Portable Water Tracking Attachment

Design Document

ECE445 Spring 2024

Project #44

Subha Somaskandan, Subhi Sharma, Cindy Su

Professor: Arne Fliflet

TA: Luoyan Li

Contents

1. Introduction

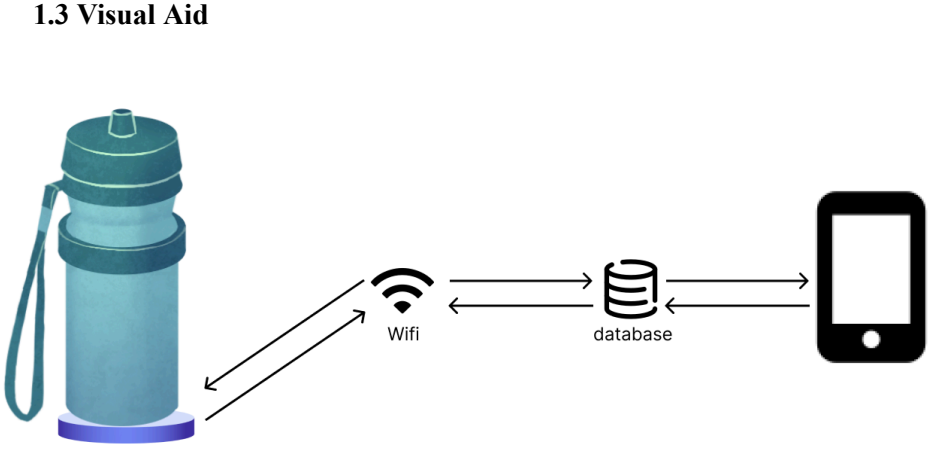
1.1 Problem

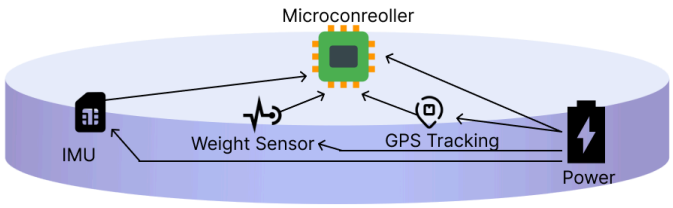
With a busy schedule, it can be difficult to remember to drink enough water during the day. Most water tracking products on the market are not customizable to one's lifestyle or daily habits. The amount of water that everyone needs to drink per day depends on their height, weight, sex, physical activity,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

