The Smart Bottle - Ensuring Proper Hydration

DUARTE GALVÃO PETHRUS GÄRDBORN
dadsg | gardborn @kth.se

May 29, 2019

Contents

1 Introduction

When Mark Weiser first proposed the concept of Ubiquitous Computing (UC), he envisioned a world in which computers would become a tool so natural to use that the user could focus solely on the task to be accomplished rather than how to handle the the technicalities of the computer [1], similar to how one does not think of how to handle a pen when writing by hand. In other words, computers should not distract the user with unrelated things. There are several aspects involved in finally reaching such a goal but one important part is that the interaction with computers should aim to be as human-friendly as possible rather than computer-friendly, e.g. you should not have to be an expert on how computers work to be able to successfully use them. Attention should not have to be on the technology used, but rather on the task one wants to accomplish. One way of making a computer more human-friendly, is to make it aware of its *context* [2], meaning that it should respond differently depending on the present environmental context and the state of the user. Moreover, the concept of *calm technology*, described in the same paper, describes how a computer program should know when to just operate in the background, without the user having to pay attention to it, and when to enter the foreground. There is a parallel with a car driver who normally is calmly driving, perhaps thinking about other things, but immediately reacts highly focused as soon as an animal jumps out in front of the car. In recent years there has also been a growing surge of another technological concept which goes well hand in hand with UC, namely, the Internet of Things (IoT), where the idea is to have all kinds of objects or *things* connected to the Internet by means of attaching radio transmitters and micro-controllers to them [3]. IoT can act as an enabler in UC since programs can easily receive direct input from sensors attached to objects and thus be able to dynamically respond to and even interact with the user.

We present the concept of a Smart Bottle which collects sensor data from the user environment - air temperature and water consumption - processes it and then tells the user whether his/her water consumption is appropriate or if there is a risk of dehydration. The Smart Bottle Application is inspired by ideas stemming from UC, using IoT as an enabler; it is easy to use and interact with for a human being and it is an application that responds to changes in the environment, e.g. air temperature and amount of water consumed. The user can interact with the bottle in an intuitive way using a simple web client on a smartphone. In Section 2, the high-level design of the concept is presented and in Section 3 the details of the prototype and especially the classification of sensor data is put forth. Finally in Section 4, we discuss the Smart Bottle's relationship with UC, its limitations as a prototype and make suggestions for the future to improve it and make it more complete.

2 Design

Below, we present a high-level view of the different parts of the Smart Bottle ecosystem: the main functionality of each part and how they interact. As seen in Fig. 1, there are three components in the Smart Bottle ecosystem which are all necessary for it to function properly:

- Smart Phone Web Client
- Smart Phone Server
- Bottle Client

The Bottle Client runs on a microprocessor on the bottle itself and is connected with two sensors measuring water weight and air temperature. Readings of the sensors are recorded on a regular basis, timestamped, and then sent in batches to the Smart Phone Server. The user needs to register the bottle at the Smart Phone Web Client. That is also where the user needs to provide age, sex and level of physical activity parameters to enable as accurate of a water consumption estimate as possible. The server performs some pre-processing of the raw sensor data and classifies it according to calculations based on water requirements for different individuals. Once the user asks about their status through the web client, the server responds with a statement letting the user know if water consumption is fine or if there is a risk of dehydration. The

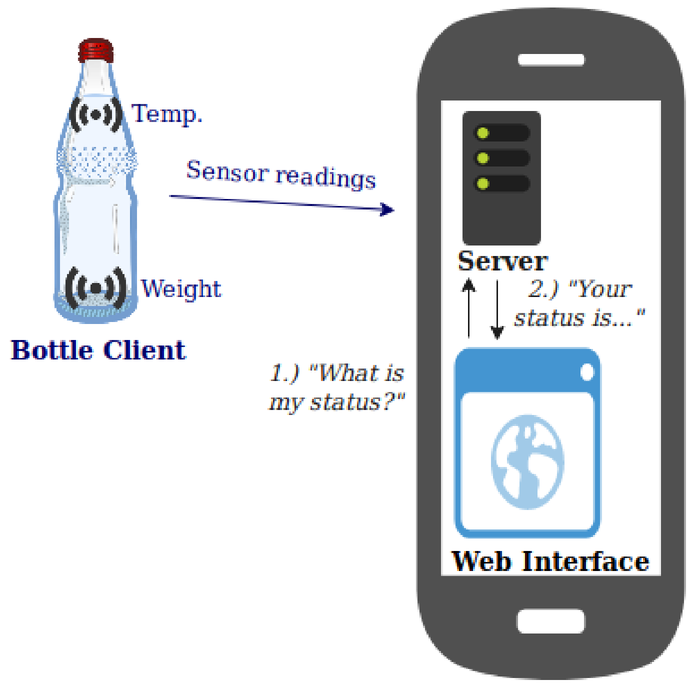


Figure 1: The different parts of the Smart Bottle ecosystem and their interactions.

