Portable Water Tracking Attachment

Proposal

ECE445 Spring 2024

Project #44

Subha Somaskandan, Subhi Sharma, Cindy Su

Professor: Arne Fliflet

TA: Luoyan Li

1. Introduction

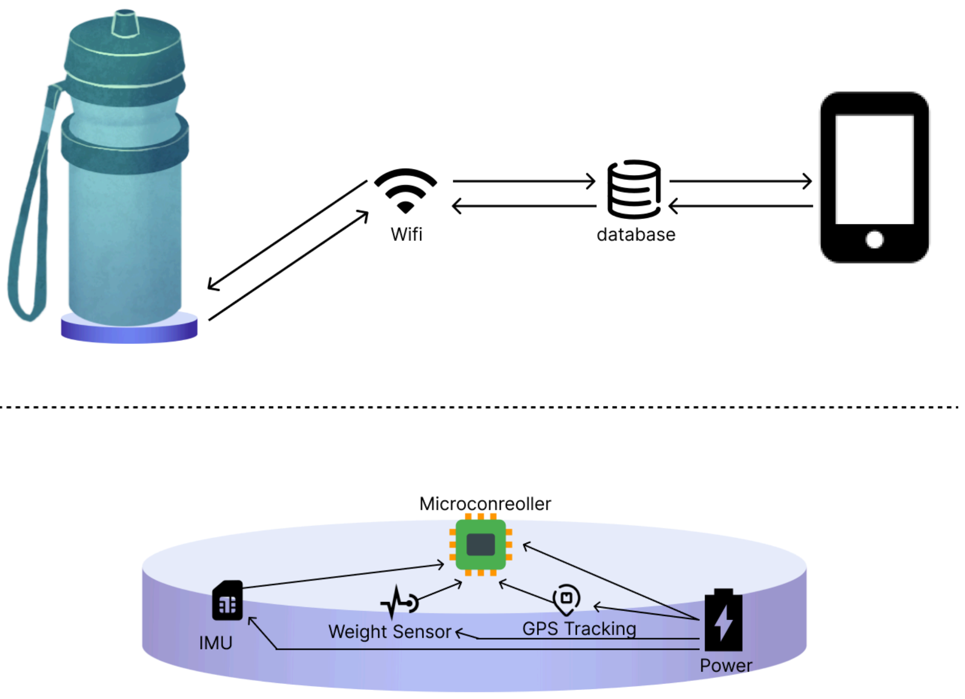
1.1 Problem

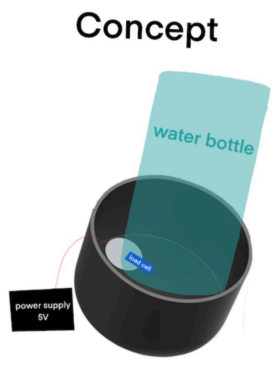
With a busy schedule, it can be difficult to remember to drink enough water during the day. Most water tracking products on the market are not customizable to one's lifestyle or daily habits. The amount of water that everyone needs to drink per day depends on their height, weight, sex, physical activity, and climate. There are a variety of water drinking apps, but these do not actually check if the right amounts of water are being drunk, and they can be ignored without completing the task. Furthermore, there are also water bottles on the market that have markings on them to indicate how much water to drink per hour, but these force one to buy a new bottle entirely.

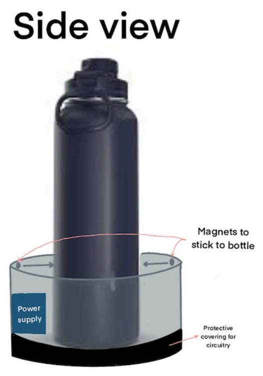
1.2 Solution

Our solution is a portable sleeve bottom attachment to a 32oz water bottle that measures how much water you are drinking, and connects to a smartphone app to remind you to do so. Within the app, you can input information such as your sex, age, activity level, and your recommended water intake will be calculated, or you can input your intake manually. Every hour, a reminder will be sent out to drink a specific amount of water, and the attachment will check using weight and inertial movement sensors if this is completed. The water drinking data will then be relayed via a microcontroller to the app, where you can see your progress for the day. In addition to water tracking, our attachment will be able to be location tracked via a transceiver module, which lets the user find their attachment or water bottle, if lost.