1.2 Solution

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

1.3 Visual Aid





1.4 High Level Requirements

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

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

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

2. Design

2.1 Block Diagram:

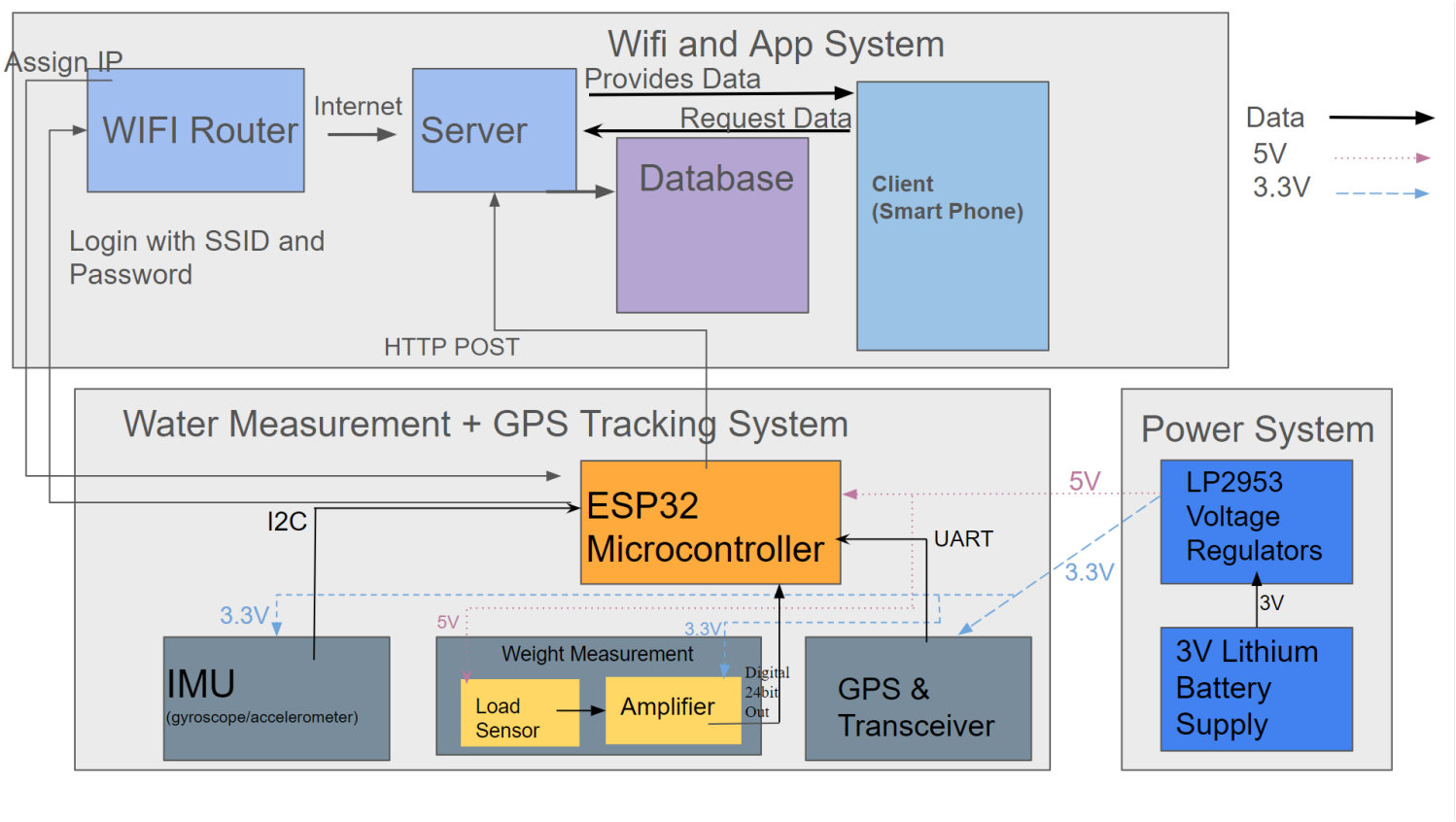


Figure 1: Block Diagram of Portable Water Tracking Attachment

Our design is made up of 3 subsystems, the Wifi and App Subsystem, the Water Measurement and GPS Tracking Subsystem, and the Power System. The Wifi and App subsystem receives data from the microcontroller through a server and wifi router, and stores the data in a database, which can be accessed from the smartphone app client. The Water Measurement and GPS Tracking system measures the weight of the water bottle when it is placed down. There is also location access through the transceiver module, and the positioning, weight, and location data is sent to the microcontroller through various protocols, listed in Figure 1. Finally, the Power System delivers both 3.3V and 5V through two voltage regulators, which are connected to a 3V Lithium battery supply. The 3.3V and 5V outputs are used to power the components in the Water Measurement and GPS Tracking system, as outlined in Figure 1.

2.2 Functional Overview & Subsystem Design

2.2.1 Power Subsystem

For the majority of our devices, a voltage input of 3.3V will suffice, but for the ESP32 Microcontroller and YZC-133 Load Cell Weight Sensor, an input voltage of 5V will be needed, as shown in Table 1. For the ESP32 Microcontroller, the development board requires 5V, but the ESP32 itself needs 3.3V, which can be directly routed from a pin on the development board.

A linear regulator, the LP2953, will be used to provide the 5V to our microcontroller and load cells as well as another regulator to provide 3.3V to our transceiver and gyroscope/accelerometer. We will use a 3V Lithium Battery, composed of two AA batteries and a clip, to power our linear regulators, as shown in the Power Subsystem in our Block Diagram Figure 1, also shown in Figure 2. A linear regulator produces a fixed output voltage via an internal feedback loop and comparators, as well as being generally more efficient in power conversion and voltage regulation when compared to resistor dividers, and buck converters. Furthermore, the LP2953 protects against reverse input battery protection and is very compatible with battery applications, which are both vital to our design.

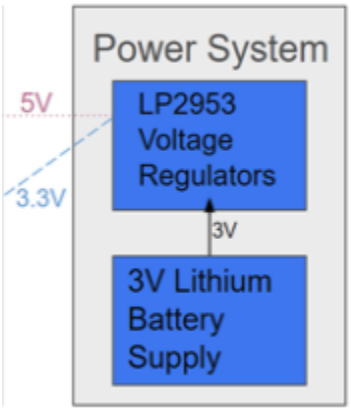


Figure 2: Power System Block Diagram

Component/Device	Voltage & Current Requirements
ESP32-WROOM-32 Microcontroller	Input Voltage: +3.3V or +5V Max Current: 250mA
YZC-133 Load Cell Weight Sensor	Input Voltage: +5V
MPU-6050 3 Axis Gyroscope + Accelerometer	Input Voltage: +2.375V - +3.46V V_LOGIC: 1.71V to Input Voltage Operating Current: 3.9mA
HX711 Load Cell Amplifier	Input Voltage: +2.7V - +5.5V Operating Current: <1.5mA
L70RE-M37 Navigation GPS Transceiver Module	Input Voltage: +2.8V - +4.3V Max Current: 16mA
W3011A RF ANT 1.575GHZ	N/A

Table 1: Power Requirements for All Components/Devices

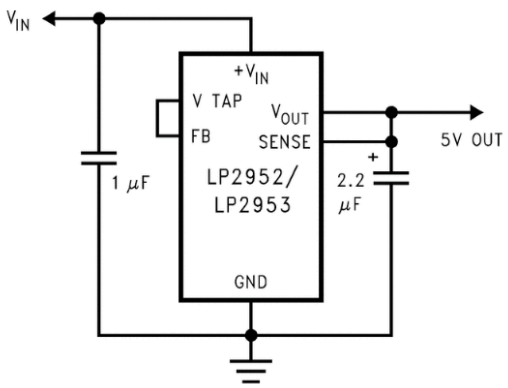


Figure 3: Linear Regulator LP2953 Schematic for 5V Fixed Output

As shown in Figure 3 above, a 1uF needs to be connected from V_in to GND since we are using a battery input, as well as tying the pins V_Tap and Feedback (FB) together, and V_Out and Sense together. For a 5V output, a 2.2uF film or aluminum electrolytic capacitor needs to be connected at the output for better filtering, and then will be routed to the microcontroller as well as the load cell. For the 3.3V output, since our current consumption is much less than 250mA, a 4.7uF to 5.5uF capacitor will suffice. This signal will then be routed to the gyroscope, load cell amplifier, and transceiver.

