Table of Content

Chapter 1

Introduction

1.1 Problem Statement

Dehydration is a condition that occurs when there is more water or bodily fluid leaving the body than entering it (Davis and Stoppler, 2019). The symptoms of dehydration are dry or sticky mouth, headache, muscle cramps and more (Khatri, 2019). Although these problems might look insignificant for some of the individuals, however, these symptoms can eventually bring serious damage to health. Dehydration can lead to serious diseases such as kidney failure, low blood pressure or even heat strokes (Crosta, 2017).

The main reason for dehydration is that individuals rely much on their inaccurate sense of thirst to drink water (Brecher, 2017). Brecher stated that according to the research conducted by Armstrong. L, individuals drink water during the time they feel thirsty and it is actually not enough for the bodies because thirst sensation appears when they are 1% or 2% dehydrated. Moreover, due to the human nervous system, individuals confused the sense of thirst with a sense of hunger. Clinical studies stated that 37% of people confuse hunger with the sense of thirst because thirst signals are weak ("Hunger vs. Thirst: tips to tell the difference", n.d.).

The term Internet of Things did not appear until 1999 by Kevin Ashton when he used it as a title of his presentation. He stated that IoT is to allow the creator to empower computers to have the ability to gather data using RFID and sensor technology that will instead let the computers understand our world without the constraints of human data entry. In order to prevent the wastage of water as natural resources and also taking care of personal wellbeing, actions needed to be taken with devices that integrate the combination of IoT concept and sensor tracking.

Therefore, the hydration tracking water bottle is introduced to overcome this problem. With the application of the Internet of Things (IoT), water intake monitoring information can be shown on users' mobile.

1.2 Customer Requirement

50 people were picked at random and were asked to conduct an online survey on a hydration tracking device anonymously. From the results of the survey, 35 out of 50 persons would like to use a 0.5-litre water bottle as a daily carry. 45 people surveyed own a smartphone and over half of the people do prefer the Android operating system over IOS. 86% of people did not use a hydration tracking device before and they opted for a led indicator as a special feature in a hydration tracking device if they have one.

A few decisions have been made based on the results of the survey, we are going to prioritize designing the device for a 1L bottle, for the time being, an app will be made since there is a high percentage of people owning a smartphone, and the app will be supported by Android first and then modifications will be made to make the app compatible with ios. One of the main features of the device will be a strip of LED indicators indicating the amount of water remaining that needs to be consumed.

1.3 Fundamental Engineering Knowledge

The fundamental engineering knowledge applied in this project includes Circuit Theory, Fundamentals of Programming, Analogue Electronics and Microprocessor and Microcontroller.

During the trial stage, the implementation of sensor detection with an Arduino module into a connection required the knowledge from Circuit Theory. The knowledge such as validating circuits using the connection of breadboard and the components are used during this stage. Meanwhile, during the coding stage, the knowledge from the Fundamentals of Programming is applied to code the electronic components using the Arduino module. The programming knowledge allows code and modifies the original coding while enabling data to be collected through the sensor and uploaded to the cloud so that the data can be processed, analyzed and sent to users' mobile through an app.

Besides, the application of analogue to digital converter (ADC) and digital to analogue converter (DAC) is learned from Analogue Electronics. The data collected by the sensor is normally in analogue signals and the microcontroller is a digital processing device, thus helping us to convert the analogue signals to digital data. The following course, Microprocessor and Microcontroller covered the knowledge of monitoring and control of microcomputer-based applications. It allows the development of

a prototype system that satisfies the microcontroller requirements and hardware specifications in this project.

1.4 Complex Engineering Problem

This project builds on the basis of water-level-detection consumed by the user that relates to hydration tracking and IoT concepts. Hydration tracking can be defined as the detection of the amount of consumable liquid that has been consumed by the person by applying one or many methods for detection and calculation. In this project, the prototype will be using sensors to detect the water consumed by the user. The water consumption data will be uploaded to the cloud for the user to review. In the process of designing and producing the prototype, the electronic and electrical application and design will be involved. The electronic device with an external power source will be assembled and programmed through application in the computer.

The main problems that will be faced throughout the process mainly is the functionality of the program code, the size of the prototype and waterproofing the electronic device. The sensor device must be compact and able to fit into the test subject which is the water bottle while the code embedded into the device must be able to work properly to ensure the correctness and connectivity of data. On the other hand, the sensor device must be water-resistant or waterproof as the device will come in close contact with liquid frequently.

1.5 Public and Health Safety

In IoT, the view of public health safety goes one notch above as everything is connected to the internet and physical contact is no longer needed. This reduces the risk of the consumer getting close to the device operation and causes injury to them like an electric shock. The electronic component of our product is going to be embedded inside the water bottle and uses wireless communication to connect to the user and not expose the sensitive component and wiring directly to the user.

Next, as our day is getting busier, people will tend to ignore or postpone the time they in-take the required water content. Hydration-Tracking Water Bottle helps to improve their health by reminding them to drink water in regular time intervals and also has features to keep track of the history of water

consumption. Hence, this project is unique in that it takes the user's health into consideration and blends with technologies which provide reminders and suggestions for timely water intake.

1.6 Environmental and Sustainability

Since the industrial revolution in the 1700's in Europe, the world went through a rapid transformation finding a lot of new elements from nature or man-made them. Some elements lead to toxic waste after being discarded, causing harm to the environment and the lives living within. The amount of carbon, fuel and greenhouse gases produced over years causes heat trapped within the atmosphere, leading to an ongoing rise of the average temperature of the Earth's climate system, damaging the environment we are living in slowly and severely. With such, the Green Internet of Things (G-IoT) is one of the ways to save Earth.

There are two types of G-IoT, one using IoT technology to monitor and reduce pollution while another is IoT systems that are able to get its role done without causing bad effects on the environment. Our project falls under the latter category as our product does not cause any negative effect to the environment. By using lesser components and thus lesser energy consumed. In doing so, batteries or power components could be used for longer times, energy will not be wasted. Saving the environment and making an effort to go green.

Wastage of water could be reduced by implementing this technology. As an example, children will not pour their water away before going back home to avoid being scolded by parents for not drinking water. By implementing IoT technology in the water bottle, parents can now monitor their water intake. Not only taking good care of children's water consumption, but they too can make sure that kids don't pour away their water even if they don't like to drink, as water consumption could be traced, thus saving water.

1.7 Objectives of the Project

The objective of this project is to design and build a weighing scale prototype that can automatically track the amount of water consumed by the users. The device will detect water consumption through the measuring of changes in water level inside a bottle. Another objective of this project is to implement the Internet of Things (IoT) for data transmission and storage. The water intake data will be collected and

uploaded to the cloud where it will create daily water intake reports to compare the previous day's performance with the latest day. Finally, this product allows users to have a better understanding of their daily water consumption. The analyzed data will be sent to users' mobile through an app to alert the users how much water they have consumed in a day and a reminder will be sent to their mobile if users do not reach the targeted water consumed.

