Smart Water Intake Tracking System for Kids

BY

YONG YUAN HUAN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONOURS) COMPUTER ENGINEERING

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Information Technology (Honours) Computer Engineering at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

First, I would like to express my sincere thanks and appreciation to my supervisor, Dr. Teoh Shen Khang, for his valuable guidance and continuous support throughout the development of this project. His constructive feedback and insightful suggestion have been essential in helping me improve and stay on track with my progress.

My deepest appreciation also goes to my family and friends, who are giving me encouragement and unlimited support for completing this project.

ABSTRACT

Dehydration in kids can pose a significant health hazard, especially when it occurs amid situations such as while they are attending kindergarten or schools and their supervisors cannot supervise them all the time. This project presents the development of Smart Water Tracking System using an ESP32-based system integrated with a load sensor (HX711) to measure real-time water intake. The data is collected and store to Firebase cloud storage when internet connectivity is available, else it will store the data locally for offline use. The purpose of this storing technique is to ensure the accurate and continuous data collection in IoT applications. A user-friendly monitoring application that involving JavaScript, HTML and CSS to allows parents and caregivers to visualize daily, weekly and monthly drinking patterns, with features such as real-time hydration tracking and refill detection. By combining offline data synchronization and cloud services, the system provides a reliable tool to encourage better hydration habits in children and lessens the risk of dehydration that may pose threats within the education sector, with a solution at low cost which enables custom form factor.

Area of Study (Minimum 1 and Maximum 2): Internet of Things

Keywords (Minimum 5 and Maximum 10): Data Collection in IoT, Monitoring Application, ESP32-based system, Real-time hydration monitoring, Smart water tracking system

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

ml	millilitre
Oz	Fluid ounce
GHz	gigahertz
g	Gram
V	Volt
mAh	Milliampere-hour

LIST OF ABBREVIATIONS

JSON	JavaScript Object Notation
API	Application Programming Interface
CPU	Central Processing Unit
GPIO	General Purpose Input Output
IOT	Internet of Things
NTP	Network Time Protocol
LED	Light Emitting Diode
Wi-Fi	Wireless Fidelity
NVS	Non-volatile Storage
ESP32	Espressif 32-bit Microcontroller
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
SPIFFS	Serial Peripheral Interface Flash File System

Chapter 1

Introduction

In this chapter, the background and motivation of the research are presented, along with the contribution to the field, and the outline of the thesis. The point to focus on this research is the design and the development of Smart Water Tracking System for kids, which is an assistant system that aims at the challenge to ensure the children have proper hydration. Hydration is crucial for maintaining children's health, blood circulation and improving cognitive abilities [1]. However, especially in young children, it is a challenge to monitor and manage their water intake due to their low awareness of hydration and the environment that they spend much more time of their day, such as school or playground.

Current research increasingly highlights the negative effects that dehydration can have on cognitive and physical development, making hydration monitoring more important than ever. With the rapid development of Internet of Things (IoT) technology, more people are using smart devices to monitor and track daily habits related to health and lifestyle. By integrating sensors, data collection methodologies and mobile applications, the IoT-based solutions can provide real-time feedback and encourage healthier behaviour. This potential makes IoT an effective approach for developing a smart tracking system that will help parents or caregivers in monitoring children's daily water intake, which tries to minimize the risk of dehydration.

1.1 Problem Statement and Motivation

Problem Statement

Nowadays, recent tragedies from different parts of the world highlight the severe consequences of dehydration and heatstroke among children. In April 2023, there are two children in Kelantan, Malaysia, had lost their lives due to heat-related illness, which an 11-year-old boy die to heatstroke after severe dehydration, and a 19-month-old girl also lost her life from severe dehydration with underlying species. Besides that, in July 2024, three children in Arizona, USA, died following heat-related emergencies. [2,3] These cases already stated that the critical need for an effective water intake monitoring system to protect children from heat-related illnesses, especially as global temperature continue to rise due to climate change. Children are particularly vulnerable due to their body size, cannot express their feeling, thinner skin and

also relatively weak immune system, which making them hard to regulate their body temperature.[4] These cases suggest that many of the children either didn't meet their daily hydration need or at the risk of drinking too much water, both will lead to potential health risks. Without effective monitoring, many children are still at risk of becoming dehydrated, which can affect their overall health.

Motivation

Maintaining good mental and physical health of children is a top priority for parents and caregivers. By achieving this, ensured proper hydration is an essential part of this, but monitoring children's water intake can be challenging. This is because children are often in an environment where they are less supervised, such as schools, making it a challenge to accurately monitor their water intake.[5] Children's low awareness of staying hydrated and inability to recognize early signs of dehydration further make monitoring their water intake more challenging. Recent incidents related to dehydration illnesses and the increasing of temperature global have highlighted the urgent need for effective solutions to monitor and ensure enough water intake in children.

Traditional ways such as reminders or manual checks often lack real-time monitoring features and accuracy. To overcome this limitation, this thesis proposes a system that involves load sensors and IoT technologies to provide a highly accurate and real-time monitoring of children's water intake. Such a system can help parents and caregivers ensure that children had meet their hydration requirements consistently, reducing the risk of dehydration and promoting better cognitive and physical development.

1.2 Objectives

The project aims to design and implement a Smart Water Tracking System for the purpose of promoting appropriate hydration among children, especially in environments where direct supervision is limited such as kindergarten and playground. From this primary objective, several sub-objectives can be derived so that the development of the device can be guided. Using IoT technology, the system will accurately track water intake in real-time and send the data to a website application so that parents or caregivers can make sure the children have proper hydration throughout the whole day at school.

The first sub-objective is to develop a device that can be attached to water bottles used by children. This attachment will use a load sensor to measure the water consumed and wirelessly transmit the consumption details via Wi-Fi to a website application. The application will display the hydration status of the children, allowing educators to monitor water intake effectively. Furthermore, the device will be designed to be portable, incorporating a power storage system in order to support full day usage of the device. This flexibility is designed to solve the compatibility problem of previous smart water bottles, since not all bottle sizes are the same.

Furthermore, the second sub-objective is to achieve accurate and stable monitoring and therefore the system will use a high-precision load sensor together with advanced data processing algorithms. This will ensure reproducibility and accuracy of the measurement, as well as minimizing errors, ensuring the system will provide reliable data for both real-time monitoring and long-term analysis.

The third sub-objective of this project is to simplify the process of monitoring children's water intake for caregivers and parents. This will be achieved by developing an easy-to-use website application that displays real-time hydration data. The system will be designed to be user-friendly, enabling the user with minimal technical expertise to easily track and manage children's hydration effectively.

1.3 Project Scope and Direction

This project is to deliver a smart water tracking system which is specially designed for monitoring and encouraging proper hydration for children. This system consists of both hardware and software part.

Hardware Scope

For the weighting part, the hardware includes an ESP32-based embedded system, a 10kg load sensor, HX711 amplifier module and ADXL345 accelerometer. The ADXL345 was used to determine whether the water bottle is standing or lying down. If it is lying down, the weighting process will be suspended. Otherwise, the system will start to detect weight changes in the water bottle to do estimation for water consumption. To ensure accurate reading, the load

sensor will undergo a calibration process at the initialization. After measurement, the raw data will the send to Firebase RealTime Database.

For the power storage system, the hardware involves a single-cell 18650 lithium battery, battery holder, voltage booster, voltage sensor , a red LED indicator and TP4056 charging module. The TP4056 sensor manages charging when connected to a micro-USB cable. Then TP4056 will pass the voltage to the voltage booster to boost the voltage from 3.7V to 6V so that it can supply enough power for the ESP32. The purpose of voltage sensor is to detect the battery voltage level and send the information to web applications, allowing users to track the ESP32 power status in real-time. If the battery voltage is too low, the red LED will be blink 3 times to informed users.

Software Scope

The software components consist of two major parts. The first is the firmware running on the ESP32. The firmware is responsible for sensor reading, reading stability and reliability, Wi-Fi handling, NTP timestamping, local storage using the Preferences library and data synchronization with Firebase.

Another major part is the web application. The web application will be developed with a combination of HTML, CSS and JavaScript programming languages. It is responsible for retrieving and processing the raw data from the database then displaying it in both graphical and numerical formats. A line chart will be used to visualize the daily water intake, meanwhile the weekly and monthly water intake consumption will be shown in bar chart. In addition, the application will also provide a summary metrics such as total refill times, total water intake, average consumption and progress to achieve hydration goals. The interface will be designed to be user-friendly and easy-to-use, allowing users with minimal technical expertise to use it effectively.

System Features

The system also supports both offline and online data collection. When the data is collected offline (not connected to Wi-Fi), the data will be stored into local storage inside ESP32 with estimated timestamp and automatically sync the data to Firebase once the connection has been

restored. This offline ensures that the data can be recorded in continuous and reliable water intake tracking.

The project demonstrates a practical application of IoT in personal health monitoring and offered a scalable model for future improvements, such as designing a mobile application or implementing more health-related sensors.

1.4 Contributions

The experiment and analysis confirm the feasibility of the proposal for Smart Water Intake Tracking System for ensured adequate hydration among children. Firstly, the system provides Automated Hydration Monitoring, where load sensors and wireless communication are used to ensure the data is accurate and in real time. This will eliminates the need for frequent manual checking and reminders. Secondly, it supports Health improveme**nt**, since children are kept in a way of well-hydrated, which is very important for children's mental and physical development, especially in hot climate or during outdoor activities. Thirdly, the design put a great emphasis on Scalability and Practicall**y**, enabling the system's easy rollout for individual and group of children, like in classrooms, thus ensured effectiveness under different scenario and usage environment. Lastly, the system includes an Offline Data Feature, where data can still be collected without Wi-Fi connection and store in local storage, then automatically sync the offline data to the database once the connection had restored, ensured continuous and reliable hydration tracking.

1.5 Report Organization

This report is organised into 7 chapters: Chapter 1 Introduction, Chapter 2 Literature Review, Chapter 3 System Methodology, Chapter 4 System Design, Chapter 5 System Implementation , Chapter 6 System Evaluation and Discussion, Chapter 7 Conclusion. The first chapter is the introduction of this project which includes problem statement, project background and motivation, project scope, project objectives, project contribution and report organisation. The second chapter will presenting some literature review about previous works on other smart water tracking devices and existing technologies that these smart water bottle uses such as hydration tracking system, embedded IoT devices, sensor that use to record the weight and other functionality that is introduced in those products. The third chapter will describe the

system methodology, where the overall approach and system model are introduced. This will be including the system architecture, use case diagrams and activity diagrams to illustrate how the user interacts with it and how the function will work. In chapter 4 it will focuses on system design by providing the details of the system block diagram, hardware component specifications, circuit design and the interaction between different system components. The goal of this chapter is to ensure that the prototype can be duplicated by following the given detail. The chapter 5 cover the system implementation, which describes the setup of hardware and software, configuration steps, system operation with screenshots and the issues or challenges that had been encountered during the development. The chapter 6 discuss the system evaluation and results, including test procedures, performance metrics, experiment results, challenged faced and evaluation of whether the objective have been achieved. The last chapter provides the conclusion and recommendations by summarizing the project outcomes, contribution and limitations, followed by suggestion for possible improvement to fix the problem and future development.

Chapter 2

Literature Review

2.1 Review of the Technologies

2.1.1 Hardware Platform

Microcontroller Selection

The ESP32 microcontroll**er** was selected as the main processing unit for this project. Below is the specification of the ESP32 microcontroller:

Table 2.1 Specification of ESP32 microcontroller

Description	Specifications
Model	Arduino® Nano ESP32
Microcontroller	u-blox® NORA-W106 (ESP32-S3)
Processor	Xtensa® Dual-Core 32bit LX7 Microprocessor
Connectivity	Wi-Fi® 4 IEEE 802.11 standards b/g/n Bluetooth® LE v5.0
Memory	512kB SRAM

Compared to other microcontrollers such as Arduino UNO, Raspberry Pi and other ESP series, the ESP32 provides several advantages. First, the ESP32 provided a built-in Wi-Fi and Bluetooth module, so it eliminates the need of external module in order to do IoT. This feature can reduce hardware complexity, reduce space requirement and lowers cost compared to Arduino Mega or Raspberry Pi 2 Model B which required external Wi-Fi module. Furthermore, the ESP32 microcontroller delivers dual-core processing with higher clock speed, which was useful for doing multitasking such as handling sensor-reading, Wi-Fi communication and offline storage simultaneously. ESP32 also supporting various development platforms that using different SDKs and programming languages including Arduino IDE, MicroPython and Mongoose OS, which allow developer to have more option while developing and save time. Its large community support and existing libraries further simplify the integration process to with sensors or database such as Firebase [6]. These features make the ESP32 an efficient and practical choice for building IoT based hydration monitoring system. The below figure 2.1.1 shows the ESP32 microcontroller.

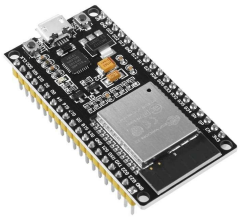


Figure 2.1.1 ESP32 microcontroller

Sensors

In this project was used a 10kg load cell combined with the HX711 amplifier module. HX711 was selected because it provided high sensitivity and precision for weight measurement. The HX711 also provided a 24-bit ADC, ensured that even small weight changes were captured accurately. Compared to other analog-to-digital conversion setups, this combination offered a cost-effective and stable reading with minimal noise. Figure 2.1.2 shows the 10kg load sensor, while figure 2.1.3 shows the HX711 amplifier module.

Figure 2.1.2 10KG load sensor

Figure 2.1.3 HX711 amplifier

Furthermore, an ADXL345 accelerometer was included to detect the orientation of the water bottle. This prevent false weight readings when the bottle is lying down. While other option accelerometer sensors such as MPU6050, the ADXL345 was selected because the project only required accelerometer data and does not need gyroscopic measurement. The ADXL345 was lightweight, low cost and consumed less power, making it an efficient choice for orientation detection without requiring complex processing. Figure 2.1.4 illustrates the ADXL345 module.

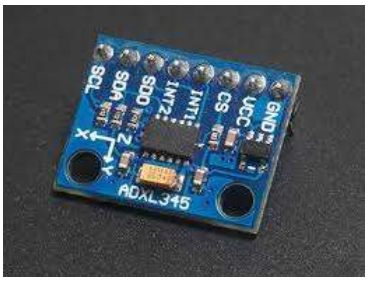


Figure 2.1.4 ADXL345

For power management, the project used a single-cell 18650 lithium-ion battery due to its rechargeability and long cycle life. Furthermore, the TP4056 charging module was employed as it supported safe recharging batteries that does not exceed 4.2V, and it provided built-in overcharge and discharge protection. A voltage booster was used in this project is to step up the 3.7V battery output to 6V required by the ESP32 [7]. Finally, a voltage sensor and a red LED indicator were included in this project to monitor the battery status, ensured that users are informed when the system required charging. Figure 2.1.5 shows TP4056 charging module, figure 2.1.6 shows voltage booster and figure 2.1.7 shows voltage sensor.

Figure 2.1.5 TP4056 module

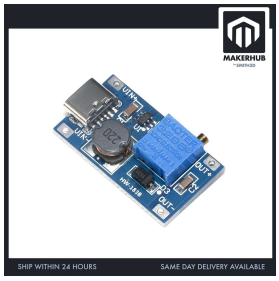


Figure 2.1.6 Voltage Booster

Figure 2.1.7 Voltage Sensor

2.1.2 Firmware/OS

The firmware for this project was developed using the Arduino IDE, which was chosen for its simplicity, availability of open-source libraries and compatibility with the ESP32 platform. The Arduino IDE provided a straightforward environment for coding, debugging, compiling and flashing the firmware to the ESP32. This firmware was responsible for sensor reading, Wi-Fi communication, timestamp assignment and local data storage. Furthermore, Arduino IDE had provided an extensive library system that created by the community, this open-source library system eliminates the need to build low-level drivers from scratch and had saved a lot of time. The key library included in this project were the HX711 Arduino Library by Bogdan Necula for load-cell interfacing, the Adafruit ADXL345 library for accelerometer processing and Firebase Arduino Client Library develop by Mobizt for uploading data to the Firebase Realtime Database. Inside the Arduino IDE, it also included some preset library such as Wi-Fi and Preferences library. The preferences library was used to store the data temporarily in the local storage during offline.

The frontend of the system was developed using Visual Studio Code (VS Code). This development platform was selected due to its flexibility where support for multiple programming languages, and strong integration with modern web development tools. In this project, HTML, CSS and JavaScript were used for the frontend development. HTML was used for the page structure, CSS for styling and layout and JavaScript handle backend tasks such as handling for data processing and fetching data from database. VS Code provided an efficient workspace where it can manage three programming languages simultaneously within a single project folder, it also provided features such as built-in debugging tools, syntax highlighting and live server preview features. These features allowed efficient testing of how the web application would look and behave in real time. In addition, VS Code supported integration with external online web services such as gstatic and chartjs**,** which allowed smooth communication with Firebase and implement chart for a better visualization. Figure 2.1.8 showed the icon of Arduino IDE while figure 2.1.9 showed the icon of Visual Studio Code.

Figure 2.1.8 Visual Studio Code

Figure 2.1.9 Arduino IDE

2.1.3 Database

The database that used for this project was Firebase Realtime Database, which was selected due to its scalability and real-time synchronization features. Firebase Realtime Database is a NoSQL cloud hosted database developed by Google. It stored the data in JSON format, enabling fast data retrieval and update. This structure was suitable for the project because the data collected, such as water intake amounts, timestamp and voltage reading, it can be organized by sorted into key-value paired.

One of the main reasons for choosing Firebase was its ability to provide real-time synchronization across devices. As soon as the ESP32 captured the data and uploaded to Firebase, the frontend was able to fetch the updated value instantly. This was particularly important for ensuring the user could monitor their children's hydration progress without noticeable delay. Another advantage of this database was its ease of integration with the ESP32 and frontend. On the ESP32 microcontroller side, the Firebase Client Library was used to establish a stable connection with Firebase, enabling read and write operation. On the frontend side, Firebase provided direct support through its JavaScript SDK, allowing data fetching and visualization on the web application.

Firebase was also chosen due to its low maintenance requirement. Unlike MySQL or MongoDB, Firebase Realtime database was a fully cloud-hosted server. This eliminated the need to configure server, manage or maintain SQL queries. This reduced the project complexity while still kept data secure and efficient data management. Furthermore, Firebase offered free usage tiers, which was sufficient for this academic project without adding more budget for sever hosting. Figure 2.1.10 shows the Firebase Realtime Database icon [8].

Figure 2.1.10 Firebase Realtime Database

2.1.4 Algorithms

Several algorithms were applied in this project to ensure accurate measurement, reliability and useability of the Smart Water Tracking System. The primary algorithm was the water intake detection algorithms, which determined drinking event by comparing the weight changes captured by the load sensor. A small weight changes that causes by the noise were filtered out, while significant changes between the current weight and previous weight were recorded as water intake or refill event. This method was chosen because it provided a simple and effective way to figure out the valid drinking action from random disturbances.

The orientation detection algorithm was implemented using ADXL345 accelerometer. By checking the orientation of the water bottle, the system was able to prevent false reading when the bottle was tilted or lying down. Furthermore, it will included a stable reading verification to ensure the weight changes was caused by movement will not recorded as valid drinking event. This ensured that only stable position will counted as valid measurement, improving the reliability of the water intake data.

An offline data synchronization algorithm was also included to address connectivity issues. When Wi-Fi was unavailable, the ESP32 temporarily store the water intake record in local storage using the Preferences library. Once connectivity was restored, the stored data was uploaded to Firebase along with correct timestamp. This ensured the data continuity and minimized the risk of data lost.

Lastly, data visualization and analysis algorithms were used on the frontend. JavaScript will be responsible to process the raw data retrieved from Firebase, analysed and sorted them into daily, weekly and monthly summaries and generated charts using Chart.js. This allowed parents and caregivers to clearly monitor hydration patterns over time.

2.1.5 Summary of Technologies Review

In summary, the technologies selected for this project were carefully chosen to balance cost-effectiveness, data accuracy and ease of implementation. The ESP32 microcontroller served as the core processing unit because of its built-in Wi-Fi and Bluetooth module, dual-core performance and wide support from the Arduino ecosystem.

For sensing, a 10kg load cell with HX711 amplifier was selected due to its high sensitivity and precision in weight measurement, while the ADXL345 accelerometer was used to detect water bottle orientation and prevent false reading. The combination of these three sensors provided reliable data for estimating water intake.

On the software side, Arduino IDE was chosen for firmware development due to its simplicity, large community library support and smooth integration with ESP32. Meanwhile, Visual Studio Code was used for frontend development as it offered flexibility, support multiple programming languages and productivity features such as live server and debugging.

For data storage and synchronization, Firebase Realtime Database was selected due to its scalability, low maintenance requirement and real-time synchronization features. This ensured that the data collected by the ESP32 can transferred to the web application without delays.

Finally, supporting algorithms such as water intake detection, orientation checking, offline data synchronization and data visualization were integrated to ensure system reliability, accuracy and usability. With the combination of these algorithms, the smart water tracking system was able provided a huge support in both real-time and offline hydration monitoring for children.