Pin Number	Function	Connection
1,8,9,16	GND, ground	Supply's Ground
2,7,10,11	NC	No Connections
15	IN, power input	3V Power Supply
3	OUT, 3.3V or 5V output voltage	SENSE, capacitor connected to ground, and to other components
4	SENSE, voltage feedback input	Tied to OUT
13	VTAP, internal resistor divider output	Tied to FEEDBACK
14	FEEDBACK, error non-inverting input	Tied to VTAP

Table 2: Pin Diagram for LP2953, for 3.3V or 5V output

Requirements	Verification
Lithium Battery Unit must supply 3V +/- 5% to the Voltage Regulator when active	1. Use a multimeter to connect across the battery unit's positive and negative leads, and measure the voltage and note any fluctuations. Ensure the steady value is within 5% of 3V.
Voltage Regulator output of 3.3V/5V must supply 3.3V/5V +/- 5%, with current output up to 250mA	1. Probe OUT and GND on the LP2953 with the positive and negative leads (respectively) of a differential probe that can be connected to an oscilloscope. 2. Once displayed on the oscilloscope, measure average voltage and ensure it is within 5% of desired output voltage. If necessary, measure maximums and minimums to this same tolerance as well. 3. Connect oscilloscope current probes to wire coming from OUT on the LP2953 to the wires going to the microcontroller and the load cell input, and verify the sum is under 250mA. Repeat on other LP2953 with output of 3.3V, but now probe the wires going to the gyroscope, load cell amplifier, and transceiver.

2.2.2 Weight Measurement and Tracking Subsystem

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

2.2.2.1 Inertial Measurement Unit (MPU-6050 Module 3-Axis Gyroscope and Accelerometer)

Input: Position of the chip, which translates to the position of the water bottle.
Output: Data regarding acceleration and 3-axis position of the bottle via an I2C protocol.

This IMU contains a triple axis gyroscope and accelerometer which have digital outputs. The unit will be connected to the microcontroller and power supply.

The IMU is necessary because it will detect when someone has picked up their bottle based on the angular position of the chip. This data will be sent to the microcontroller using I2C protocol so that it can be used to indicate when the bottle has been placed back down to initiate the weight sensor. The range of the gyroscope is user-programmable from 200, 500, 1000, and 2000 degrees/second. The unit has a 3.9 mA operating current when the available 6 degrees of motion are all enabled.

Pin	Name	Connection
6	AUX_DA	I2C Master Serial Data to external sensors
7	AUX_CL	I2C Master Serial Clock to external sensors
8	VLOGIC	Digital I/O Supply Voltage to VDD
9	AD0	I2C Slave Address LSB
12	INT	Interrupt Digital Output (totem pole or open drain)
13	VDD	To Power Supply Voltage and Digital I/O Supply Voltage
18	GND	To Power Supply Ground
23	SCL	I2C Serial Clock to Microcontroller SCL

24	SDA	I2C Serial Data to Microcontroller SDA

Table 3: IMU Pin Connections

2.2.2.2 Weight Measurement Using a Load Cell and HX711 Amplifier
Amplifier Inputs: Load Cell's channels S- and S+
Amplifier Outputs: Analog Output to the Microcontroller

The HX711 is necessary for the load cell because it ensures that the output voltage of the strain gauge is proportional to the excitation voltage. This helps to make the final voltage produced by the load cells accurate and free from noise. This unit is a 24 bit analog-to-digital converter meant for weighing sensors.

The ESP32 microcontroller has a HX711 library in order to effectively calibrate the load cell, set any offsets, and obtain any weight data. Commands within the ESP32 IDE such as read() and get_value(number of readings) obtain the raw readings and the average of the last defined number of readings minus the tare weight, respectively.

The load cell is 12.7mm x 12.7mm x 80mm and will be mounted on a small platform. The range of weight it can measure is from 0-5kg, and it requires a 5-10V drive voltage. The rated output is 1.0±0.15 mV/V and it has a recommended excitation voltage of 5V. The input resistance is 1066±20Ω and the output resistance is 1000±20Ω.

