Table of Contents

Project Development 3
Introduction 3
Ideation 3
Concept 7
Design 7
Prototype 25
Summary 26

Project Development

Introduction

This chapter outlines the project's evolution, beginning with the initial ideation phase for both the team and the product. After establishing the core concept, including its physical structure, smart systems, and packaging, the focus shifts to the prototyping stage. Here, we detail the key hardware and software adjustments made to the original design, concluding with an analysis of the test results and final performance.

Ideation

The first Ideation of the team idea can be found on Figure 1

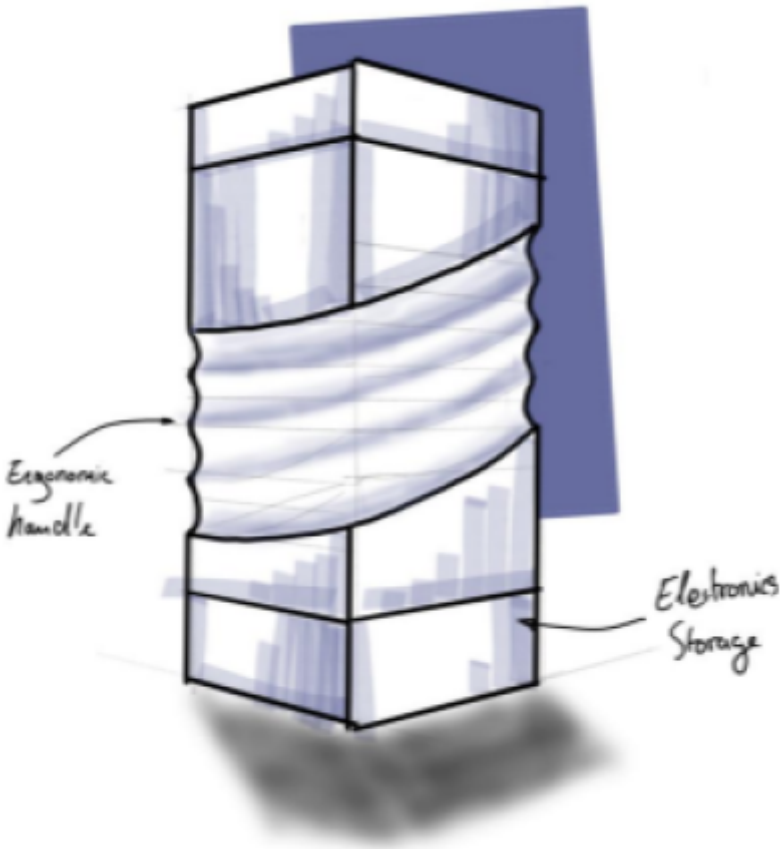

Figure 1: TRAQUA first Design idea

Here the team already had the idea of keeping the components on the bottom of the bottle, but a more square bottle was kept in mind with a handle that would be comfortable for the user's hand.

The second ideation of TRAQUA already has a more “classic” bottle shape 2.

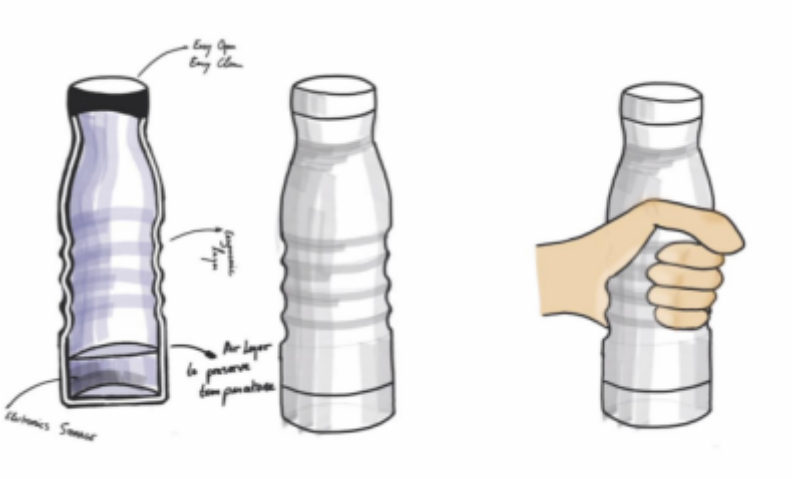

Figure 2: TRAQUA second bottel design

For the third design, the team tried to come up with a concept that would either easy fold or just fit inside a backpack very easily. This was quite hard to accomplish, not because of the design but more in terms of the electronic components. It would be very hard to put them into the bottle if the bottle were shaped like in FIGURE 3.

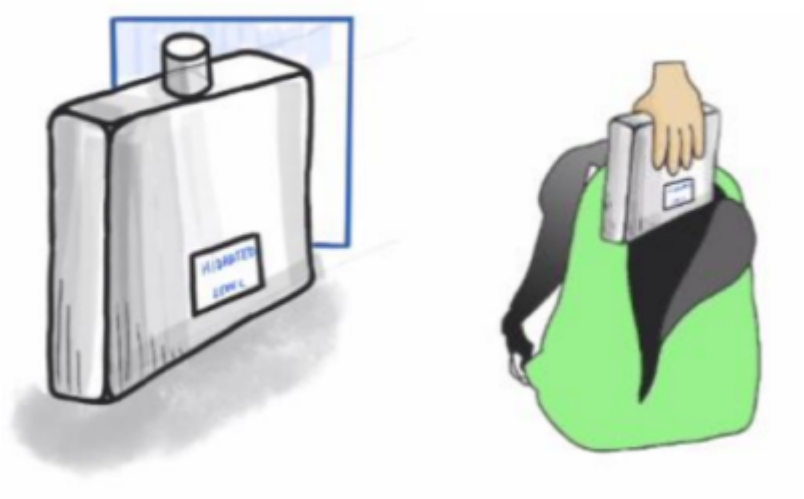

Figure 3: TRAQUA third design

Flyers

The team also had a few changes in their flyer before coming up with the definite solution. Below we will explore the different flyers the TRAQUA design team came up with.

The team's first flyer had no QR code; the text idea was already there, but the text just lies in boxes that feel out of place alongside the TRAQUA water bottle 4.

Figure 4: First design

Choice of the subject

Immediately following the initial presentation, the group gathered to evaluate the proposed project tracks and reach a consensus. Our approach was rooted in an open brainstorming session where we weighed our individual strengths against the potential of each topic. It quickly became clear that the “Smartification of Objects” was the path that generated the most genuine enthusiasm among us.

More specifically, we decided to focus on a smart water bottle. This choice wasn't just based on personal interest; after conducting a preliminary market analysis, we identified a significant gap in the current landscape. We felt we could differentiate our product by consolidating multiple features and services into a single, cohesive solution. This sense of opportunity, combined with the fact that every team member felt comfortable and motivated by the technical challenges involved, made the smart

bottle the definitive choice for our project.

Brainstorming

Once we settled on the smartification of objects, specifically the smart water bottle, the team held its first dedicated brainstorming session. Because the potential features for a “smart” container are so vast, we needed a way to filter the noise and focus on what actually added value. We turned to Miro to map out these early thoughts, using the boards shown to organize our ideas into functional clusters. This collaborative space was essential not just for tracking our suggestions, but for refining a set of concepts that eventually became the foundation of our final proposal. Figure 5 represents one of the brainstorming activities we did in Miro.

Figure 5: Brainstorming in Miro

Design Thinking

Figures 6, 7 and 8 showcase the result of the initial brainstorming sessions conducted as part of the Design Thinking process.

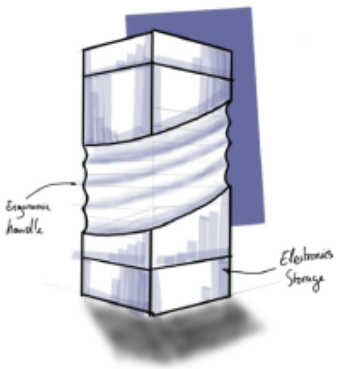

Figure 6: Add caption

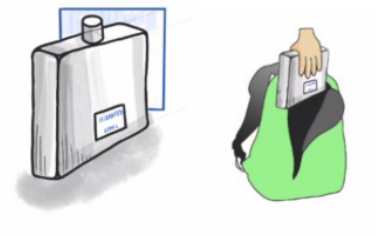

Figure 7: Add caption

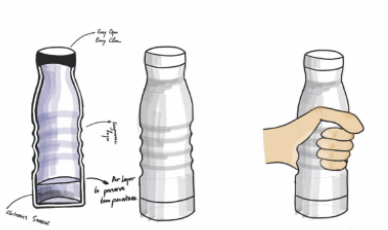

Figure 8: Add caption

Concept

The primary function of the smart bottle is to encourage consistent and healthy water consumption while, in the meantime, eliminating the hygiene concerns commonly associated with reusable water bottles. To achieve this, the bottle integrates five core features into a single device

First, a built-in UV-C sterilization module automatically disinfects the water and the interior surface of the bottle at regular intervals, removing the need for frequent manual cleaning and reducing bacterial buildup. Second, a temperature sensor continuously monitors the drink's temperature and makes this data available to the user in real time. Third, a hydration tracking system records water intake throughout the day and sends customizable reminders when the user falls behind their hydration goal.