temperature, weight and time readings are used together to deduce the consumption of the user at different air temperatures. In the prototype implementation, the Smart Phone Web Client and Smart Phone Server are both implemented whereas the Bottle Client is simulated.

3 Implementation

First, the retrieval of actual water consumption and air temperature at different times is described. Then, the process of estimating an individual's recommended water consumption is presented together with the sources used to produce the estimate. Finally, the last step of comparing the actual consumption with the recommended consumption and the generation of a score and recommendation is described.

The source code for a demo of this solution, implemented in Python, is available on GitHub:

`https://github.com/duartegalvao/Smart-Bottle`

3.1 Record Actual Water Consumption

The idea of a smart bottle requires a real bottle equipped with a microprocessor and two sensors which regularly sends its readings to the Smart Phone Server. However, in our prototype, the Bottle Client functionality is simulated as a Python program which generates probabilistic temperature and weight readings from a randomized wave function at regular intervals (by default, every 5 minutes), as shown in Fig. 2. When it is time to register new readings of temperature and weight, they are timestamped and stored locally at the bottle. A few times of day, batches of gathered readings are sent over HTTP/REST to the Smart Phone Server which handles the processing of data.

3.1.1 Weight Data Pre-Processing

The sensor readings are expected to have some variation and be quite erratic, especially since we are relying on a weight sensor, which will give nonsensical values whenever the bottle is not sitting down flat. To solve this, we chose to do server-side pre-processing of the weight data. To find stable points, we look for groups of consecutive points with low internal variance (< 0.1), and if a group contains more than three points its