2.2 Review of Existing System

2.2.1 HidrateSpark Pro 21Oz

The HidrateSpark Pro 21Oz smart water bottle has implemented several advance technologies to track and manage water intake effectively. The most important feature of this smart water bottle is using SipSense technology, which is developed by HidrateSpark themselves, it is a precise method of measuring water consumption cased on weight. A Bluetooth technology is also used in this smart water bottle to upload the water intake data to HidrateSpark application every time when the bottle is within range of the phone, allowing user to monitor their water intake in real time.[11]

The SipSense technology is using weight-based measurement, which commonly referred to as a load sensor, to track every sip taken, providing a highly accurate reading to the amount of water consumed. This sensor system can track the water intake in mL/Oz , with a 97% accuracy compared to manual recordings.[11]

Figure 2.2.1 Exploded view of HidrateSpark Pro SipSense technology sensor module

Other than weight-sensing capabilities, this smart water bottle also cooperate with popular health and fitness platform, such as Apple Health, Fitbit, Google Fit and Withings Health Mate. The HidrateSpark app not only can track the water intake, but it also providing a function which user can adjust their hydration goals based on the factors like user's height, weight, age. This function even can change the goal based on environmental conditions like temperature, elevation and activity level if the location service is enabled or sync with fitness app.[11]

Figure 2.2.2 HidrateSpark Smart Bottle and App

Furthermore, the water bottle also includes glow reminder system, which used LED light to remind the user to drink water throughout the day. Using the HidrateSpark application, user can customize the glow setting, including colour, glow frequency and intensity, making it a high interactive tool to encouraging consistent hydration.[11]

Figure 2.2.3 HidrateSpark Glow Colours

2.1.2 EQUA Smart Water Bottle

Compared to HidrateSpark Pro 21Oz, EQUA smart water bottle is using motion sensor technology to track user's water consumption throughout the day. The motion sensor will detect movement of the bottle, to recognize all the user movement and calculate the amount of water intake. This technology provides an efficient, which is hands-free approach to hydration tracking, removing the need for manual input of data. [12]

Bluetooth connectivity is another crucial component of the EQUA Smart Water Bottle. It is allowing the smart water bottle to sync with their application, EQUA hydration app, on

user's smartphone. By implementing Bluetooth technology, data that contain user daily water intake can be transfer from the bottle to the app in real-time. Moreover, the application also providing features for user to set their personalized hydration goals based on the factors like user's weight, age and physical activity level. This application provides users with easy access to their hydration data and enabling them to monitor their progress and adjust their habits accordingly.[12]

The most special features of EQUA smart water bottle are EQUA also implemented machine learning to learn the user's behavior to improve the accuracy of hydration recommendations over time. Besides that, by using machine learning, the bottle and its application can adjust the user daily water intake goals based on various factors, such as user's physical attributes, daily activity and environmental conditions. By using machines learning can provide users a personalized advice tailored to their specific needs to promoting a better health outcome. [12]

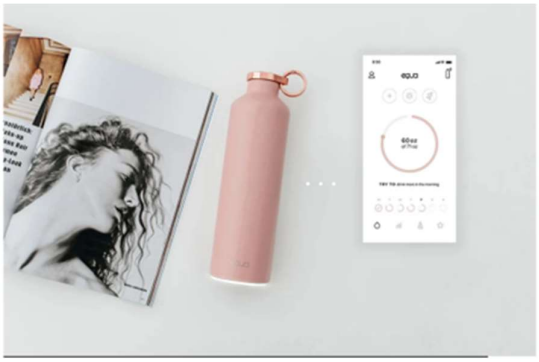


Figure 2.2.4 EQUA smart water bottle and app

The EQUA smart water bottle come equipped with a visual reminder system that encourages users to maintain hydration throughout the day. The bottle's embedded glow feature illuminates at present intervals, reminding users to drink water when they have not consumed enough during a certain period. This reminder system is particularly beneficial for user who may forget to drink water due to busy schedule. [12]

Figure 2.2.5 EQUA glow reminder

2.1.3 Trago Smart Water Bottle

In this smart water bottle , Trago is using ultrasonic sensor, which offer a unique approach to measuring liquid consumption. The working principle for ultrasonic sensors is to emit an ultrasonic pulse at 40kHz, which move through the air inside the bottle. When the pulse is emitted and hits the surface of the liquid, it will reflect and go back to the sensor, thus the water intake can be calculated by measuring the pulse return time and the speed of the sound.[13]The benefit of using ultrasonic sensor is the sensor are able to provide an accurate measurement of the water intake within 0.5Oz, regardless of the type of liquid inside the bottle. The developers that design the Trago smart water bottle explained that their choice of using ultrasonic sensor for measuring water intake, stating that other sensors, such as weight sensor, pressure sensors and accelerometers were found to be extremely inaccurate. Being able to measure any liquid and guarantee an accurate reading is a big plus in the field.[14]

Figure 2.2.6 Trago smart water bottle with ultrasonic technology

Other than used of ultrasonic sensor, Trago smart water bottle also using motion-sensing technology for a better accuracy and power efficiency. By using motion sensor, the system can know the bottle are at rest or the user are taking a drink. In this way instead of continuously transmitting data, it will save batteries to work for longer. The integration of motion sensor ensures that only relevant drinking events are captured such as drinking water to avoid false reading during instances when the bottle would idle.[14]

Trago also develops their application, Trago app, which providing real-time monitoring and personalized hydration recommendations. The app calculates a user's optimal daily water intake based on the user input, such as age, weight, activity level and environmental condition. This capability enables users to dynamically adjust their hydration goals, making the product suitable for a wide range of individuals, including athletes and fitness enthusiasts. The integration with other health and fitness platforms, such as MyFitnessPal, Apple Health, and Under Armour Record, enhances the app's utility by linking water intake data with broader health metrics.[14]

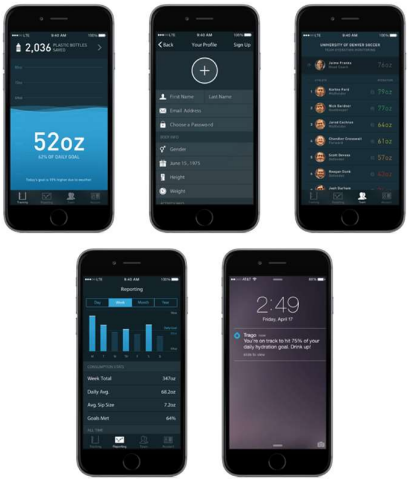


Figure 2.2.7 Trago app interface

Moreover, Trago app also supports a group setting such as teams and athletic programs. This feature providing coaches, trainers and parents can monitor their athletes' hydration through Trago app, ensuring the team member have proper hydration throughout training and competitions.[14]

Figure 2.2.8 Trago App in Athletic and Group settings

One of the standout features of Trago smart water bottle is universal cap, which can be fitted onto any standard wide-mouth water bottle, including brands like Nalgene, Camelback and Hydro Flask. This flexibility allowing the user to use the Trago system when continue using their preferred water bottle.[14]

Table 2.2.1 Comparison of Tracking Methods and Features in Smart Water Bottles

Feature/Method	HidrateSpark Pro 21Oz	EQUA Smart water bottle	Trago smart water bottle
Technology Used	SipSense (weight-based technology)	Motion sensor	Ultrasonic sensor
Water Consumption Tracking	Load sensor measuring each sip	Motion sensor detects drinking motion	Ultrasonic pulse measures liquid volume
Connectivity	Bluetooth	Bluetooth	Bluetooth
App Integration	HidrateSpark App, syncs with Apple Health, Fitbit	EQUA Hydration App, syncs with health platforms	Trago App, syncs with Apple Health, MyFitnessPal
Additional Features	LED glow reminders	Glow reminder system	Motion-sensing technology for accuracy
Compatibility	Specific to HidrateSpark bottles	EQUA Smart Water Bottle	Universal cap, fits Nalgene, Camelback, Hydro Flask

2.3 Limitation of Previous Studies

Both of the HidrateSpark and EQUA smart bottle have a same limitation, which is their incompatibility with different bottle design. Unlike Trago smart water bottle, this brand had offer a universal cap that fits onto standard wide-mouth water bottle, including brand like Nalgene and Hydro Flask. This feature is absent in HidrateSpark and EQUA smart water bottle, it need user to purchase their own brand-specific bottles in order to utilize the tracking features. This lack of flexibility can limit their use, especially where children will use a smaller, themed bottle that is suitable to their age group and preference, such as in kindergartens or schools.[15]

Additionally, although Trago smart water bottle had offer a universal cap, but it is still a challenge when used in environments like kindergartens. Trago designs the universal cap is more faced to adult user, which is using a large capacity water bottle, but it does not along with the needs of young children, who require smaller bottle for handling.[15] As a result, Trago smart water bottle is not suitable for younger age group especially for the young group who are studying in kindergarten, its design is more toward for adult users.

Cost is another limitation that applies to all three smart water bottles. Three of the smart water bottles are using premium materials, such as stainless steel for insulation and durability, along with advanced design element such as double-walled thermos cups to keep liquids at the desired temperature for several hours, which will increase the production cost of the water bottles.[11,12] This features will give a benefit of maintaining the temperature of beverages, but drive up the price, making them less affordable compared to traditional plastic or simpler water bottles. Additionally, the use of machine learning adds more cost in the development. Implement this technology, it may help to enhancing the accuracy of water consumption tracking, but it also will increase the overall price of the product, making it more challenging to justify in environments such as kindergartens.

In environments like schools and kindergartens, children's water bottles often having the wear and tear issues because of frequent use and handling method by young children. With kids frequently lose or damaging their bottles, regular replacement becomes necessary. Kid's water bottle is being suggested to be replaced at least a year, and sometimes more often, due to inappropriate wear and tear. [16]

2.4 Summary

In this literature review, three existing smart water bottle bottles were examined, which are HidrateSpark Pro 21Oz, EQUA Smart Water Bottle and Trago Smart Water Bottle. Each of these three products use different technology to track the water intake, where HidrateSpark uses a weight-based SipSense technology, EQUA relies on motion sensor and deep learning algorithms. Meanwhile, the Trago is using ultrasonic sensor to detect the water level.

While these smart water bottles provided several advanced features like Bluetooth connectivity, mobile app integration, customize hydration goals and visual glow reminder, they also have limitations. HidrateSpark and EQUA require their own-brand specific bottles which is not flexibility, while Trago offers universal cap design, but it only supports larger bottles which are not suitable for young children. Besides that, their product is more costly for using premium materials, advanced sensor and deep learning technology for EQUA smart water bottle, which is not suitable for environments like schools or kindergarten, where frequent and affordability is more important.

In summary, existing smart water bottles such as HidrateSpark, EQUA and Trago demonstrated different methods for hydration tracking through weight-based sensors, motion sensors with deep learning and ultrasonic level detection. While these systems offered advanced features, but they still faced limitations in terms of flexibility, suitability for children and cost-effectiveness in school environments. To address these limitations, this project proposed a low-cost attachable smart water tracking device that used ESP32 microcontrollers, load sensors and Firebase Realtime Database. Meanwhile, lightweight algorithms were designed for water intake detection, orientation checking, offline synchronization and data visualization. The system was developed to provide a reliable, affordable and practical solution for monitoring children hydration both in real-time and offline.

Chapter 3

System Methodology/Approach

3.1 System Design Diagram/Equation

In this section, the technical and mathematical foundation that supporting the smart water tracking system was presented.

3.1.1 Load Sensor Calibration Equation

The weight of the water bottle was measured by used a 10kg load cell interfaced through HX711 amplifier. Before the load sensor can work, the system must undergo calibration, as the load sensor initially produces raw, unscaled data upon startup. This step was crucial because incorrect calibration factor will affect the data accuracy. Furthermore, calibration can prevent other environmental issues such as electrical drift, environment factor and etc to affect the data accuracy.

The formula to get the calibration is:

$$calibration\ factor = \frac{reading}{known\ weight(gram)}$$

3.1.2 Water Intake Estimation

The system will take the water intake based on the difference between old reading and new reading.

The formula is:

$$diffrent\ (g)\ = new\_reading - current\_reading$$

Furthermore, a threshold is set to ignore environmental issues such as vibration or user handling.

3.1.3 Timestamp Estimation using millis()

When the system is offline and the timestamp was unknown, the system will used millis() to estimate the time for each water drinking event. Once the system is connected to Wi-Fi, it will sync the NTP to get the current timestamp and do calculation to get the estimated timestamp.

The formula is:

$$T_i = T_{NTP} - \frac{(millis_{system} - millis_i)}{1000}$$

Where :

- T_i: Estimated timestamp of the i-th offline data
- T_{NTP}: Time from NTP that get from internet
- $millis_{system}$: Milliseconds of the total system running time
- $millis_i$: Milliseconds stored when data was recorded
- The division of 1000 is to convert the millisecond difference to seconds.

3.1.4 Power Consumption and Battery Life Estimation

The system is powered by a 3.7V 3800mAh 18650 lithium-ion battery, connected through a TP4056 charging module and a voltage booster, which provides a regulated 5V supply to the ESP32 microcontroller, HX711 load cell amplifier, ADXL345 accelerometer, voltage sensor and red LED indicator. Since the system operates continuously with Wi-Fi enabled, both the steady-state current and the startup current burst must be considered.

Table 3.1 Current Consumption of Components

Component	Typical Current Consumption
ESP32 microcontroller (Wi-Fi active)	~200 mA (average)
ESP32 microcontroller (Wi-Fi burst at startup)	up to 400-500 mA (peak)
HX711 Load Cell Amplifier	~1.7 mA
Voltage sensor	~3 mA
ADXL345 accelerometer	~0.14 mA
Red LED indicator	~4 mA

The ESP32 faced a short burst of current during Wi-Fi initialization, where current can peak at 400-500 mA for a few seconds while ESP32 looking for available Wi-Fi. After stabilization, the device typically draws ~200 mA on average during continuous Wi-Fi usage.

The power requirement was calculated as:

$$P = V \times I = 6\,V \times 0.209\,A = 1.254W$$

Accounting for a boost converter efficiency of approximately 80%, the equivalent current from the battery is:

$$I_{bat} = \frac{P}{V_{bat} \times \eta}$$

$$I_{bat} = \frac{1.254}{3.7 \times 0.8} = 0.423\,A$$

Thus, the battery supplies ~346 mA on average, with brief peaks above this value during Wi-Fi setup.

The expected runtime of the system is expressed as:

$$T = \frac{Capacity}{I_{bat}} = \frac{3800}{423} \approx 8.98\,h$$

3.1.5 Battery Voltage Estimation

The ESP32 microcontroller monitor the battery level using a voltage sensor. The voltage sensor operates based on a voltage divider to step down the battery voltage before measurement. The raw ADC reading that provided from the ESP32 was 12-bit resolution , ranging from 0-4095, which corresponding to 0-3.3V. Since the actual battery voltage exceeds 3.3V, the divider ensured that safe measurement by scaling the input voltage.
The formula is:

$$V_{bat} = \left(\frac{ADC_{raw}}{4095} \times 3.3V\right) \times 6$$

Where:

- ADC_{raw}: Raw reading from 0-4095
- 3.3V: Reference voltage of ESP32 ADC
- 6: Divider scaling factor
- V_{bat}: Estimated battery voltage

3.2 System Architecture Diagram

The system architecture of the smart water tracking system for kids is designed to operate as an embedded system that support for both offline data logging using NVS and online data synchronization using Firebase. The architecture follows a layered structure where sensor data is collected, processed, transmitted and visualized at web application.

At the sensing stage, the load cell with HX711 amplifier measured weight changes, while ADXL345 accelerometer monitored the orientation of the water bottle to avoid false reading when the bottle was tilted or lying down. Other components such as voltage sensor and red LED indicator ensured that the ESP32 battery status could also be tracked.

The ESP32 microcontroller served as the CPU, which responsible for managing sensing operations, executing algorithms and handling local data storage when Wi-Fi is unavailable. It also managed communication with the Firebase Realtime Database once the ESP32 microcontroller was connected to Wi-Fi.

Finally, the frontend web application which connected to the database, it will processed and visualize the hydration data in real time. The interface displayed daily, weekly and monthly consumption patterns, as well as summary such as total intake, refill time and goal achievement.

This layered design was chosen to balance low cost, reliability and scalability, ensured that the system could function effectively in environments with limited supervision such as schools or playground while maintaining accurate tracking even during the system went offline.

Below figure shows the system architecture diagram.

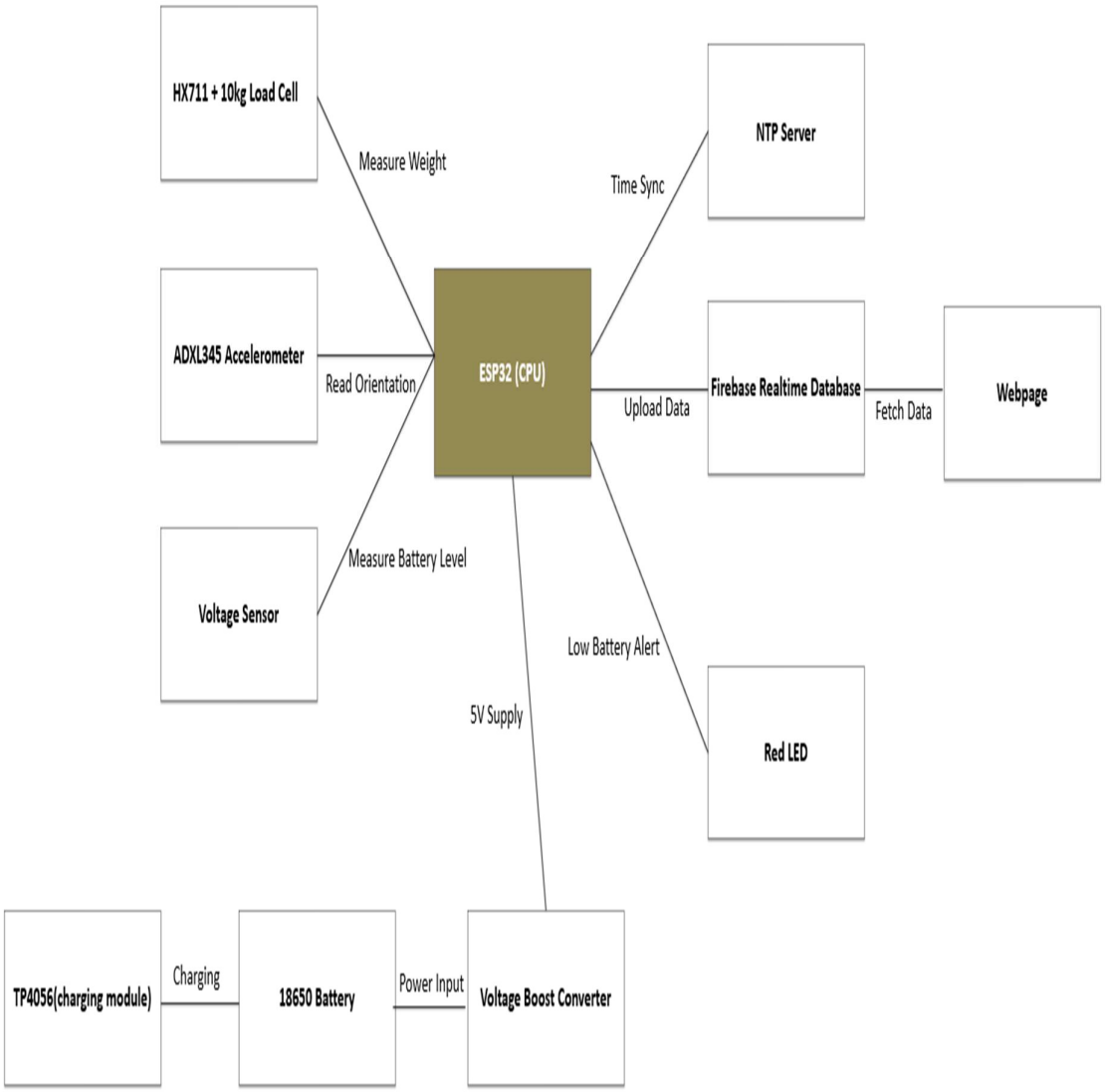


Figure 3.1 System Architecture Diagram

3.3 Use Case Diagram and Description

The figure below shows the use case diagram. The use case diagram illustrates the interactions between the Smart Water Tracking System and its external actors. In this project, two primary actors were involved, which was the child, who is interacting with this system by drinking water and refilling the water bottle. Another actor was parent/teacher, who will monitor the child's water intake, set hydration goal and check water intake progress. The diagram highlights the various function available to each actor, while the use case description will provide further explanation of the system functionality.

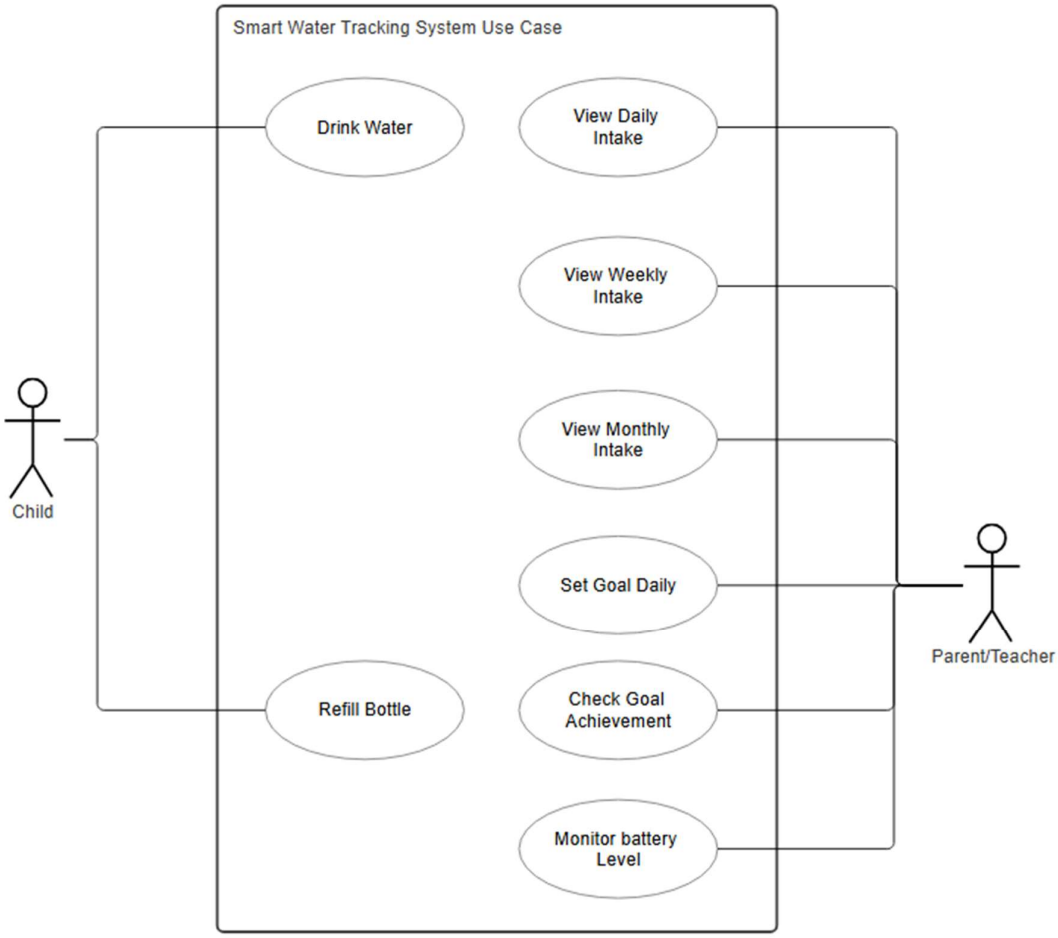


Figure 3.2 Use Case Diagram

Use Case Description

The following table describes of each use case show in the diagram. Each use case will highlight the functionality of the system, the actor involved and the expected outcome. This provides a clear view of how the Smart Water Tracking System operate from both chid and parent/teacher perspective.

Table 3.2 Use Case Description

Actor	Use Case	Description
Child	Drink Water	The child drinks water from the bottle. The system will detects the weight change and records the intake

Child	Refill Bottle	The child refills the water bottle. The system updates the recorded weight to database to reflect the refill
Parent/Teacher	View Daily Intake	The parent/teacher view the total water consumption by the child in a single day via web dashboard
Parent/Teacher	View Weekly Intake	The parent/teacher reviews the child water consumption over the past week
Parent/Teacher	View Monthly Intake	The parent/teacher reviews the child water consumption trends over a month to track long-term hydration patterns.
Parent/Teacher	Set Goal Daily	The parent/teacher sets a daily, weekly and monthly water intake target for the child
Parent/Teacher	Check Goal Achievement	The parent/teacher check whether the child has meet the hydration goal
Parent/Teacher	Monitor Battery Level	The parent/teacher can check the system battery level on the web application to ensure uninterrupted monitoring

3.4 Activity Diagram

The activity diagram below illustrates one of the workflows of the Smart Water Tracking System. The process begins when the child drinks water, followed by orientation checking and weight measurement. The system then checks the Wi-Fi connection. If Wi-Fi is unavailable, the data will be store in local storage. If Wi-Fi is available, the data will uploaded to Firebase, where parent or teacher can view it through the frontend application and check goal achievement.

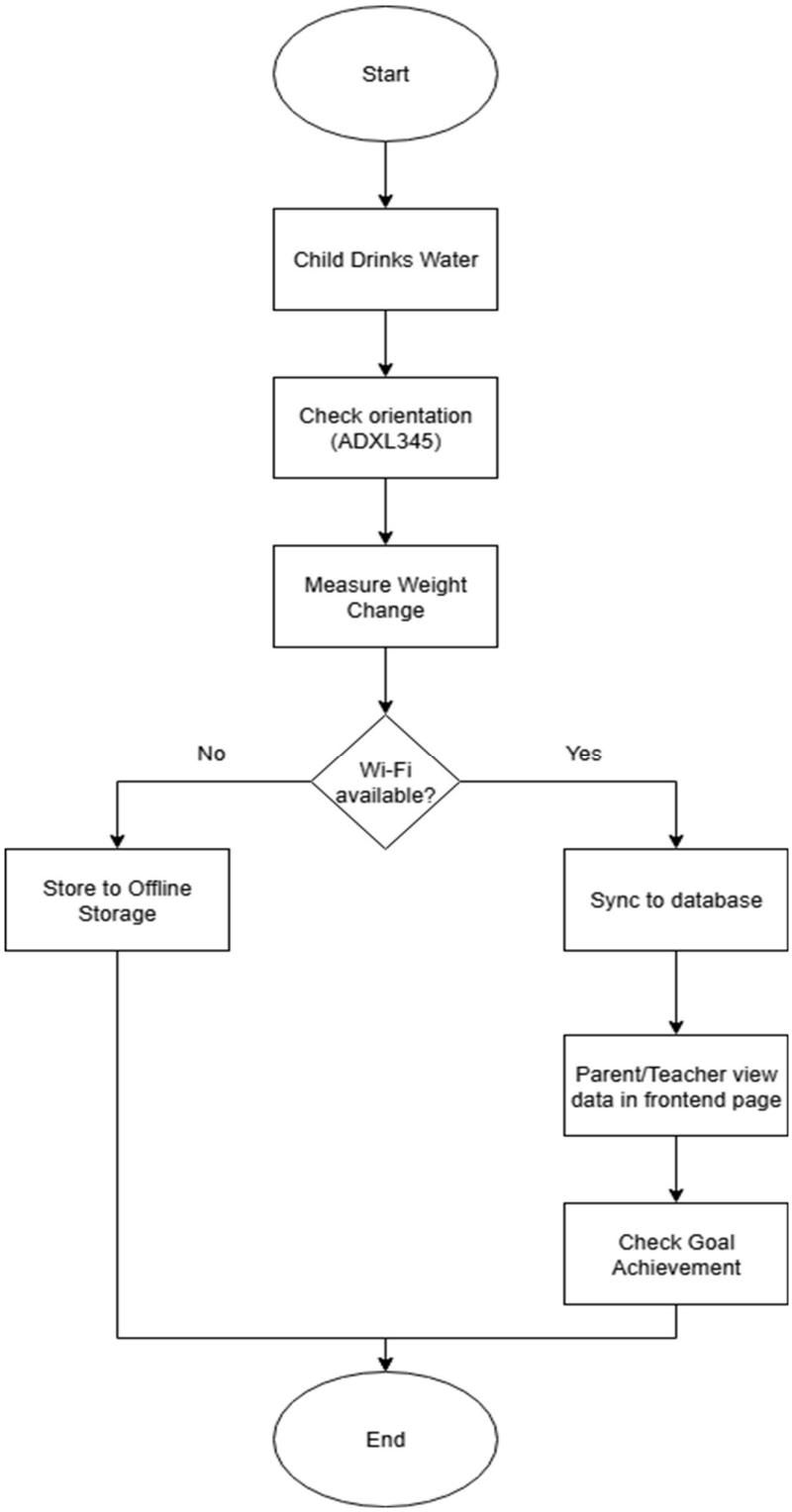


Figure 3.3 Activity Diagram

Chapter 4
System Design

4.1 System Block Diagram

The block diagram of the Smart Water Tracking System is shown in Figure 4.1. The system is divided into five layers, which were Power Layer, Sensing Layer, Processing Layer, Connection Layer and Interface Layer. Each layer is responsible for different specific functions of the system.