Pin	Name	Connections
3	AVDD	To Power Supply of 2.6-5.5V and E+ of the Load Cell
5	AGND	To Power Supply Ground
7	INA-	To Channel S- of the Load Cell
8	INA+	To Channel S+ of the Load Cell
9	INB-	To Power Supply Ground
10	INB+	To Power Supply Ground

Table 4: Pin Assignments of HX711 Amplifier

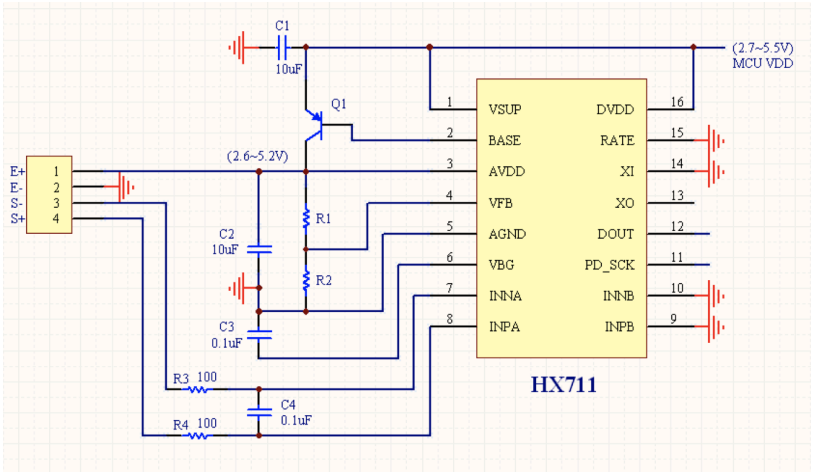


Figure 4: HX711 Module connected to Load Cell

In order to connect the load cell to the amplifier and microcontroller:

- Connect the red wire (E+) to the E+/GND of the ESP32 and HX711 modules
- Connect the black wire (E-) to GND for the HX711 amplifier and GPIO 16 pin of the ESP32 module
- Connect the white wire (S-) to the INA- pin for the HX711 amplifier and GPIO 4 pin of the ESP32 module
- Connect the green wire (S+) to INA+ of the HX711 amplifier and the 3.3V pin of the ESP module.

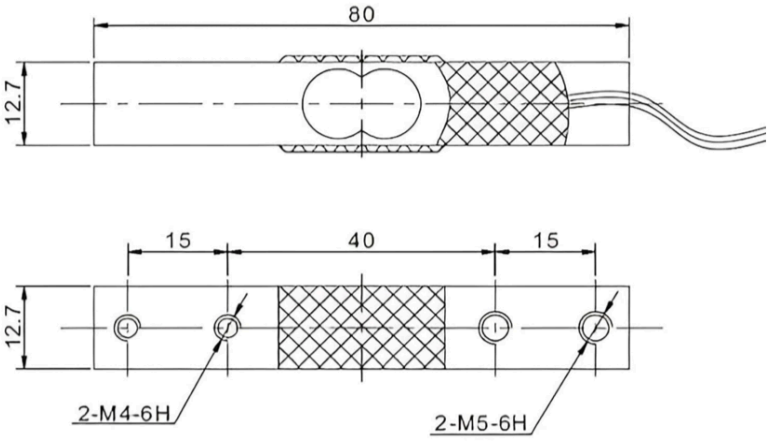


Figure 5: Dimensions of the Load Cell (mm)

2.2.2.3 GPS Chip and Antenna (L70RE-M37 GPS Transceiver Module, W3011A Antenna)
The GPS unit utilizes an L1 Band Receiver at 1575.42 MHz. The serial interface used is UART, and the default number of bits transmitted during one pulse (one baud) is 9600bps. The RF antenna must be connected to the GPS module to receive data. This antenna can operate between 1559-1606.6MHz, which is compatible with this GPS module.

A GPS is more for a convenience element; if one loses their device, they will be able to track it. Consequently, they will also be able to track their bottle's location if the device is attached. The GPS chip has to be surface mounted onto the PCB with the Antenna. This module has ultra low consumption, so during standby it will consume 500uA, 8uA during backup, and 13mA during tracking.

In order to access data from the GPS chip, the ESP32 module can be connected via a UART serial connection. The protocol used by this chip is NMEA-0183 and PMTK, and thus a NMEA-0183 parser can be used to sift through geographical data points collected. The parser is based on ESP UART Event driver and an ESP event loop library.

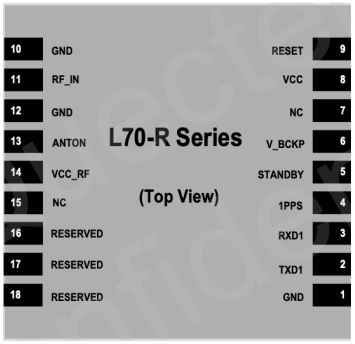


Figure 6: Quectel L70-R Series GPS Module

Pin	Name	Connection
2	TXD1	To the transmit pin of the ESP32 Module
3	RXD1	To the receiving pin of the ESP32 Module
8	VCC	To Power Supply (2.8-4.3V)
9	RESET	To Power Supply Ground. If this pin is unused then keep it open or connect it to VCC.
11	RF_IN (characteristic impedance of 50 ohms)	To the Antenna (Antenna is unpolarized)
13	ANTON	To a second power supply. Keep the pin open if unused.