All sensor data is transmitted via Bluetooth to a companion mobile application that serves as the central interface for the end user. The app displays all key aspects mentioned above. . This ensures that the user remains actively involved in managing their hydration while the bottle handles the more technical tasks autonomously.

The bottle will be prices at approximately 330 €. For mass production the team could look to reduce the production price up to 100 € per bottle. Meaning the end price of the bottle would be around 250 €.

Design

MATERIAL SELECTION : “The materials for each component of the smart water bottle were selected based on both functional requirements and aesthetic considerations, ensuring a balance between performance and visual appeal. Polypropylene (PP) is used for the cap due to its durability and ability to withstand repeated use, while also providing a clean and smooth finish that contributes to a modern appearance. The bottle body is made from polished aluminum, chosen not only for its lightweight structure and corrosion resistance, but also for its sleek, reflective surface, which enhances the overall aesthetic and gives the product a premium look. For the compartment containing the electronic components, a plastic material such as polycarbonate is used. This ensures proper electrical insulation and waterproof protection, while also allowing for a precise and refined design that integrates seamlessly with the rest of the bottle. Overall, the combination of these materials supports a design that is durable, safe, and visually appealing for everyday use.”

Structure

Figure 9 illustrates the initial structural draft of the product concept. The sketches explore the basic 3-piece concept.

The design contains the main body, which serves as the holder for the water. The top, a removable cap. The cap design also ensures the product can be safely transported without leakage or contamination. A filter integrated at the nozzle is proposed, indicating that the product may be used for dispensing liquids while controlling flow or removing particles.

Last, the bottom part that holds all the components.

The sketch also demonstrated how the different parts are assembled together, showing connection points and structural alignments.

Overall, these drafts focus on defining the product's core structure, usability, and functional components before moving into more detailed design stages.

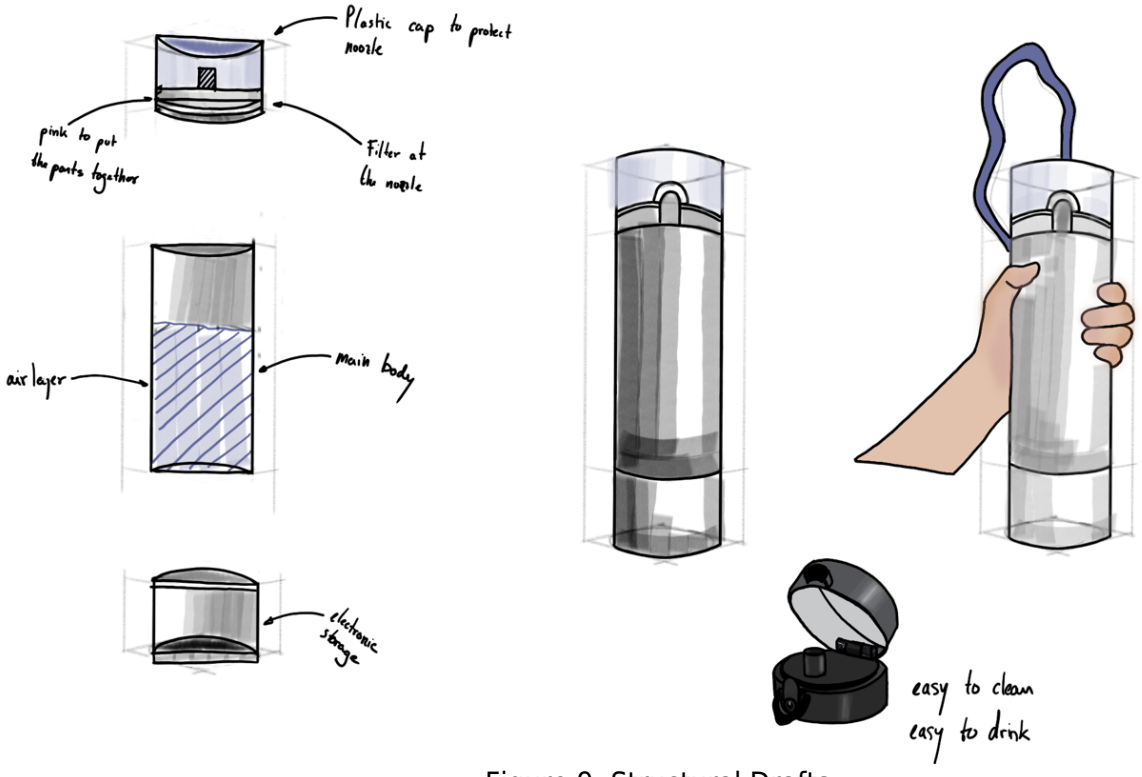

Figure 9: Structural Drafts

To ensure a coherent and professional presentation of the project, a visual identity was established for TRAQUA. This identity provides consistency across all project deliverables, including the report, presentation, and prototype interfaces.

Logo: The TRAQUA logo combines a water droplet icon with the project name, reflecting the system's core focus on water quality monitoring.

Tagline: *"Know your water, trust your bottle."* — summarizing the project's goal of giving users reliable, real-time insight into their water quality.

Color Palette (Monochrome): Three colors form the visual foundation:

- #6377BD — primary dark blue, used for headings and key UI elements
- #D0E3FF — light blue, used for backgrounds and secondary elements
- #FFF9F0 — off-white, used for neutral backgrounds and contrast

Typography: The project uses the **Big Shoulders** font family, chosen for its clean, technical aesthetic that aligns with the engineering nature of the product.