1.8 Planning and Project Milestone

Project scopes are defined to set a clear goal and distribute tasks reasonably in between, ensuring the project to complete in time. The task-based approach is preferred to plan the project, defining works to be done to meet the requirements and complete the project. Tasks to be done are determined by breaking down the project into crucial tasks. Each task should be done within the time limit given. The project milestones are established following the tasks in Table 1.

Table 1.1 Duration of Tasks taken

Tasks	Descriptions	Durations/Days
A	Problem identification	21
B	Project planning	21
C	Literature Review	21
D	Methodology Research	14
E	Interim Report Writing	63
F	Selection of sensors	14
G	Materials Purchase	14
H	Design of Electronic Circuit	14
I	Design of Data Monitoring System	28
J	Design of Reminder System	14
K	Building of Prototype	28
L	Arduino Programming	28
M	Data storage Implementation (IoT)	21

N	Improvement of Design	56
O	Report writing	63

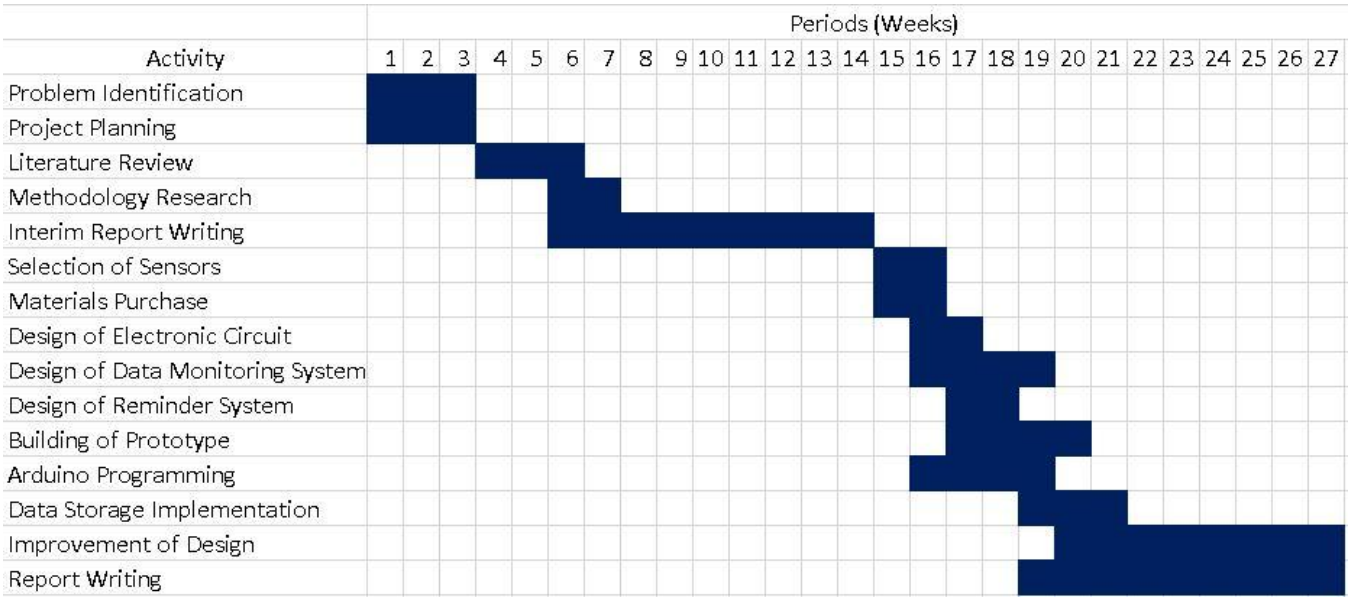


Figure 1.1 Gantt Chart for Project

Chapter 2

Literature Review

2.1 Review of Problems and Solutions

Hydration tracking or sensing can be defined as the measurement of the fluid using a sensor or technique. Hydration sensing is done by applying support vector machine models with the measurement of the inclination of the specific container. (Griffith et al, 2019) In contrast to the other method of measuring the volume of the water consumed, Griffith and Biswas proposed that by using the support vector machine classifiers that collect data via inertial measurement unit sensors they developed and trained under two specific scenarios by using MATLAB.

Griffth and other researchers commented that water containers equipt with sensing ability have limits of tracking functionality to only that container, unable to transfer sensor or device to multiple containers and also the sensing is restricted to the single container. Prototype proposed and demonstrated by Griffith and Biswas not only able track the water consumed by the user but also can identify the type of container the user is using if the inertial measurement unit is attached to the container.

Based on the journals and researches of Jovanov and his team, the personalized health monitoring and inventions can be supported by Big Data analytics. In the purpose of recording water consumed of the sensor implemented smart bottle, Jovanov, Nallathimmareddygari and Pryor proposed several versions of the smart water bottle including two configurations which are WiFi cloud system integration and body area network integration. With this, the monitoring of activity and physiological status with different sensors and water consumed by the users can be sent to the cloud with WiFi connectivity without the need of considering the Bluetooth limited range (Jovanov et al, 2016).

The controller for the cloud-based solution is implemented using Photon Wi-Fi platform for IoT applications. This approach allows a large number of intelligent devices integrated within the existing infrastructure. (Emil Jovanov, 2016) Both integrations can send out alerts and notifications to users. A 3D accelerometer is used to detect the use of the bottle while Touch and Pulse Sensor (TaPS) is implemented for physiological monitoring. The sensor uses capacitive sensing to detect when the user

touches the bottle and physiological monitoring using a photoplethysmographic sensor (PPG). PPG could be turned on and off according to the results of feedback from the TaPS. (Emil Jovanov, 2016)

Next, Grow is a conceptual smart bottle that uses bottle surface as an ambient display with a liquid level sensor inside the bottle to measure and give feedback about water consumption (Kaner et al, 2018). Kaner stated that the Grow smart bottle was equipped with a liquid level sensor that calculates the capacitance at the levels of six copper plates placed between the walls of the bottles and later the capacitance values are interpreted by the Raspberry Pi and correspond to the resistance wire. However, Grow does not equip with IoT technology which means the amount of water intake cannot be transferred to the mobile phone through an app. According to the interview, the participants' reactions were positive except a few expectations were pointed out by the users including battery time, smart bottle size, durability and ease of use (Kaner et al, 2018).

Hydration-Tracking Water Bottle or Smart Bottle is a water bottle fitted with sensors. These sensors will provide the technological addition to the bottle which makes it "Smart" (Tommy et al., 2018).In order to accurately estimate the water consumption for an individual, an initial input of gender, age, and weight is needed. The Android application connected via Bluetooth will display the real-time bottle water indicator, history of water consumption and a notification for reminding the user to drink water according to the user's water in-take habits. To track hydration, it can be done by analysing the user and water consumption data. Moreover, it has a feature that can predict the hydration needed based on the user's environment can be achieved by using an atmospheric temperature and humidity sensor (Tommy et al., 2018). (Closest to the current project, with extra features to read external atmosphere and humidity to supply water.)