Table 5: Quectel L70-R Series GPS Module Pin Assignments

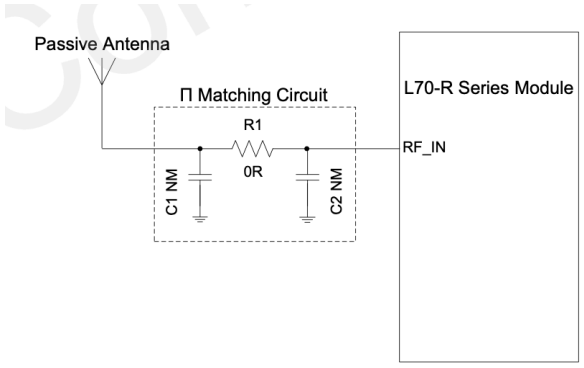


Figure 7: Reference Design for a Passive Antenna

In the figure above, the antenna directly receives power from the VCC_RF. The L70-R series module provides a power supply for an external antenna via VCC_RF, with a range of 2.8-4.3V. The typical value used is 3.3V.

ANTON is an optional pin which can be used to control the power supply of the active antenna. This will not be used in our design.

The Antenna can be connected to pin 11 of the GPS module via surface mount. The Antenna has two contact pads, and it is not polarized. Either terminal can act as a ground or feed.

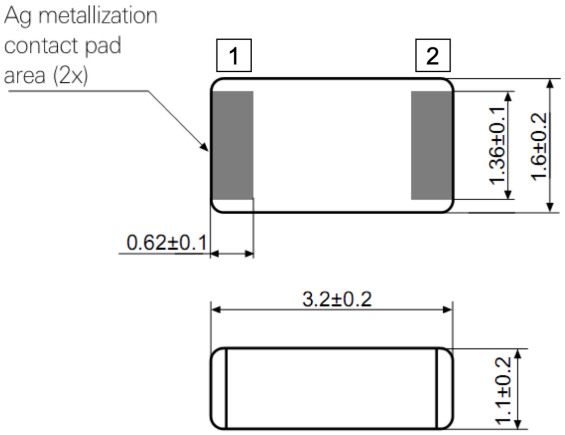


Figure 8: Antenna Mechanical Drawing

No.	Terminal name	Terminal Dimensions
1	Feed / GND	0.62 x 1.36 mm
2	Feed / GND	0.62 x 1.36 mm

Antenna is symmetrical.
Either of terminals 1 or 2 can be feed / GND

Figure 9: Antenna Dimensions and Terminal Definitions

Requirements	Verification
The IMU (Inertial Measurement Unit) sensor data, specifically x, y, and z angular velocity and acceleration, must be able to be read on the microcontroller IDE.	1. Program ESP32 Board with Arduino IDE, and download the MPU6050 library. 2. Ensure pin connections to IMU and ESP32 are correct as outlined in 2.2.2.1. 3. In the code, initialize an object of the MPU6050 library, and follow the documentation to set ranges for the accelerometer and gyroscope. 4. Using the MPU6050 documentation, write code to get the values for x, y, and z acceleration and angular velocity, and upload code. 5. Move and rotate the water bottle that the IMU is attached to in all three directions, and ensure the display readings correspond accordingly.
The Load Cell and Amplifier Combination must be able to detect changes in weight as little as 2.6oz[1] +/- 5%, as the user drinks water, and displays it on the microcontroller IDE	1. Program ESP32 Board with Arduino IDE, and download the HX711 library. 2. Ensure pin connections to the load cell and amplifier within themselves as well as the amplifier to the microcontroller are correct based on 2.2.2.2. 3. Following the documentation for the HX711 Library, calibrate the sensor with the water bottle, which has a known weight of 0.590kg. 4. After calibration, fill up the water bottle to full capacity with water. 5. Using the HX711 Library, write code to get and display the weight of the water bottle, and verify it is heavier due to the added weight of the water. 6. Drink or pour out amounts close to 2.6oz and place the water bottle back down flat, and read the weight again, and ensure it is within 5% of 2.6oz

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

	subtracted from the original weight of the water bottle. Repeat for bigger amounts of water to verify.
The GPS and Antenna units must retrieve geographical data and submit this information to the microcontroller IDE.	1. Connect the GPS module to the ESP32 microcontroller using a UART serial connection. This can be done by connecting the transceiving and receiving pins of the GPS module to the respective pins on the ESP32 board using surface mounting techniques. 2. The data can be accessed by implementing an NMEA-0183 parser to parse through data streams outputs from GPS modules based on ESP UART libraries.

2.2.3 Wifi and App Subsystem

When the weight sensor updates the weight of the water bottle, the value will be transmitted to the microcontroller. Through the WiFi Module, this data will be transmitted to a database hosted on an online server. When the end-user accesses the application on their smartphone, the application interfaces with the server to retrieve and display the most recent updates.