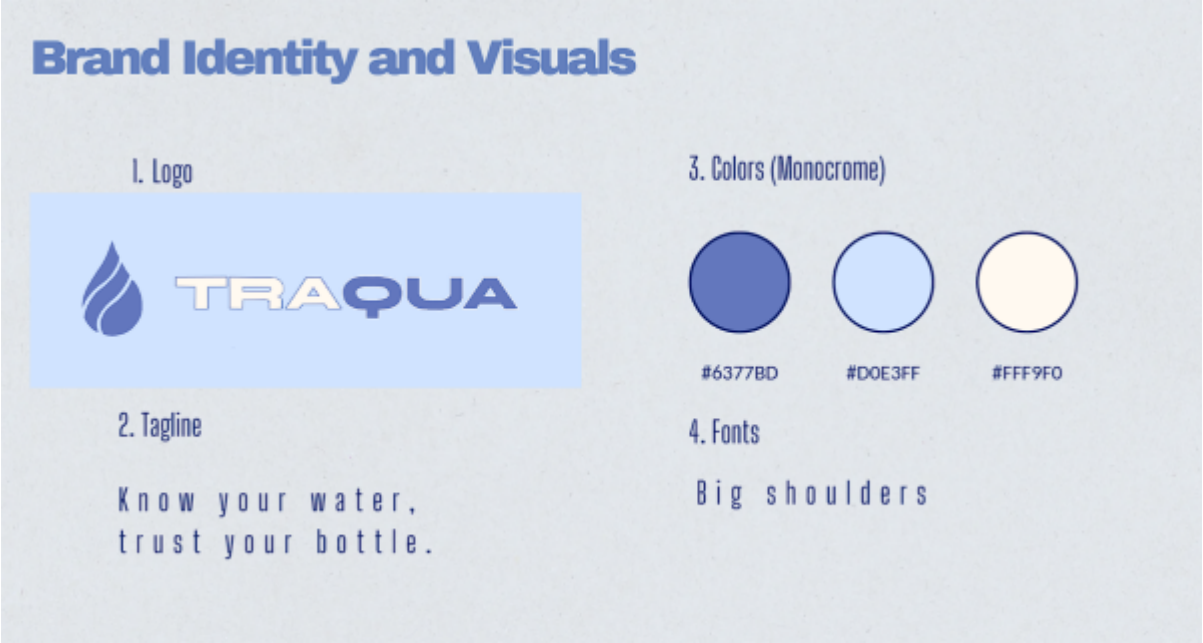

Figure 10: Branding

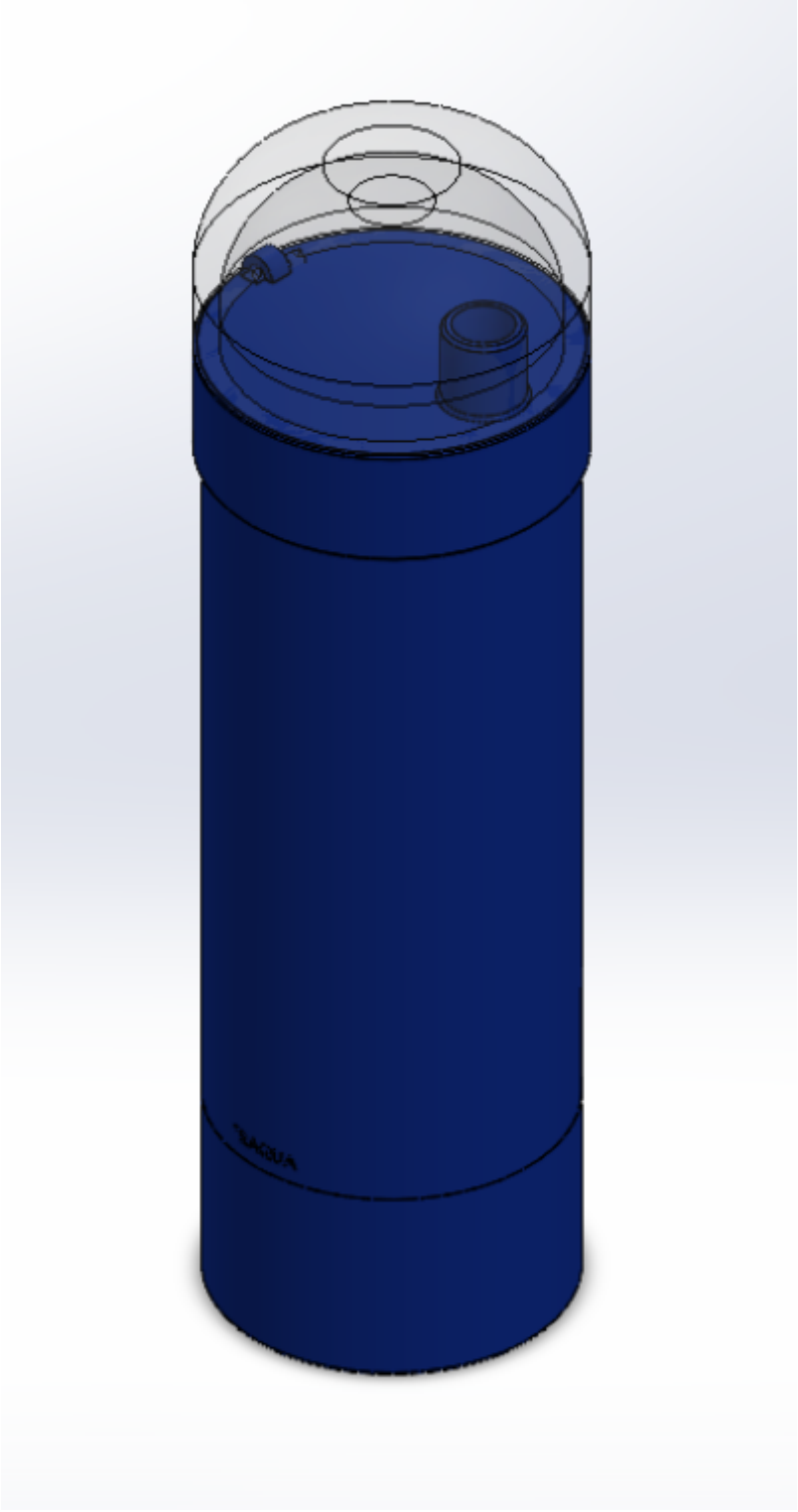

Figure 11: fullbody_3D


TRAQJA

Figure 12: fullbody_2_3D

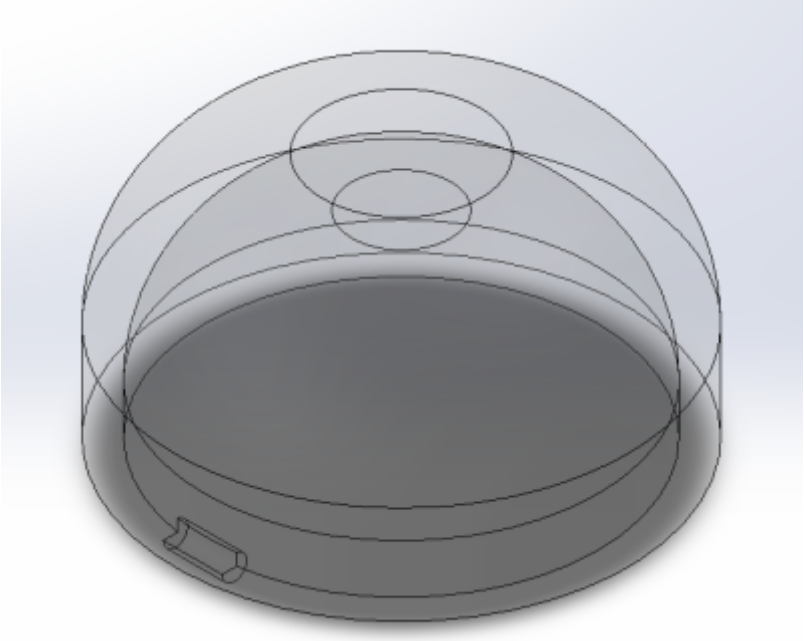

Figure 13: toppart_3D

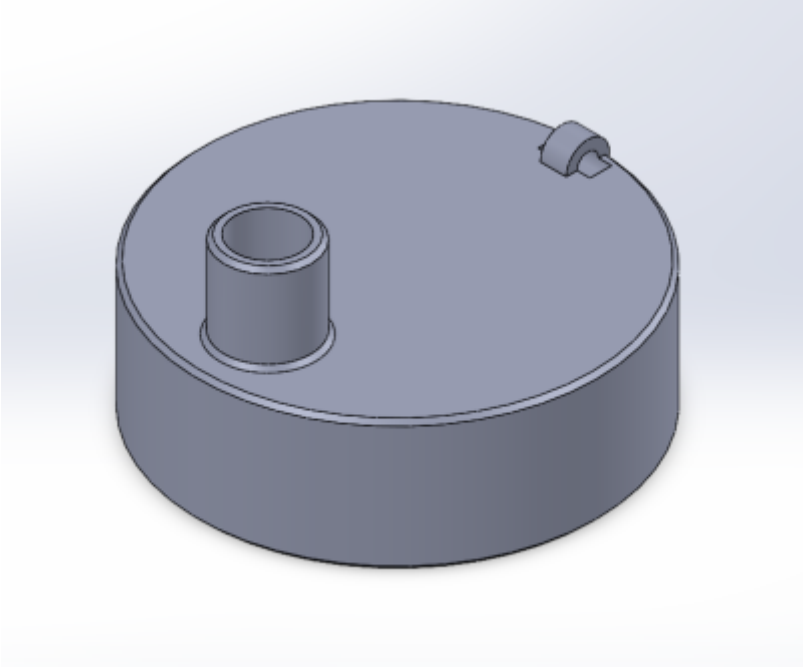

Figure 14: drinkpart_3D