A lot of apps and gadgets developed are not able to measure the actual daily fluid intake and also rely on users to understand the daily activity. A hydration monitoring system is then developed which detects drinking and physical activities automatically. The system consists of an embedded computer, a throat microphone and a smartphone application. The embedded computer is used to collect and process the signals in real-time. The throat microphone records sound from the throat area and the smartwatch is used to collect body activity data. The hydration monitoring algorithm runs on the embedded computer and real-time feedback is transmitted to the smartphone via Bluetooth. The system has a very minimal form factor and can be worn conveniently for longer periods. (Mengistu, Do, & Sheng, 2016)

2.2 Review of Components

Review of components allows flexibility in the design and development of the hydration tracking device in order to meet the different requirements of the users. It clarifies the differences and similarities of two or more components available in the market. By considering the most suitable components based on the project specifications and budgets, the most cost-effective solution can be applied. The comparison of components based on their prices, characteristics, pros and cons are as shown below so that the decision on components used in hydration tracking device can be made easily.

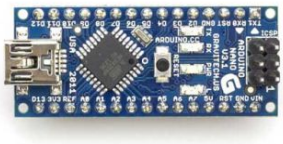
Comparison of Development Boards

Arduino's microcontroller development boards are commonly available worldwide and open-source developed. Their boards are widely used in developments of prototype and projects both for students and professionals. These boards not only are manufactured under Arduino but also in other branded clones which still commonly referred to Arduino boards. These open-sourced microcontrollers are programmed with Arduino IDE (Integrated Development Environment) available on IOS, Linux and Windows where the programming language is similar to C++, providing a more accessible platform for all users. On hardware, these boards usually have pins for inputs and outputs, buttons and connection jacks for direct outputs and power. Among all, Arduino Uno is the most widely used among the other microcontrollers.

In this project, Arduino Uno, Arduino Nano and NodeMCU are considered. The most prominent difference between these Arduino microcontrollers is the size. Arduino Uno has a larger and bulkier body than Nano, it contains higher support across multiple platforms while sacrificing the bulking body. It is more popular among hobbyists and students as it is the entry level of microcontroller. Arduino Nano and NodeMCU have the advantage of smaller size with mostly the same functions and number of pins compared to Uno.

On the other hand, NodeMCU is a development board that applies ESP8266 SoC developed by Espressif System, it combines the functions of an Arduino board and IoT ESP8266 module that will enable the user to connect to the Internet. The only cons of NodeMCU is that it only provides 1 Analog I/O pin although this can be fixed if a multiplexer IC is connected to it. (Arduino vs ESP8266 vs ESP32 Microcontroller Comparison - DIYI0T, 2020)

Table 2.1 Table of Comparison of Development Boards

Specifications	Arduino Uno	Arduino Nano	NodeMCU Devkit V2
Physical Appearance			
Size	68.6mm * 53.4mm	18mm * 45mm	49mm * 24.5mm
Microcontroller	ATmega328P	ATmega328	Tensilica 32-bit RISC CPU Xtensa LX106
SRAM	2 KB	2 KB	64 KB
Flash Memory	32 KB (0.5 KB used for bootloader)	32 KB (2.0 KB used for bootloader)	4 MB
EEPROM	1 KB	1 KB	0.5 KB
Clock	16 MHz	16 MHz	80MHz
Pins - PWM - Digital - Analog	20 pins - 6 - 14 (6 include PWM) - 6	22 pins - 6 - 14 (6 include PWM) - 8	30 pins - 16 (same as DIO) - 16 (1 include Analog) - 1
USB- Serial	ATmega16U2	FTDI FT232RL	CP2102
Cost	RM 15 - RM 22	RM 12 - 20	RM 17 -20

Comparison of Connection Module

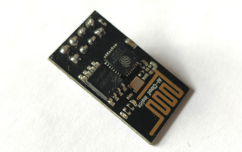
With the purpose of mixing the elements of IoT into the project, there are several different modes of connecting the prototype. The usage of the development board for this project requires a connecting module for this purpose. Each of the modules serves different purposes and provides different modes of connection to the internet. The modules are HC-05 Bluetooth, NodeMCU ESP8266, ENC28J60 Ethernet module and ESP8266 module.

As the name suggests, HC-05 uses Bluetooth 2.0 to achieve connection with full-duplex 2 way connection using development board with other devices with it's Master and Slave modes. It has the highest operating voltage with 5V and low operating current on 30mA compared to other modules, causing it to have higher power consumption. The low transmission rate of 3Mpbs and effective range of around 10 meters makes it not very suitable for constant data transmission with the targeted device.

ENC28J60 Ethernet LAN is a connection module that requires active connection with the target device in order to establish connection. It contains on board MAC & PHY, 8 Kb RAM and a Serial Peripheral Interface (SPI) that is used to connect Ethernet cable. This module is commonly used for hosting web servers combined with an Arduino board with its high transfer rate and reliability.

On the other hand, NodeMCU development boards is a microcontroller equipped with an on-board ESP8266 which is a relatively inexpensive WiFi module. ESP8266 module can work with other Arduino boards while NodeMCU already has the core functions of development boards integrated together. ESP8266 WiFi module and NodeMCU provide the same transmission rate of 11 - 54 Mbps depending on the setting and modes of the device via serial communication. The major downside of using a single ESP8266 module is that it is very sensitive to operating voltage, hence the input voltage for Vin and Rx are needed to be closely adjusted.

Table 2.2 Table of Comparison of Connection Module

Module	HC-05 Bluetooth	NodeMCU Devkit V2	ENC28J60 Ethernet LAN	ESP8266 WiFi Module
				
Operating voltage	4 V to 6 V	3.0 V to 3.6 V	3.1 V to 3.6 V	3.3 V only
Operating current	30 mA	Below 80 mA	120 mA to 180 mA	100mA
Transmission Speed (up to)	3 Mbps	11 - 54 Mbps	10 Mbps	11 - 54 Mbps

Transmission Method	Bluetooth V2.0+EDR (Enhanced Data Rate)	Wifi 802.11 b/g/n	Ethernet through SPI port (Serial Peripheral Interface)	Wifi 802.11 b/g/n
Size	12.7 mm x 27.0 mm	46 mm x 26 mm	55 mm x 36 mm	25mm x 14.5mm
Price	RM 11 -16	RM 17 -20	RM 12 - 14	RM 7- 12

Comparison of Load Cell

The way of measuring water level inside the hydration tracking water bottle is using the weight of water. Hence, a weight sensor needs to be implemented in order to determine the water inside the bottle. A load cell works by using a variable resistor which changes based on the load applied. Inside the load cell comprises a Wheatstone bridge with 3 fixed resistance resistors and one variable resistor. As the resistance of the variable resistor changes, the Wheatstone bridge will output a voltage difference in between the resistors. Then the voltage will be amplified through the HX711 load cell amplifier before sending to the Arduino. There are a few types of load cell weight sensors which can be used in our project. The comparison of the load cell is shown in the table below.