1.3 Visual Aid







1.4 High Level Requirements

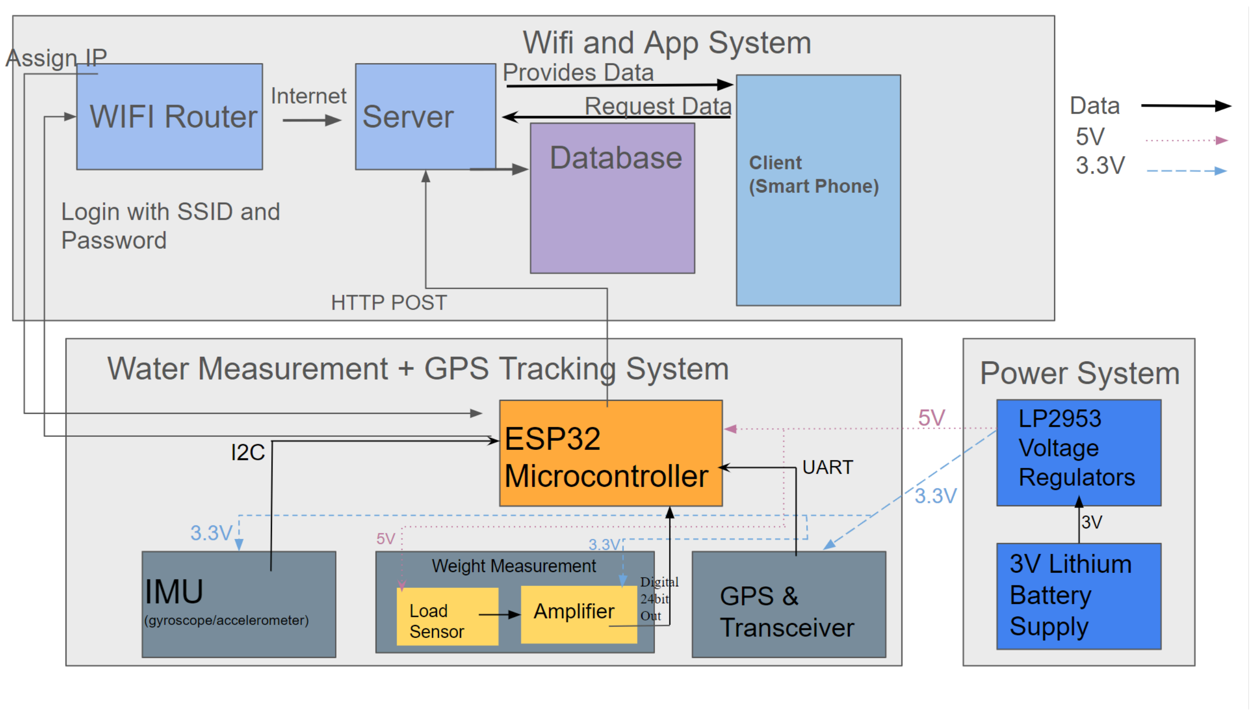
1. The application must set water drinking habits and send reminders to drink calculated amounts per hour based on the user's age, sex, activity level, and hours of sleep.

2. If the user finishes the water requirement before the reminder is sent, the application must not send a reminder until the next hour that the user does not meet the water drinking requirement. Similarly, if the user finishes their water requirement for the day, the application must not send any notifications after this point.

3. The application must track and log the user’s daily water consumption and present the data in an easy-to-understand chart format for the user to check daily water intake.

2. Design

2.1 Block Diagram:



2.2 Subsystem Design

2.2.1 Power Subsystem

The Power Subsystem is made up of a 3V Lithium Battery supply made up of 2 AA batteries, and two voltage regulators. The 3V from the batteries will be able to power the voltage regulators, and one regular will output 3.3V, and the other 5V. As outlined in the block diagram above, the voltage outputs go to the Water Measurement & GPS Tracking subsystem, to power the microcontroller and load sensor with 5V, and the IMU, amplifier, and GPS/Transceiver with 3.3V. The Lithium batteries must be able to supply 3V +/- 5% to the voltage regulator when active, and the voltage regulator outputs must be within 5% of the expected 3.3 or 5V value, with a current limit of 250mA from each regulator. These requirements are vital to our project since steady voltage to power our devices as well as microcontroller guarantees the closest measurement with little error.

2.2.2 Water Measurement + GPS Tracking Subsystem

This system contains an IMU, transceiver, and a load sensor with an accompanying amplifier which all communicate with the ESP32 microcontroller. This system measures the weight of the water bottle through the load sensor and amplifier, specifically when it is placed down, which is indicated by the IMU. This system also enables location tracking through the transceiver and accompanying antenna. This data is sent to the microcontroller through a digital bit signal, I2C protocol, and UART protocol, respectively. This system gets both 5V and 3.3V from the power subsystem as input to send to the load sensor, and the amplifier/IMU/transceiver respectively.