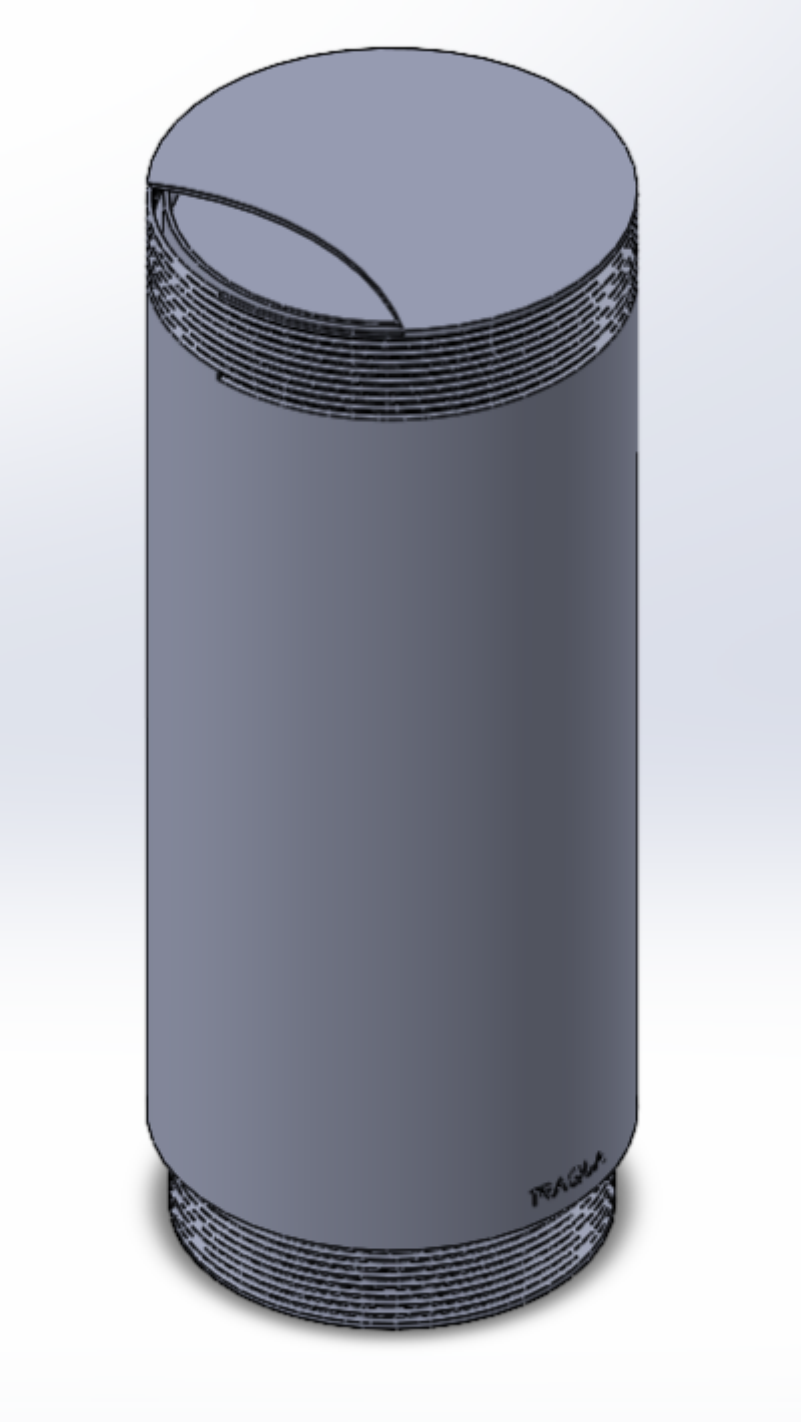

Figure 15: midpart_3D

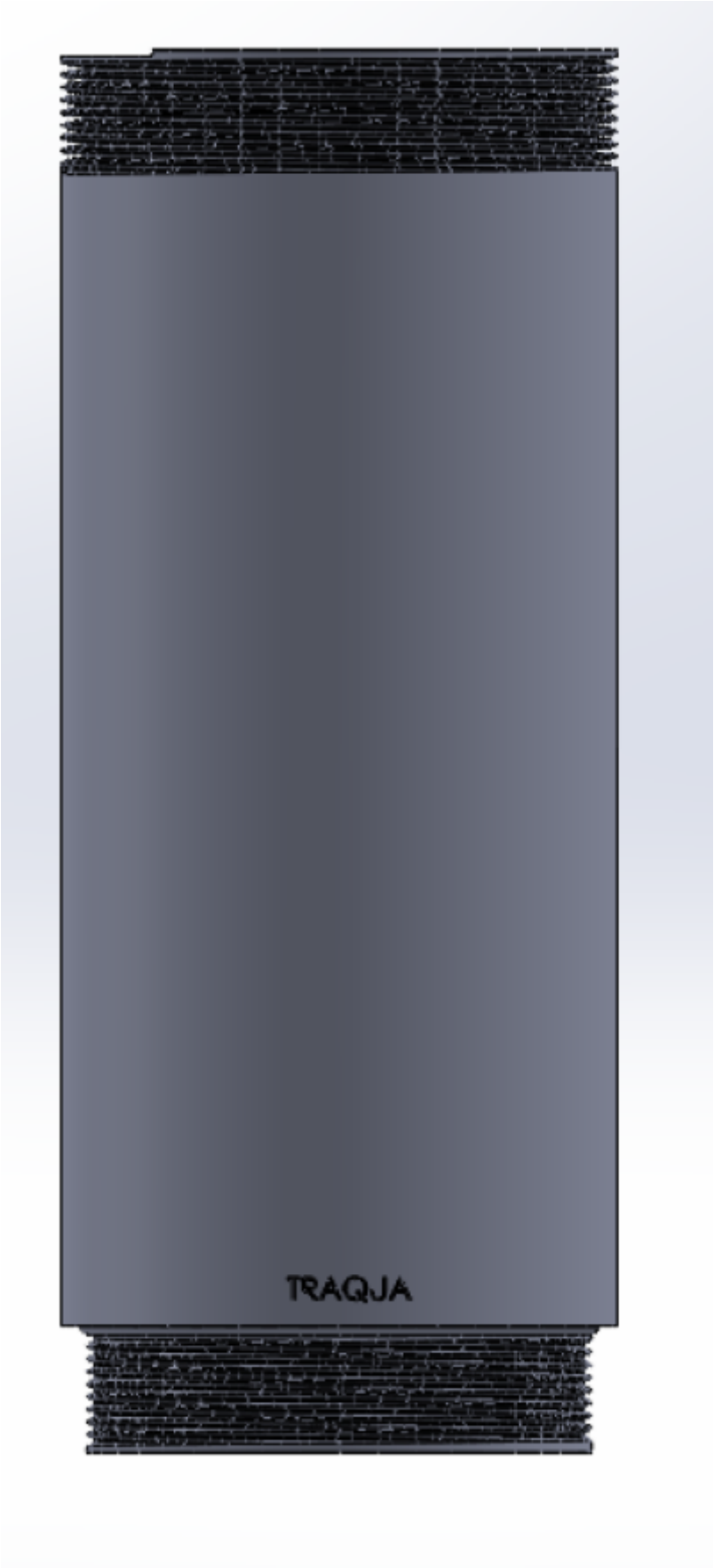

Figure 16: midpart_2_3D

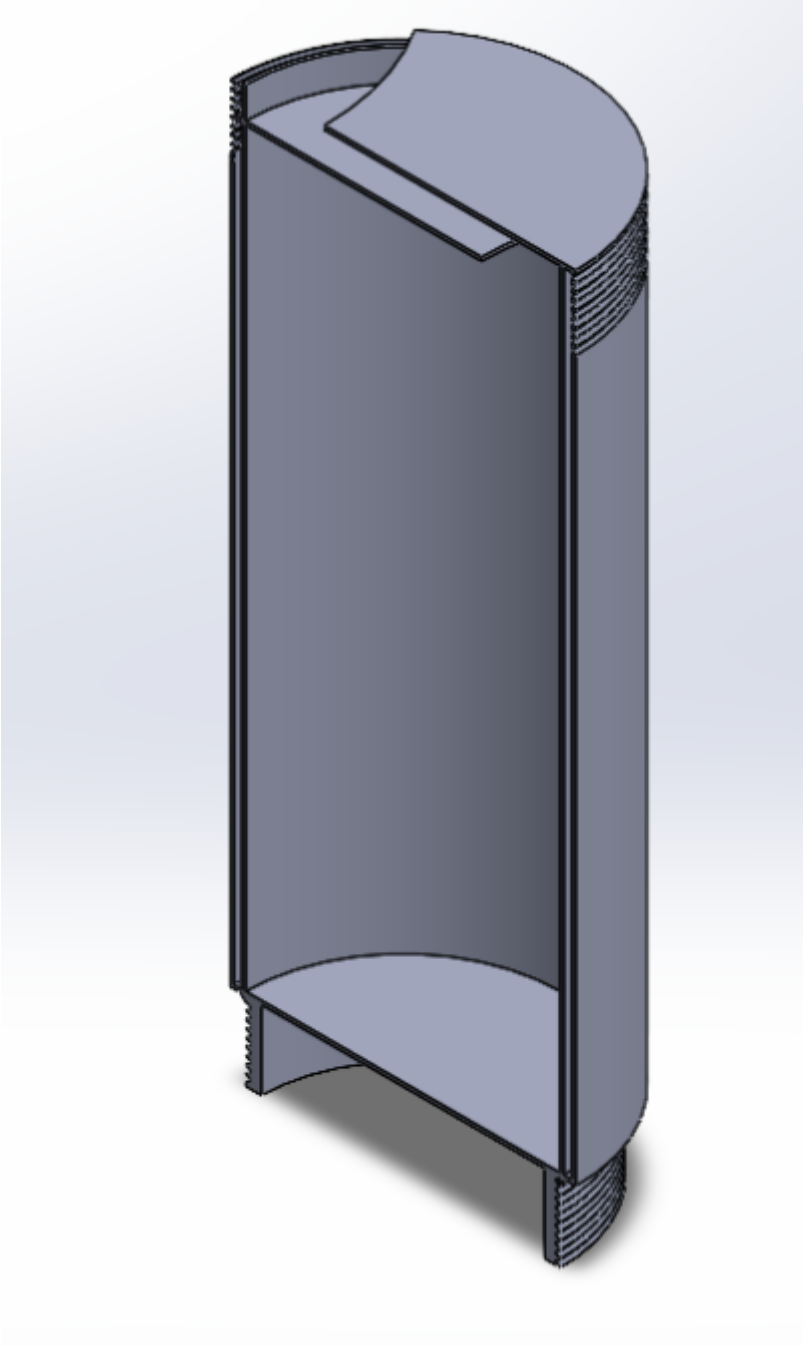

Figure 17: midpart_3_3D

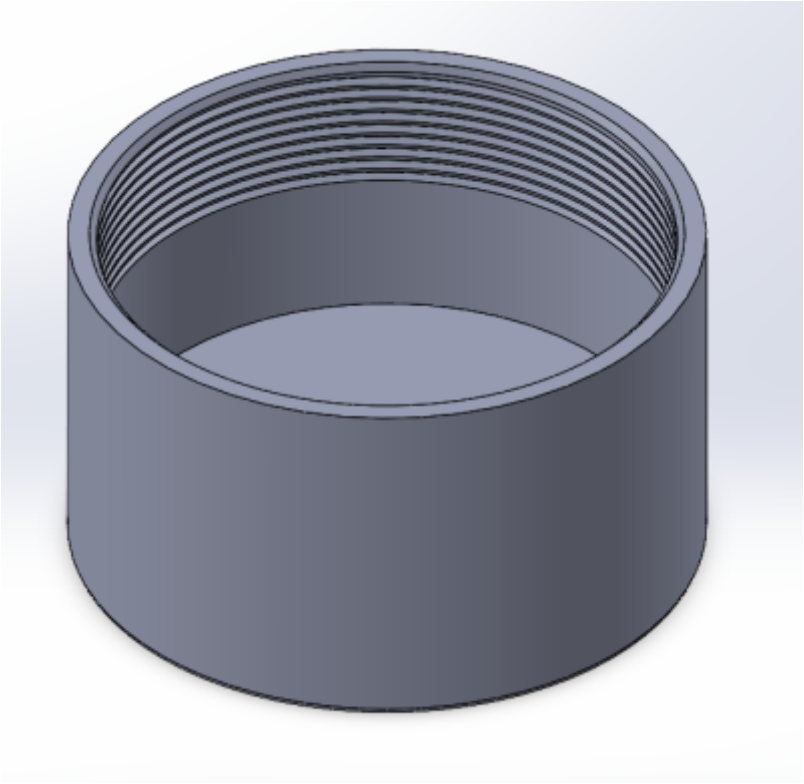

Figure 18: bottompart_3D

(*iv*) 3D model with load and stress analysis; (*v*) colour palette.