Table 2.3 Table of Comparison of Load Cell

Comparison	Load cell strain gauge	Load cell straight bar
Physical appearance		
Size	34 mm x 34 mm x 3 mm	58 mm x 12.7 mm x 12.7 mm
Load range	0 kg to 50 kg	0 kg to 5 kg
Rated output sensitivity	1.0 ± 0.1 mV/V	1.0 ± 0.1 mV/V
Zero output	± 0.1 mV/V	± 0.1 mV/V

Input impedance	1000 ± 20 Ω	1066 ± 10% Ω
Output impedance	1000 ± 20 Ω	1000 ± 10% Ω
Exciting voltage	≤ 10 V	3 V to 12 VDC
Operating Temperature	0 to +50 °C	-20 to 65 °C
Price	About RM8.40	About RM14.10

Comparison of Power Supply

For the hydration tracking water bottle, the components used in this project consume power. Therefore, the electricity source is an essential element in hardware designing. The voltage and current of the power supply will be considered to support the prototype. As the size of the tracking device should not be too large and able to attach to the water bottle, small batteries should be the main considerations for the project.

A battery is a portable chemical powerpack that can produce a limited amount of electrical energy. Button cell has the smallest size that is the easiest to be deployed into the prototype but the current would be too low to drive the MCU and other components. Alkaline and Lithium batteries are in the same size and both require at least 2 cells for the prototype. The powerbank is the biggest among all considerations but it provides stable current and voltage of 5V and 2.1A respectively, it is also rechargeable means that it can be reused again multiple times, able to reduce waste. The table below shows the comparison of different types of batteries.

Table 2.4 Table of Comparison of Load Cell

	Button Cell	Alkaline	Lithium(Li-FeS2)	PowerBank
Physical Appearance				
Capacity(mAh)	500	800-1200	1200	5000
Nominal Voltage(V)	3	1.5 each	1.5 each	5V

Discharge rate	Low	Low	Medium	Medium
Rechargeable	No	No	No	Yes
Shelf life	10 years	7 years	10-15 years	5-10 years
Cost	About RM4.00	RM10.80 for 4 pieces Approximately RM2.70 per one.	About RM5.50	RM 40

2.3 Summary

From the journals reviewed, there are different approaches and methods to obtain information on personal hydration level and IoT application. By utilizing support vector machine classifiers that is machine learning, different models of containers and drinking volume can be obtained. This method has the highest working potential with the least amount of sensors applied but it needs extensive knowledge on machine learning and requires long development time. Project Grow does the similar way of connecting IoT with current projects by connecting the device to the microcontroller motherboard, the product includes a beautiful physical view for the aesthetic which is a late-product physical enhancement after finishing the prototype.

Journals from Jovanov and Mengistu applied that same approach to user hydration data by integrating sensors into the human body. Integration using a personal body area is quite power efficient compared to other sensing methods but requires extra deep knowledge on biomedical and physiology which need further studies. A cloud-based solution provides seamless integration for a large number of users using a standard infrastructure but requires high power consumption. At last, the Smart Bottle project conducted by Tommy and others is the closest to what the current project is going toward.

NodeMCU is chosen as the microcontroller to achieve this project. NodeMCU development boards is a microcontroller equipped with an on-board ESP8266 which is a relatively inexpensive WiFi module. NodeMCU already has the core functions of development boards integrated together. NodeMCU provides a transmission rate of 11 - 54 Mbps depending on the setting and modes of the device via serial communication. We chose a power bank as the load cell to supply the required power for the system. This is to ensure that enough current and voltage is provided to the system and no power shortage will occur anytime soon.

Chapter 3

Methodology and Work Plan

3.1 Introduction

Hydration Tracking Water Bottle is to be designed and built to monitor and track the water consumption of the user and send the data to the cloud for analysing before displaying it to the user's smartphone by wifi connection. The raw data that the sensor outputs are processed by an IoT platform and sent to an Arduino module for data review. Our workflow is represented by the flowchart below. The general flow is as follows, planning, executing, testing, improving and finishing.

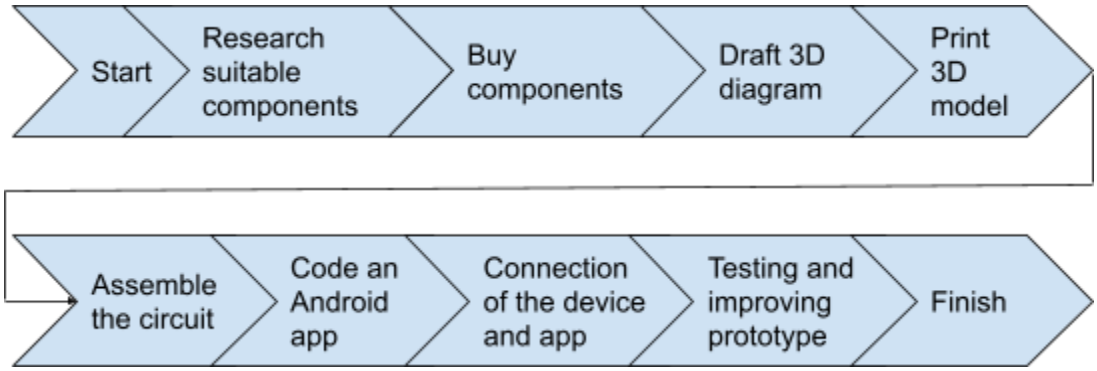


Figure 3.1 : Methodology of Hydration-tracking Water Bottle

3.2 Requirements and Specification

In order to track a person's water consumption, we need to obtain the water intake which a person drinks. This can be done by implementing a device on the water bottle to measure the amount of water left.

3.2.1 Methods of Obtaining Data

The first method that we considered is to use an accelerometer and gyroscope that is attached to the water bottle. The sensor will detect when the user uses the bottle to drink water. Both sensors will be

connected to an Arduino board. Due to this method using the sensor to estimate water level, many errors might occur during the application and usage as there is no clear indication of water being consumed or not.

The second method is to use an ultrasonic sensor together with an Arduino module placed on the bottle cap. The ultrasonic sensor is able to measure the water level accurately by sending a signal pulse and measuring the time taken for the sensor to reflect back after hitting the water surface. After obtaining the time for the signal to reflect back to the sensor, we can use physics properties and programming in Arduino to determine the water level. The crucial drawbacks of this method are the ultrasonic sensor does not blend well with humidity inside the bottle and the device needs to be redesigned for different types of water bottle as different bottles have different volumes.

The third method is to use a weighing scale sensor and an Arduino board. This device will be attached to the bottom of the water bottle. For the scale to accurately estimate the water level in the bottle, it needs to do calibration by taking the weight of the bottle when empty. Since 1g = 1 ml, the data from the device can be directly converted to the amount of water consumed by the user in milliliters. Since the device operates on the exterior of the water bottle, there is no need for waterproofing. Moreover, if we use elastic material to wrap around the base of the bottle, it will be universally compatible with other water bottles as it is stretchable.