1. Power Layer

The power layer consists of the TP4056 module, an 18650 lithium-ion battery and a voltage boost converter. The TP4056 module allows safe charging of the battery and provides protection against overcharging or discharging [9]. The 18650 battery provide power to the system, while the voltage boost converter regulates the output to provide a stable 6V for the ESP32 microcontroller.

2. Sensing Layer

This layer includes the load cell with HX711 amplifier for measuring weight differences, the ADXL345 accelerometer for orientation checking and a voltage sensor to monitor battery level. These sensors collected raw data when the child drinks or refills water, providing the necessary input for further processing.

3. Processing Layer

The ESP32 microcontroller act as the CPU of the system. Several processing modules were implemented within the ESP32, such as weight measurement module, orientation module, voltage sensor ADC converter module, data handling module and Firebase setup module. These modules will process the raw data that received from the sensing layer and store or transmit the data.

4. Connection Layer

The connection later managed the network communication between internet and ESP32 microcontroller. The ESP32 built a Wi-Fi connection to synchronize data with Firebase

database. The NTP server is used to obtain accurate timestamp, which are attached to the recorded data. If Wi-Fi was unavailable, data will temporarily store in offline storage until synchronization was possible.

5. Interface Layer

This layer will provided feedback and visualization for users. A red LED was used to indicate the system battery status, while the processed data was transmitted to the frontend application. The frontend will displays daily, weekly and monthly hydration trends along with battery status. Parent and teacher can monitor the hydration habits of children through this interface.

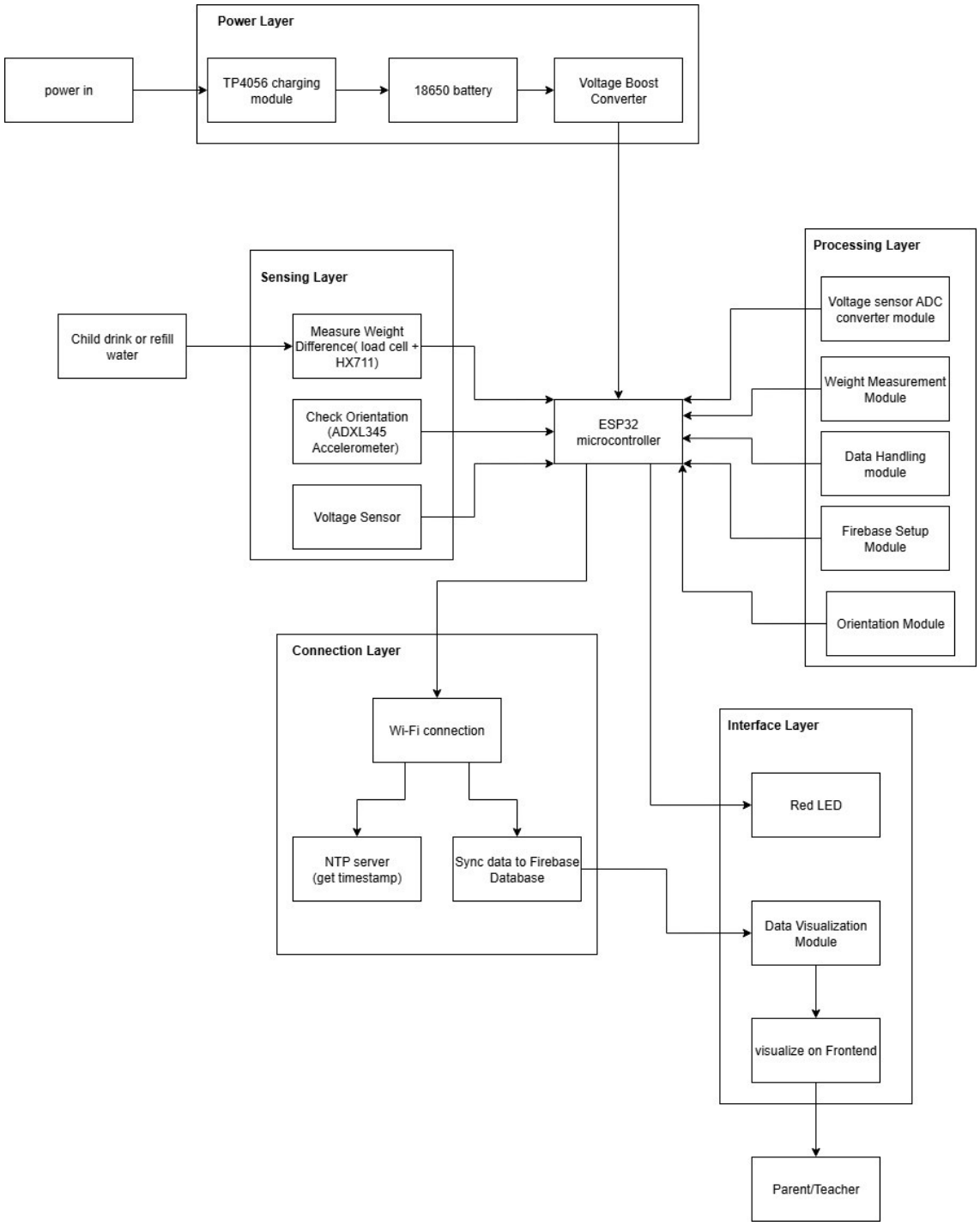


Figure 4.1 System Block Diagram

4.2 System Component Specifications

The smart water tracking system was built using several hardware components. Table 4.1 summarizes the specifications and functions of the selected components that used in this system implementation.

Table 4.1 Specifications of System Components

Component	Specifications	Function in the System
ESP32-WROOM32	-3.3V operating voltage -Dual-core 32bit MCU -Built-in Wi-Fi & Bluetooth	Main component for data processing, local storage management and communicate with Firebase
HX711 Load Cell Amplifier	-24-bit ADC resolution -low noise, high precision -built-in conversion from ADC to digital value -Operating voltage: 2.6 to 5.5V	Auto convert analog signals from load cell into digital values for water weight measurement
10kg Load Cell	-weight capacity :10kg	Detect water bottle weight changes to estimate water consumption
ADXL345 Accelerometer	- ±2g, ±4g, ±8g, ±16g selectable range -Operating voltage: 3 to 5V -3 axes(X,Y,Z)	Detect bottle orientation to avoid false readings when bottle is tilted or lying down
Voltage Sensor	-Input voltage range:0-25V -Divider Ratio: ~1:6 -12-bit resolution(0-4095)	Monitor battery voltage to ensure reliable system operation
18650 Li-ion Battery	-capacity: 3800mAh -rechargeable -Nominal voltage: 3.7V	Supply power to the system
TP4056 Charger Module	-Input voltage range: 4.35-6V -charging current: 1A	Provides safe charging for 18650 battery

	-protection: overcharge & discharge	
Voltage Boost Converter	-input voltage: 3V-6V -output voltage range: 5-28V DC -max output current: 2A	Steps up battery voltage to supply stable 6V to ESP32
Red LED	-Current: ~4-10 mA -Forward voltage: ~2V	Light up when voltage sensor detects low battery level

4.3 Circuits and Components Design

The smart water tracking system integrated several electronic components to enable sensing, processing, communication and power management. Each component is connected to the ESP32 microcontroller, which acts as the central processing unit of the system. The design ensured efficient data acquisition, stable operation and reliable power delivery.

Load Cell with HX711 Amplifier

The load cell was connected to the HX711 module, which amplifiers the small voltage changes generated by the strain gauge when water weight changes. The connection of the load cell with HX711 amplifier were red wire to E+(VCC), black wire to E- (GND), green wire to A+ and white wire to A-. The channel A was selected due to it offers higher amplification gains of 64x or 128x to ensure more accurate weighting result [10]. The HX711 is then interfaced with the ESP32 using digital pin(DT and SCK), where DT was connected to GPIO16 and SCK was connected to GPIO4, for continuous weight measurement.

ADXL345 Accelerometer

The accelerometer was connected to the ESP32 cis the I^2C communication protocol. In this project only four pin will be used, which were VCC, GND, SDA and SCL. VCC was connected to 3.3V, SDA to GPIO21 and SCL to GPIO22. GPIO21 and GPIO22 was selected was because both of them was the default 1^2C pins supported by the ESP32 [22]. ADXL345 was used to monitor the water bottle orientation to prevent false reading from the load cell while the bottle was tilted or lying down.

Voltage Sensor

A voltage sensor module is connected to the ESP32 analog input pin to measure battery voltage. The VCC and GND pin of the module ware connected to the 18650 battery while the signal (S) pin was connected to GPIO32 of the ESP32 to provide the ADC reading. With the assist of voltage sensor, the system was able to monitor the battery level and alert the user if charging is needed.

Power Supply Design

The system is powered by a rechargeable 18650 Li-ion battery. A TP4056 charging module manages battery charging via micro-USB input. The battery output is connected to a DC-DC boost converter, which provided a regulated 6V to power the ESP32 microcontroller. Additionally, a red LED indicator was integrated into the system so it will provide a visual alert when the battery voltage dropped below a predefined threshold, as detected by the voltage sensor.

ESP32 Microcontroller

The ESP32 integrated all sensor inputs, executes the data processing logic and manages Wi-Fi communication. When Wi-Fi is unavailable, data was temporarily stored in offline memory. When available, the ESP32 will transmitted the store records to Firebase Realtime Database.

Frontend Connection

Processed and uploaded data can be accessed by parents or teachers through the frontend web application. The ESP32 ensured that all data stored locally is consistent with Firebase once synchronization was completed.

This circuit and component design ensures proper integration between sensing, processing, power and communication units, providing a reliable and efficient smart water tracking solution.

4.4 System Components Interaction Operations

From the system block diagram, the Smart Water Tracking System was divided into five main layer, which were sensing layer, power layer, communication layer, processing layer and interface layer. In this subchapter, the interaction and operation of these layers were described in more detail to explain how the system functions as a whole.

4.4.1 Sensing Layer

The sensing layer consists of a load cell connected to HX711 amplifier, the ADXL345 accelerometer and voltage sensor. The load cell with HX711 amplifier was to measure the weight of the water bottle and detects water intake. Furthermore, the ADXL345 monitor the water bottle's orientation to prevent false reading when the bottle was tilted or lying down. The last component, voltage sensor, will continuously monitor the battery level and provides the result to the ESP32. These three sensors provided the raw data required for further processing.

4.4.2 Power Layer

The power layer included the 18650 Li-ion battery, TP4056 charging module and DC-DC voltage boost converter. The 18650 battery supplies the main power to the system, while the TP4056 charging module allow the battery to be recharged via micro-USB cable. When the battery had fully charged or being disrupted while charging, the TP4056 will have a built-in protection circuit to protect the battery.

With only using the 3.7V battery was not enough to handle the ESP32 task and maintain stable Wi-Fi connection, thus the need of DC-DC voltage boost converter was crucial in this project. It will increase the 3.7V to stable 6V to meet the requirements for ESP32 to handle it tasks. When the 6V was entered into ESP32 V_{in} pin, it will auto regulated the 6V to 3.3V to power itself and the connected sensors. This ensures that all components receive stable voltage for continuous operation.

4.4.3 Processing Layer

In this layer, ESP32 microcontroller was acting as the core of this system, responsible for executing the main logic, performing calculation and managing the data flow between components. When the ESP32 initial, in setup() function, it will tared the load sensor,

initialized the ADXL345 accelerometer and captured a weight reading as current reading. The ADXL345 accelerometer operate on the principle of detecting changes in capacitance along the x, y and z axes to measure acceleration. In this system, the accelerometer was using static forces (gravity) for orientation checking. When the water bottle was placed in upright, the z-axis will be showing the values in the range of 9 to 10, while x and y axes remain close to 0. If the z-axis was not in this range, the system recognizes that the water bottle was tilted or lying down and the weight measurement will temporarily disabled to prevent false reading until the water bottle was placed back upright.

Once the orientation was confirmed, the load sensor will perform a new weight measurement and compare to the current weight every 3 seconds. If the difference more than 30 (which was to prevent moving or shaking the water bottle accidentally), then it will take 3 additional weight reading every 0.3 second. Only if the reading remained stable and the variation does not exceed ±2 compared to the initial difference, the system recorded the event as a valid water intake event.

Next, ESP32 will determine whether the system was operating in online or offline mode. In offline mode, the data reading will be stored into local storage (NVS). By using Preferences library to store the data is because the system only needs to store 2 data points, which are weight changes and timestamp. The fixed data structure makes NVS a suitable and efficient storage option. Furthermore, NVS has a feature to prevent power-lost which the data will be stored into local storage although the microcontroller suddenly shut down. Since the amount of data being stored is minimal, there is no need to implement larger file systems such as SPIFFS, making NVS the most lightweight and effective solution for this application. [17]

If a valid timestamp was available and the system was disconnected from Wi-Fi, it will store the weight changes and the valid timestamp to the local storage. Otherwise, the system will store the weight changes with the system runup time as a temporarily timestamp in another local storage. Once Wi-Fi connectivity was restored, the system will process the offline data with unknown timestamp first, by using timestamp that get from NTP server and the recorded runtime offset to get an estimated timestamp.

When then system was connected to Wi-Fi, ESP32 will initialize the Firebase configuration and begins multitasking to handle additional system functions. Alongside weight monitoring, the microcontroller executes a task for battery voltage measurement. In this task, the voltage sensor takes 20 consecutive readings, averages the result to reduce the noise and upload the battery level together with a timestamp to Firebase every 5 minutes. This ensured that power status information remains accurate and up to date, allowing users to be alerted when charging was required.

4.4.4 Communication Layer

In this layer, it responsible for internet communication and online database storing. The ESP32 microcontroller connect to the internet via Wi-Fi. After connected to Wi-Fi, it retrieves the real-time timestamp using NTP. The use of NTP is essential because the ESP32 don't have internal RTC. Without internal RTC, the ESP32 would require an additional module, such as the DS3231 to track the time[18]. By using NTP, this project eliminates the need for extra hardware while still ensuring accurate time synchronization through the internet[19].

After connected to Wi-Fi and get the timestamp, ESP32 will upload the data log which contained weight changes and timestamp to Firebase Realtime Database in JSON format. Furthermore, if there are data entries stored at the local storage due to a previous disconnection, the system will automatically commit these records to Firebase once the Wi-Fi was connected. This can ensure that the data reliability and stability during network outages.

4.4.5 Interface Layer

The interface layer consists of 2 part; the hardware-based low-battery indicator and the frontend web application hosted on Firebase.

On the hardware side, when the voltage level that detected by the voltage sensor was lower than the threshold of 3.5V, the red LED indicator alerts the user to recharge the battery by blinking 5 time per second.

On the software side, the frontend web application processes raw data from Firebase before displaying it to the user. The system grouped data by date and classifies water intake record into negative and positive values. The negative values water data represent the children actual

water intake, meanwhile the positive value represent the amount of water refilled into the bottle. Furthermore, the system also displays the battery status at the top-right corner of the webpage. This is achieved by comparing the most recent uploaded timestamp from the ESP32 with the current time and battery level. If the timestamp exceeds a threshold of 10 minutes, the system is shown as offline. Otherwise, it is indicated as online.

In the webpage, parent or teacher can view daily, weekly and monthly water intake summaries. The interface also highlights the refills times, total intake and goal achievement. Furthermore, the webpage also offers several personalization features, including the ability to change the font size, clear all the data in database and set the hydration goal separately for daily, weekly and monthly intervals. By providing data visualization and user customization, the system ensures hydration monitoring was convenient, user-friendly and adaptable to different user needs.

Chapter 5

System Implementation

5.1 Hardware Setup

In this section, it will described the hardware structure of the system. To house and organize all the components at one place, a custom case was designed and fabricated using 3D printing technology. The 3D design was created in SolidWorks and divided into 5 layer, where were battery layer, power layer, microcontroller layer, sensor layer and platform layer, all enclosed within an outer case. The 3D printer used in this project was CREALITY Ender-3 V3 KE to print the case. The specifications of the printer are shown in Table 5.1.

Table 5.1 3D Printer Specifications

Model	Ender-3 V3 KE
Printing Technology	Fused Deposition Modelling (FDM)
Build Volume	220 x 220 x 240 mm
Maximum Printing Speed	500 mm/s
Input Printing	Support for high-quality printing

The outer case was designed with 3 vertical rods with 6.2cm each to lock and stabilize the inner layers. Two external openings were included: one for the charging port and another for the LED indicator. The figures 5.1 and 5.2 shows the outer and inner views of the case while figure 5.3 illustrate the outer case in 2D view.

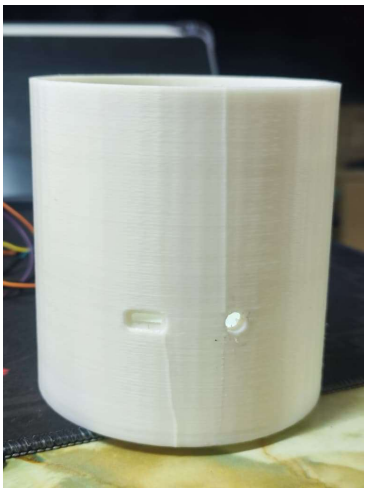


Figure 5.1 Outside View

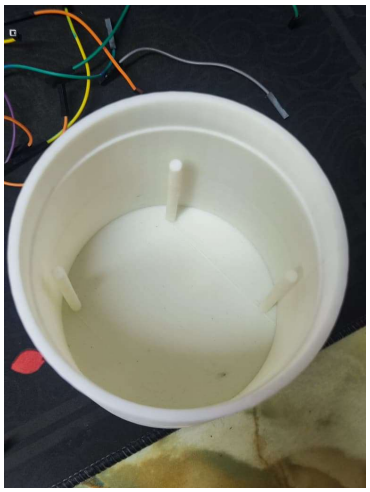


Figure 5.2 Inside View

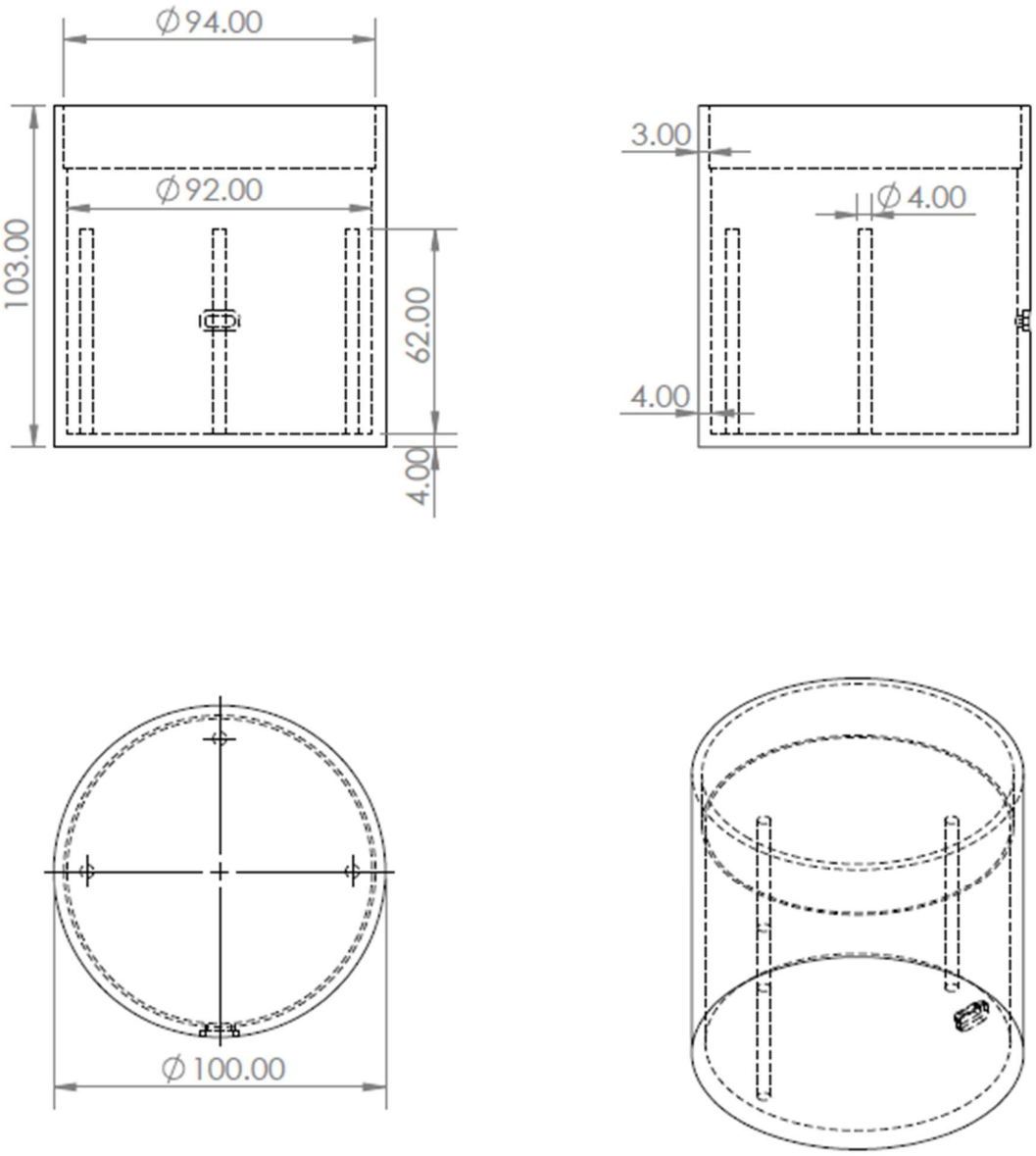


Figure 5.3 Outer Case (2D View)

- **Battery Layer:**

  The battery layer holds the 18650 Li-ion battery and its battery holder. A cut-out hole was designed to allow battery wires pass to power layer. The battery holder was placed under the power layer for easy replacement. Figure 5.4 shows the battery layer.

Figure 5.4 Battery Layer

- **Power Layer:**

  In power layer, it included the TP4056 charging module, voltage sensor, DC-DC voltage boost converter and red LED indicator. The TP4056 sits on a 27 x 17 x 6 mm platform to secure its position. The figure 5.5 and 5.6 illustrate the power layer in 2D design and real-world implementation.