Smart System

Hardware

Figure 19 This is a system architecture/block diagram for a smart water bottle. At the core is an ESP32 DEVKIT V1 microcontroller that connects to several peripherals: an LED light, accelerometer (LIS3DHTR), TDS sensor (SEN0244), UV-C light, and a pressure sensor (FSR406) with a capacitor. Power is supplied by a battery through a charging circuit. The ESP32 communicates with a mobile application over BLE (encrypted + bonded) using notify/read/write operations. The legend color-codes the lines: red for energy, green for information, and blue for water-related connections.

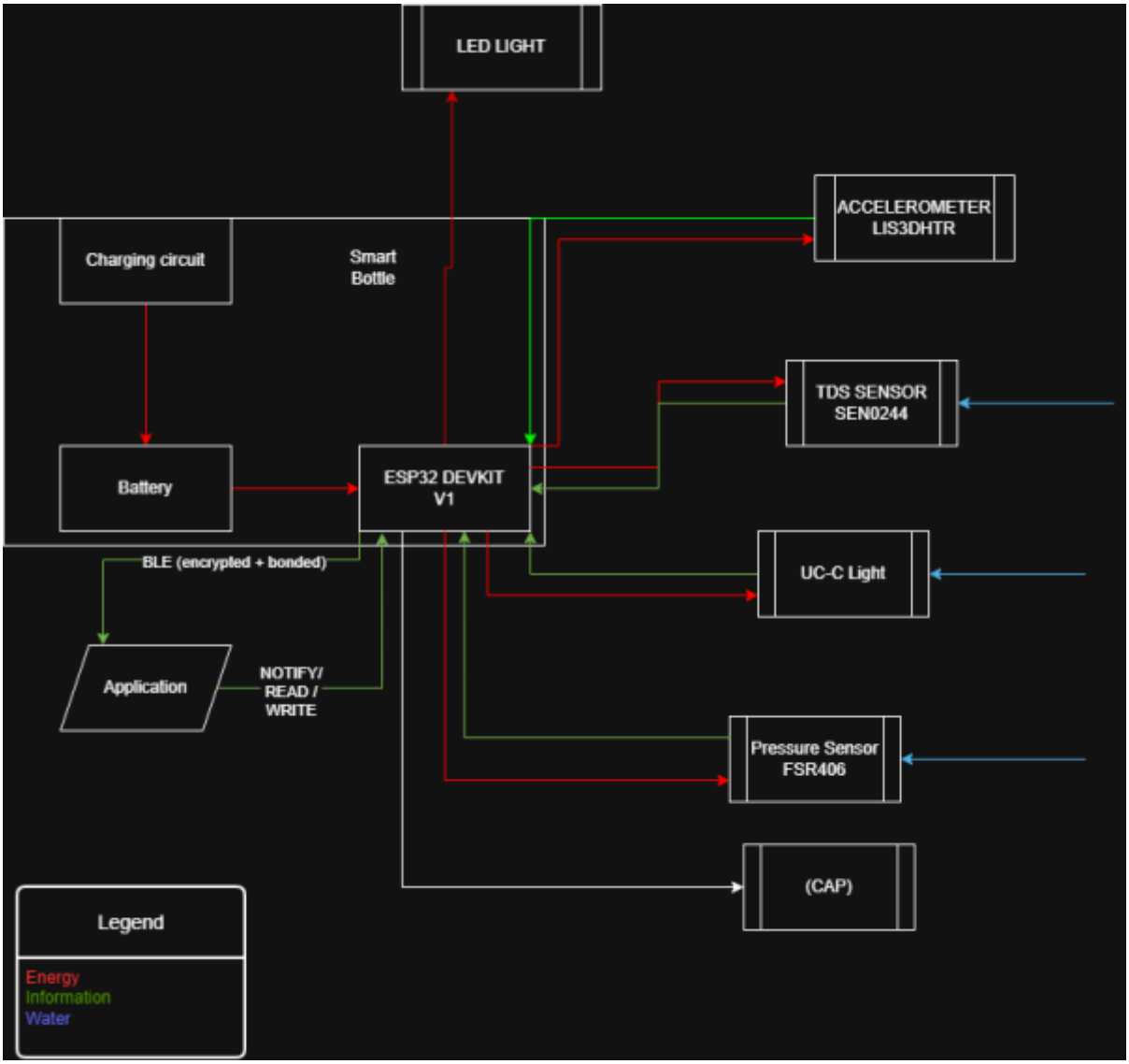

Figure 19: Black box diagram underling components

Figure 20 presents the initial electronic system architecture of the smart bottle prototype. The diagram outlines the integration of the main components and their interactions, serving as a first iteration of the system design. At the core of the system is the ESP32 microcontroller, which manages data processing and communication between all connected components. Several sensors are included to support the intended functionality of the product. A TDS sensor is used to measure water quality, while a temperature and humidity sensor (DHT11) monitors environmental conditions. Additionally, a motion sensor and a force-sensitive resistor (FSR) are incorporated to detect user interaction and usage patterns.

The system also includes a UV-C light module, controlled via a transistor circuit, which is intended for sterilization purposes. A display module (SSD1306) is connected via I²C to provide real-time feedback to the user.

Power is supplied through a battery management system (BMS) connected to multiple lithium-ion cells. Voltage regulation is handled by a boost converter and linear regulator to provide stable 5V and 3.3V outputs required by different components. A USB-C interface is included for charging.

It is important to note that this design represents an initial draft, created to explore component selection and system integration. Certain aspects, such as power efficiency, component optimization, and circuit simplification, will be revised in later iterations.

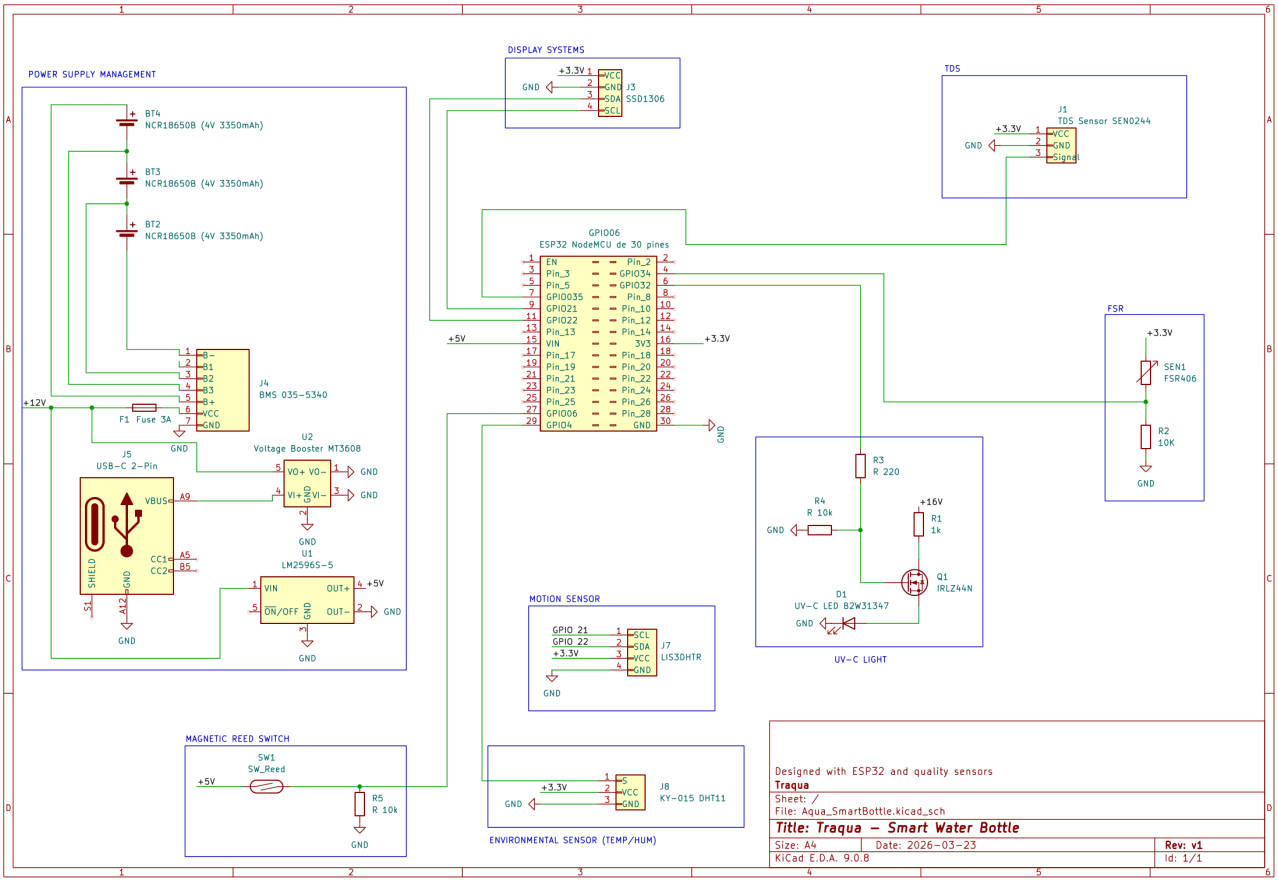

Figure 20: Detalied schematics

To ensure the system operates reliably, a power budget was established for all electronic components. The table below outlines the voltage, normal and maximum current draw, and resulting power consumption for each component 21.