3.2.2 Hardwares and Sensors

Development Board and Connection Module

NodeMCU DevKir V2 is chosen for this project among other types of development boards. This development board not only comes with Tensilica Xtensa 32-bit LX106 RISC microprocessor with ESP8266 chip, also with adjustable clock frequency ranging from 80Hz to 160Hz. The input voltage for this development board is regulated to 3.3V with the application of voltage regulator, or else any input voltage higher than 3.3V will damage the ESP-8266, rendering the board broken and unusable. NodeMCU also exempted from using an additional connection module as this development board is built on ESP8266 WiFi module, enabling direct 2-way communication between the IoT platform and the chip without massive delay.

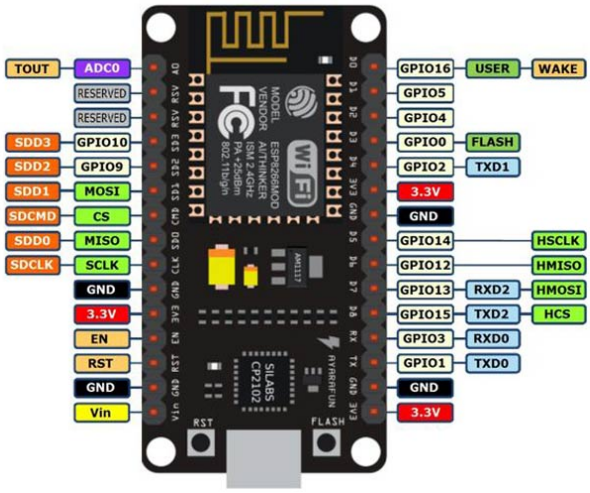
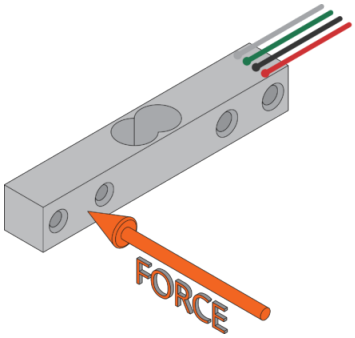


Figure 3.2 The pins of NodeMCU Devkit V1

Load Cell Sensor

Between the two load cells discussed, a Straight Bar load cell with range of 0-5kg is chosen for the prototype. The reason why this type of load cell is chosen is because it is easier to deploy into the project rather than the other type. Straight bar load cell only needs a few M3 * 12mm Phillp screws installed onto 3D printed platform and structure of the prototype while using a strain gauge load cell require 3B printed platform with accurate design that will fit it in. The accuracy of detected range is also considered as the volume of water detected is needed to be accurate, straight bar type supports a smaller range with 0 - 5 kg while strain gauge type supports range of 0 - 50 kg.

Next, each of the straight bar load cells is configured with a wheatstone bridge setup. With such, the user of this type of load cell does not have to configure a wheatstone bridge on their own with the same type of additional load cell. On the other hand, strain gauge load cell require wheatstone bridge configuration for reading the weight data. The wheatstone bridge works by reading the varying voltage of the wheatstone bridge.



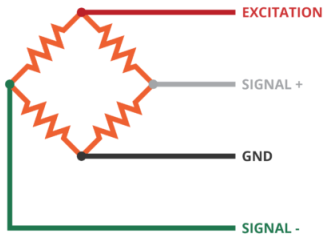


Figure 3.3 The configuration of Wheatstone Bridge inside straight bar load cell

Table 3.1 The nodes of Wheatstone Bridge and the wire color represented

Wheatstone Bridge Nodes	**Wire color**
Excitation + (E+) / Vcc	Red
Excitation - (E-) / Ground	Black
Output Voltage - (O-)	White
Output Voltage + (O+)	Green

Load Cell Amplifier

To increase the accuracy and sensitivity of the weight of the water, a load cell amplifier, HX711 is implemented into the prototype. This load cell amplifier is a 24-bits analog-to-digital converter (ADC) to have a direct interface with wheatstone bridge seput for weight scaling application. (24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales, n.d.) According to the datasheet provided by Avia Semiconductor, the current consumption during normal operation is lower than 1.5mA while the normal operating voltage is between 2.6V to 5.5V. This amplifier will be connected to the 3.3V output pin of the NodeMCU that provides maximum controller current dissipation of 12mA. So, it is safe that NodeMCu will not overcharge the amplifier and break the HX711 circuit board.

Besides, HX711 also has adjustable sampling values ranging from 2 to 128 that can be adjusted via the configuration file in the Arduino library. With lowering sampling value, the memory usage and the sampling time are lowered while sacrificing the smoothing of the returned value. In this project, the sampling value is set to 2 so that the sampling value will be read slower to suit the interval of ThingSpeak data upload.

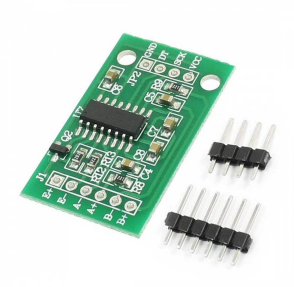


Figure 3.4 The physical appearance of HX711

Power Supply

The power supply selected for this prototype is the power bank with 5000mAh supply. The reason why the power bank is selected rather than the batteries is because that the batteries does not meet the requirement of the current to power up NodeMCU and HX711 load cell amplifier. NodeMCU Devkit V1 used in this project requires operating current of 215mA. (ESP8266EX Datasheet, 2015) Due to this, we have to use a larger power supply which is the power bank to power up the prototype. It supplies a constant 5V and 2.1A which fulfil the requirement of voltage and current of NodeMCU to constantly supply power to itself and the load cell amplifier. The circuit board of NodeMCu is connected to a stripped wire of USB connector to connect to the power bank externally.

Figure 3.5 The physical appearance of power bank used

Proteus Circuit Diagram

The prototype is connected and configured according to the Proteus diagram below. To finalize the project, the components are going to be solder on the veroboard for smaller size and easy implementation.