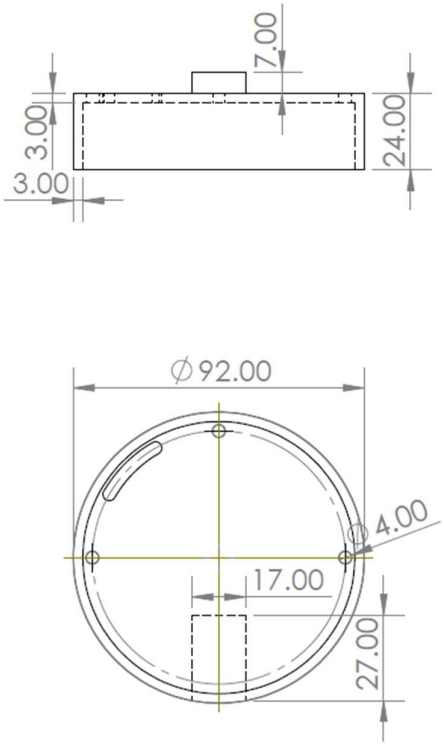


Figure 5.5 Power Layer (2D view)

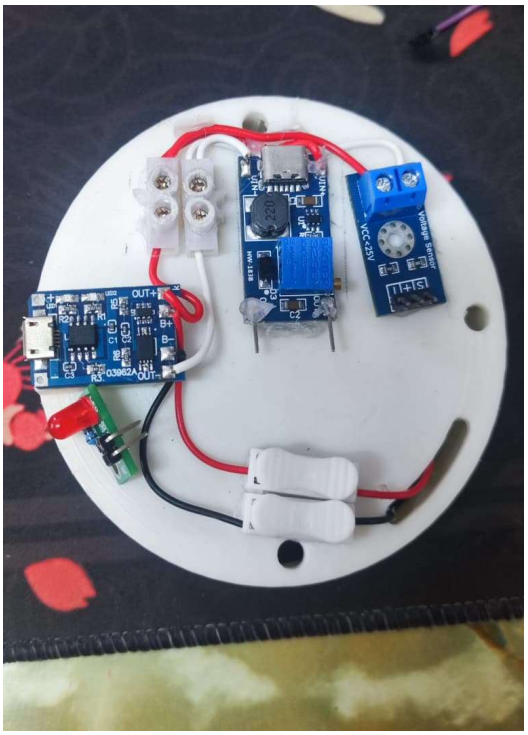


Figure 5.6 Power Layer (Real-World View)

- **Microcontroller Layer:**

  The microcontroller layer houses the ESP32 microcontroller and 2 quick wire terminal connector. The 2-terminal connector was separately distributed the 3.3V pin and ground pin to connected sensor components. A38 x 20 x 10 mm platform was designed for the ESP32, beside it was having 2 hole for wiring connections. Figures 5.7 and 5.8 show the design and assembled layer.

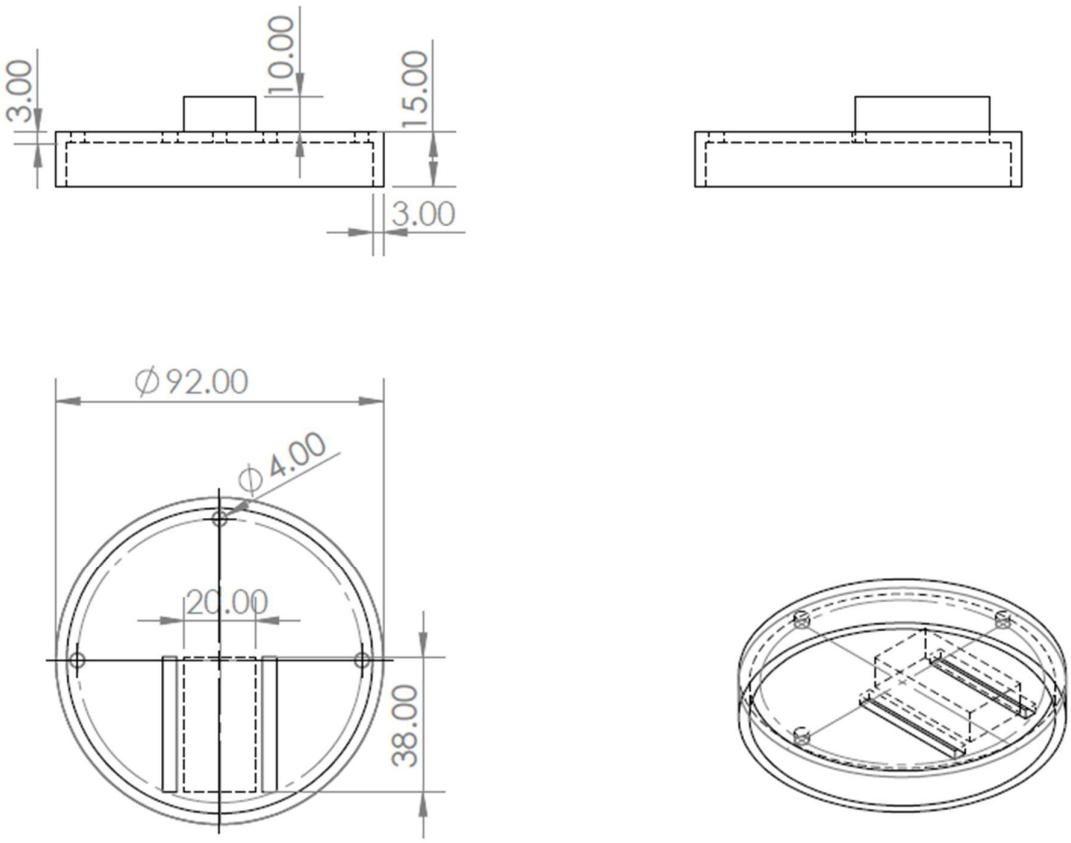


Figure 5.7 Microcontroller Layer (2D View)

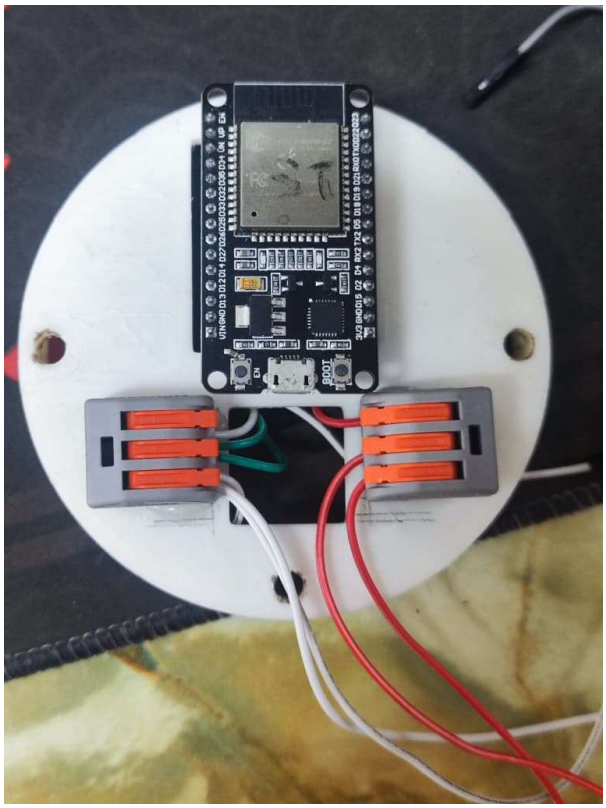


Figure 5.8 Microcontroller Layer (Real-World View)

- **Sensor Layer:**

  In sensor layer it includes the 10kg load cell, HX711 amplifier and ADXL345 accelerometer. Platforms of 20 x 15 x 11 mm for HX711 amplifier and 20 x 17 x 11mm for ADXL345 were designed to mount the modules. Beside the platform, it will have a hole to let jumper wire connect to the sensor pins. Additionally, a 3mm height was designed under the load cell to prevent the strain gauge from touching the surface and producing false readings. Figures 5.9 and 5.10 show the design and implementation.

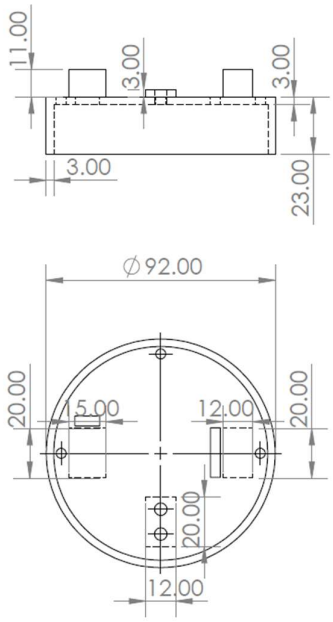


Figure 5.9 2D View of Sensor Layer

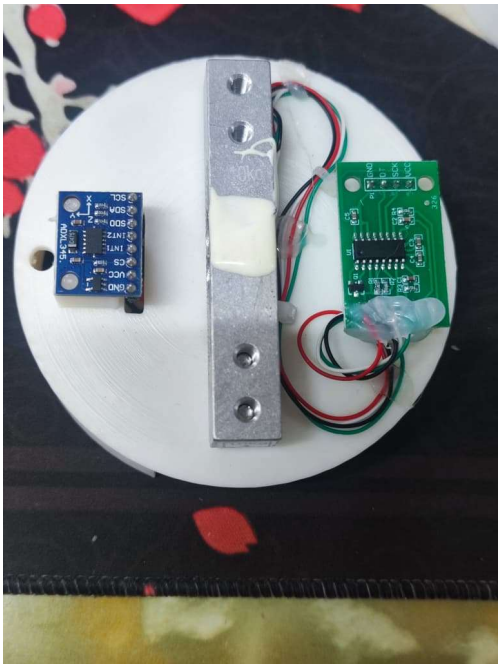


Figure 5.10 Real-World View of Sensor Layer

- **Platform Layer:**

  This was the top layer, designed as a flat platform to place the water bottle. The outer case will have an approximately 2cm raised edge to prevent water bottle moving and keep the water bottle centered on the load cell. A second 3mm height block was also integrated into this layer, positioned above the load cell. This ensures the strain gauge remains properly elevated when the water bottle was placed on top to improve the accuracy of the weight measurement. Figure 5.11 show the platform layer in 2D view.

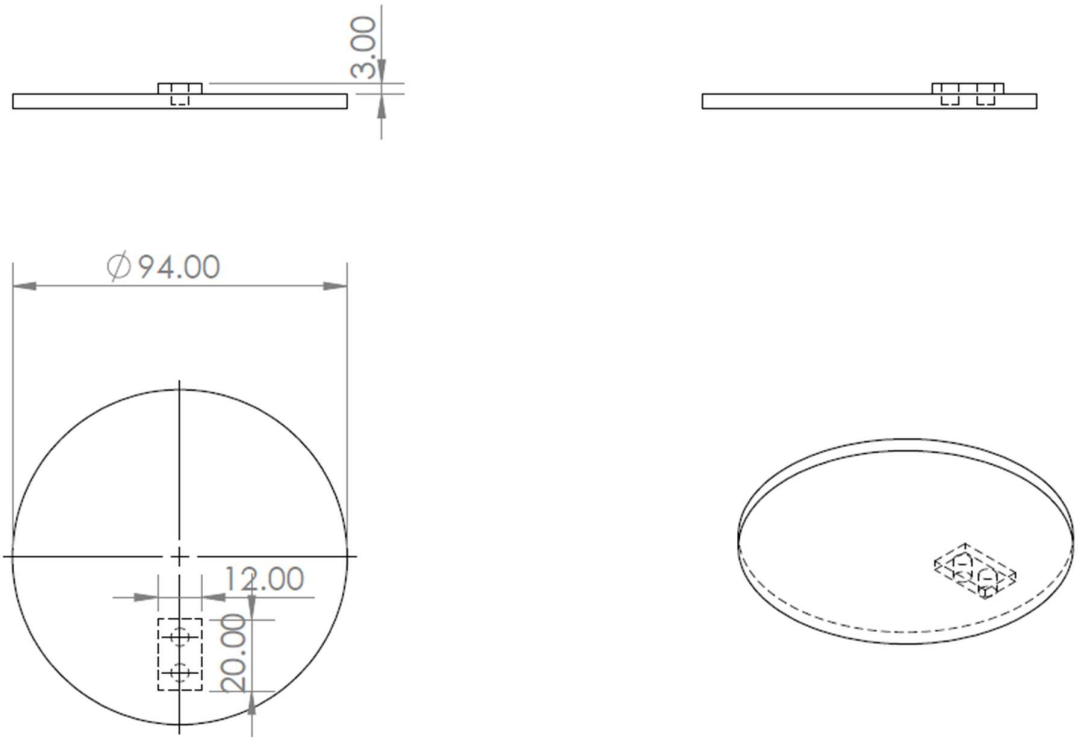


Figure 5.11 Platform Layer (2D View)

Finally, Figures 5.12 and 5.13 show the assembled system in both 2D views and real-world, combining all five layers into a single unit. Figure 5.14 shoes the assembled system together with the outer case.

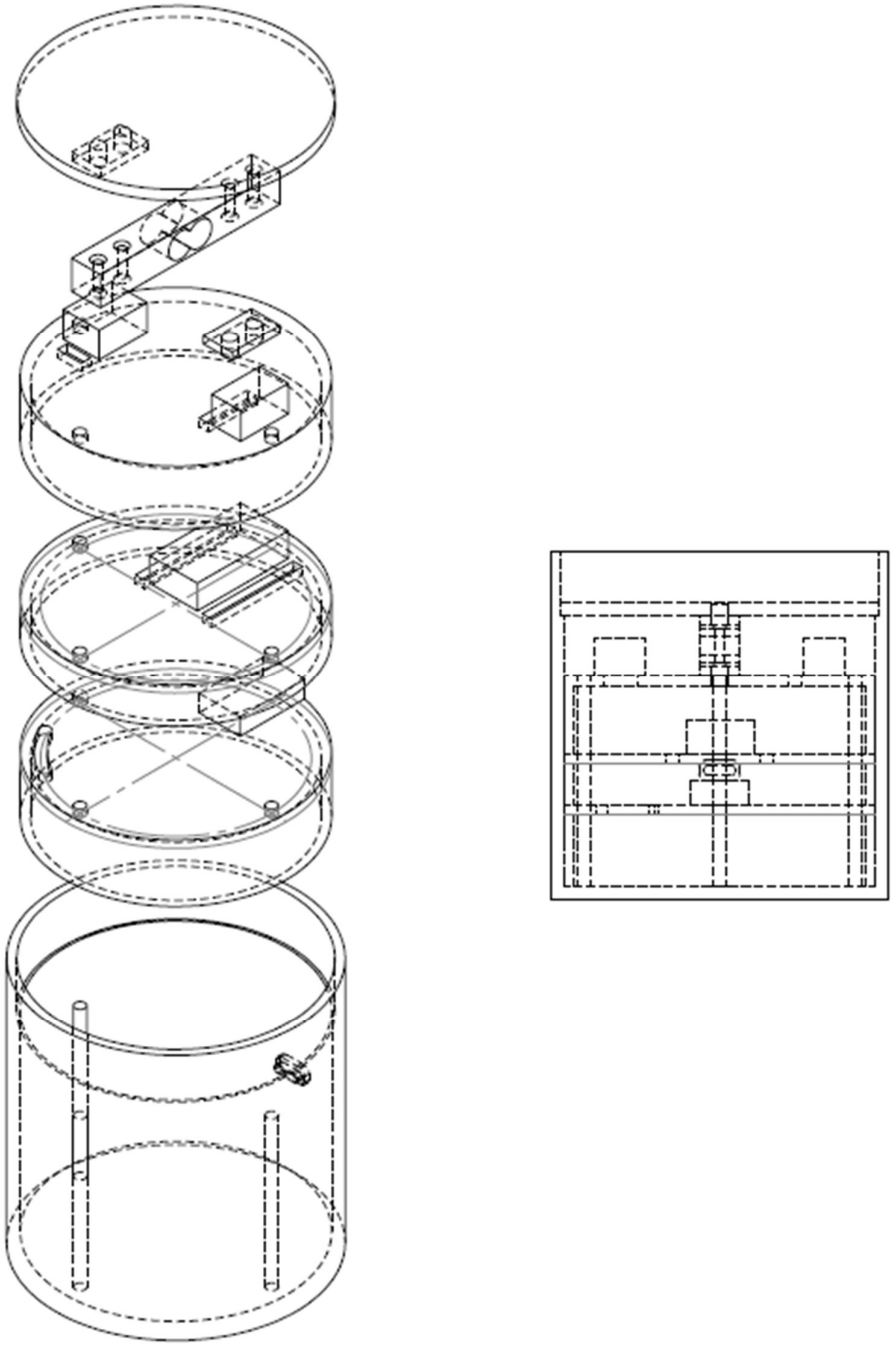


Figure 5.12 Assembled System (2D View)

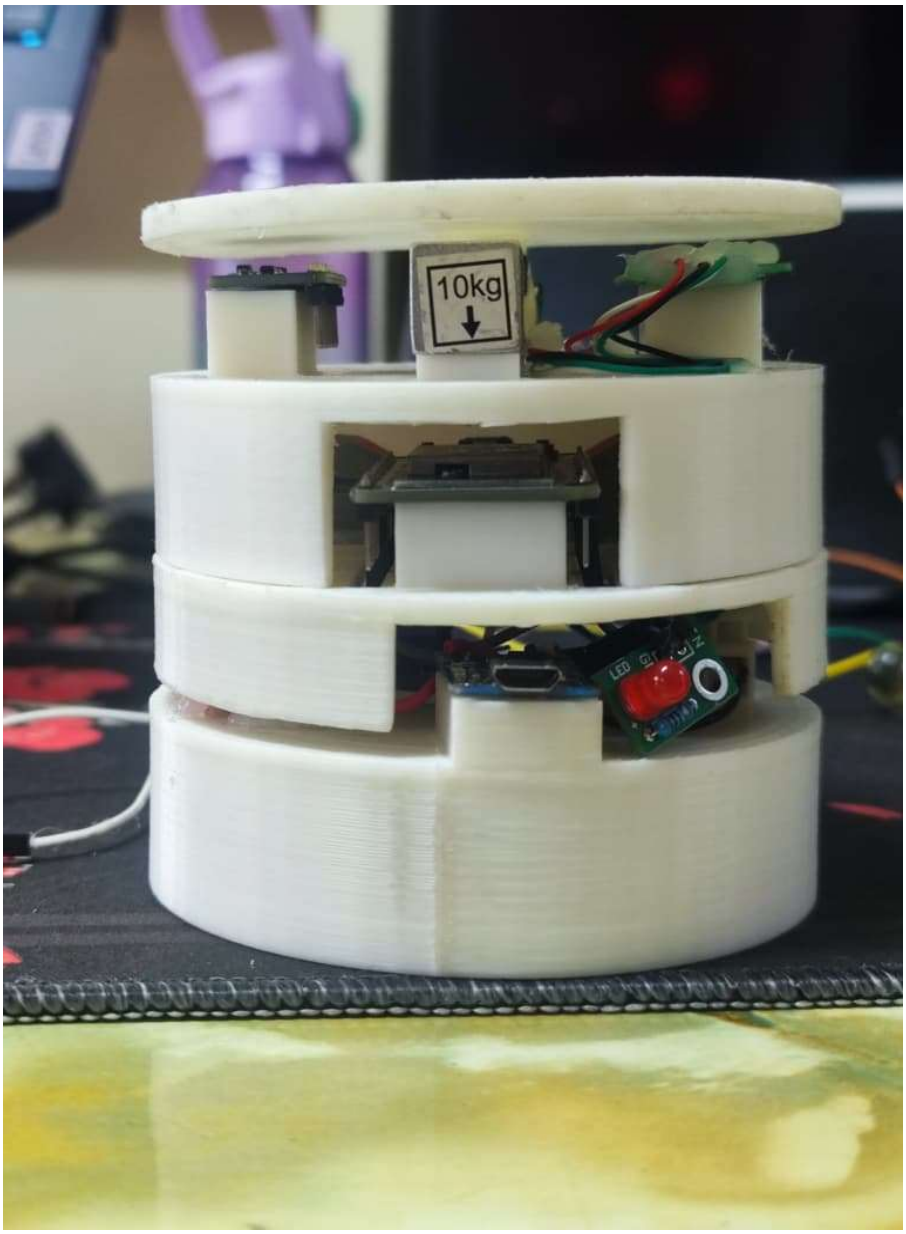


Figure 5.13 Assembled System without Outer Case (Real-World View)

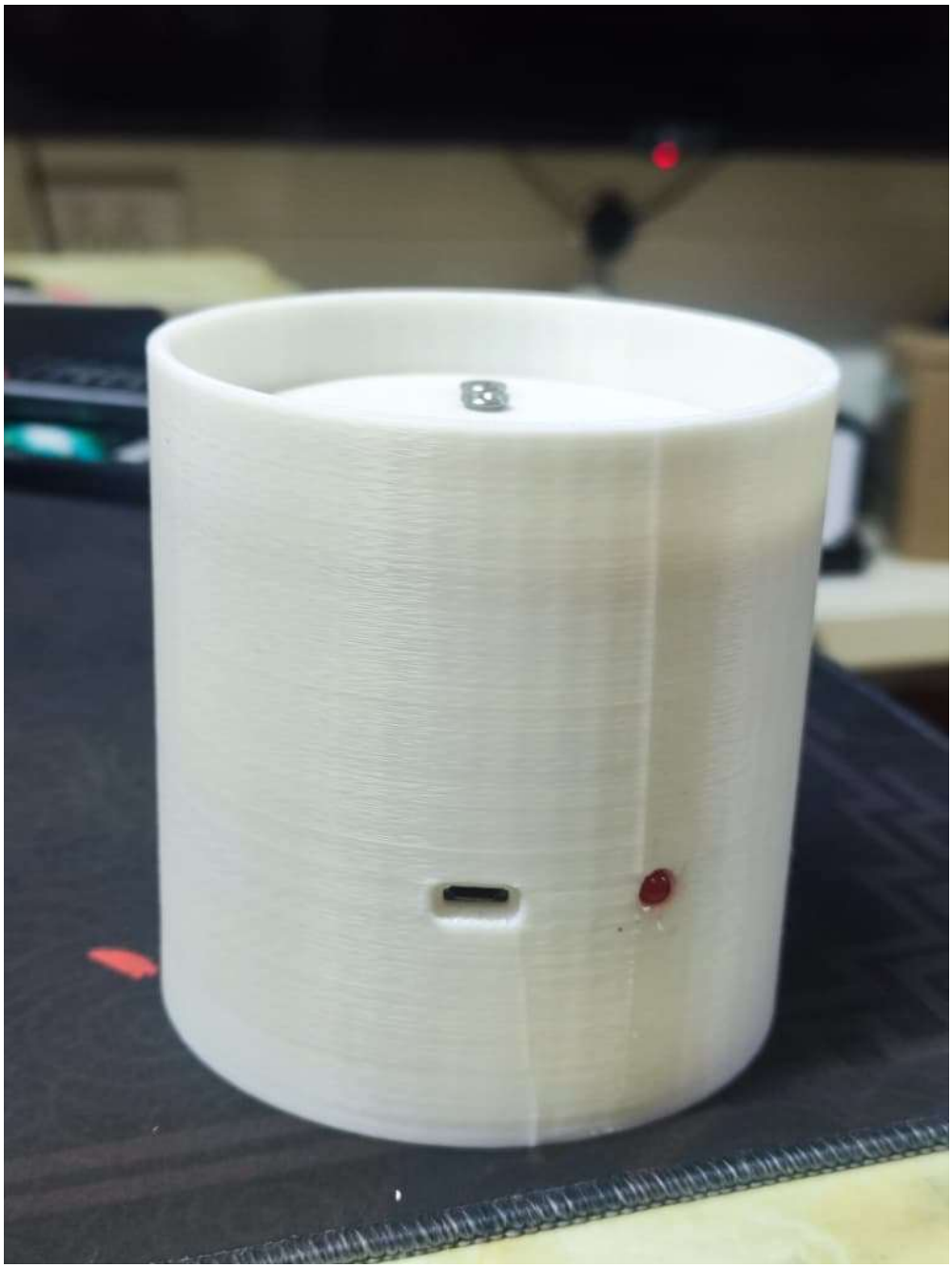


Figure 5.14 Assembled System with Outer Case (Real-World View)

5.2 Software Setup

The software setup involves preparing both the microcontroller programming environment and the web-based frontend hosting environment. The development platform used in this project was the Arduino IDE, which provides a user-friendly interface for coding and uploading firmware to the ESP32 microcontroller.

In this project, a laptop was used to develop the code for the ESP32 using Arduino IDE. The laptop also served to monitor the ESP32 output using the Serial Monitor and flash the firmware into ESP32 through COM7. The table 5.2 shows the specifications of the laptop used in this project.