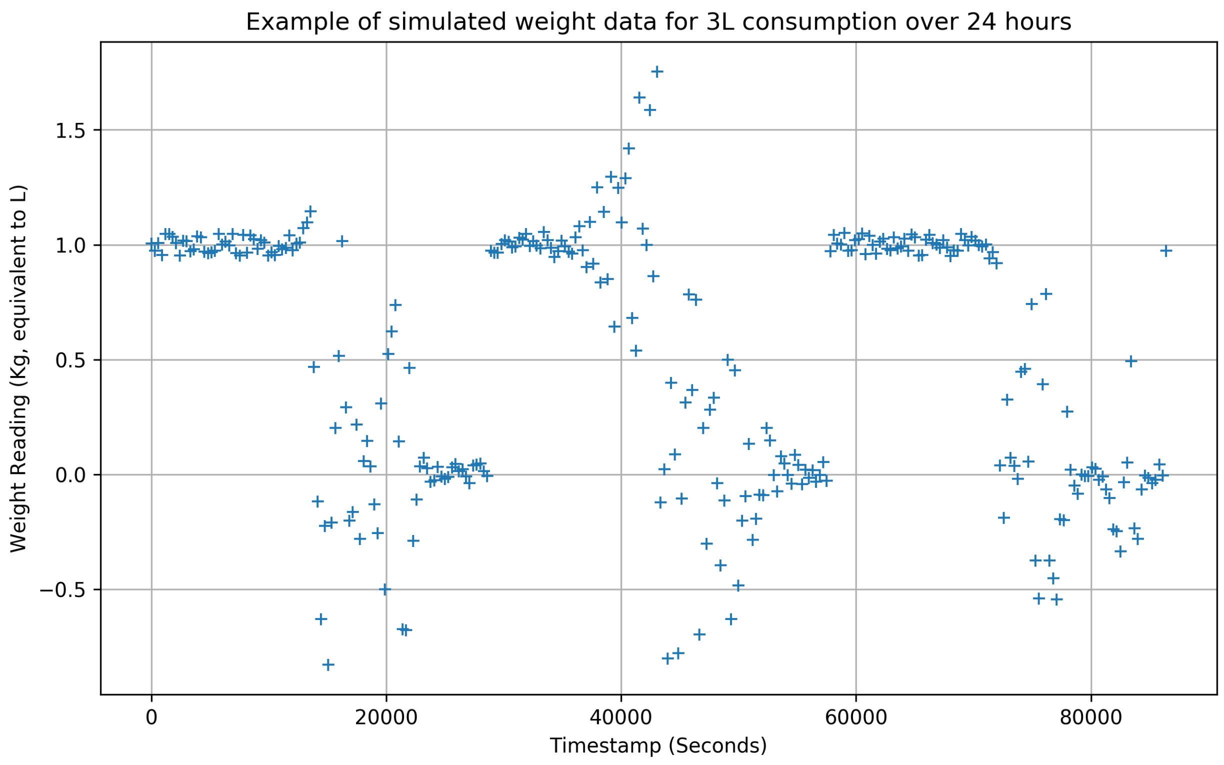


Figure 2: Example weight output of the simulated bottle.

median weight and time is saved. Then, to calculate consumption, for every two consecutive stable points w_i and w_{i+1}, if $w_i > w_{i+1}$, we add $w_i - w_{i+1}$ to the total consumption value.

3.2 Estimate Recommended Water Consumption Based on Individual Needs

The Smart Phone Server, implemented in the Django framework, receives data sent from the Bottle Client and stores each individual reading into a database sorted according to their timestamps. Once the user requests to know the water consumption status, an appropriate response is given based on different calculations which are explained in further detail below. The basis for the calculations are [4]. According to the authors, the average water requirements for males and females are 3.7 L and 2.7 L respectively. However, individual variance is high and many factors affect the actual individual need. Other than sex, the level of physical activity as well as air temperature are two of the main factors affecting water consumption. Therefore, our application takes into account all of these factors when estimating an individual's water needs. Several different formulas are used and joined together to form the final classification mechanisms. We will now explain, step by step, how the classification formulas were constructed and joined together.

The first step was to generate reference data based on required water consumption under different activity levels (a_L) and air temperatures (T). In Fig. 3 of [4], the different levels of daily water requirements are compared based on four different activity levels under different air temperatures. Since we did not have access to the original formulas used, we have replicated this figure through visual readings of its values and an approximation using a polynomial function which very closely resembles the original figure. This resulted in the function fit_1, which is plotted in Fig. 3.