POWER CALCULATIONS

EQUIPMENT	Voltage [V]	I_normal [A]	I_max [A]	P_normal [W]	P_max [W]
TDS sensor	3.3	0.003	0.006	0.0099	0.0198
Accelerometer	3.3	0.000002	0.000004	0.0000066	0.0000132
UV-C LED module	12	0.1	0.11	0.8	0.88
Pressure sensor	3.3	0.000264	0.00022	0.000726	0.000726
Temperature sensor	5	0.0001	0.0025	0.0005	0.0125
Microcontroller	5	0.08	0.24	0.4	1.2
TOTAL			0.358724	1.2111326	2.1130392

Figure 21: Power Budget table

Software

The following sub chapter will introduce and explain the reason why TRAQUA decided to program the application system the way it is, talk about its components and the tools used to do it.

Figure 22 we can see an easy entry flow of how the application works. The application was made with one key concept in mind. Keep most of the “important” values one click away. With that mindset in mind. The application works in the following way. If the users log in for the first time, they enter their height and weight. This will help us determine how much water that person needs to intake per day. Then with 1 click that person can see the water purity and his or her total water intake. He can also start the cleaning process from the app by clicking on the navigation menu on the bottom of the app.

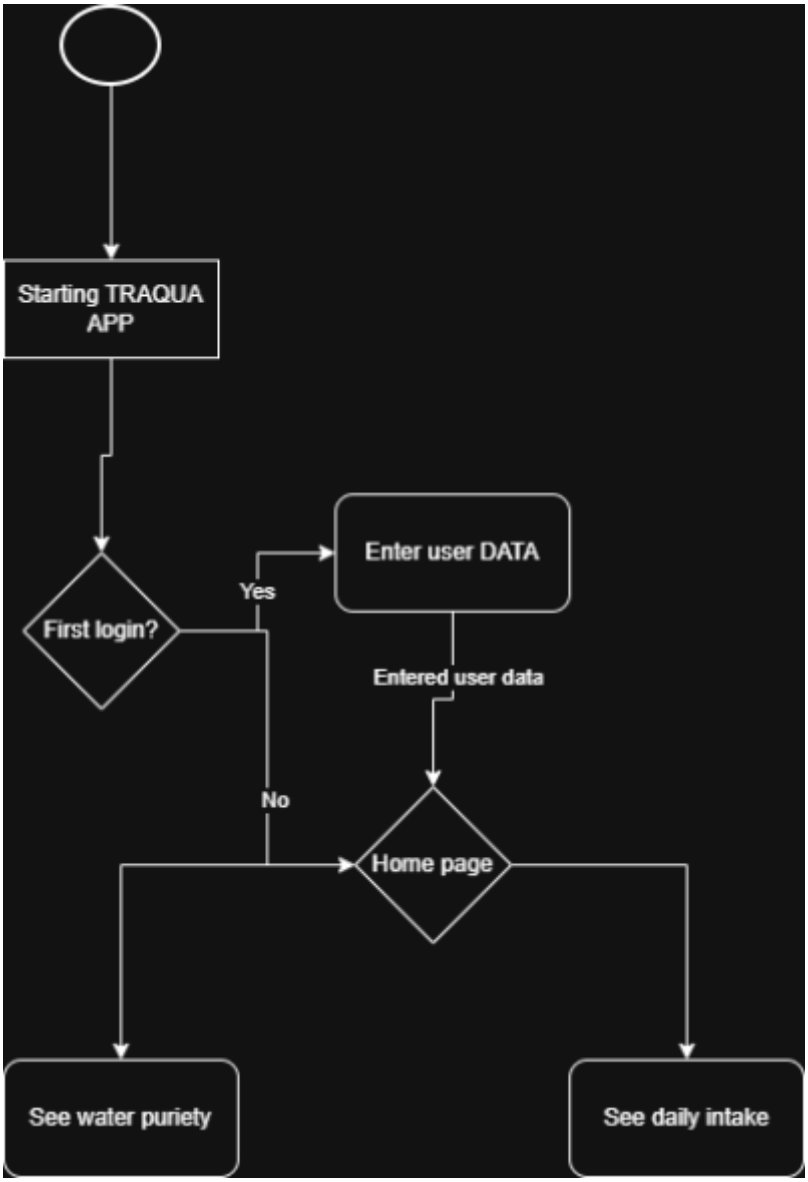

Figure 22: Use case Diagram of the system

Figure 23 This is a Level 1 Data Flow Diagram (DFD) for a hydration tracking system. It shows an external entity (Water bottle) sending raw readings to a Sensors process (1.0), which passes data to a Process Data process (1.5). That process sends processed data to a Data store, which feeds into a Display on app process (2.0) that delivers hydration info to the User external entity. There's also a legend in the bottom-left defining the DFD notation (rectangles for external entities, circles for processes, open rectangles for data stores).

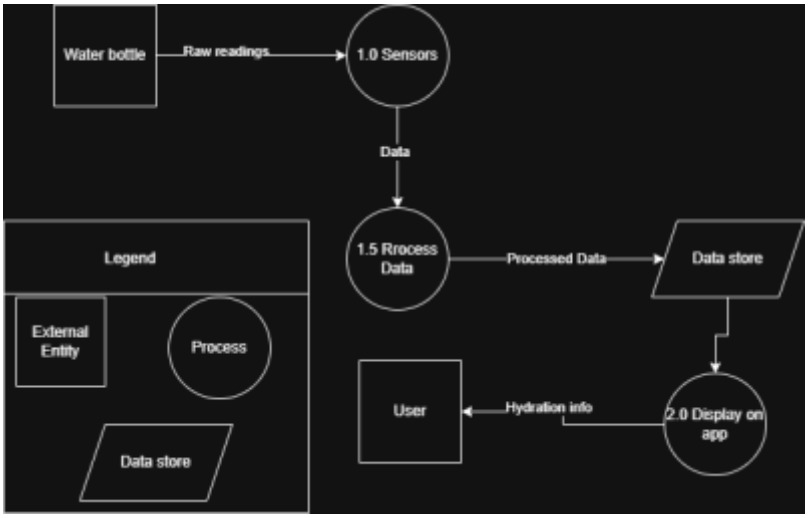

Figure 23: Level 1 DFD of the system

Software choices

The system consists of four main components. A mobile application, a web application a backend service and an embedded device based on ESP32

- Mobile Application

The application was developed using EXPO Go, a framework built on top of React Native. The choice was made due to its rapid development capacity, its cross-platform support (iOS and Android), its ease of testing without the requirement of complex native configuration, and the fewer risks of vulnerabilities that React has. Expo Go also enables efficient prototyping and provides access to device features such as Bluetooth, which is essential for communicating with the ESP32.