2.2.3.1 Wifi-Module (ESP32 WROOM 32)

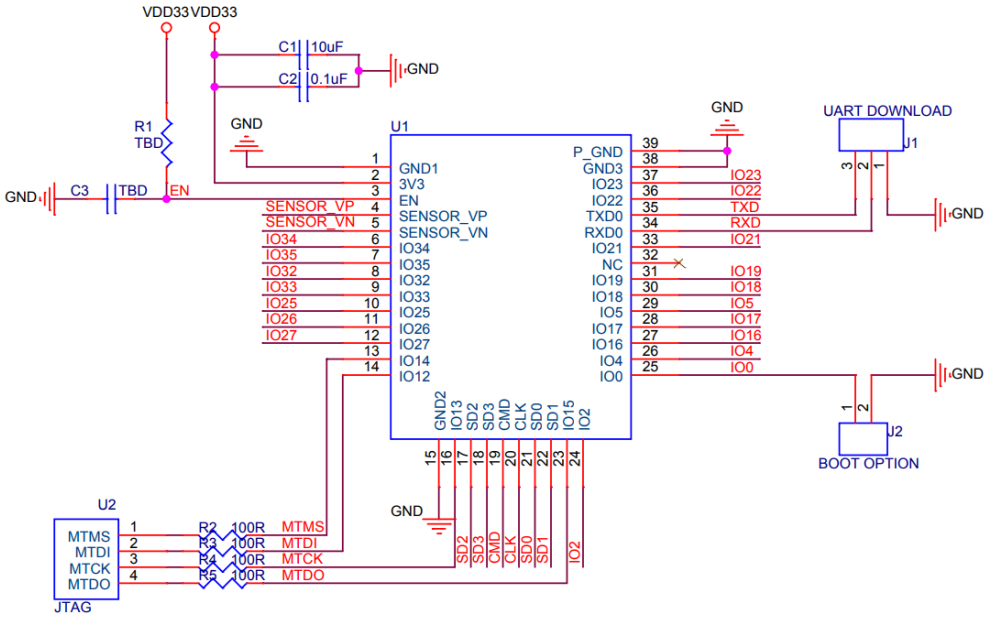


Figure 10: Peripheral Schematics

Pin number	Name	Usage
35	UART TXD0	Data transmission, will be connected to the TX pin of the USB to Serial port convertor
34	UART RXD0	Data reception, will be connected to the RX pin of the USB to Serial port convertor
25	GPIO0	Determine if code is run from memory or loaded via UART
24	GPIO2	Determine if code is run from memory or loaded via UART
3	EN	When Programming the esp32, we will pull EN pin low briefly to trigger reset
1, 15, 39	GND	Ground
3	3V3	3.3V power

Table 6: ESP32 Pin Diagram

Configuration

To program the ESP32 via the UART terminal and ensure it reads new code instead of executing preloaded code from its memory, it is necessary to manipulate specific GPIO pins to induce the correct boot mode. Specifically, GPIO0 must be pulled low during the reset to enter the bootloader mode. After the new code has been transmitted to the ESP32 via UART, a subsequent reset is initiated. For the ESP32 to boot from its updated internal memory and execute the new code, GPIO0 should be set high before this reset. This sequence ensures that the ESP32 exits the bootloader mode and starts running the newly uploaded program from its flash memory upon reboot.

Booting Mode			
Pin	Default	SPI Boot	Download Boot
GPIO0	Pull-up	1	0
GPIO2	Pull-down	Don't-care	0

Figure 11: Strapping Pins table from esp32 datasheet

Wi-Fi Configuration

To set up Wi-Fi connectivity on the ESP32, the Arduino IDE will be used to program and deploy the necessary configuration script. This script includes the SSID, password, and the server's URL for the Wi-Fi network that the ESP32 is intended to connect to. After the script is successfully transferred to the ESP32, it establishes a connection to the designated Wi-Fi network with the specified credentials and sends POST requests to transmit data to the database.

Requirement	Verification
A laptop must be able to transmit code to the ESP32 through a USB Type-C connector to UART	Connect the ESP32 to a LED and to the laptop using a USB Type-C cable and upload a led blinking test code. Confirm the successful code transmission by observing the expected behavior of the blinking LED.
The ESP32 must successfully connect to a specific Wi-Fi network using credentials provided in the Arduino IDE code, which includes the SSID, password, and server's URL .	After uploading the Wi-Fi configuration code to the ESP32, verify the connection by checking the ESP32's serial output for a confirmation message indicating successful connection to the Wi-Fi network
Once connected to the Wi-Fi network, the ESP32 must handle POST requests to send data to an online database.	Execute a test script that triggers the ESP32 to send mock data to the online database using POST request. Verify the data by checking the entries. Measure the response time over multiple transmissions to ensure reliability and accuracy.

2.2.3.2 Front-End Application

When opening the app, the user will see a homepage with a water bottle at the center, representing the current water level within your actual water bottle. The homepage also has a button that navigates to the statistics page. This page shows the analysis of the user's water intake and displays the intake data in a chart presented through weekly, monthly, and yearly statistics, allowing users to track and manage their hydration habits. Another feature accessible from the home page is the location page. This page provides the functionality of showing the real time location of the user's water bottle to ensure that it can always be found easily.