Table 5.2 Specification of Laptop

Description	Specifications
Model	Lenovo Legion 5i
Processor	Processor: AMD Ryzen 5 5600H
Operating System	Windows 11
Graphic	NVIDIA GeForce RTX3060 6GB
Memory	16GB DDR4 RAM
Storage	1.5TB SSD

For the ESP32 setup, the ESP board package was first installed in the Arduino IDE through board manager by adding the Espressif repository link: "https://raw.githubusercontent.com/espressif/arduinoesp32/ghpages/package_esp32_index.json". After installation, select the ESP32 Dev Module from boards manager and set the upload speed to 115200 baud rate [20]. The required libraries were then installed using the Arduino Library Manager and internal library. The key libraries used in the project include:

1. Adafruit ADXL345 – to handle accelerometer for orientation detection
2. ArduinoJson – to transmit data to Firebase in JSON format
3. Firebase Arduino Client Library for ESP8266 and ESP32 – to send data to Firebase Realtime Firebase
4. HX711 Arduino Library – to interface with the load cell amplifier
5. WiFi and WiFiMulti – to establish and manage multiple Wi-Fi connections
6. Preferences – to store offline data in ESP32's non-volatile memory
7. Wire – to configure the I^2C for ADXL345 accelerometer

In addition, the frontend interface was hosted on Firebase Hosting. To set up this, Node.js and Firebase CLI were installed on the development computer. Once the Firebase initialized, it will generate the Firebase configuration that link to the created Firebase project. The HTML, CSS and JavaScript files for the dashboard were placed in the hosting folder and the deployment was performed using the "firebase deploy" command.

Once completed, the ESP32 firmware and the frontend hosting were connected through the Firebase Realtime Firebase, enabling the hardware to upload water intake and voltage data, while the frontend retrieved and visualized the information in real-time.

5.3 Setting and Configuration

After completing the installation of the development environment and deployment of the frontend, the system required further customization and tuning to ensure the system functionality. This section describes the configuration steps carried out for the ESP32 microcontroller, database, synchronization, sensors, frontend and system threshold.

5.3.1 ESP32 Wi-Fi Configuration

The ESP32 was configured to connect to a Wi-Fi network by embedding the SSID and password inside the program code. To support multiple network sources, the WiFiMulti library was utilized, enabling the device to automatically switch to other alternative connections in case of disconnection. This ensured continuous data transmission to the Firebase database without manual setup. The system can setup multiple Wi-Fi connections with the command: wifiMulti.addAP("SSID","Password");

5.3.2 Firebase Database Configuration

The ESP32 was linked to Firebase by insert the project URL and authentication key into the program code. Two structured data paths were define: one was waterIntake with having two key parameter: weight and timestamp, another was voltageBattery with having two key parameter: battery voltage and timestamp. Database rules were also set to regulate read and write permissions, ensuring secure communication between the microcontroller and Firebase. To enhance data security, authentication can also be enabled and only verified accounts can access to the Firebase. The code to set up the Firebase in ESP32 as follow:

```
  config.api_key = API_KEY;
  config.database_url = DATABASE_URL;
  auth.user.email = "Email Account";
  auth.user.password = "Password";
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
```

5.3.3 NTP Configuration

NTP was implemented to get an accurate timestamps for recorded data. The time zone was configure to GMT +8 (Malaysia Time) and the synchronization interval was set to periodically update the ESP32 internal clock. This allows the offline data, can store the timestamp as real-world time in offline storage.

The NTP configuration was as follows:

```
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = 8 * 3600;  //GMT+8
const int daylightOffset_sec = 0;

configTime(gmtOffset_sec,daylightOffset_sec,"pool.ntp.org","time.google.com","time.cloud
flare.com");
struct tm timeinfo;
int retry = 0;
while (!getLocalTime(&timeinfo) && retry < 5) {
  Serial.println("Failed to obtain time");
  delay(1000);
  retry++;
}
```

5.3.4 Sensor Calibration

Calibration was performed on all sensing module to improve measurement accuracy. The HX711 load cell amplifier was tared using calibration factor to establish a zero baseline before water intake measurements. The ADXL345 accelerometer was tested in multiple orientations, with the z-axis expected to produce reading of approximately 9-10 while water bottle was in

upright position. The voltage sensor was calibrated by mapping the ADC readings to actual battery voltage levels using known reference values.

5.3.5 Frontend Configuration

The dashboard was linked to Firebase using the JavaScript SDK. The system was configured to retrieve new data and update the water intake chart to ensure real-time visualization. Default goal values for daily, weekly and monthly water intake goals can also be adjusted by the user as needed. Additionally, the system also included a personalization feature that allowed users to convert the data from ml to Oz.

5.3.6 System Thresholds

Several thresholds were implemented to enhance data accuracy and reliability. A minimum change of 30 ml was required in load cell reading before registered as a valid water intake event, effectively filtering out noise caused by minor movement. To further enhance the data reliability, the system will take 3 additional reading and compared to the initial difference weight, if the variation between these readings did not exceed ±2g, it will only registered as a valid drinking event. The low battery threshold was defined at 3.5V, triggering a warning when the battery voltage dropped below this level. Additionally, the frontend application was set to detect offline conditions if no updates were received in voltageBattery structure data within 10 minutes, ensuring timely alerts for connectivity issues.

5.4 System Operations

This section will demonstrates the working process of the Smart Water Tracking System, covering both the hardware operation and the frontend interface. Screenshots and photos will included to illustrate each stage of process.

5.4.1 System Startup

When the ESP32 boots up, the system first tares the load sensor and initializes the voltage sensor and the ADXL345 accelerometer. After initialization, the ESP32 attempt to connect to available Wi-Fi connection. If there was available Wi-Fi, the ESP32 communicates with NTP server to synchronize the system time then upload the stored offline data to Firebase database if exist. If Wi-Fi connection fails, the system will switch to offline mode, skipping the Wi-Fi connection and Firebase configuration. In this mode, valid water intake event were stored

locally in NVS. Figure 5.15 shows the ESP32 connected to a mobile phone hotspot. Furthermore, Figure 5.16 show the Serial Monitor output during successful Wi-Fi connection , while figure 5.17 shows the ESP32 operating in offline mode.

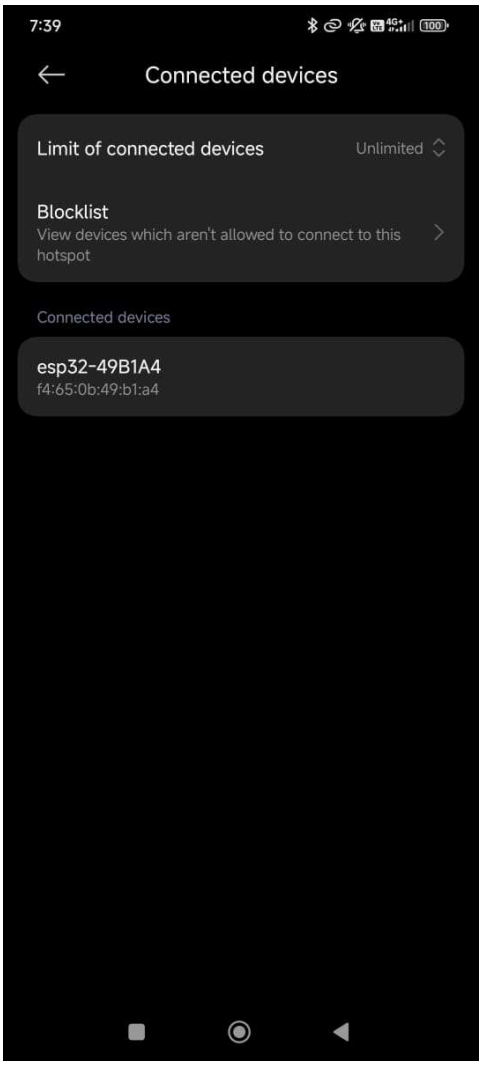


Figure 5.15 ESP32 connected to phone hotspot

```
ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:4888
load:0x40078000,len:16516
load:0x40080400,len:4
load:0x40080404,len:3476
entry 0x400805b4
Initializing the scale
Connecting Wifi...

WiFi connected
10.145.254.2

Firebase connected
Failed to obtain time
Failed to obtain time
Failed to obtain time
Time initialized successfully
Current time: 2025-09-13 21:32:38
No offline data to process
No local data to upload.
```

Figure 5.16 ESP32 Serial Monitor during successful Wi-Fi connection

```
ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:4888
load:0x40078000,len:16516
load:0x40080400,len:4
load:0x40080404,len:3476
entry 0x400805b4
Initializing the scale
Connecting Wifi...
WiFi not connected. Trying to reconnect...

WiFi not connected. Skipping firebase and time sync
Wifi not connected
No significant change | Current: -0.75
```

Figure 5.17 ESP32 Serial Monitor output in offline mode

5.4.2 Orientation Detection

During normal operation, the ADXL345 accelerometer continuously check the water bottle orientation. If the z-axis value falls outside the expected range of 9-10, the system interprets the bottle as tilted or lying down form and weight measurement are temporarily disabled. If the bottle was in upright form (z-axis within the range), it will proceed to load cell reading. Figures 5.18 and 5.19 show when the ADXL345 accelerometer placed in a vertical orientation and the output of ESP32 in Serial Monitor. Meanwhile, Figures 5.20 and 5.21 shows when the ADXL345 accelerometer back to upright form and the corresponding output in Serial Monitor.

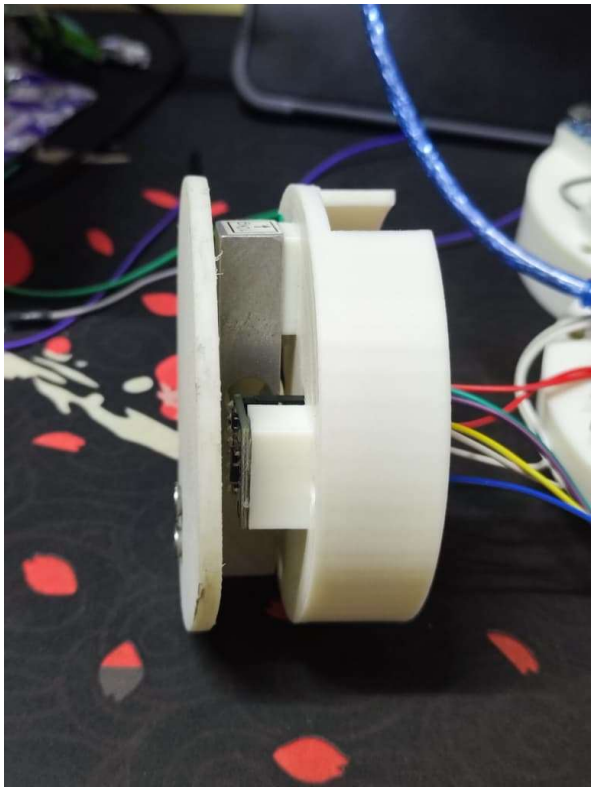


Figure 5.18 ADXL345 accelerometer in vertical orientation (real-world)

```
No significant change | Current: -0.07
Water Bottle was not in upright, weight measurement process suspend.
Water Bottle was not in upright, weight measurement process suspend.
```

Figure 5.19 ESP32 Serial Monitor output for vertical orientation

Figure 5.20 ADXL345 accelerometer in upright orientation (real-world)

```
Water Bottle was not in upright, weight measurement process suspend.
No significant change | Current: -0.07
No significant change | Current: -0.07
```

Figure 5.21 ESP32 Serial Monitor output for upright orientation

5.4.3 Water Intake Detection, Data Logging and Uploading

HX711 amplifier and load cell were responsible for measuring the bottle's weight every 3 seconds. If the weight difference that bigger than 30g was detected, the system performs additional 3 more reading at 0.3 second interval to ensure data accuracy and reliability. If the variation between these readings within ±2g of the original weight difference, the system will recorded the event as a valid drinking/refilling action. The negative values data represents as the child drink amount and positive value represent the amount of water refilled into the bottle. Figure 5.22 shows the Serial Monitor of ESP32 together with the laptop system time during a weight change event.

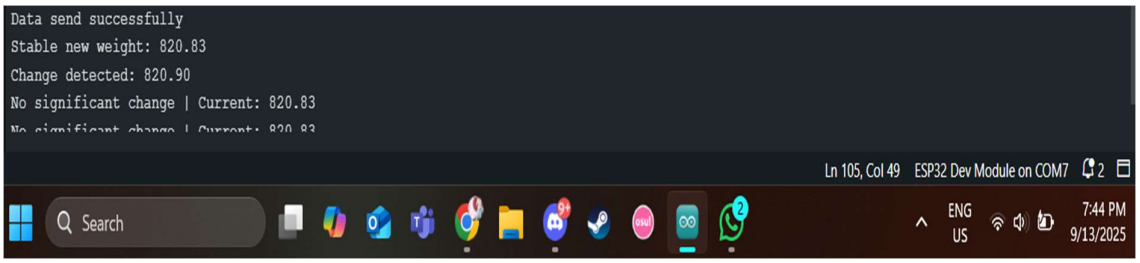


Figure 5.22 Serial Monitor output showing water intake detection and laptop system time

The water intake measurement in Figure 5.22 was done in online mode, the valid event was immediately uploaded to the Firebase database. The logging result at the Firebase console was illustrated in Figure 5.23.

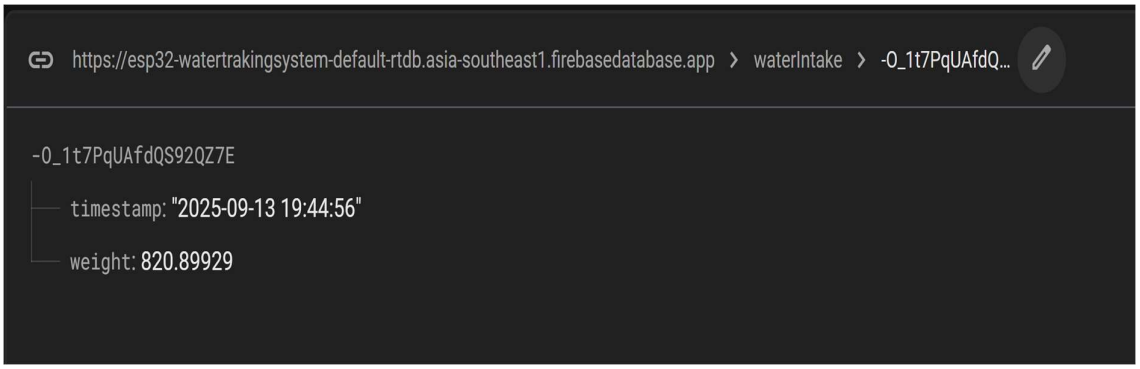


Figure 5.23 Water intake event successfully uploaded and logged in Firebase Console

In addition to online logging, the system also supports offline data storage when Wi-Fi is not available. The system relies on the NTP as the system time. To validate this offline storage mechanism, 2 water intake events were recorded while operating in offline mode. Figure 5.24 and 5.25 shows the valid water intake event, message that mention the data was stored into local storage with valid timestamp in Serial Monitor and the laptop time where the measurement was taken.

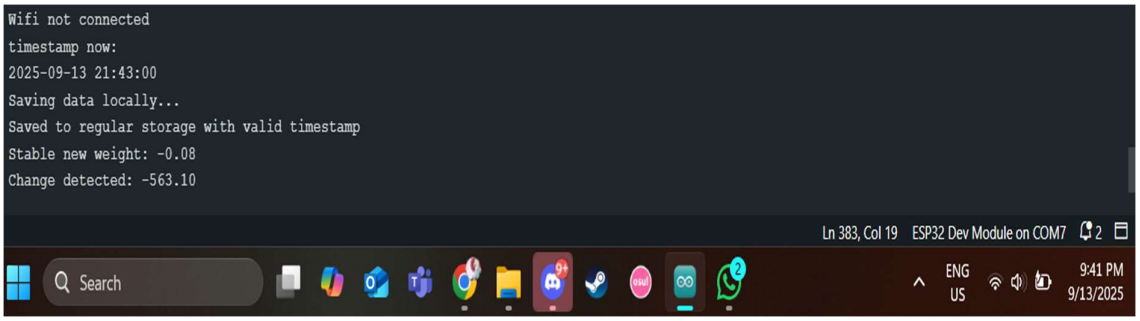


Figure 5.24 Serial Monitor output showing valid drinking event stored locally with NTP timestamp

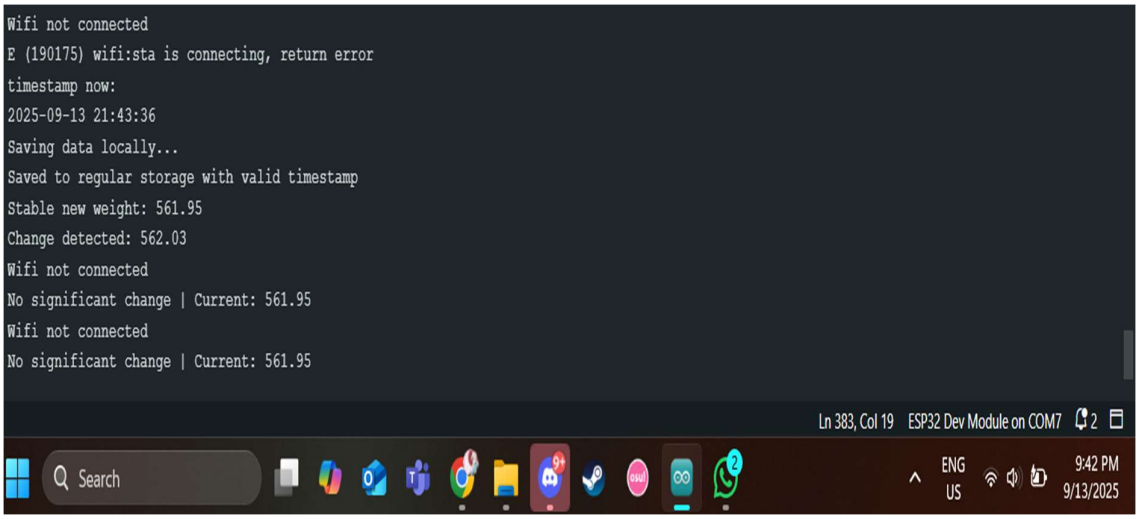


Figure 5.25 Serial Monitor output showing valid refill event stored locally with NTP timestamp

Afterward, the ESP32 was switch back to online mode. The system will automatically detected the stored event in offline storage and upload them to Firebase Database. The Serial Monitor output of this process was shown in Figure 5.26, while Figure 5.27 presents the successfully logged data in Firebase Console.

```
Wifi reconnected

Firebase connected
Time initialized successfully
Current time: 2025-09-13 21:44:05
No offline data to process
Uploading 2 local entries in main storage...
Uploaded data0 successfully.
Uploaded data1 successfully.
Local storage cleared.
```

Figure 5.26 Serial Monitor output showing offline data being uploaded when Wi-Fi reconnects

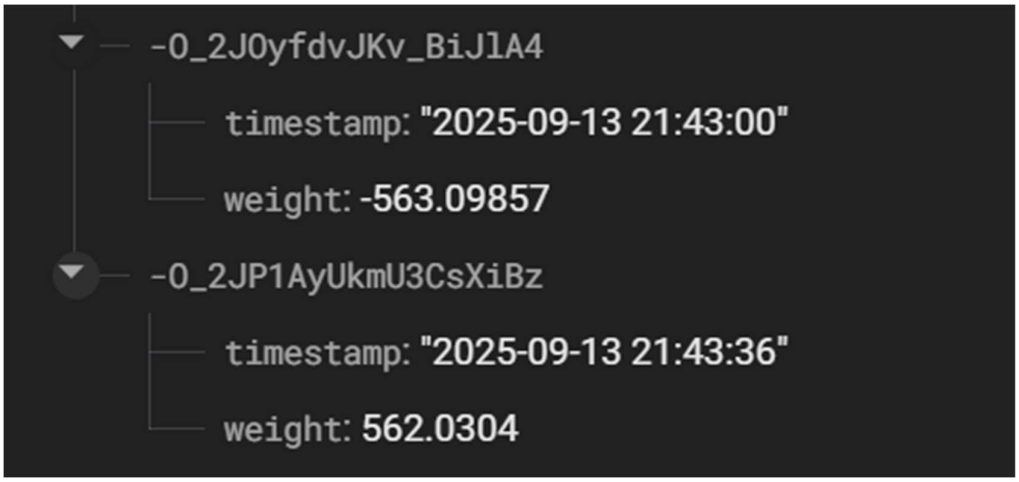


Figure 5.27 Firebase console showing uploaded offline events

Lastly, the system was also tested under a difference offline condition where the ESP32 was initialized without Wi-Fi and Firebase configuration just like Figure 5.17. In this scenario, NTP synchronization could not be perform since it need Wi-Fi connection and therefore the system will temporarily store the ESP32 runup time as timestamp of the water event. In the testing setup, 2 data were collected by ESP32 under this situation. Figures 5.28 and 5.29 illustrated the Serial Monitor Output with laptop system time, showing how the data was captured with unknown timestamp.

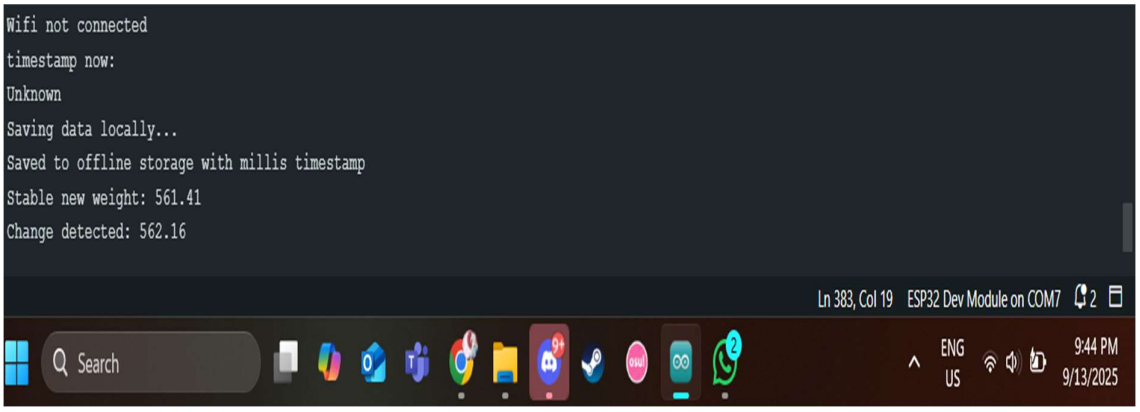


Figure 5.28 Serial Monitor output showing offline data stored with temporary run-up timestamp (event 1)

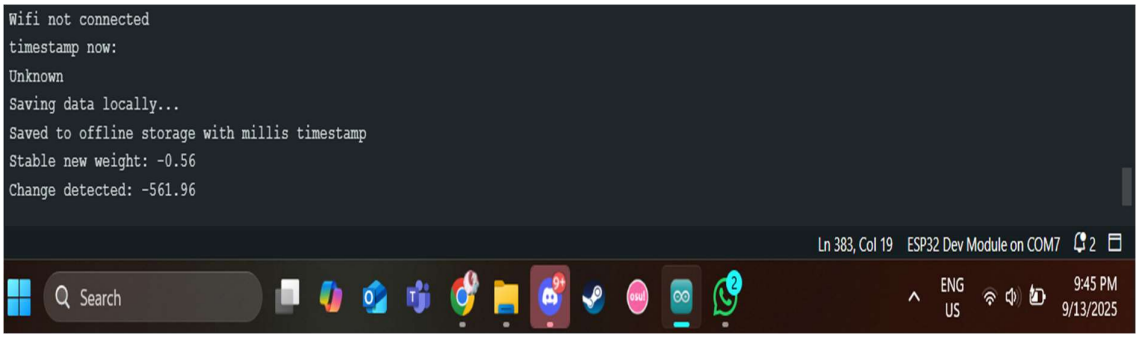


Figure 5.29 Serial Monitor output showing offline data stored with temporary run-up timestamp (event 2)

Once Wi-Fi connectivity was restored, the ESP32 configured Firebase and synchronized time from NTP server. After time had been initialized, it will processed the data with system runup time as temporarily timestamp. Figure 5.30 the Serial Monitor output of ESP32 during this process. Figure 5.31 shows the corrected time events successfully logged in Firebase.