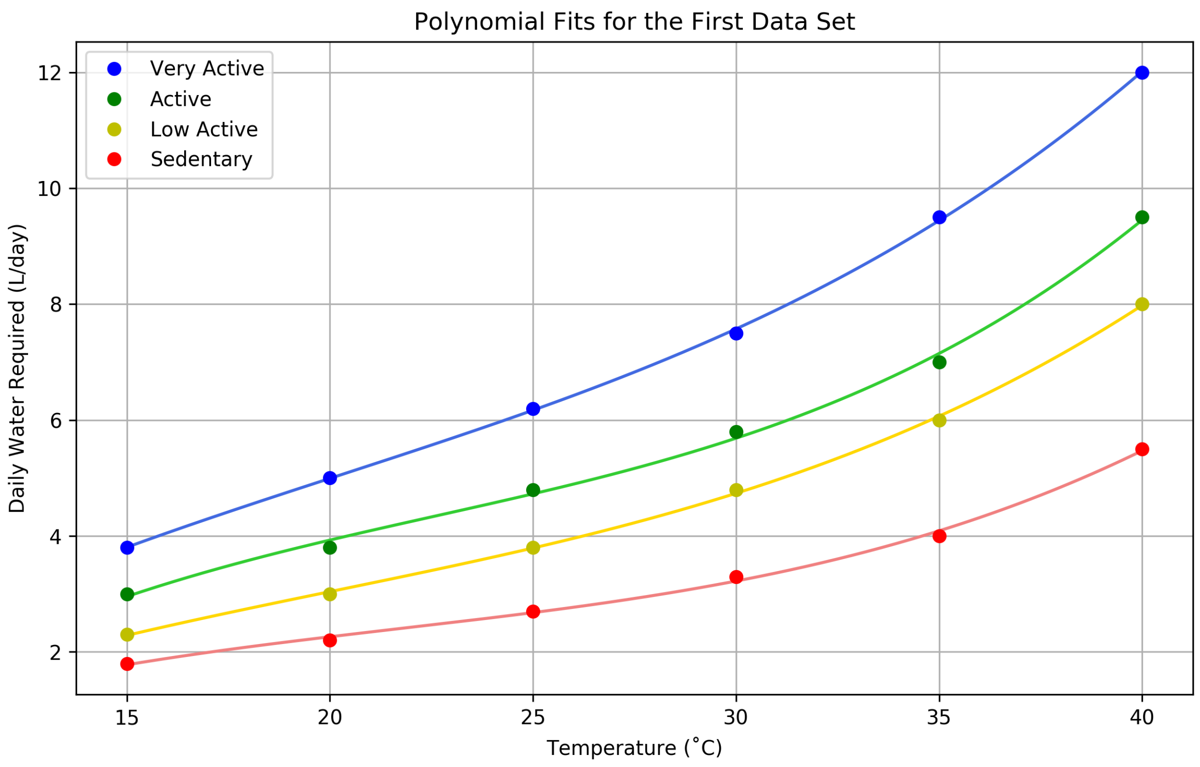


Figure 3: Water requirement under different activity levels and air temperatures.

$$fit_1(T,a_L) = \begin{cases} -2.017 + 0.448T - 0.0169T^2 + 2.6\times10^{-4}T^3 & a_L = \texttt{"Sedentary"} \\ -1.938 + 0.457T - 0.0156T^2 + 2.6\times10^{-4}T^3 & a_L = \texttt{"Low Active"} \\ -4.412 + 0.849T - 0.0306T^2 + 4.5\times10^{-4}T^3 & a_L = \texttt{"Active"} \\ -2.219 + 0.620T - 0.0193T^2 + 3.2\times10^{-4}T^3 & a_L = \texttt{"Very Active"} \end{cases}$$

The second step was to also take the user's age (A) and sex (S) into account. These calculations are based on Fig. 2 of [4] which shows average required water consumption in different age groups based on sex. The data was now fitted to another curve, namely fit_2 - a logarithm and a sigmoid - and the resulting graph can be seen in Fig. 4.

$$fit_2(A,S) = \begin{cases} 0.426\times\log(1.186A) + 1.010 & S = \texttt{"Female"} \\ fit_2(A,\texttt{"Female"}) + \frac{-1.001}{1+\exp(A-12.1)^{0.768}} + 0.999 & S = \texttt{"Male"} \end{cases}$$

Finally a "master" formula combining fit_2 and fit_1 calculates the final estimate. The starting point is an average consumption ($c_{average}$) of 3 L/day (a rounded value from the figures mentioned in [4]), disregarding sex, age and level of physical activity. Then, the calculation based on the specific air temperature and activity level is added. Finally, sex and age are added to produce the final estimate for the specific user.

$$c_{ideal}(T,a_L,A,S) = c_{average} + \widehat{fit_1}(T,a_L) + \widehat{fit_2}(A,S)$$

where

$$\widehat{fit_n}(x_1,\ldots,x_k) = fit_n(x_1,\ldots,x_k) - c_{average}$$

Additionally, to avoid erroneous outputs, all the input parameters were bound to realistic values.