- Web application

Link to website: https://alvesbernardo.github.io/TRAQUA/

The shop of TRAQUA. This was done using SVELTE. SVELETE was chosen because of its lightweight nature, fast performance, and simple reactive model. Unlike traditional frameworks, Svelte compiles components at build time, resulting in highly optimized and efficient code

Figure 24, Figure 25, Figure 26, Figure 27, Figure 28 are screenshots of the website

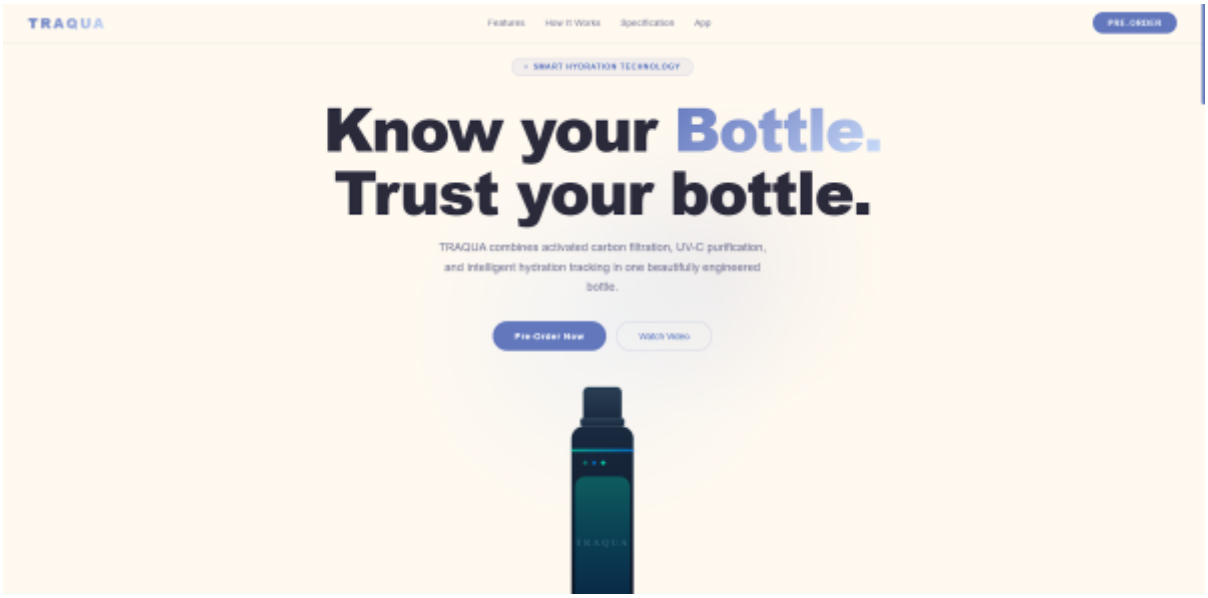

Figure 24: Website landing

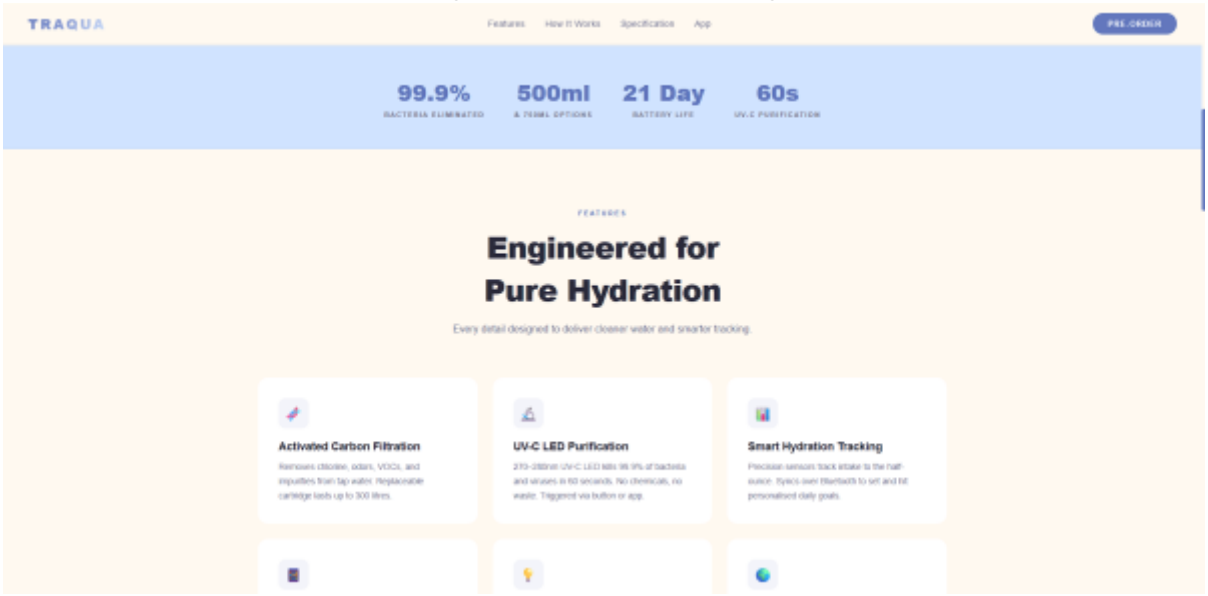

Figure 25: Website section 2

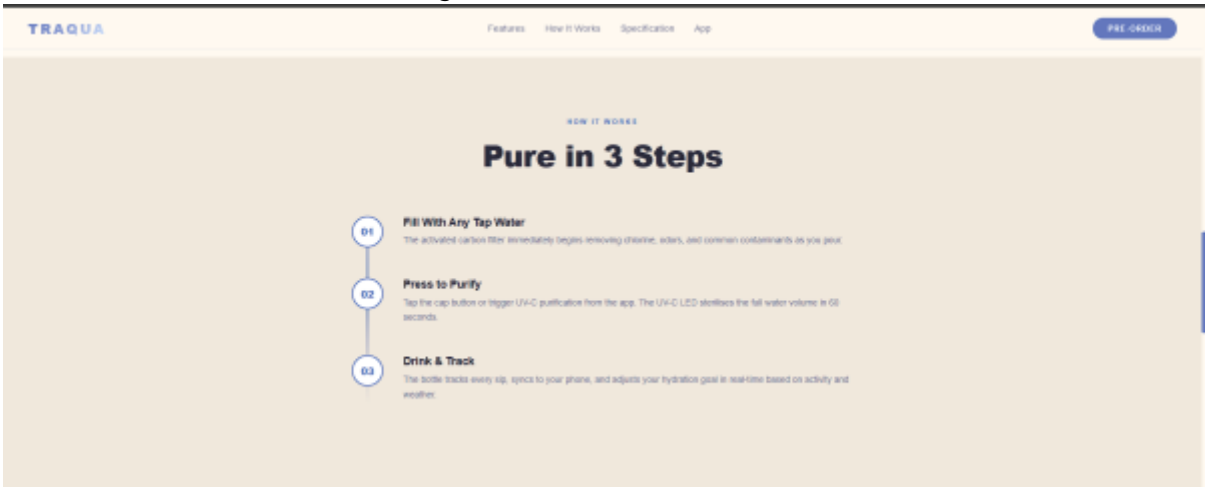

Figure 26: Website section 3

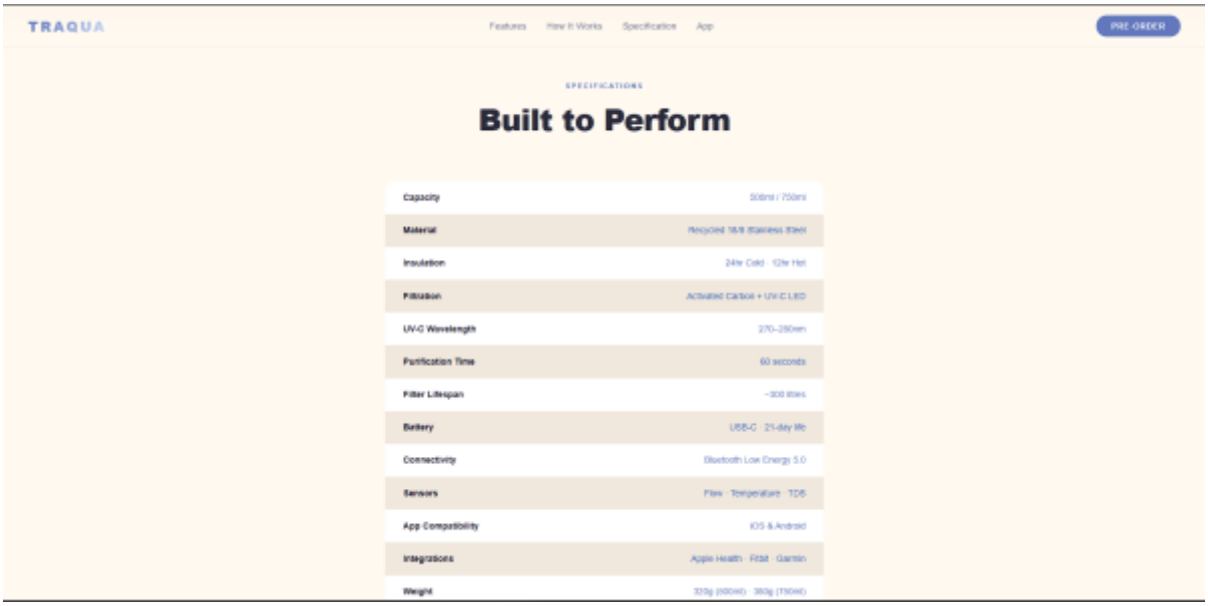

Figure 27: Website section 4

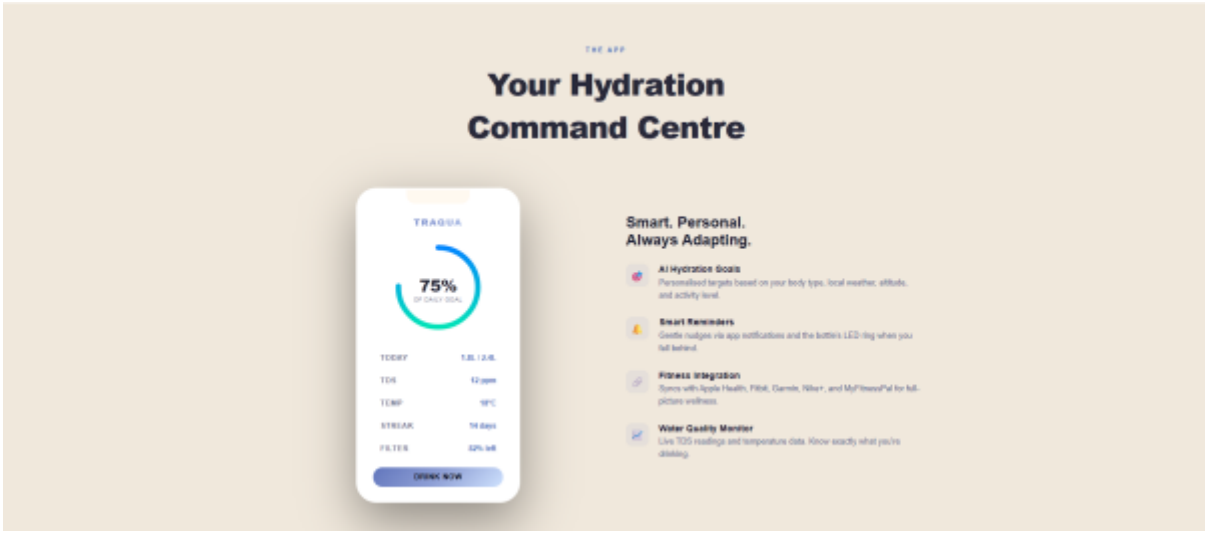

Figure 28: Website section five

- Backend

The backend is planned to be developed using Python in combination with FASTAPI. Due to its simplicity, readability and extensive ecosystem. The backend is responsible for handling business logic, data storage, and communication between the web application and other system components.

- ESP32 The system uses an ESP32 microcontroller, which is well-suited for IoT applications due to its low cost, built-in Wi-Fi and Bluetooth capabilities, and sufficient processing power. The ESP32 is responsible for interacting with hardware components and executing real-time tasks while communicating with the mobile application.
- Communication Communication between the mobile application and the ESP32 is established using Bluetooth Low Energy (BLE). BLE was selected because it is energy-efficient, widely supported, and ideal for short-range communication between devices
- Security Considerations

Security is an important aspect of the system, particularly in Bluetooth communication. To ensure secure data exchange, several measures are implemented:

Secure pairing and bonding between devices Encryption of Bluetooth communication to prevent interception Authentication mechanisms to restrict unauthorized access Input validation across all

components (mobile app, backend, and ESP32)

Additionally, secure development practices are followed in both the frontend and backend to minimize potential vulnerabilities.

Comparing Mobile framework

Table 1: Mobile Framework Comparison

Criteria	**Expo Go (React Native)**	**Flutter**	**Native (Swift/Kotlin)**
Cross-platform	iOS + Android from one codebase	iOS + Android from one codebase	Separate codebases needed
BLE Support	Yes (expo-ble / react-native-ble-plx)	Yes (flutter_blue)	Full native BLE APIs
Development Speed	Fast — hot reload, JS/TS	Fast — hot reload, Dart	Slower — two separate apps
Testing/Prototyping	Very easy — scan QR to test	Requires emulator or device build	Requires full build per platform
Community/Ecosystem	Large (React/JS ecosystem)	Growing rapidly	Mature but platform-specific
Learning Curve	Low (JS/TS knowledge)	Medium (Dart)	High (two languages)
Chosen	**Yes**	No	No

Web Framework

Table 2: Web Framework Comparison

Criteria	**Svelte**	**React**	**Vue**
Bundle Size	Very small (compile-time)	Larger (virtual DOM runtime)	Medium
Performance	High — no virtual DOM overhead	Good — virtual DOM diffing	Good
Learning Curve	Low — simple reactive model	Medium — JSX, hooks, state mgmt	Low-Medium
Build Output	Highly optimized vanilla JS	Requires runtime library	Requires runtime library
Community	Smaller but growing	Largest	Large
Chosen	**Yes**	No	No

Backend Framework

Table 3: Backend Framework Comparison

Criteria	**FastAPI (Python)**	**Django (Python)**	**Express (Node.js)**