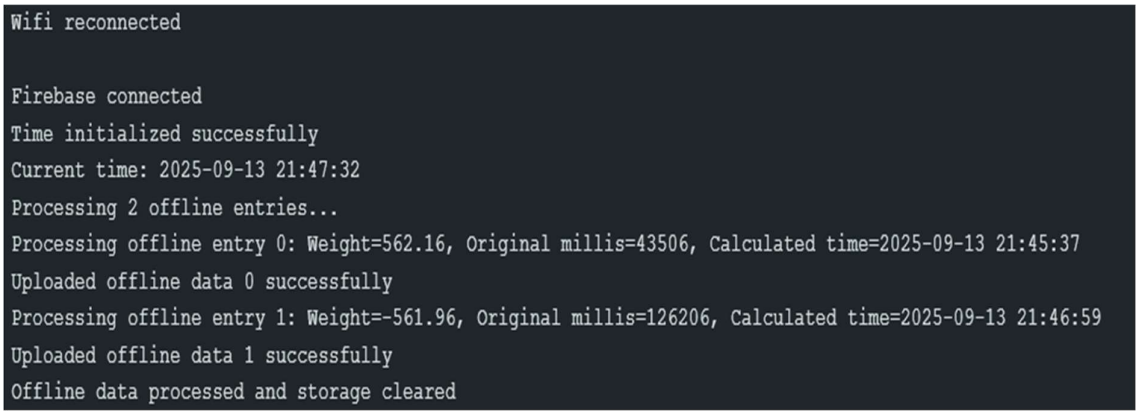


Figure 5.30 Serial Monitor output showing offline data being processed and timestamp corrected

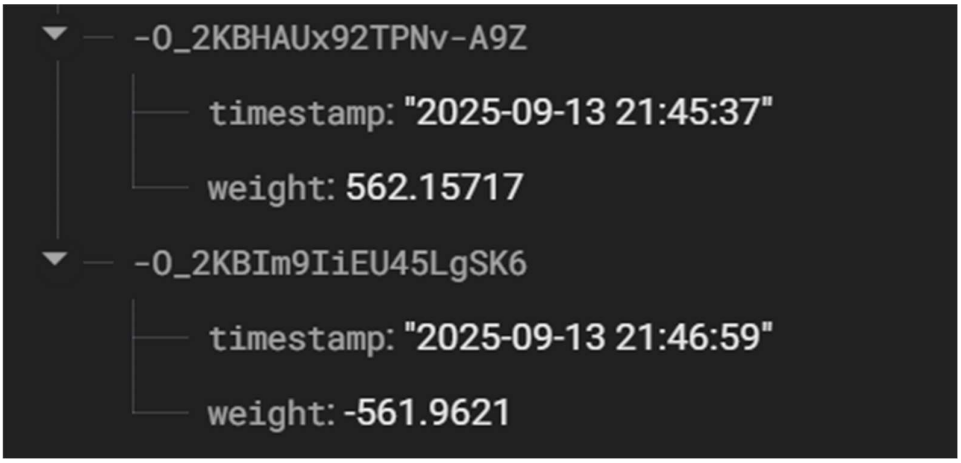


Figure 5.31 Firebase console showing offline data uploaded with corrected NTP timestamps

5.4.4 Battery Monitoring

When the ESP32 work in online mode, it starts multitasking by running task to continuously monitor the battery voltage. The voltage sensor read the battery level through ESP32 GPIO32 pin. If the detected voltage was lower than the threshold of 3.5V, the system triggers a low-battery alert. The red LED indicator will blinks 5 times per second to notify the user of low voltage condition. The voltage battery level and timestamp will uploaded to Firebase. Figure 5.32 shows Serial Monitor output when battery level low than 3.5V while Figure 5.33 illustrated the red LED indicator blinking as a visual alert for the user. Figure 5.34 presents the logged battery voltage data uploaded to Firebase console.

```
Battery Voltage: 2.74 V
No significant change | Current: -0.76
Voltage data sent to Firebase successfully.
```

Figure 5.32 Serial Monitor output when battery voltage drops below 3.5 V

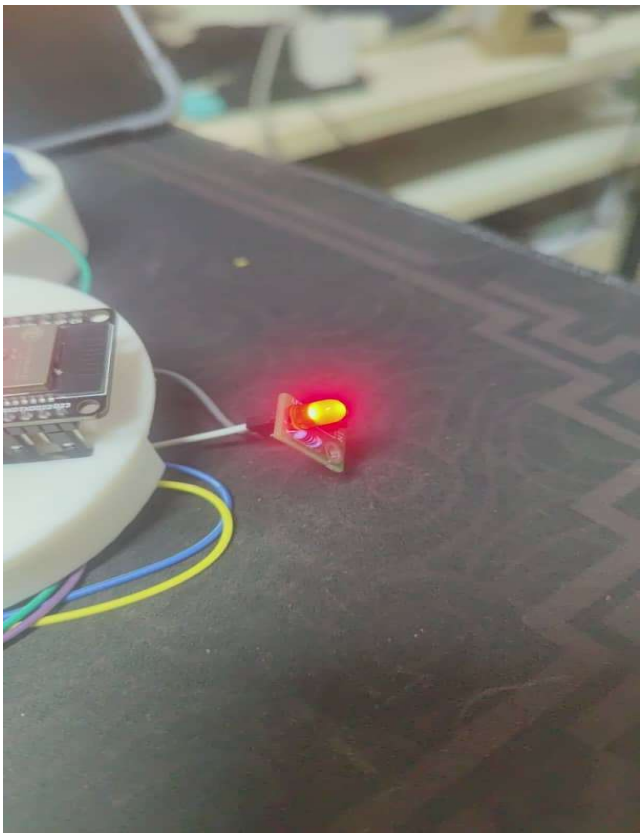


Figure 5.33 Red LED indicator blinking (low battery alert)

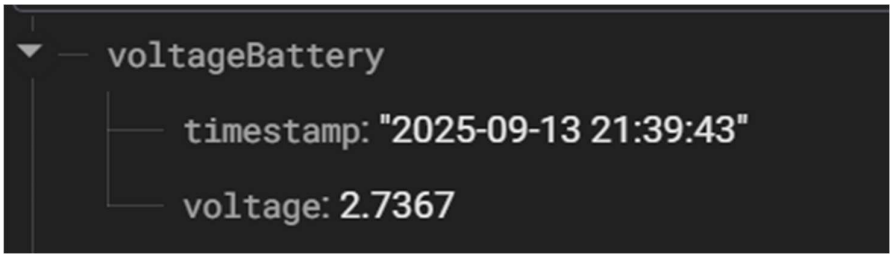


Figure 5.34 Battery voltage data logged in Firebase console

5.4.5 Frontend Display

The frontend dashboard was designed to provide real-time visualization of the water intake data retrieved from the Firebase database, Once the ESP32 uploaded the valid drinking and refill events, the data was automatically synchronized with the frontend using the Firebase JavaScript SDK.

The main page of the frontend displayed a welcoming message. In the navigation bar, the user allows to switch to daily, weekly and monthly page and view the summary. On the right-hand side, a setting button was provided for user to do customization. Besides, it also included a battery indicator to show the current battery status and an online/offline status icon to indicate the connectivity of the ESP32 system. Figure 5.35 show the main page layout and the the navigation bar.

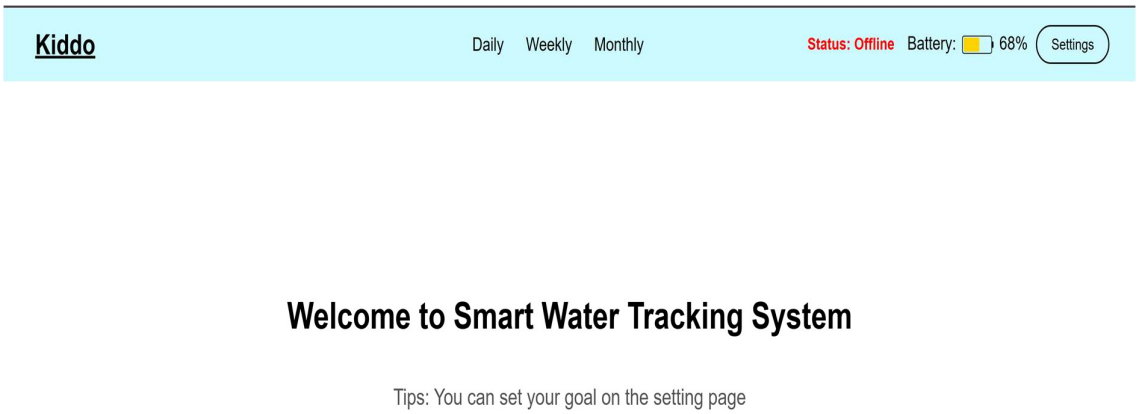


Figure 5.35 Main page layout and navigation bar

The dashboard included multiple features to improve user interaction and monitoring. Daily, weekly and monthly charts were plotted to provide a clear overview of hydration patterns across different timeframes. User was allowed to select specific time or time ranges to view the hydration patterns and data either in ml or Oz form. The goal achievement was also employed to track the hydration status, while data that exceed the goal target will be show in green colour, the data that below the goal target but within 500ml of reaching it were shown in pink colour and the data that did not achieved the goal target was shows in red colour.

In the daily view, the chart shows the water intake event only excluding refills for clarity. In the summary section, it displayed the number of refills, total water consumed and goal achievement status. Figure 5.36 illustrates the daily summary view.

Figure 5.36 Daily summary view (ml)

For weekly and monthly views, the hydration goal target was distributed across 7 and 30 days respectively. The frontend compared the amount of water intake with the goal target for the selected period and highlighted the hydration status for each day. Furthermore, the summary box in weekly and monthly shows the total amount of water drinks, average daily

intake and goal achievement status. Figure 5.37 and 5.38 shows the view of weekly and monthly page.

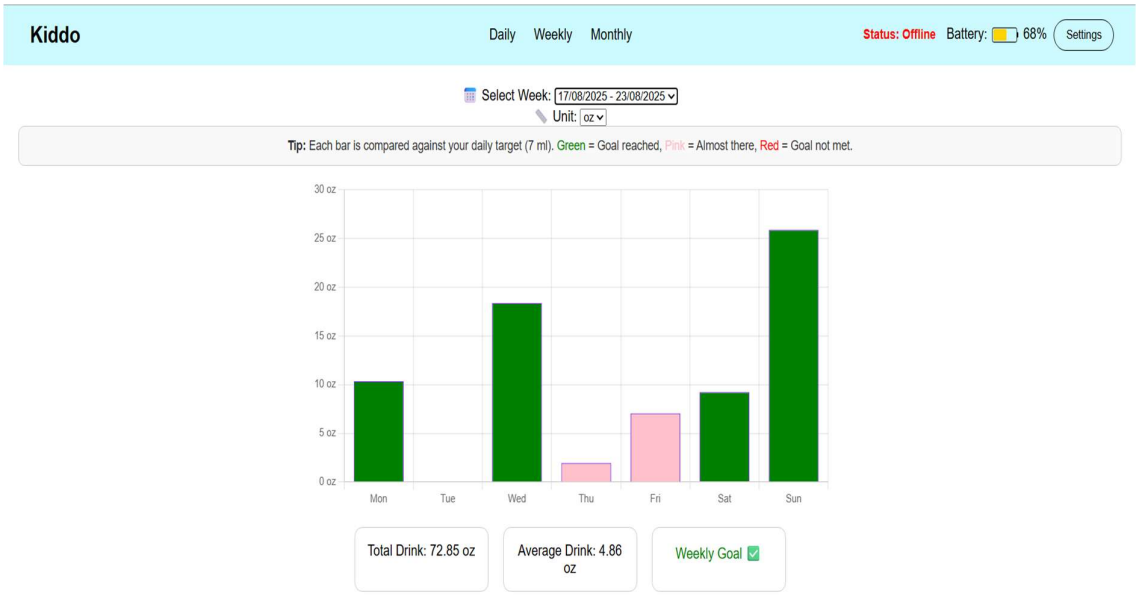


Figure 5.37 Weekly summary view(Oz)

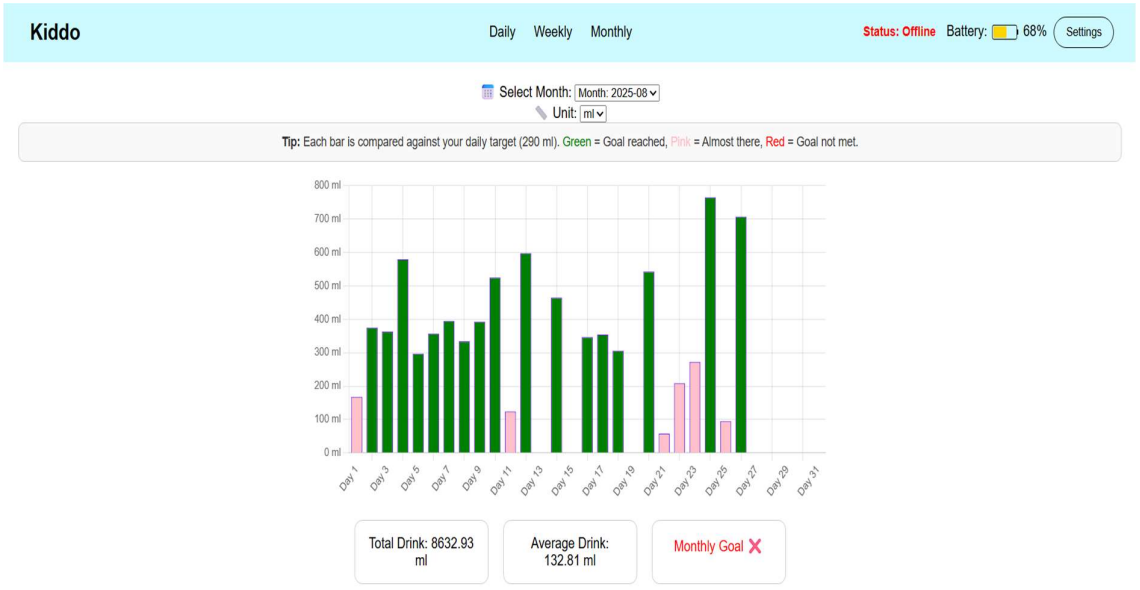


Figure 5.38 Monthly summary view (ml)

In setting page, the users were also allowed to change the font size of the page for better readability. Figures 5.39 and 5.40 demonstrate the difference between the medium and large font size options.

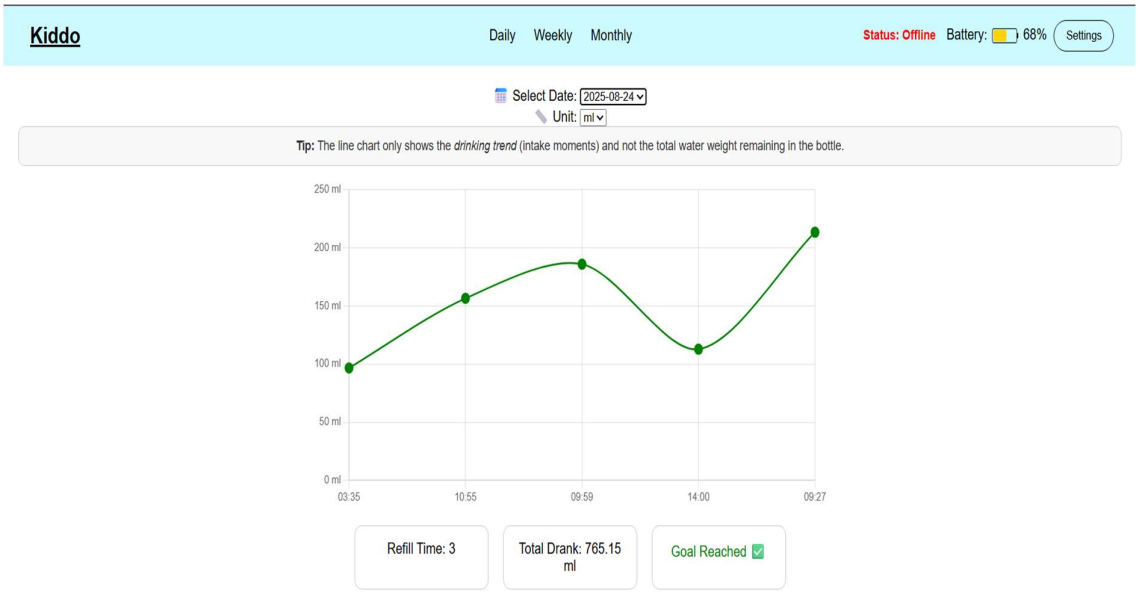


Figure 5.39 Font size in medium form

Figure 5.40 Font size in large form

Additionality, the frontend provided customization options for user to change the hydration goal target for daily, weekly and monthly. When a value was changed, a save button appeared to update the goal setting and store in local storage. Figure 5.41 shows the goal customization interface.

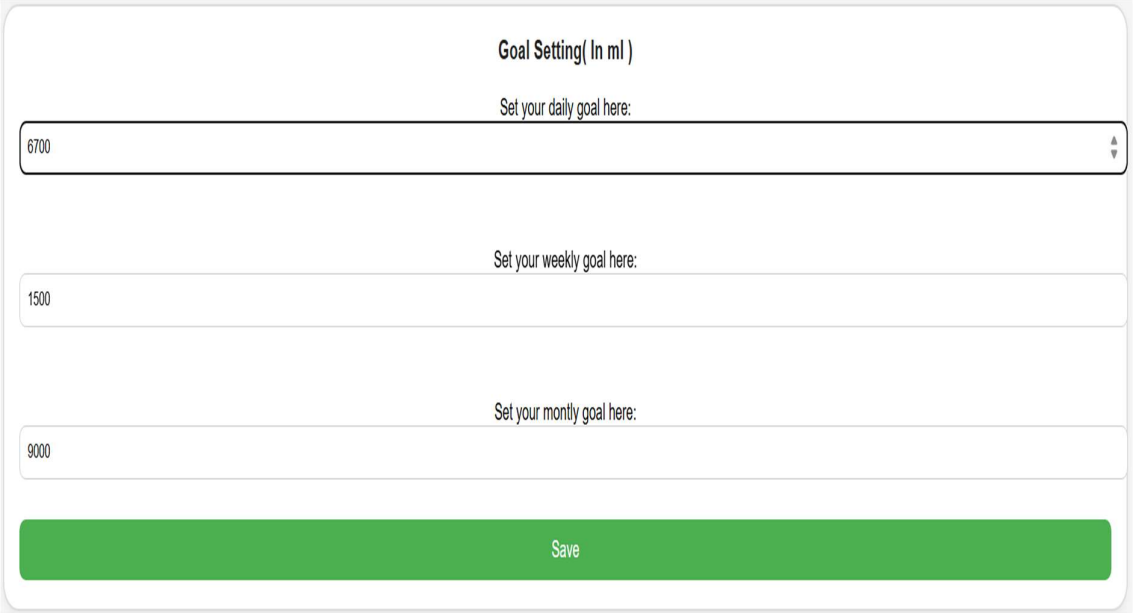


Figure 5.41 Goal customization setting

The frontend also provided a feature for deleting all cloud data. For testing purpose, instead of using the actual data structure, a new data structure call “sample” was created to verify the functionality. The Figures 5.42 shows the new data structure in Firebase Console and figure 5.43 displayed the confirmation dialog on the frontend page.

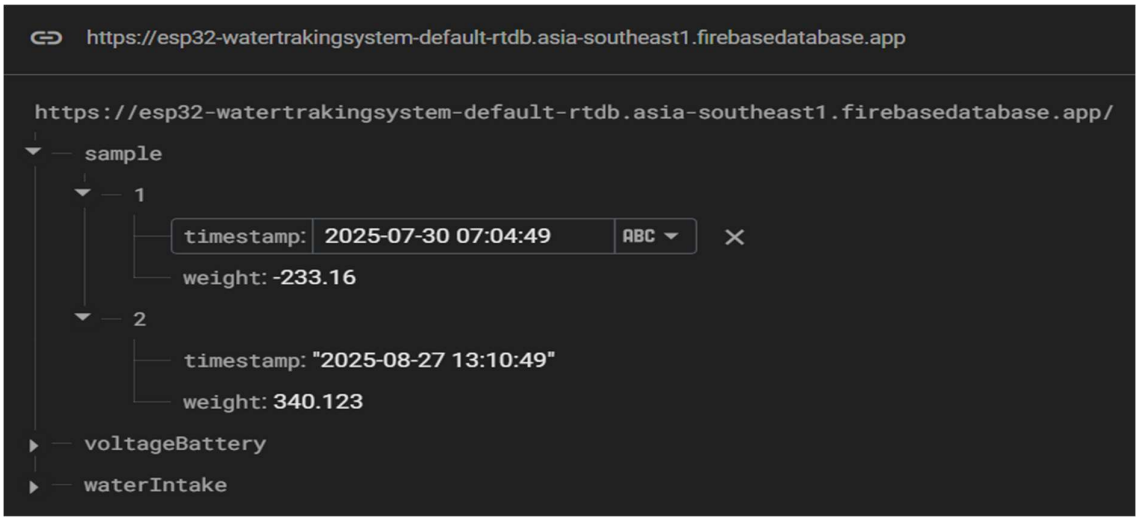


Figure 5.42 New sample data structure with data in Firebase Console

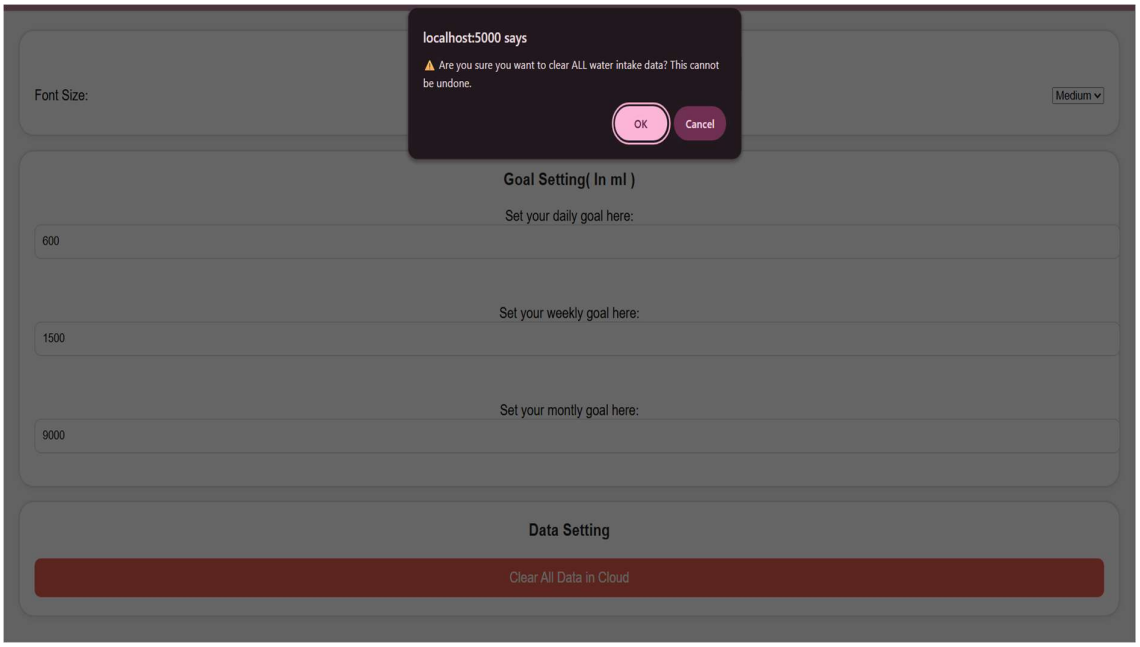


Figure 5.43 Confirmation Message for data deletion

Once confirmation, the data structure was permanently deleted and could not be be restored. Figure 5.44 shows Firebase Console after the data structure had been cleared.

Figure 5.44 Firebase Console after clear the data structure

5.5 Implementation Issues and Challenges

During the development of the Smart Water Tracking System, several issues and challenges were encountered during hardware and software implementation. These challenges required several time of testing and design adjustment to ensure the system could operate reliably.

Sensor Calibration and Stability

The HX711 load cell measurement was affect the data accuracy by environment factors. The load cell measurements will having a tolerance of +/- 5% due to several factors such as drift, high surface temperature and electromagnetic interference from nearby electronic devices.[21]

The system has implemented a stability check mechanism to ensure the data reliability and reduce the impact of these environmental factors .

False Reading When Bottle Was Tilted

In certain condition, user did not always place their water bottle in upright position, such as during sporting activities or putting inside a bag, false reading was captured and uploaded to frontend, leading to confusion for users. To solve this issue, ADXL345 accelerometer was implemented into the system to monitor bottle orientation and prevent captured false reading.