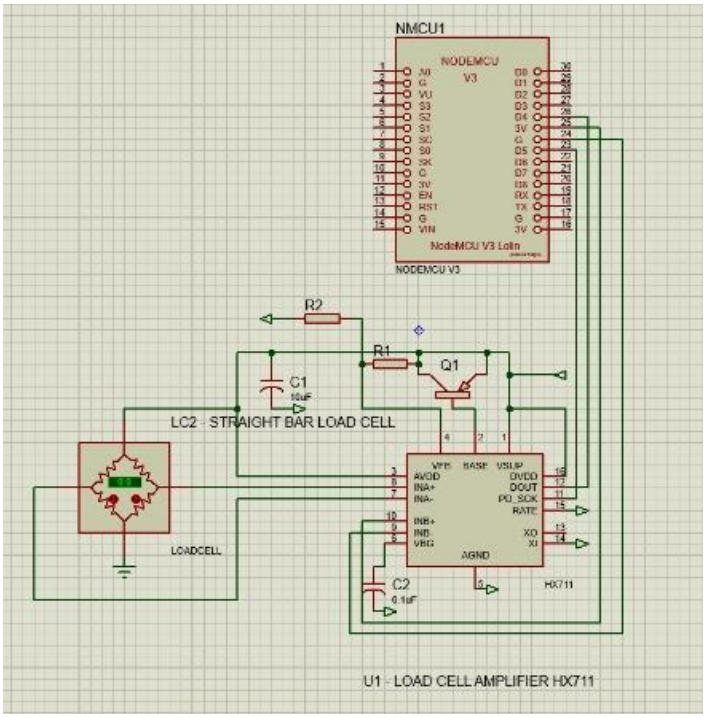


Figure 3.6 The circuit diagram of the prototype simulated using Proteus

3D Model Design

The 3D model is constructed using Tinkercad. This platform was chosen because it is relatively easier to model a 3D model for beginners as compared with using professional software like Solidworks. Not only that, this modeling application is an online platform, it does not require any installation and it is open-sourced which is free to use. The diagram of the 3D model shown below is captured using an online free STL viewer website (https://www.viewstl.com/).

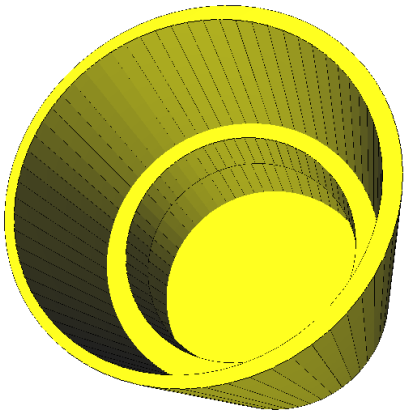


Figure 3.7: 3D Model of the Outer Structure

The outer structure is an emptied out cylinder. There is an inner ring for the placement of the load cell. We started out constructing the outer structure because it is the main structure of this project. The following part is designed and measured based on this cylinder.

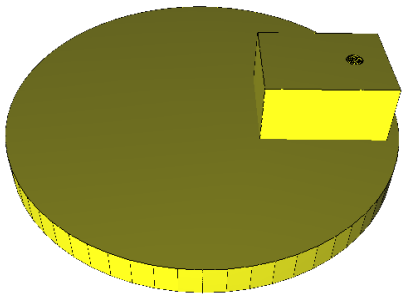


Figure 3.8: Bottom plate with a protruding rectangle block

The components are hidden beneath the load cell to make the product more aesthetic. The protruding rectangle block contains a M3 screw hole for attaching the load cell weight sensor. One end of the load cell will be attached to the screw hole.

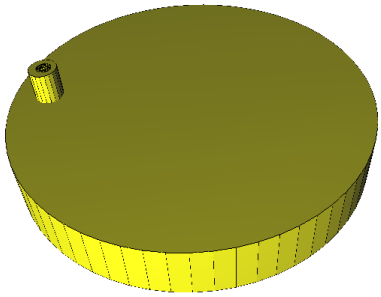


Figure 3.9: Upper plate with a protruding cylinder block

The cylinder block projecting out of the plate consists of another M3 screw hole for attaching the other end of the load cell weight sensor. It is supposed to be facing down. The flat plate is faced up for supporting the water bottle. Note that the upper plate is thicker than the bottom plate due to the weight of the water and the bottle.

3.2.3 IoT Implementation - Data Storage and Monitoring

After the load cell collects water weight data, the program will process the data in real-time and send it to the cloud every 15 seconds using the ThingSpeak cloud computing platform. There are a lot of data storage and monitoring platforms available on the Internet such as Blynk, adafruit.io and Google Cloud but ThingSpeak is chosen to be used in this project.

ThingSpeak is an IoT analytics platform service which allows users to aggregate, visualize and analyze live data streams in the cloud (Anon., n.d.). It is an IoT platform developed by MATLAB® from MathWorks. The reasons for ThingSpeak to be used in this project is because of its easy device configuration with NodeMCU Devkit V2 and being able to analyze the project's real-time data. After that, the data collected can be visualized in graph or chart forms for monitoring purposes. In this project, a ThingSpeak account is created and the data of water consumption are received from the ESP8266 WiFi connected to the Thingspeak Server. The hydration tracking project will be assigned with a channel and a WriteAPI whereas the variables can be viewed from the graph below after the configuration is completed.

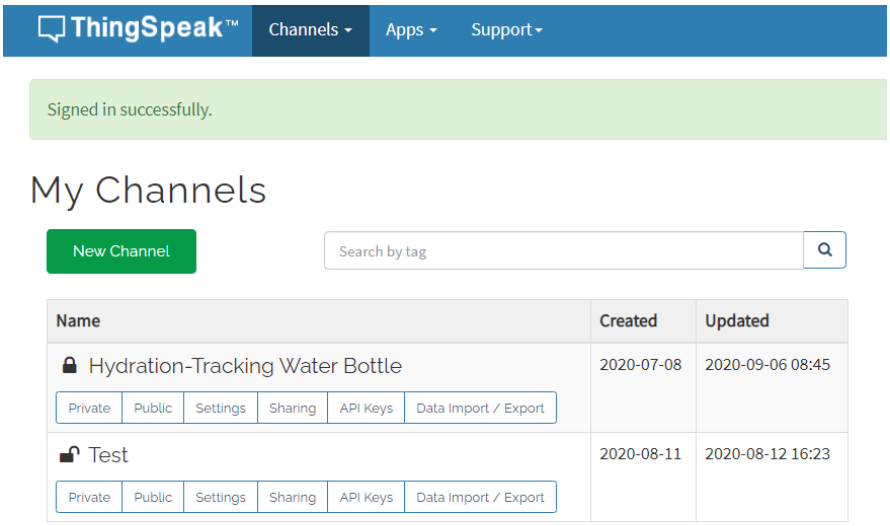
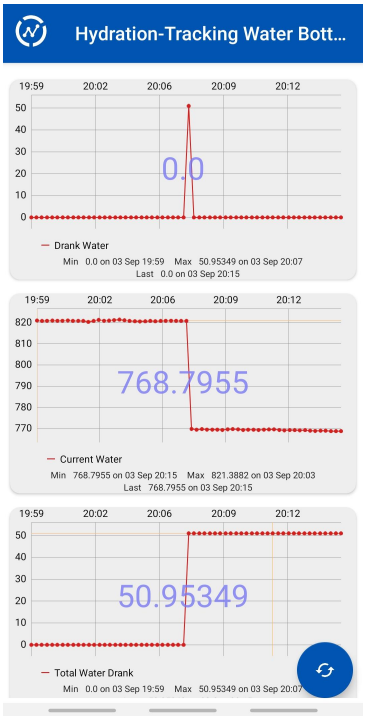