Performance	High — async, built on Starlette	Moderate — synchronous by default	High — async, event-driven
API Development	Built for APIs — auto Swagger docs	Heavier (DRF needed)	Flexible but manual setup

Criteria	FastAPI (Python)	Django (Python)	Express (Node.js)
Learning Curve	Low	Medium (opinionated, ORM, admin)	Low
Data Validation	Built-in (Pydantic)	Manual or via serializers	Manual or via libraries
Ecosystem	Growing	Very mature	Very large (JS ecosystem)
Chosen	**Yes**	No	No

Microcontroller

Table 4: Microcontroller Comparison

Criteria	ESP32	Arduino Uno	Raspberry Pi Pico
Wi-Fi	Built-in	No (shield needed)	Pico W only
Bluetooth/BLE	Built-in	No (module needed)	Pico W has BLE
Processing Power	Dual-core 240 MHz	Single-core 16 MHz	Dual-core 133 MHz
Cost	Low (~5-8 €)	Low (~5 €)	Low (~4-6 €)
GPIO Pins	34	14 digital, 6 analog	26
IoT Suitability	Excellent — designed for IoT	Limited — no wireless	Good but less mature
Chosen	**Yes**	No	No

Below you can find a detailed use case explanation of how the user interacts with the system. Our user being the actor in this case. Figure 29 ...

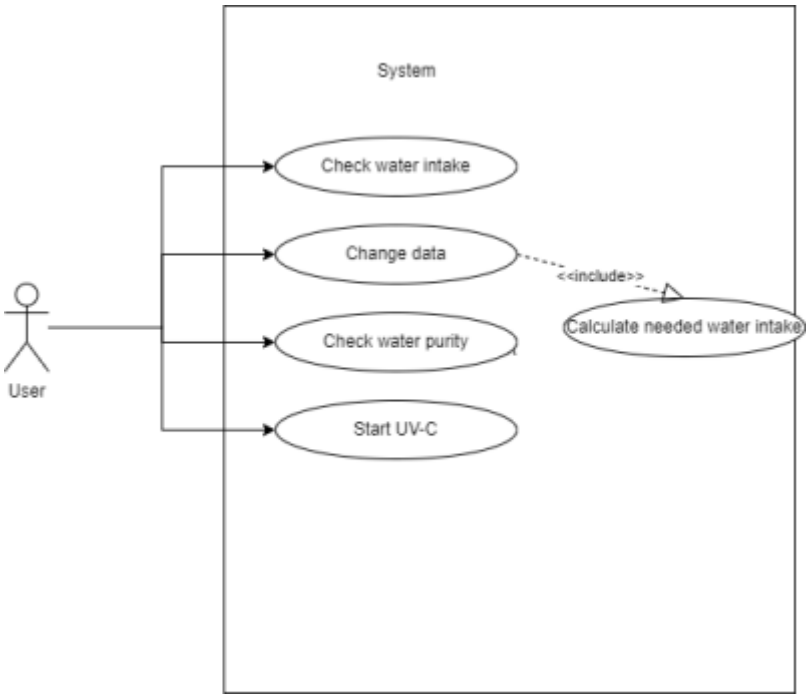

Figure 29: Detail use case

Packaging

Present and explain the: (*i*) initial packaging drafts; (*ii*) detailed drawings; (*iii*) 3D model with load and stress analysis, if applicable.

Prototype

For the prototype, we use a rigid plastic bottle as the main structure. This material is easy to obtain, lightweight, and resistant, making it suitable for testing our concept. Inside the bottle, we add aluminum foil sheets. The aluminum acts as a reflective surface, allowing the UV-C light emitted by the LED to be reflected throughout the interior of the bottle. This improves the distribution of the light and increases the efficiency of the water disinfection process. We also include a plastic separator at the bottom of the bottle to protect the electronic components. This separator isolates the battery, sensors, and LED from direct contact with the water, ensuring safety and proper functioning during testing. This simple prototype allows us to validate the concept and test the performance of the UV-C cleaning system before developing the final product.

Refer main changes in relation to the designed solution.

Structure

Detail and explain any changes made in relation to the designed solution, including structural downscaling, different materials, parts, etc.

Hardware

Detail and explain any change made in relation to the designed solution. In case there are changes regarding the hardware, present the detailed schematics of the prototype.

Software

Detail and explain any changes made in relation to the designed solution, including different software components, tools, platforms, etc.

The code developed for the prototype (smart device and apps) is described here using code flowcharts.

Tests & Results

Hardware tests

Perform the hardware tests specified in Tests. These results are usually presented in the form of tables with two columns: Functionality and Test Result (Pass/Fail).

Software tests

Software tests comprise: (i) functional tests regarding the identified use cases / user stories; (ii) performance tests regarding exchanged data volume, load and runtime (these tests are usually

repeated 10 times to determine the average and standard deviation results); (iii) usability tests according to the System Usability Scale.

Summary

Provide here the conclusions of this chapter and make the bridge to the next chapter.