Delay on getting timestamp from NTP server

The whole system was heavily rely on NTP for getting timestamp. However, during initialization, the NTP server may experience delay or failure due to poor internet connection or the NTP server was having peak usage time. To address this challenge, a retry mechanism was implemented, although it can solve the issues but it still having a long-time delay to get the time.

Power Supply and Battery Monitoring

The ESP32 required a stable 6V input from the 18650 battery to initialize the Wi-Fi connection and power other components. Therefore, a DC-DC voltage boost converter was used to step up the 18650 Li-ion battery to 6V and voltage divided was implemented to monitor the battery status. Calibration was needed for voltage sensor to accurately captured low-battery conditions.

Frontend Visualization and Customization

Initially, both positive and negative value were shown in the chart, resulting in refill and intake actions was being shown together. This caused confusion in data interpretation. The data handling logic was refined to filtered out the positive value and only show negative values data in the chart. Furthermore, goal customization values were not saved after page refresh. This problem was solved by using local storage in the frontend to prevent data missing.

User Interface Accessibility

The default font size was not suitable for all users. Thus, a customization feature was added to the setting page allowing user to adjust the font size of the page, improving accessibility for children and parents.

Overall, these challenges were being solved during the development through testing, debugging and implement new sensor. This solution contributed to a more stable and user-friendly system.

5.6 Concluding Remark

This chapter show the detailed implementation of the Smart Water Tracking System, covering both hardware and software aspect. The discussion in this chapter included system setup, configuration and workflow of the system followed by the frontend webpage development for data visualization and user interaction. Implementation challenges were also being identified, along with the solutions adopted to ensure system full functionality, accuracy, reliability and user-friendly.

Overall, the system was successfully implemented and integrated, achieving the functionality of monitoring water intake, handling offline and online data synchronization and providing an interactive webpage for users.

The following chapter will evaluate the system's performance through testing and validation. Metrics such as data accuracy, response time, reliability and user interface functionality will be examined to ensure the system meet the project objectives.

Chapter 6

System Evaluation And Discussion

6.1 System Evaluation and Performance Metrics

To evaluate the effectiveness of the Smart Water Tracking System, a set of tests were carry out. The testing aimed to verify both the functional correctness and the performance of the hardware, software and data visualization components. Performance metrics were defined to evaluate how well the system achieved it objectives.

Performance Metrics Considered

- **Accuracy (%)** = $\frac{Measured\ Value}{Actual\ Value} \times 100$
- **Data Loss Rate (%)** = $\frac{Missing\ Entries}{Total\ Entries} \times 100$
- **Synchronization Delay (s)** = Time required to upload offline data once Wi-Fi reconnected

6.1.1 Load Sensor Accuracy Test

The HX711 load sensor was tested to determine measurement accuracy under different known weight. The actual weight was measured manually using electronic scale and compared with sensor readings.

Table 6.1 Load Sensor Accuracy Test

Trial	Actual Weight(g)	Sensor Reading (g)	Error (g)	Accuracy(%)
1	44	43.82	-0.18	99.6
2	232	233.23	1.23	100.53
3	486	488.56	2.56	100.53
4	57	57.94	0.94	101.65
5	217	211.24	-5.76	97.35
Avg	-	-	-0.242	99.93

The average accuracy was 99.93%, which was acceptable for hydration tracking purpose. Errors observed (± 0-6) were negligible in comparison to daily hydration goals, confirming the data accuracy and stability of the load sensor with calibration used.

6.1.2 Data Synchronization Test

The system supports both offline logging using NVS storage and online synchronization with Firebase. Test were conducted to check if the number of data that stored in offline storage can be successfully upload to Firebase once the Wi-Fi connection was restored.

Table 6.2 Data Synchronization Test

Condition	**Number of Data Entries taken in offline**	**Number of Data Uploaded after Reconnect**	**Data Loss(%)**	**Average Sync Delay**
5 mins offline	15	15	0	3
15 mins offline	30	30	0	2.5
30 mins offline	40	39	2.5	4.8

The synchronization process achieved a 97.5% reliability rate, with only one data loss at the 30 minutes offline condition. The delay in syncing was short, ranging between 2.5 to 4.8 seconds, showing the efficiency of the retry mechanism. The 5-minute offline condition sync delay was greater than 15 mins offline was due to the delay of NTP server to get timestamp to update the ESP32 system time.

6.1.3 Orientation Test

The ADXL345 accelerometer was also be tested to verify its ability to detect bottle orientation. Test were conducted by placed the ADXL345 accelerometer in different position and the rate of successfully disabled the weight measurement was not in upright position.

Table 6.3 Orientation Test

Test Case	**x-axis**	**y-axis**	**z-axis**	**Measure Result**
Upright	0	-0.67	9.38	Weight recorded
Vertical	-9.61	-2.16	-0.67	Measurement disabled

Downward	0.27	-0.39	-10.16	Measurement disabled
Tilted (45 Degree)	-6.55	-0.67	6.75	Measurement disabled

The ADXL345 accelerometer successfully distinguish between upright and non-upright orientations. Weight reading was only taken when the orientation was detected at upright position, preventing false water intake events. This confirmed that the orientation checking mechanism able to enhance the system reliability in real-word usage.

6.1.4 Frontend Goal Achievement Test

The frontend was tested to verify whether goals status and water intake summaries were displayed correctly. The system should clearly indicate whether the user had met their hydration goal.

Table 6.4 Frontend Goal Achievement Test

Type	**Day**	**Goal (ml)**	**Actual Intake (ml)**	**Goal Reached**
Daily	24/8/2025	600	765.15	Yes
Weekly	17/8/2025– 23/8/2025	1500	2154.3	Yes
Monthly	08-2025	9000	8632.93	No

The frontend was able to display correct goal achievement status. In case where the total water intake was slightly below the target, the system also can highlighted it appropriately. This ensures parents or teacher can receive clear feedback on keeping hydration progress.

6.1.5 Water Intake Event Detection

The raw data process in frontend was tested to verify the system can whether separate the negative and positive value correctly. This mechanism was tested by using an amount of 100 data entry, where 50 for positive values and 50 for negative values.

Table 6.5 Confusion Matrix

		Actual Values	
		Positive	Negative
Predicted Values	Positive	50	0
	Negative	0	50

The confusion matrix had shown the raw data processing mechanism had successfully identified all the data into positive and negative value. This ensures the system can provide a reliable visualization data for user to view.

6.1.6 Overall Findings

The above test demonstrates that the Smart Water Tracking System had meets the expected performance requirements with:

- Data Accuracy exceed 99%
- Data synchronization was highly reliable and minimal risk of data loss
- Orientation checking was reliable and prevent false reading
- Frontend feedback was accurate and easy to interpret

These outcomes confirm that the system was suitable for real-world use and follow the project objective of promoting proper hydration for children.

6.2 Testing Setup and Result

6.2.1 Testing Environment

The smart water tracking system was tested using the following setup:

1. Hardware Components: HX711 amplifier, 10kg load cell, ADXL345 accelerometer, voltage sensor, TP4056 charging module, DC-DC voltage boost converter, 18650 Li-ion Battery
2. Software Component: Arduino IDE for ESP32 programming, Firebase Realtime Database as cloud storage, frontend webpage connected via Firebase SDK
3. Testing environment: indoor lab setting with stable Wi-Fi connection. For offline test, Wi-Fi was temporarily disabled to evaluate local storage and synchronization performance.

4. Reference Tools: Electronic Scale for weight comparison, laptop system time for timestamp verification, Serial Monitor for debugging logs.

Testing was conducted in different conditions to simulate real-life usage, including online mode, offline mode and low battery condition.

6.2.2 Load Cell Accuracy Test

The purpose of this test was to measure the accuracy of water intake/refill action. The method was used a known weight object to place on the load sensor and compared with the actual values. Figure 6.1 shows the water bottle weight using electronic scale.

Figure 6.1 Water bottle weight using electronic scale

Then, we placed the water bottle with known weight on the load sensor and view the data in Serial Monitor. Figure 6.2 illustrates the water bottle weight by using load sensor.

```
No significant change | Current: 0.30
Data send successfully
Stable new weight: 969.14
Change detected: 968.83
No significant change | Current: 969.14
```

Figure 6.2 Water Bottle Weight using load sensor

From the figure, the load sensor will having a error of 4.83ml compared to the actual weight of the water bottle. The accuracy of the data was 100.5%, which was acceptable for hydration monitoring.

6.2.3 Data Logging and Offline Storage Test

The test was conducted to verify the reliability of data recording in both online and offline modes. The weight was continue using from section 6.2.2. The water intake event was stimulated at online mode. Figure 6.3 showed that 87.64ml had been consumed along with the laptop system time.

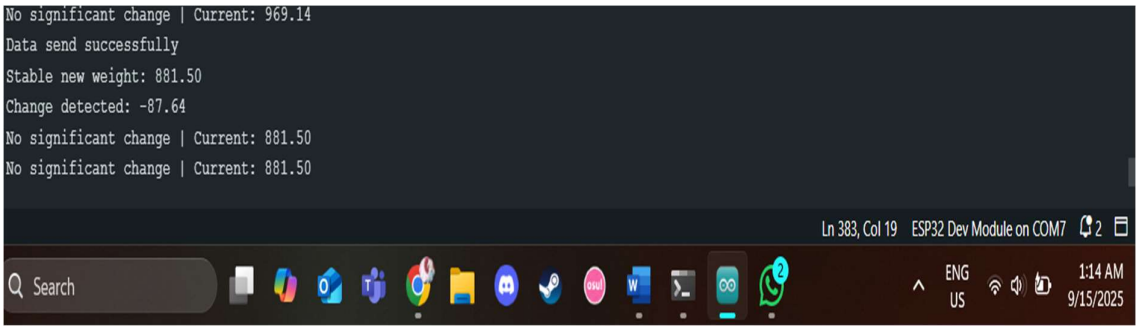


Figure 6.3 Amount drink and system time

Figure 6.4 illustrate the data had been synchronized to Firebase successfully at Firebase Console.

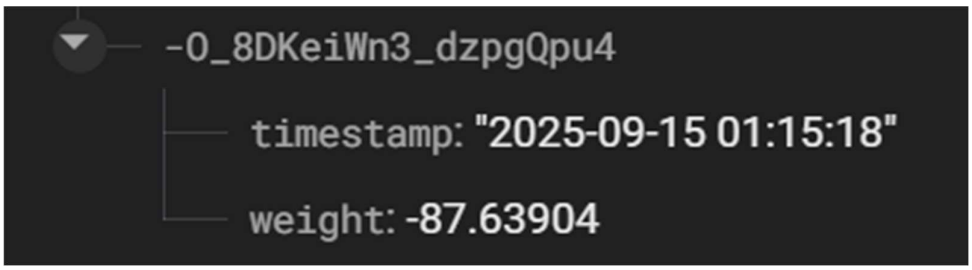


Figure 6.4 Data Uploaded to Firebase

The timestamp between the firebase uploaded and laptop system time was difference by approximately of 1 minute, which was acceptable for hydration monitoring.

For the offline test, Wi-Fi was disconnected for validate offline storage functionality with valid timestamp that get from NTP server. Figure 6.5 shows the ESP32 had disconnected from Wi-Fi.

```
No significant change | Current: 881.50
Wifi not connected
No significant change | Current: 881.50
Wifi not connected
No significant change | Current: 881.50
```

Figure 6.5 ESP32 disconnected from Wi-Fi

A water intake event occurred at offline mode. The system stored the data in NVS temporarily and sync to Firebase once Wi-Fi was reconnected. Figure 6.6 shows the water intake event was recorded in offline mode.

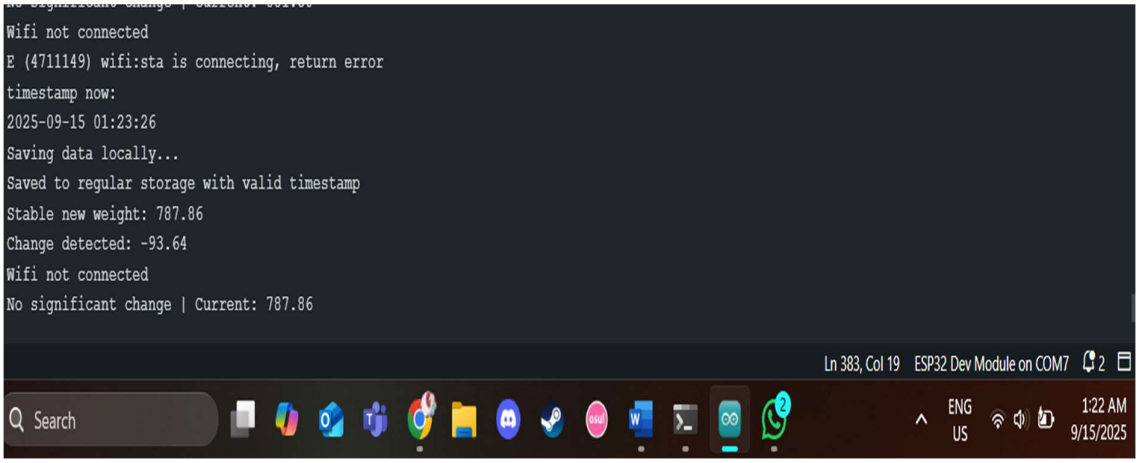


Figure 6.6 Water intake event (offline mode)

From the figure above, the system had detected a weight changed of 93.64ml and the timestamp at ESP32 was 1:23:26 which was slightly ahead of the laptop system time, which was expected since NTP provides more accurate network-based time compared to laptop clock. Figure 6.7 shows the ESP32 successfully reconnected and uploaded the offline data to Firebase, while Figure 6.8 illustrates the Firebase Console with the uploaded data.

```
Wifi reconnected

Firebase connected
Time initialized successfully
Current time: 2025-09-15 01:27:28
No offline data to process
Uploading 2 local entries in main storage...
Uploaded data0 successfully.
Uploaded data1 successfully.
Local storage cleared.
```

Figure 6.7 ESP32 reconnected and sync data

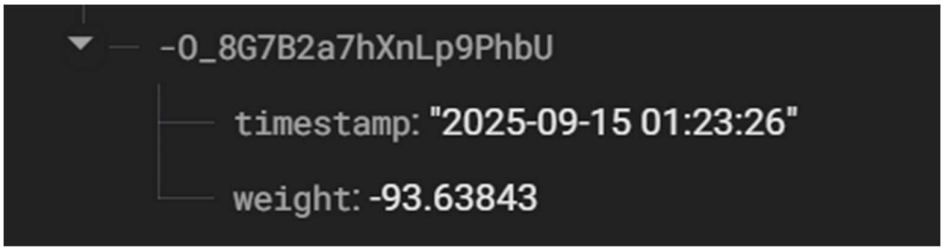


Figure 6.8 Firebase Console showing uploaded offline data

From both online and offline test, the system shows the ability to maintain continuous operation and preserve data integrity to ensured that no water intake event data was lost even when temporarily disconnected.

6.2.4 Low Battery Condition

The test was conducted to verify the functionality of voltage sensor in monitoring the battery status of the ESP32. When the detected battery voltage was lower than the defined threshold of 3.5V, the system triggered a visual alert by blinking the red LED indicator 5 times per second to inform the user to recharge the battery. Figure 6.9 shows the low battery at Serial Monitor while figure 6.10 illustrate the red LED indicator light up to inform user.

```
No significant change | Current: 787.86
Battery Voltage: 2.66 V
Voltage data sent to Firebase successfully.
No significant change | Current: 787.86
```

Figure 6.9 Low battery voltage detected (Serial Monitor Output)

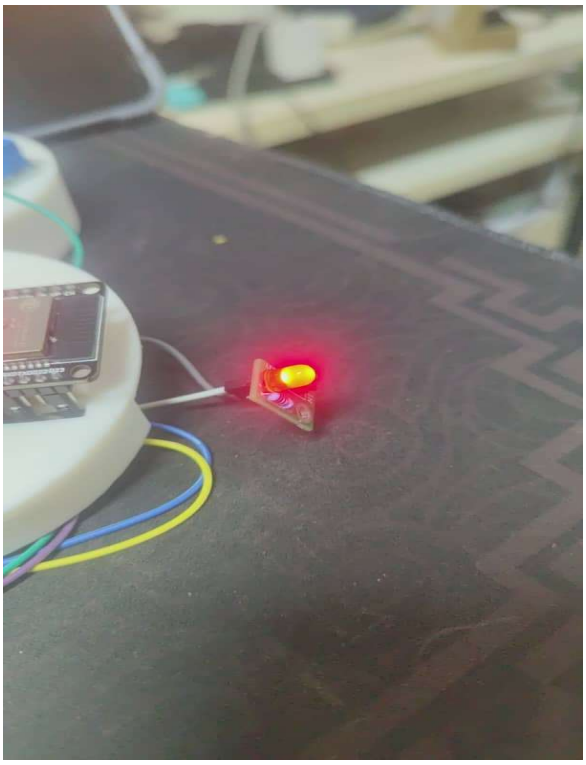


Figure 6.10 Red LED indicator activated under low battery condition

The results confirm that the system was able to detect low battery conditions in real time and providing a clear visual alert that tried to inform the user to prevent device shutdown, which affected the data reliability.

6.2.5 Frontend Visualization and Customization Test

This test was carried out to verify whether the frontend webpage was able to visualize the hydration data retrieved from Firebase correctly and allow user customization for improved accessibility and personalization.

The first part of the test was focused on real-time data visualization. Once the ESP32 uploaded the water intake event, the dashboard displayed the data in daily, weekly and monthly charts. The daily chart only showed water intake events, while the weekly and monthly show the data in bar chart and the summary section shows total water drinks, average intake and goal achievement. Figure 6.11 show the daily hydration line chart, while Figures 6.12 and 6.13 illustrates the weekly and monthly page.

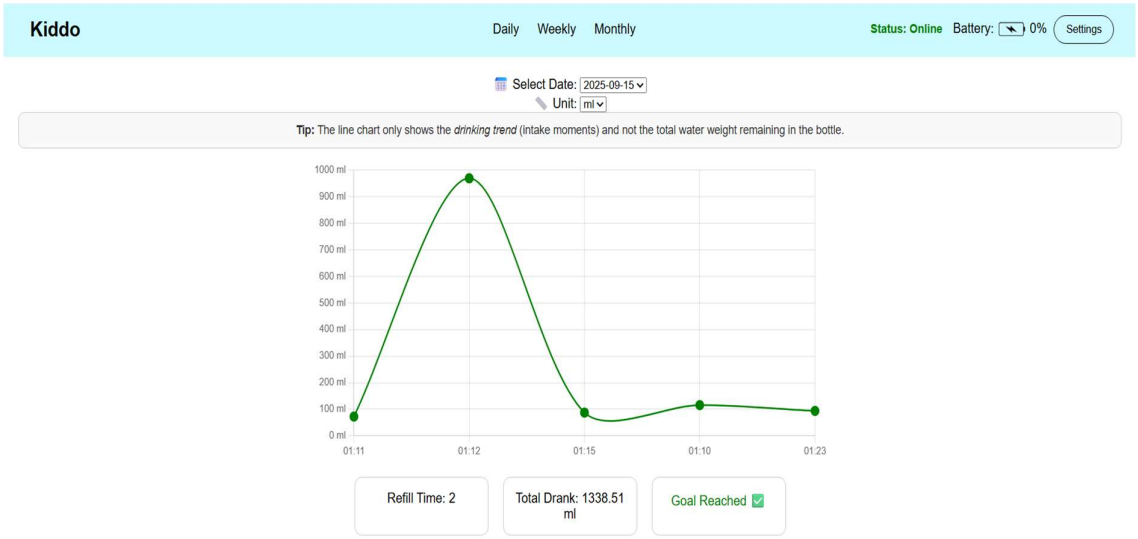


Figure 6.11 Daily summary page

Figure 6.12 Weekly summary page

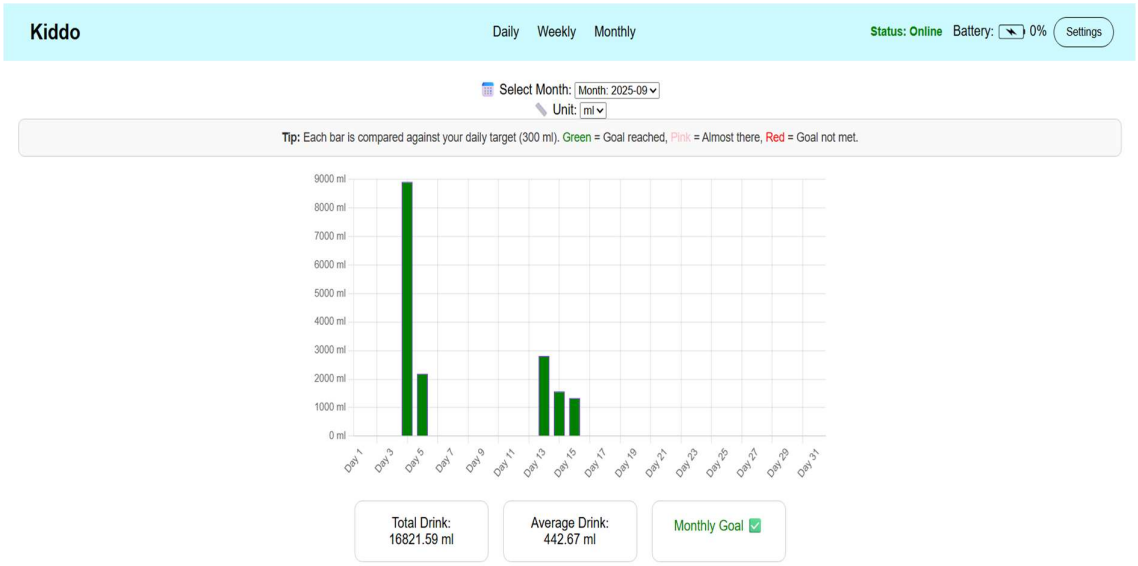


Figure 6.13 Monthly summary page

The second part of the test evaluated customization features. In the setting page, the user was able to adjust the font size for better accessibility. Figure 6.14 and 6.15 shows the difference between medium and large font size options.

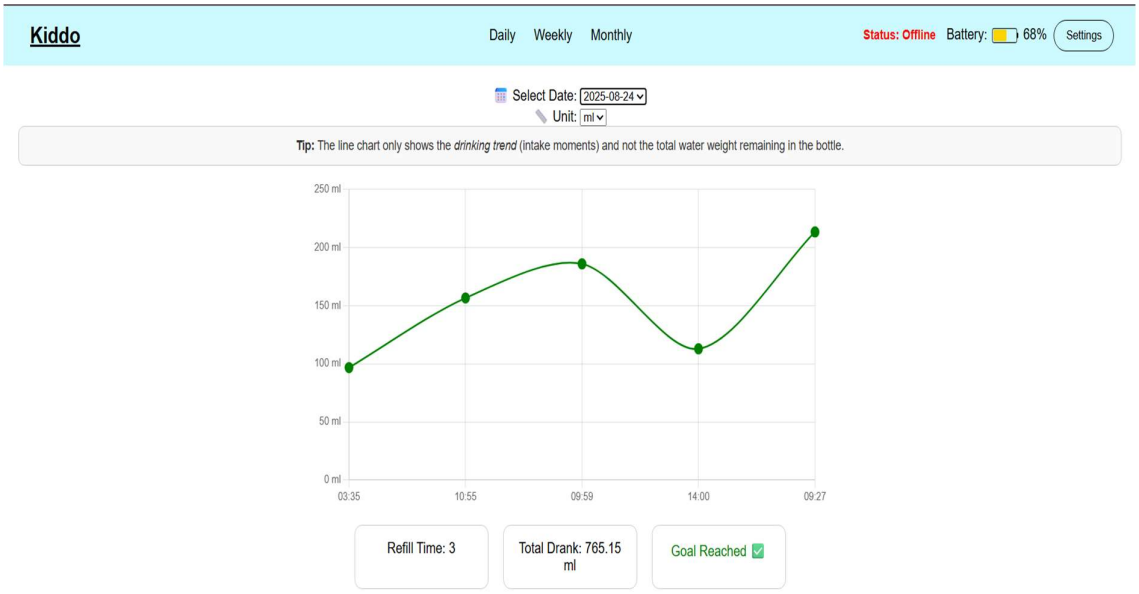


Figure 6.14 Font size in medium setting

Figure 6.15 Font size in large setting

Additionally, the user could modify the goal target separately for daily, weekly and monthly. Once the goal was modified, a save button will appear, and the updated goal will store in the browser local storage, ensuring persistence across sections. In default, the daily water intake goal was 600ml. For example, when user need to change the daily goal to 1000ml, after entering the value the save button will show. Figure 6.16 shows the modified goal setting with save button appear.