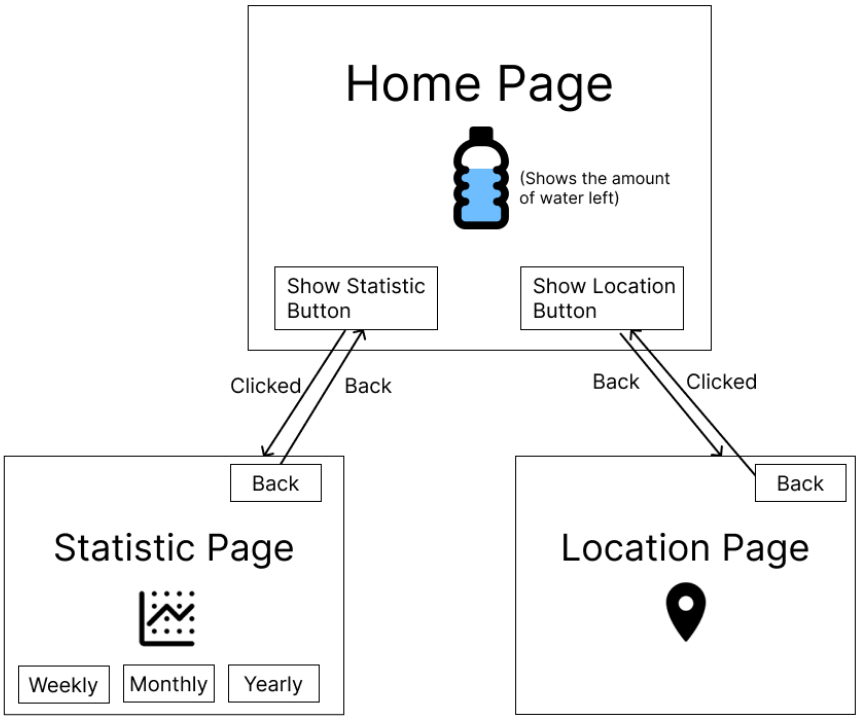


Figure 12: Front-End Application flowchart

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

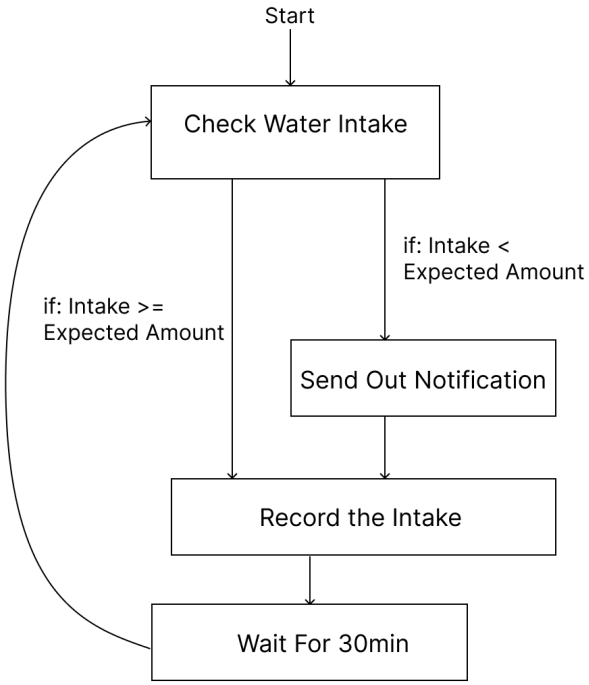


Figure 13: App Notification System flowchart

2.3 Tolerance Analysis:

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

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

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

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

3. Cost and Schedule

3.1 Cost Analysis:

The total component and part cost is listed below in Table 7, excluding shipping and taxes.. With shipping and taxes included, our final total for components and parts is $153.06. For labor costs, we are assuming a salary of $35/hour for each teammate, and estimating 200 hours each, this comes out to $7,000 per person. Having 3 people, the total labor cost for the team comes out to 3 x &7,000 = $21,000. Furthermore, we will also be utilizing the machine shop, with an estimated quote of 5 hours, assuming a salary of $20 per hour. Thus, our project comes out to a grand total of $21,253.06.

Component/Parts Description and Part Number	Manufacturer	Vendor	Quantity	Extended Price
ESP32-WROOM-32 Microcontroller	Espressif	Amazon	Pack of 3	$12.99
USB to Serial port CP2102	HiLetgo	Amazon	1	$7.49
YZC-133 Load Cell Weight Sensor	Geekstory	Amazon	Pack of 2	$10.99
MPU-6050 3 Axis Gyroscope + Accelerometer	DIANN	Amazon	Pack of 5	$12.99
HX711 Load Cell Amplifier	Stemedu	Amazon	Pack of 5	$9.49
L70RE-M37 Navigation GPS Transceiver Module	Quectel	DigiKey	2	$18.60
W3011A RF ANT 1.575GHZ	Pulse Electronics	DigiKey	5	$8.45
LM2953 Low Dropout Voltage Regulator	Texas Instruments	DigiKey	1	$6.54
TPCL225K016R5	KYOCERA AVX	DigiKey	5	$14.35

000 CAP TANT 2.2UF				
C4AQOBU4520 M18J CAP FILM 5.2UF	KEMET	DigiKey	4	$6.92
32oz Water Bottle	N/A	Amazon	1	$12.99
Water Bottle Boot (estimated)	N/A	Amazon	1	$10.99
TOTAL	—		—	**$132.79**