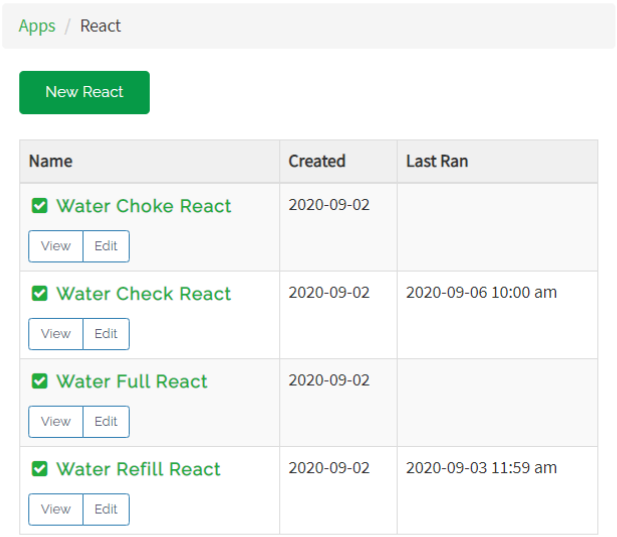
Figure 3: The Hydration Tracking Channel and graph preview created in ThingSpeak

For the monitoring system, a free applet named 'ThingsView' is installed for smartphones available on Google Play Store and IOS App Store. With ThingsView app, the data from the channel can be accessed through a mobile device instead of by web on PC. Providing an easier access for the user to view their water intakes and embracing the concept of IoT.



For the IoT alert system, React and Twitter are applied for the project. The prototype's alert system will notify the user whether their water intake reaches the average daily target or remind them to refill their water bottle when it is near empty. React is another application developed by MathWorks for the intention of enhancing IoT usability with other applications. React can work with other applications of theirs such as ThingHTTP, ThingSpeak, MATLAB Analysis and ThingTweet which will be implemented into the alert system. This application works by processing the received field data from ThingSpeak and performing actions based on the preset condition.

As mentioned above, ThingTweet application is used in collaboration with React to develop the alert system. ThingTweet allows the user to connect a twitter account, linking the account to React will enable tweet posting on the twitter account when the condition is met on React. This allows the user of the prototype to get various updates on the water consumptions throughout their usage.



3.2.4 Arduino Programming

The program for the hydration tracking water bottle has two parts which are calibration and main program. Before running the main program, the calibration program needs to be run first so that a calibration scale factor of the load cell can be obtained. The purpose of sensor calibration is to minimize any measurement uncertainty that may affect the accuracy of water weight taken in the main program. An EEPROM library is used in both the calibration and main program where it is a memory to save the values even when the microcontroller board is turned off. This library allows users to read and write those values.

When running the program, messages will be shown in the serial monitor to notify users about the beginning of calibration. The load cell is tared before measuring and it will require users to remove any load applied to the load cell. After taring, a message will tell the users to place a known mass on the load cell so that the new calibration value can be recorded. The program will ask the permission of users whether they want to save the last calibration value to the EEPROM address and users can reply by typing a ‘y’ or ‘n’ in the serial monitor. If users typed ‘y’, the value will be saved, else the value will be ignored. The program can be set to re-calibrate by sending an ‘r’ from the serial monitor after the calibration end message is shown.

The function of changeSavedCalFactor can be called out by sending a ‘c’ from the serial monitor. This function allows users to edit the calibration value manually. In the function changeSavedCalFactor(), the value obtained from the getCalFactor() will be set as the oldCalibrationValue. A new value is sent from the serial monitor and declared as newCalibrationValue. Then, the program will check whether the new value is larger than zero. If yes, the program will ask the users’ permission to save the value to EEPROM, else the new value is discarded. The program saves the new calibration value to the EEPROM after acquiring the agreement from users.

The main program running the hardware is to measure the amount of water consumed by the consumer each day. The ESP8266Wifi and ThingSpeak libraries are defined in the program so that the data is able to be sent to ThingSpeak IoT platform for further data analysis. The wifi name and password need to be set first for NodeMCU to connect to the Internet via WIFI. The calibration factor will be declared and set to the value obtained from the calibration program previously.

A float variable ‘current_water1’ will be declared initially taking the value of 0. The main program runs in a loop. In the loop, the float variable ‘current_water2’ is to introduce the value obtained from the load cell and it will then be printed in ml. After delaying for 1 second, calculate drank_water, where the function cal_water() is executed. The cal_water() function is used to find the difference between cuurent_water1 and current_water2. An if statement is used to check the condition whether the difference between cuurent_water1 and current_water2 is less or equal 10 or current_water2 is more than current_water1. The drank_water will become 0 if any one of the conditions is fulfilled.

By obtaining the value for drank_water, the above value will be added to the already existing variable ‘total_water’, and print the value obtained by ‘total_water’ in 2 decimals places. By an if statement of prevtime = curTime, the ‘total_water’ variable saved will be reset, thus achieving resetting the data stored for consuming water on a daily basis. Whether resetting or not the loop will run again from the start. Then, the values for drank_water, current_water2 and total_water will be sent through ThingSpeak via the Internet. In ThingSpeak, three fields are created for these three values in order to perform data analysis and monitoring.

Chapter 4

Results and Discussion

4.1 Physical Configuration and Connection

Before assembling the components into one final body, various tests and configurations are run on the pre-assembled prototype. The data from the tests showed no error and ready to be assembled as a finalized prototype. There are multiple changes on the 3D printed case of the prototype, the upper and lower platforms of the prototype are sanded flat to make room for cables and wire to pass through. After assembly, the prototype is tested for acquiring data and charts as results for discussion.

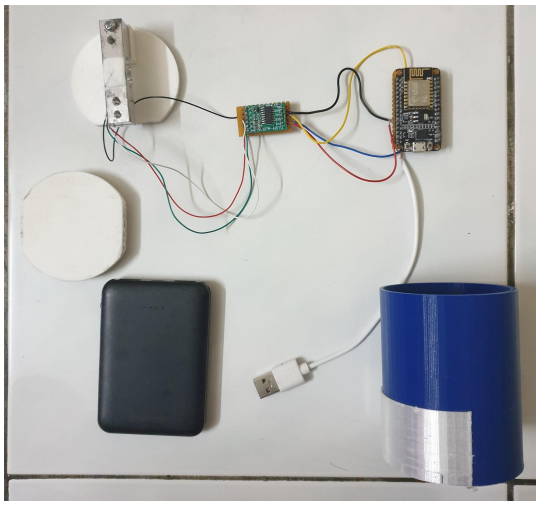


Figure 4.1 The pre-assembled prototype

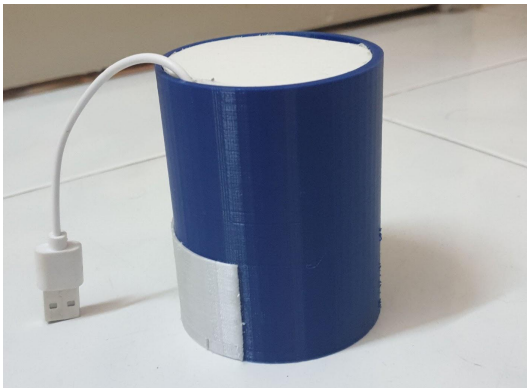


Figure 4.2 The finalized prototype

4.2 Arduino Serial output

Arduino IDE is used for programming the prototype that uses NodeMCU as the development board. To cross-check the value of the variables parsed to the ThingSpeak Channel, the program will display the value of the variables in the Arduino Serial output while connected to the development board.