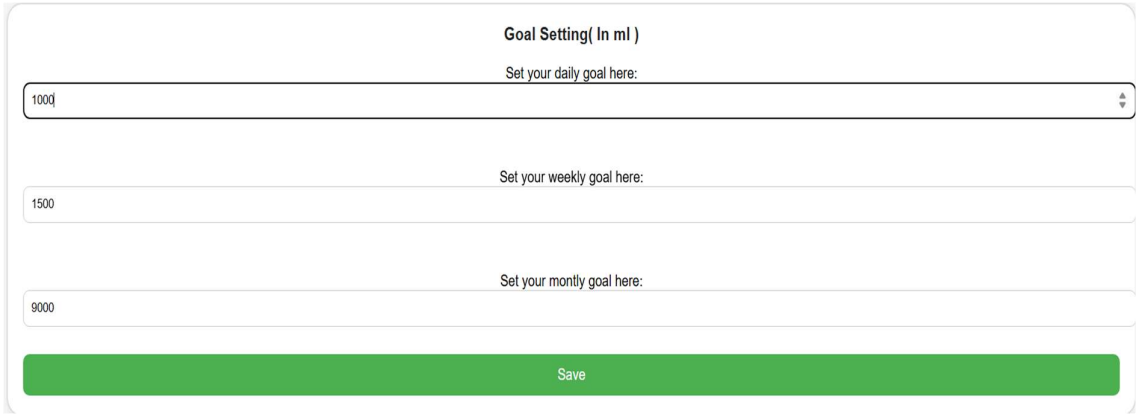


Figure 6.16 Modified goal setting

After the user click save, the goal setting updated in the local storage. Figure 6.17 show the local storage updated message.

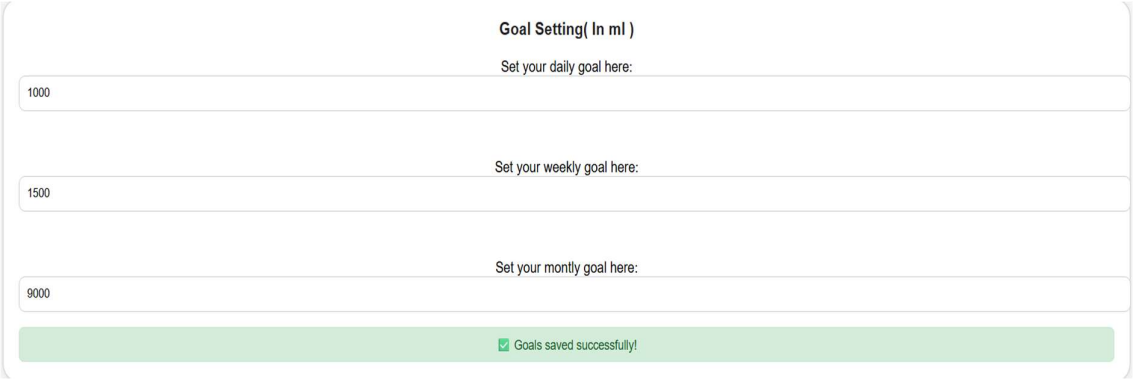


Figure 6.17 New goal being saved

Figure 6.18 shows the daily page after goal was updated, where the hydration progress indicator reflected the new target.

Figure 6.18 Daily page with new goal

6.3 Project Challenges

Although most implementation issues were resolved through testing or redesign, several challenges had remained unsolved or only partially being solved during the project.

The first challenge was about SSL connection Error. During data synchronization with Firebase, the ESP32 sometimes produced SSL connection errors, resulting in data upload failure. The error logs indicated as unstable secure socket initialization. This problem was suspected to be cause by unstable Wi-Fi connection, as the ESP32 only supports the 2.4GHz

Wi-Fi band, which was more easier being disturbed compared to 5GHz. In environment of using 2.4GHz channel, the reliability of the connection may be reduced, leading to SSL handshake failure. The proposed solution was changing the microcontroller than support 5GHz Wi-Fi band to enhance the stability of the connection between microcontroller and Firebase. Figure 6.19 shows the error message of SSL connection timeout.

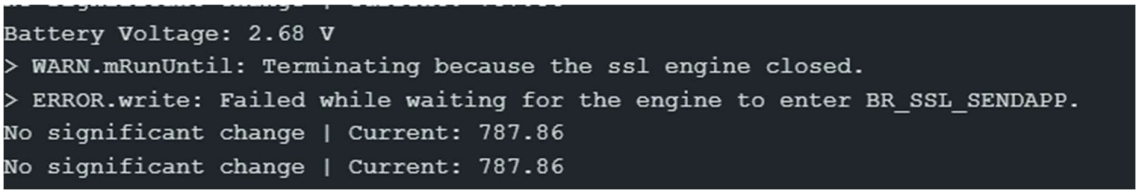


Figure 6.19 Error Message of SSL connection

Another unresolved issue was the ESP32 failing to initialize Wi-Fi when powered by the 18650 Li-ion battery boosted to 6V. In this project, the system could only operate Wi-Fi reliably when connecting a micro-USB to the TP4056 charging module. This suggested that the battery alone was unable to provide stable and sufficient current for Wi-Fi initialization, although the battery capacity was stated as 3800mAh. A possible reason was that the battery may be a counterfeit product, as many low-cost 18650 battery on market was a rewrapped battery cell with lower actual capacity than the labelled value. This limitation reduced the portability of the system, as it required external charging to maintain Wi-Fi connectivity.

In summary, these unsolved challenges highlight potential limitations in hardware quality and operating system specification, which should be addressed in future development of this system.

6.4 Objectives Evaluation

The objectives established in Chapter 1 were evaluated against the final implementation and testing results of the Smart water Tracking System for kid. Table 6.6 summarizes the achievement status and the evidence.

Table 6.6 Objectives Evaluation

Objective	**Evaluation**	**Evidence**
Primary Objective: Design and implement a Smart Water Tracking	Achieved	A functional system was successfully develop consisting of ESP32, HX711 load cell,

System to promote hydration among children		ADXL345 accelerometer, Firebase integration and web-based frontend
Sub-objective 1: Develop a device that can be attached to children's water bottles using a load sensor to measure consumption and wirelessly transmit the data to a web application.	Achieved	HX711 load cell measures water intake events and ESP32 transmits the data to Firebase via Wi-Fi.
Sub-objective 1a: Ensure portability by incorporating a rechargeable power supply to support full-day usage.	Partially Achieved	Device runs on 18650 battery with TP4056 charging module, but Wi-Fi initialization fails without constant charging, likely due to insufficient current.
Sub-objective 2: Achieve accurate and stable monitoring using a high-precision load sensor and processing algorithms.	Achieved	HX711 calibration test achieved <5% error compared to actual weight. Orientation detection (ADXL345) reduces false reading
Sub-objective 3: Simplify the monitoring process for caregivers and parents with an easy-to-use web application.	Achieved	The frontend application provides real-time visualization by distributing into daily, weekly and monthly summary page, support goal customization and font-size adjustment, improving usability of non-technical users.

6.5 Concluding Remark

This chapter presented the testing, performance evaluation and overall assessment of the Smart Water Tracking System. A series of experiments were conducted to test the system functionality, such as verify the accuracy of load sensor, data logging in both online and offline modes, low battery monitoring and frontend visualization. The result demonstrates that the system was able to capture and synchronize water intake events, visualize of hydration data and provide customization for user usability.

Although certain limitation was identified, such as SSL connection and insufficient battery current, but most of the project objectives were successfully achieved. The system fulfilled its purpose of promoting proper hydration among children by providing real-time monitoring and interactive dashboard for caregivers. The next chapter will conclude the project by summarizing key contributions and providing recommendations for future development and improvement of this project.

Chapter 7
Conclusion and Recommendations

7.1 Conclusion

The project was carried out with the main objective of designing and implanting the Smart Water Tracking System to promote healthy hydration among children due to most of the children does not know the important of hydration, especially in environment with limited direct supervision, such as kindergarten or playground. Through out the development process, both hardware and software components were successfully integrated to achieve a functional and reliable system.

On the hardware side, the ESP32 microcontroller was used together with HX711 amplifier, 10kg load cell, ADXL345 accelerometer and voltage sensor. This combination allowed the system to take accurate weight measurement, do orientation checking to reduce false reading and monitor the system battery level for safe operation. Furthermore, a DC-DC voltage boost converter was used to step the 18650 battery to power the ESP32 and the system also implemented a red LED indicator to provide a visual alert to inform users in low battery conditions.

For the software implementation, the system was designed to work in both offline and online modes. Valid drinking and refill event were recorded and synchronized with Firebase database in real-time when Wi-Fi was available. When offline, the data stored at NVS storage locally and automatically upload once the connection was restored. This ensured that the data continuity and the risk of data loss. On the frontend side, a web-based application was developed using Firebase JavaScript SDK to visualize the hydration patterns. The dashboard provided daily, weekly and monthly view, customization of hydration goal, font size adjustments for accessibility and real-time indicators for system connectivity and battery status.

The system objectives outlined in Chapter 1 were largely achieved. The device was able to capture water measurement with acceptable accuracy, synchronize offline effectively and provide a user-friendly interface for caregivers. Several tests were conducted and showed the

system could reliably capture hydration event and prevent meaningful summaries for different timeframes. Although several technical challenges were found, such as SSL connection instability and power supply limitations, but the project demonstrated the practical feasibility of a smart hydration tracking solution.

In conclusion, the smart water tracking system successfully addressed the problem of monitoring children's hydration status by providing accurate, reliable and accessible data. Although improvements can still be made, but the current prototype had proves the system capability and have a strong foundation for further refinement.

7.2 Recommendations

Although the system was achieved its primary objectives, several area of improvement were identified during development and testing.

The first area was hardware improvement. During testing, the 18650 battery could not reliably support Wi-Fi initialization, suggesting wither poor-quality of battery cell or insufficient current to support. Future versions should be use higher-grade batteries with verify capacity or alternative energy solutions to power the system. Furthermore, the SSL connection failures between ESP32 and Firebase was observed, due to the ESP32 only support 2.4 GHz Wi-Fi, connection drops was more likely. The recommendation of this issue was trying to improve the retry mechanism or considering alternative microcontroller with better Wi-Fi stability as CPU of the system. Lastly, the current prototype uses multiple sensors module stacked in layer, resulting in a relatively bulky 3D-printed case. A more compact design can be achieved by developing a custom PCB that integrate the microcontroller, HX711, voltage sensor and ADXL345 accelerometer onto a single board. This would reduce the overall size, simplify wiring, lower power consumption and allow the case to be redesigned into a smaller and portable form factor that suitable for children daily use.

Lastly was the recommendation for Frontend enhancement. While hydration goals and font size can be adjusted, additional personalization such as theme colour, reminder notification and the option to hide the summaries section can also be added to the system for better usability. Furthermore, design the dashboard layout for mobile devices would allow users to access the system conveniently on smartphones or tablets.

REFERENCES

[1] “Choose water for healthy hydration,” HealthyChildren.org. https://www.healthychildren.org/English/healthy-living/nutrition/Pages/Choose-Water-for-Healthy-Hydration.aspx

[2] S. Online, “Health DG: 11-year-old boy died from heatstroke, toddler from dehydration,” The Star, Apr. 28, 2023. [Online]. Available: https://www.thestar.com.my/news/nation/2023/04/28/health-dg-11-year-old-boy-died-from-heatstroke-toddler-from-dehydration

[3] K. Davis-Young, “3 children have died after heat emergencies in the last 2 weeks in Arizona,” KJZZ, Jul. 11, 2024. [Online]. Available: https://www.kjzz.org/news/2024-07-10/3-children-have-died-after-heat-emergencies-in-the-last-2-weeks-in-arizona

[4] “A Child’s Health is the Public’s Health | CDC,” Centers for Disease Control and Prevention, Oct. 24, 2022. https://www.cdc.gov/childrenindisasters/features/children-public-health.html

[5] J. Warren et al., “Challenges in the assessment of total fluid intake in children and adolescents: a discussion paper,” European Journal of Nutrition, vol. 57, no. S3, pp. 43–51, Jun. 2018, doi: 10.1007/s00394-018-1745-7.

[6] Nick, “The ESP32 Chip explained: Advantages and Applications,” DeepSea, Jun. 24, 2025. https://www.deepseadev.com/en/blog/esp32-chip-explained-and-advantages/

[7] Tech Explorations, “How to power your ESP32 development kit, options,” Tech Explorations, Mar. 15, 2024. https://techexplorations.com/guides/esp32/begin/power/

[8] Your Application’s Backend, Simplified, “Firebase Advantages and disadvantages,” Back4App Blog, Apr. 22, 2024. https://blog.back4app.com/firebase-advantages-and-disadvantages/#Serverless_Platform

[9] Jason, “Understanding the TP4056: A complete guide to Single-Cell Li-Ion battery charging.” https://www.ic-components.com/blog/understanding-the-tp4056-a-complete-guide-to-single-cell-li-ion-battery-charging.jsp

[10] Simplediyguy, “HX711 weighting scale AD, using both channels,” Simple DIY Circuits, Oct. 24, 2022. https://simplediycircuits.wordpress.com/2022/10/24/hx711-weighting-scale-ad-using-both-channels/

[11] “PRO 21 oz,” HidrateSpark. https://hidratespark.com/products/hidratespark-pro-21oz-smart-water-bottle

[12] “Everything you need to know about EQUA Smart Water Bottle,” EQUA - Sustainable Water Bottles, May 07, 2018. https://myequa.com/blogs/blog/everything-you-need-to-know-about-equa-smart-water-bottle?srsltid=AfmBOopo6FbyiES0MzWDENGZtjsnhFZtUloDf--A-ge5C5-n6xFDXqcD

[13] C. Carlson, “How ultrasonic sensors work,” MaxBotix, Mar. 01, 2023. https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work

[14] “Trago - the world’s first smart water bottle,” Kickstarter, Dec. 26, 2016. https://www.kickstarter.com/projects/905031711/trago-the-worlds-first-smart-water-bottle

[15] “What is the best type of water bottle for kids?,” BOTTLEPRO. https://www.bottlepro.net/hydration-blog/what-is-the-best-type-of-water-bottle-for-kids

[16] G. D. D. N. CoLtd, “How often do kids water bottles need to be replaced? - Knowledge - Guangzhou Diller Daily Necessities Co.,Ltd,” Dillerbottle, Oct. 07, 2020. [Online]. Available: https://www.dillerbottle.com/info/how-often-do-kids-water-bottles-need-to-be-rep-50289367.html#:~:text=As%20we%20all%20know%2C%20plastic,affecting%20the%20use%2C%20therefore%2C%20the

[17] “File System Considerations - ESP32 - — ESP-IDF Programming Guide v5.4.1 documentation.” https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/file-system-considerations.html

[18] M. Alam and M. Alam, “ESP32 & DS3231 based Real Time Clock (RTC) on OLED,” How to Electronics, Aug. 22, 2022. https://how2electronics.com/esp32-ds3231-based-real-time-clock/

[19] M. Cheich, “Using time features with your ESP32 [Guide + Code],” Programming Electronics Academy, Jun. 12, 2024. https://www.programmingelectronics.com/esp32-time-servers/

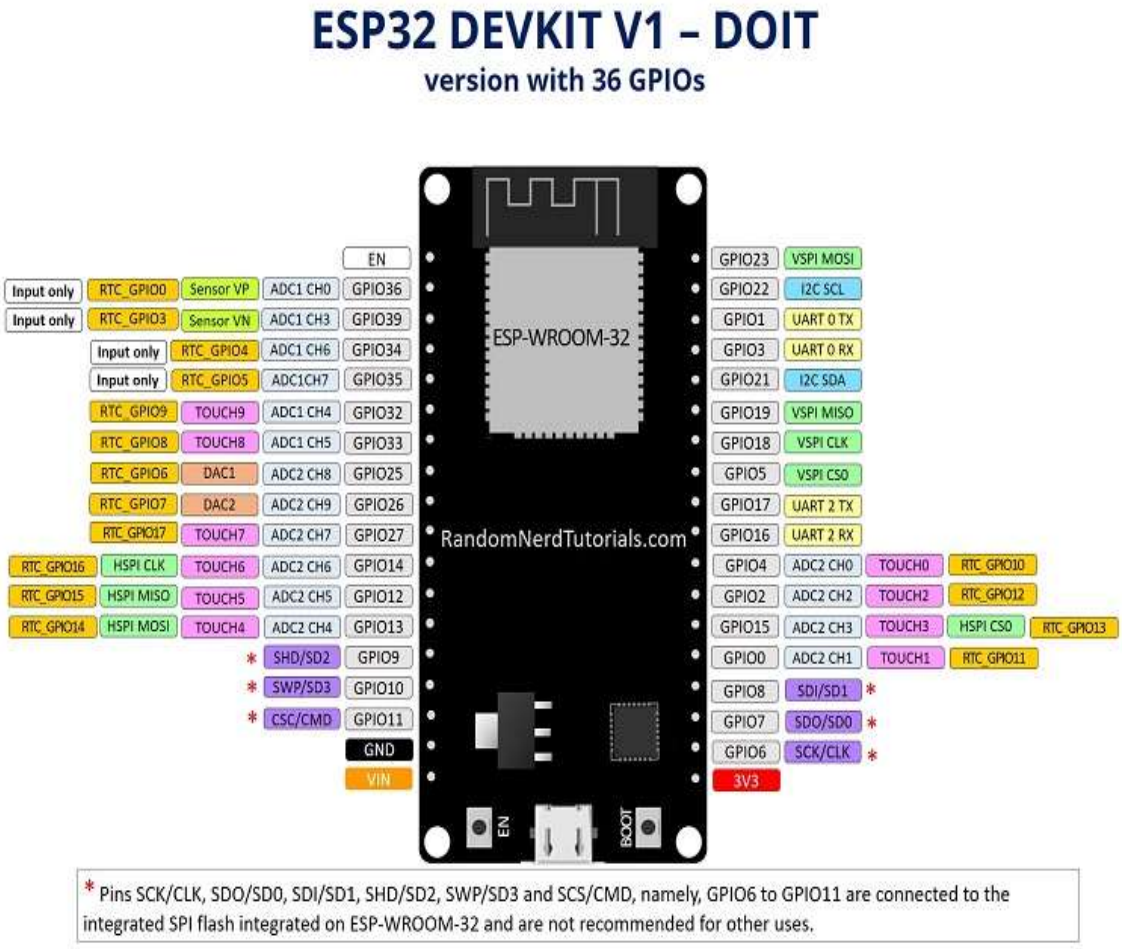
[20] R. Santos and R. Santos, “Installing ESP32 in Arduino IDE (Windows, Mac OS X, Linux) | Random Nerd Tutorials,” Random Nerd Tutorials, Feb. 28, 2024. https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/

[21] “Load Cell Amplifier HX711 Breakout Hookup Guide - SparkFun Learn.” https://learn.sparkfun.com/tutorials/load-cell-amplifier-hx711-breakout-hookup-guide

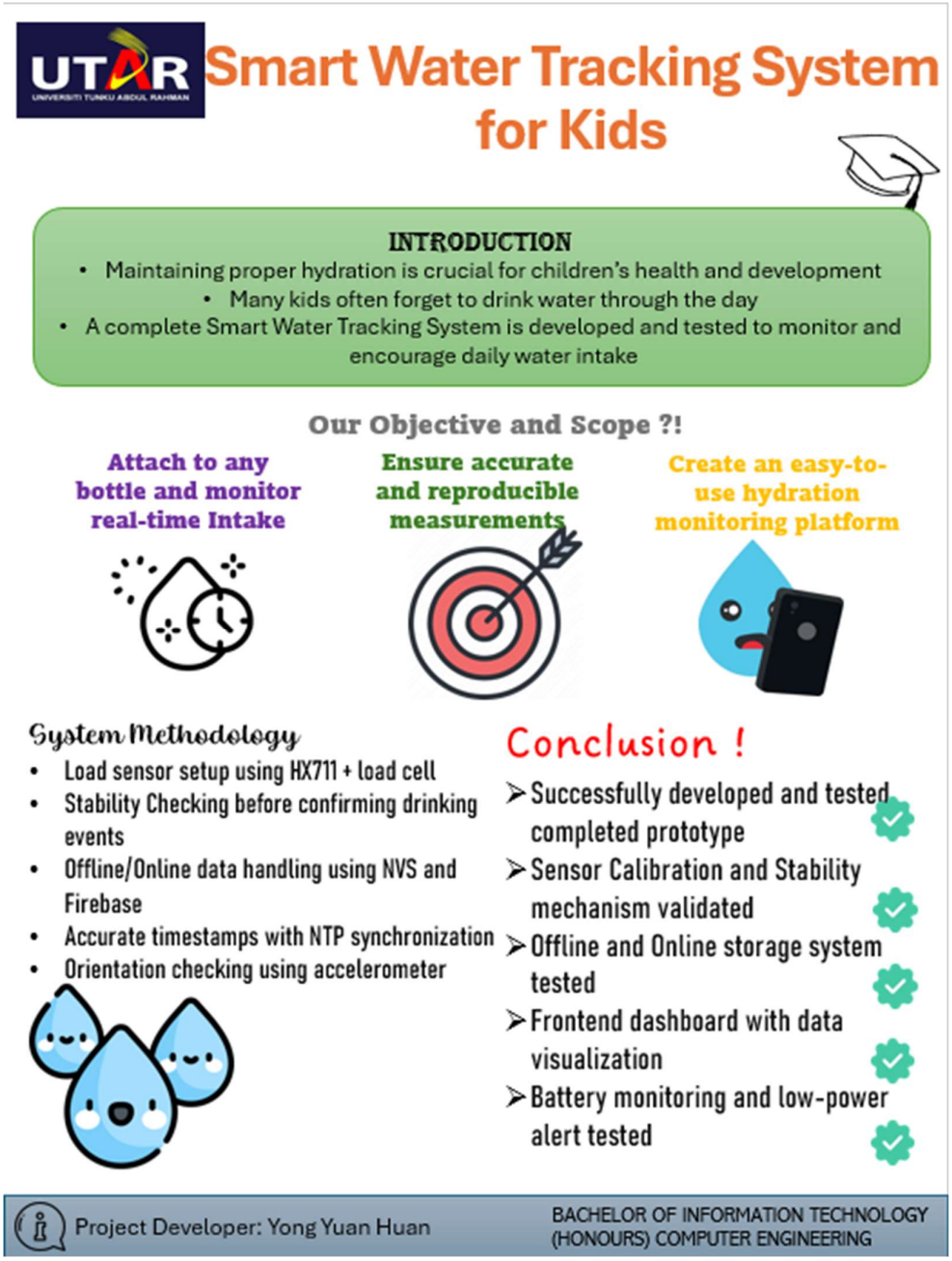
[22] “ADXL345 Accelerometer Interfacing with ESP32 | ESP32,” © 2018 ElectronicWings. https://www.electronicwings.com/esp32/adxl345-accelerometer-interfacing-with-esp32

Appendix

ESP32 GPIO Pin Diagram



POSTER


UTAR
UNIVERSITI TUNKU ABDUL RAHMAN
Smart Water Tracking System for Kids
INTRODUCTION
• Maintaining proper hydration is crucial for children's health and development
• Many kids often forget to drink water through the day
• A complete Smart Water Tracking System is developed and tested to monitor and encourage daily water intake
Our Objective and Scope ?!
Attach to any bottle and monitor real-time Intake
Ensure accurate and reproducible measurements
Create an easy-to-use hydration monitoring platform
System Methodology
• Load sensor setup using HX711 + load cell
• Stability Checking before confirming drinking events
• Offline/Online data handling using NVS and Firebase
• Accurate timestamps with NTP synchronization
• Orientation checking using accelerometer
Conclusion !
➢ Successfully developed and tested completed prototype
➢ Sensor Calibration and Stability mechanism validated
➢ Offline and Online storage system tested
➢ Frontend dashboard with data visualization
➢ Battery monitoring and low-power alert tested
Project Developer: Yong Yuan Huan
BACHELOR OF INFORMATION TECHNOLOGY (HONOURS) COMPUTER ENGINEERING