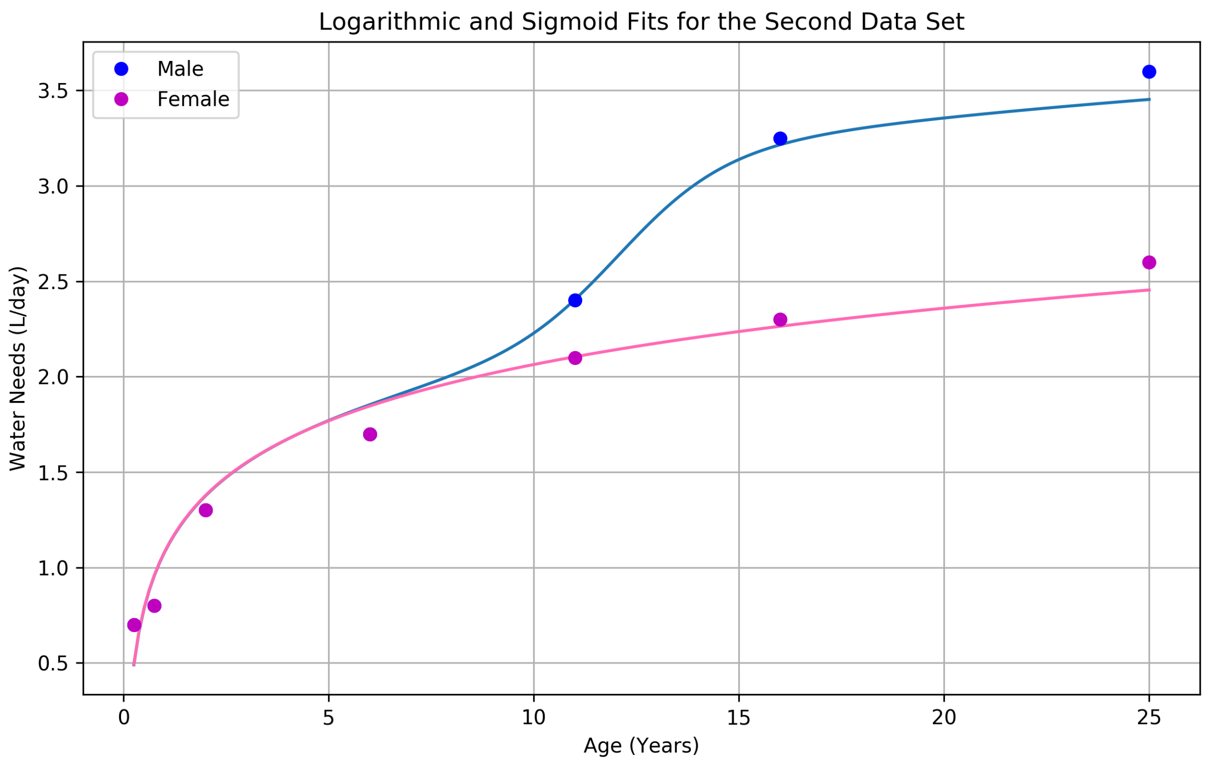


Figure 4: Water requirement as a function of age and sex.

3.3 Compare Actual Consumption with Estimate and Present Recommendation

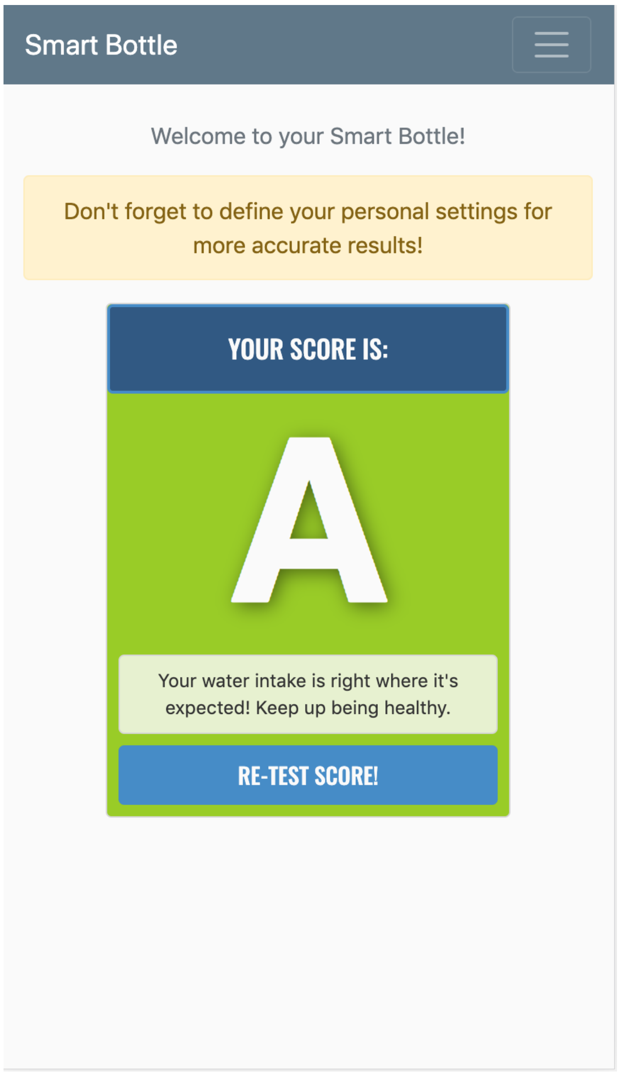
By all means, classification is the most important part in generating an appropriate response to the user. However, based on the final ”master formula”, it is necessary to interpret the result in a well-founded manner and present it in a user-friendly way. Towards the end of [4], there is a discussion about what ranges can be suitable recommendations of requires water intake based on average values. The authors say that it is possible to deviate as much as 1 L from the average value and consumption might still be fine. We have decided that beginning with a 0.5 L deviation a warning can be given to the user that consumption *might* be too low to stay hydrated. In total, there are five different recommendations that are given as seen in Table 1.