Table 7: Total component quantity and cost

3.2 Schedule:

Week of	Task	Team Member
February 26	Unit test individual components and connect them to the microcontroller to get sample information. Attend a design review with the instructor to get feedback and make any necessary changes before drafting the PCB.	Everyone
March 4	Begin designing the PCB and prepare to order the first board at the end of the week. Begin writing code for the microcontroller's communication with the IMU, GPS, and weight sensor.	Everyone
March 18	Test and debug PCB, and order the second board by the end of the week. Begin and continue to work on the mobile application, and	Subhi Sharma, Subha Somaskandan Cindy Su

	make adjustments to the microcontroller's code. Begin to prototype the physical design and consult the machine shop for assistance.	
March 25	If PCB does not work, retry and order the third board. If PCB works, integrate the board and code together and test its functionality. Adjust the microcontroller's code and ensure that it works with the new board. Continue to work on the mobile application and have main frameworks outlined (inventory system and a portal to input personal information).	Subhi Sharma, Subha Somaskandan Everyone Cindy Su
April 1	If PCB does not work, retry and order the fourth board. If PCB works, integrate the board and code together and test its functionality. Adjust the microcontroller's code and ensure that it works with the new board. Begin to assemble the complete device and test its functionality. Mobile application must be able to make necessary calculations to show how much water one needs to drink. More complex functionality such as the reminders and data collection must also be finished or close to completed.	Subhi Sharma, Subha Somaskandan Subhi Sharma, Subha Somaskandan Subhi Sharma, Subha Somaskandan Cindy Su
April 8	If PCB does not work, retry and order the fifth board. If PCB works, integrate the board and code together and test	Everyone

	its functionality. Adjust the microcontroller's code and ensure that it works with the new board. Test the final product's functionality and make final changes if needed. Work on presenting at the final demo.	
April 15	Present the product's functionality at the mock demo. Use the feedback from the mock demo to make any necessary changes for the final demo. Begin working on the final presentation and final report.	Everyone
April 22	Present at the final demo. Work on the final presentation and report.	Everyone
April 29	Present product at the final presentation.	Everyone

Table 8: Schedule for the Building Process

4. Discussion of Ethics and Safety

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

According to the IEEE Code of Ethics, Section I, Point VI, maintaining technical competence and undertaking tasks only after clear limitations have been outlined as mentioned, could result in potential similarities. However, our design does have clear differences as it is portable and removable from the bottle. This device can be connected to any 32 oz bottle, rather than being built into one. We understand the ethical standard of not infringing on any existing patents and we will not be repeating any existing processes already done by HidrateSpark.

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

Our team also upholds all necessary safety accommodations, as we have all completed the required lab training.

Citations

"NMEA Parser Example." *GitHub*, github.com/espressif/esp-idf/blob/master/examples/peripherals/uart/nmea0183_parser/README.md. Accessed 22 Feb. 2024.

Dave, et al. "ESP32 with Load Cell and HX711 Amplifier (Digital Scale)." *Random Nerd Tutorials*, 23 Apr. 2022, randomnerdtutorials.com/esp32-load-cell-hx711/#hx711-amplifier. Accessed 22 Feb. 2024.

HidrateSpark. "Hidratespark pro 32 Oz: Bluetooth Smart Water Bottle & Hydration Reminder App: Insulated Stainless Steel." *HidrateSpark*, hidratespark.com/products/hidratespark-pro-32oz-smart-water-bottle?utm_source=google&utm_medium=cpc&utm_campaign=Performance_Max-Shopping&campaignid=19636654461&adgroupid=&adid=&gad_source=1&gclid=Cj0KCQiAzoeuBhDqARIsAMdH14HqBSS0eOD4fHKBRN9bdDQNBT1iFPscshHAkXgEQpV4WaNAqpC_7EsaAsVrEALw_wcB. Accessed 22 Feb. 2024.

Pulseelectronics, productfinder.pulseelectronics.com/api/open/part-attachments/datasheet/W3011A. Accessed 22 Feb. 2024.

L70 Hardware Design - Rs Components, docs.rs-online.com/74ea/0900766b8147dbe2.pdf. Accessed 22 Feb. 2024.

LP295x Adjustable Micropower Low-Dropout Voltage Regulators Datasheet, Texas Instruments, https://www.ti.com/lit/ds/symlink/lp2952-n.pdf?ts=1708657346374&ref_url=https%253A%252F%252Fwww.google.com%252F. Accessed 22 Feb. 2024.

How to Convert a Load Cell Reading into Total Weight, Hunker, https://www.hunker.com/13408598/how-to-convert-a-load-cell-reading-into-total-weight. Accessed 22 Feb. 2024.

How to connect ESP32 for programming, https://www.lab4iot.com/2019/07/14/tutorial-on-how-to-program-the-esp32-wroom-32-or-esp32f/ Jul 14, 2019. Accessed 22 Feb 2024.

ESP32-wroom-32 Datasheet, https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. Accessed 22 Feb. 2024

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