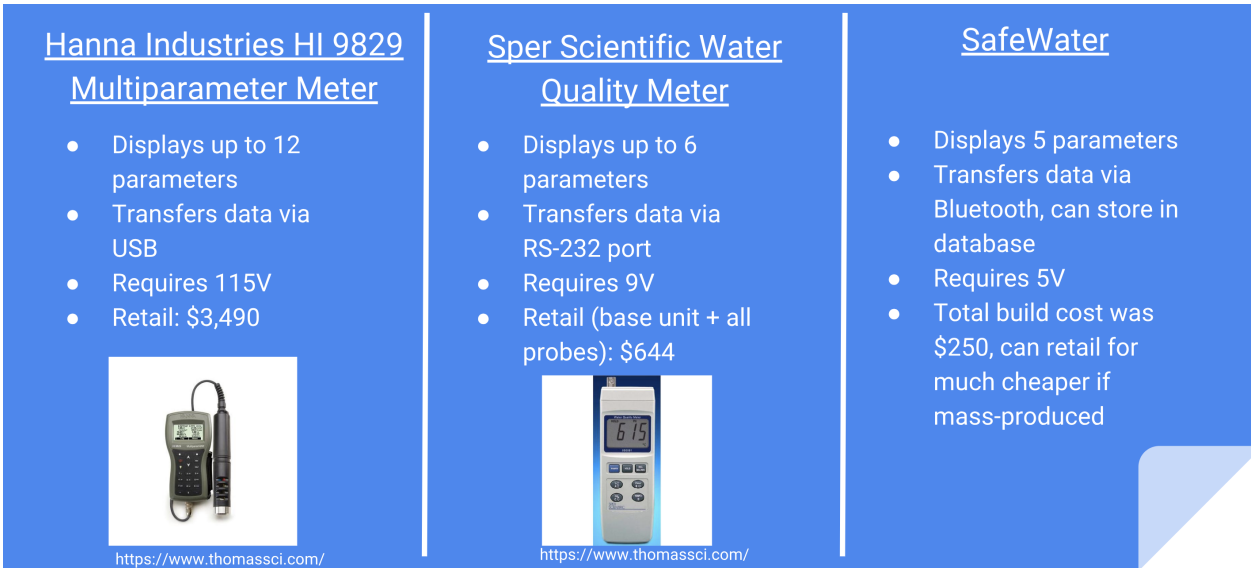

Figure 8: Costs vs. Capability

Possible future work for Project SafeWater includes but are not limited to:

- Improved hardware form factor: 3D-printable waterproof housing design, merging all sensors into one probe
- More sensors: dissolved oxygen, bacteria, conductivity, salinity, ions
- User login and account management
- Directions to nearest clean water source based on mapped data points
- Collaborating with apps like Jay Lee's ViewShare for a more comprehensive mapping of the environment

The possibilities are endless. Here's one small step towards a better planet!

GitHub repo: https://github.com/gnuoyohes/SafeWater

7. Acknowledgments

I would like to thank my advisor Dr. Kaplan, seminar TA Molly Pan, and colleagues of IW 05 for their invaluable ideas and support for Project SafeWater. I would also like to thank Dean Bogucki and the School of Engineering and Applied Science for funding this project. In addition, this project would not have come to fruition without various online API guides and docs and their selfless contributors, and the Princeton Towpath, which provided me with such valuable data and a beautiful morning bike ride.

References

[1] Government of Northwest Territories, "Oxidation-Reduction Potential (ORP)," [Online; accessed 8-January-2019]. Available: https://www.enr.gov.nt.ca/sites/enr/files/oxidation-reduction_potential.pd

[2] J. Jiang, X. Wang, R. Chao, Y. Ren, C. Hu, Z. Xu, and G. Logan Liu, "Cellphone based Portable Bacteria Pre-Concentrating microfluidic Sensor and Impedance Sensing System," 2013, [Online; accessed 6-January-2019]. Available: https://arxiv.org/pdf/1312.0329.pdf

[3] Lishtot, "TestDrop Pro," 2017, [Online; accessed 6-January-2019]. Available: https://www.lishtot.com/TestDrop-Pro.html

[4] Public Lab contributors, "Riffle," 2018, [Online; accessed 6-January-2019]. Available: https://publiclab.org/wiki/riffle

[5] Safe Drinking Water Foundation, "TDS and pH," 2017, [Online; accessed 8-January-2019]. Available: https://www.safewater.org/fact-sheets-1/2017/1/23/tds-and-ph

[6] Safe Drinking Water Foundation, "Water Temperature," 2018, [Online; accessed 8-January-2019]. Available: https://www.safewater.org/fact-sheets-1/2018/8/15/water-temperature-fact-sheet

[7] S. Schmidt, "Puerto Rico After Maria," 2018, [Online; accessed 6-January-2019]. Available: https://www.washingtonpost.com/news/national/wp/2018/09/12/feature/water-is-everything-but-for-many-in-puerto-rico-it-is-still-scarce/?noredirect=on&utm_term=.eee30c3458d0

[8] Wikipedia contributors, "Turbidity," 2018, [Online; accessed 8-January-2019]. Available: https://en.wikipedia.org/wiki/Turbidity

[9] Wikipedia contributors, "Flint Water Crisis," 2019, [Online; accessed 6-January-2019]. Available: https://en.wikipedia.org/wiki/Flint_water_crisis