Table 1: Different recommendations depending on classification

Variance [L]	**Score**	**Recommendation**
$< +0.6$	O	”Your water intake seems higher than expected.”
< -0.2	A	”Your water intake is right where it’s expected!”
< -0.5	B	”Your water intake seems slightly lower than expected.”
< -1.0	C	”Your water intake seems lower than expected.”
> -1.0	D	”Your water intake seems a lot lower than expected.”

To display the results to the user, the Web Client offers three different views: Home, Analytics and Settings. The Settings screen is simply where the user enters age, sex and level of physical activity and is able to reset all measurement readings. The Home screen displays the last tested score together with a recommendation, as seen in Fig. 5a. To get the most up-to-date score, the user simply pushes the ”Re-Test Score” button.

To receive more details on the reasons for the particular score, the user can press the menu button in the top right corner and press the ”Analytics” button which will result in the view seen in Fig. 5b. The top


Smart Bottle

Welcome to your Smart Bottle!

Don't forget to define your personal settings for more accurate results!

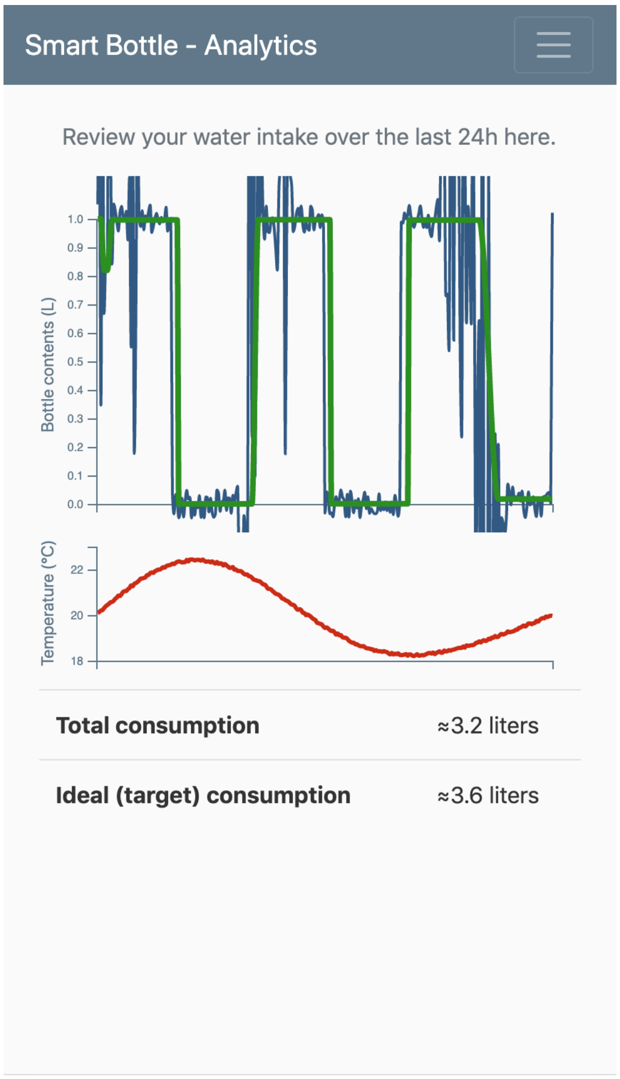
YOUR SCORE IS:

A

Your water intake is right where it's expected! Keep up being healthy.

RE-TEST SCORE!

(a) Home screen


Smart Bottle - Analytics

Review your water intake over the last 24h here.

Total consumption	≈3.2 liters
Ideal (target) consumption	≈3.6 liters

(b) Analytics screen

Figure 5: Screenshots from the demo application.

chart displays water consumption and bottle refills as a function of time, showing both the actual readings and the pre-processed measurements. The chart below displays the temperature recorded throughout the day. Finally, the estimated ideal water consumption (from the formulas in Section 3.2) is presented together with the actual consumption so that users can see how far off or close to target they are.