Once the data is received by the microcontroller, there are libraries that we can download through the Arduino IDE, made by the manufacturers of the components, that will help us grab data that we need, like getting the weight from the load sensor, the location from the transceiver, or the positioning of the bottle from the IMU. Requirements of this subsystem include that the IMU data (x,y,z angular velocity and acceleration) must be able to be read on the microcontroller IDE, the load sensor and amplifier combination must be able to detect changes in weight of about 3oz, and the GPS/antenna unit must retrieve geographical data and submit this information to the microcontroller.

2.2.3 Wi-Fi and App Subsystem

When the sensor detects a change in the weight of the water bottle, it updates the measured weight in real-time. This updated weight value is then transmitted to the microcontroller (ESP32) via a digital signal, The microcontroller processes this information and sends it to a remote database through a WiFi connection. And the real-time updates are pushed to a MySQL database hosted on a cloud server.

On the user's end, when they open the application on their smartphone, which will be developed using React Native. The app will initiate a request to the server and this request will be handled through a RESTful API, designed to fetch the latest weight of the water bottle from the MySQL database. The server responds with the most recent data, which is then rendered in the application's UI, providing the user with up-to-date information on their water intake.

The application is designed to comprise three pages:

1. Main Page: This is the application's home screen, with an image of a water bottle in the center that dynamically updates to reflect the current volume of water remaining in the bottle, which offer a easy way for users to access their water consumption at a glance
2. Statistics Page: This page will provide users with data fetched from the SQL database regarding their hydration habits with a line graph, which will display the daily water intake of users, allowing them to track their hydration over time and identify patterns or areas for improvement.

3. Location Page: This page provides the functionality of showing the real time location of the user's water bottle to ensure that it can always be found easily.

Every 30 minutes, the app will check to ensure the user's water intake aligns with their pre-set hydration goals. If the user has not consumed the desired amount of water within this timeframe, the app will send out a notification to the user as a reminder, encouraging the user to drink water and ensure they maintain high hydration levels throughout the day.

2.3 Tolerance Analysis:

The block critical to our project's success is the weight sensor load cell. The readings must be accurate in order to assess how much water the user has drunk, and is vital to sending reminders to the smartphone application. Routed from the HX711 amplifier, the 24 bit digital signal will be encoded in the microcontroller interface, and this reading is given in volts and can be displayed in the microcontroller IDE.

The relationship between the weight that the load cell is measuring and the reading in volts is given by

$$Weight\ Measured\ =\ \frac{Reading * Rated\ Weight\ of\ Sensor}{Sensitivity * Excitation\ Voltage}$$

In our application, the smallest weight that needs to be measured is the weight of the water bottle itself added to the smallest increment of water one can be reminded to drink. This is outlined in footnote 1[1], but the smallest increment is 2.6oz. Converting to kg, the smallest total weight is 0.590 kg + 0.073708 kg = 0.6637088 kg. The rated weight of the sensor is 5kg, the sensitivity is 1+/- 0.15 mV/V, which ranges from 0.85 mV/V to 1.15 mV/V, and the excitation voltage is 5V. Solving the equation, this gives a reading of about 0.5mV. This is a very small voltage, but this can be displayed on the IDE properly. Furthermore, this is the smallest increment, and on average, the increment and the weight needing to be measured will be higher and less error prone.

[1] This quantity is derived from the minimum amount of water the user can manually set, which is 48oz, spread over 24 hours minus the minimum hours of sleep a user can input, which is 6 hours. This equates to 2.6oz per hour the user is awake. Note that the calculated water intake in a day based on user input of age, sex, and activity will be in the range of 60-80oz.

3. Ethics and Safety:

The closest mechanism to our design is a water bottle specifically designed to be “smart” and has app connectivity and location tracking, linked here. Our weight system could be very similar to this smart water bottle, and this could result in accidental misuse.

In the IEEE Code of Ethics, Section I, Point 6, maintaining technical competence and undertaking tasks only after clear limitations have been outlined is mentioned, and our design coming somewhat close to the HidrateSpark water bottle could be an issue here. We believe our design does address some limitations of the water bottle, and our design is significantly different as it is portable. However, we do not want to be infringing on any patents because of the limitations and just repeating the process already done by HidrateSpark.

Furthermore, according to IEEE Code of Ethics Section II, Point 7, discrimination based on age, gender, sex, ethnicity is not tolerated and innovation should reflect this as well. With our product, we do have to make calculations based on metrics such as these, so we have to be careful to not assume anything or be offensive. One idea we have to combat this is to allow an option for the user to set their water intake manually, as this may be what works the best for them. We can also put forth our calculations as recommendations or suggestions rather than facts, as everybody will be different.

Citations:

IEEE Code of Ethics, IEEE, https://www.ieee.org/about/corporate/governance/p7-8.html. Accessed 22 Feb. 2024