4 Discussion and Future Work

First, the implementation and its relationship with UC concepts is discussed. Then follows a discussion on the limitations and possible improvements of the prototype both with regards to specific technical aspects as well as matters of user interaction where we make suggestions of how to make it even more of a UC application.

4.1 Relationship to Ubiquitous Computing

As mentioned in Section 1, there are certain features, such as *context awareness*, that make an application adhere to the concept of UC. The Smart Bottle Application is all about tapping into its environment to be able to make appropriate recommendations to its user. In fact, the different air temperatures which are registered by the bottle coupled with the user's water consumption are what determines the output that the user will get when using the UI. The static information given at registration, such as age, sex and activity level are all necessary as well. But it is the dynamically gathered temperature and consumption data which makes it a UC application. Another way of doing it could have been to let the user manually type in their daily consumption coupled with temperature data at different times and then run the same classification as we are using now. However, then it could not have been considered a UC application, because the application itself would not have been aware of its environment. Even though the Smart Bottle App displays UC characteristics, there is still room for improvements also in this regard as we will see in section 4.2.

4.2 Limitations and Future Work

Perhaps the most obvious area of future work is the bottle itself which is only simulated in our prototype. A real bottle needs to be equipped with temperature and weight sensors and a microprocessor which can then perform the sensor readings and send the information which are now only sent from a Python process. When the real bottle is implemented, it is also necessary to change the protocols used for communicating with the server. Right now, the prototype uses HTTP. However, HTTP is not optimal for IoT devices. There are many IoT protocols out there that could be used such as MQTT (Message Queuing Telemetry Transport), Bluetooth, CoAP (Constrained Application Protocol) and so forth. We would propose looking into CoAP [5], since it uses methods similar in names and function to HTTP like GET, POST etc and is also supported by the Django framework. Using CoAP instead of HTTP would drastically reduce bandwidth usage and number of transmissions since it uses the connection-less UDP (User Datagram Protocol) instead of the connection-oriented TCP (Transmission Control Protocol).

The method which we use to provide the estimate could probably be fine-tuned. The problem is that an acceptable deviation of water consumption of up to 1 liter is rather big and it is therefore difficult to be very precise in the recommendations. By taking even more sensors into account, perhaps a smart watch which can measure physical activity in a more dynamic way, even more precise recommendations could be produced.

An interesting scenario for the smart bottle is that of using it on a bigger scale with multiple users - students in a school, for example. To enable that, the application needs to be extended with analytic tools which can look at a collection of smart bottle data to see whether a group as a whole is staying hydrated or not.

Finally, there is another important aspect that we would have liked to improve but had to leave for future work. It concerns the way that the user interact with the application. According to the current design, the user always have to poll the application to be able to receive a recommendation. But what if the application itself would ”poll” the user instead? The idea is that the application would take the initiative and raise a notification or something similar on the user’s smart phone once the app has detected that the user is running the risk of dehydration to remind them of drinking more water. To add that functionality would also make the application more UC-like since the application not only dynamically reacts to changes in the environment but also dynamically communicates those changes with the user. According to the concepts of ”calm computing” [2], the app would normally stay in the background unnoticed but move into the foreground once it has good reasons to do so. Detecting possible dehydration would probably count as a good reason to move into the foreground to gently remind the user of drinking some more water. In that way, the Smart Bottle App would be even more naturally integrated into its environment, and be even more in accordance with Mark Weiser’s original vision.

References

[1] M. Weiser, “The computer for the 21st Century,” *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 19–25, Jan. 2002.

[2] M. Weiser and J. S. Brown, “The Coming Age of Calm Technology,” in *Beyond Calculation: The Next Fifty Years of Computing*, P. J. Denning and R. M. Metcalfe, Eds. New York, NY: Springer New York, 1997, pp. 75–85. [Online]. Available: https://doi.org/10.1007/978-1-4612-0685-9_6

[3] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128610001568

[4] M. N. Sawka, S. N. Cheuvront, and R. Carter, “Human Water Needs,” *Nutrition Reviews*, vol. 63, no. suppl_1, pp. S30–S39, Jun. 2005. [Online]. Available: https://academic.oup.com/nutritionreviews/article/63/suppl_1/S30/1927756

[5] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP).” [Online]. Available: https://tools.ietf.org/html/rfc7252