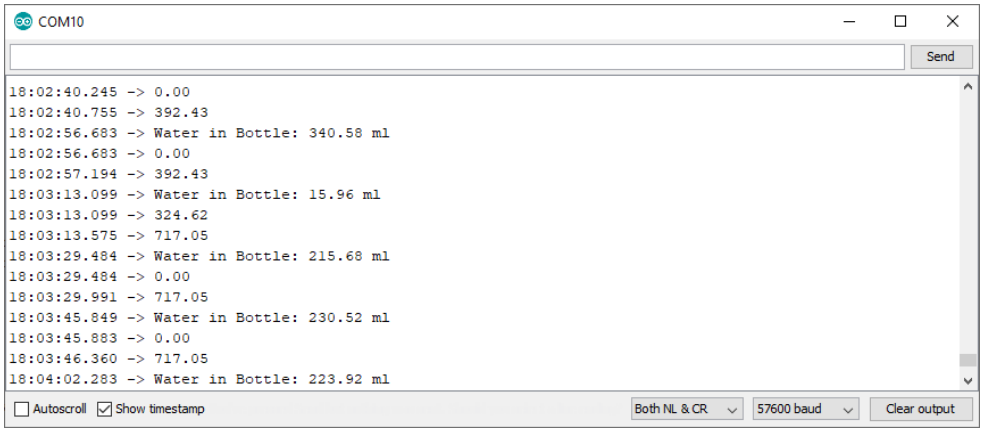


Figure 4.3 The serial output of Arduino IDE

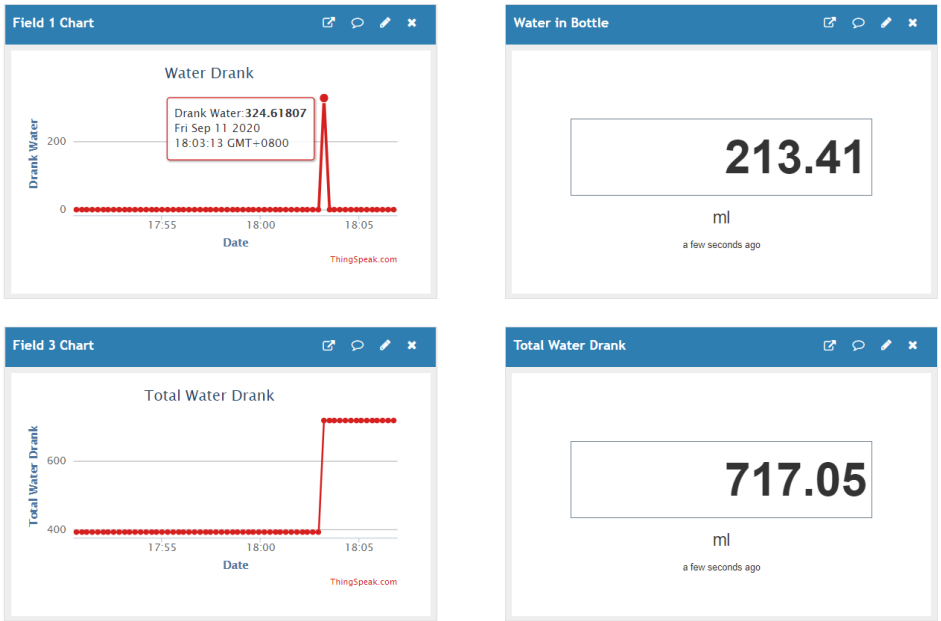


Figure 4.4 The charts of ThingSpeak channel

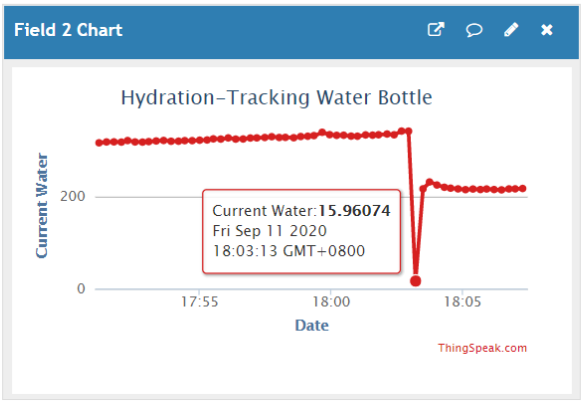


Figure 4.5 The chart of current left in water bottle

To test on the accuracy of data parsed, the output of Arduino Serial and ThingSpeak channel data are compared. In Figure 4.3, Arduino Serial output displayed an original water level of 340.50 ml. It is then reduced to 15.96 ml as the water has been drunk by the tester, the variable 'drank_water' has been assigned to 324.62 ml as result. Referring to the ThingSpeak channel response, the water drank chart has the same value of 'drank_water' as in the arduino serial, the data input time is also the same for both platforms. The same conditions also happened on the 'total_water_drank' variable, the value of this variable has increased to 717.05 ml on ThingSpeak Total Water Drank chart and numeric display and so does at for the value of the same variable in Arduino Serial. Besides, the value of water left inside the bottle for both platforms has the same dip on instance of 18:01:13.

Based on the conditions stated above, the data passed from the NodeMCU to the ThingSpeak channel is accurate and done successfully. The NodeMCU established connections to the ThingSpeak channel with the use of 'ThingSpeak' library that can be downloaded inside Arduino's Library manager. To initiate a connection between the two platforms, the program needs to include the SSID which is the WiFI name and the password of the WiFi as parameters. After connecting to the local WiFi, the channel ID and WriteAPI are also needed to be declared in the program. To pass the value of the variables to the channel, the variables are assigned with different fields that are declared in the ThingSpeak channel. The function of passing the variables data is obtained from the 'ThingSpeak' library.

4.3 Thingspeak

Thingspeak is an online open-source Internet of Things (IoT) platform. Users can upload their data to the cloud service for real-time analysis. Moreover, Thingspeak has a trigger and act function where the data will trigger a response after reaching some threshold set by the user. This response can switch on/off an actuator or a switch. In our case, the response is to send out a tweet on Twitter.

Steps to transmit data to cloud via Thingspeak:

1. Go to https://thingspeak.com/ and create an account.

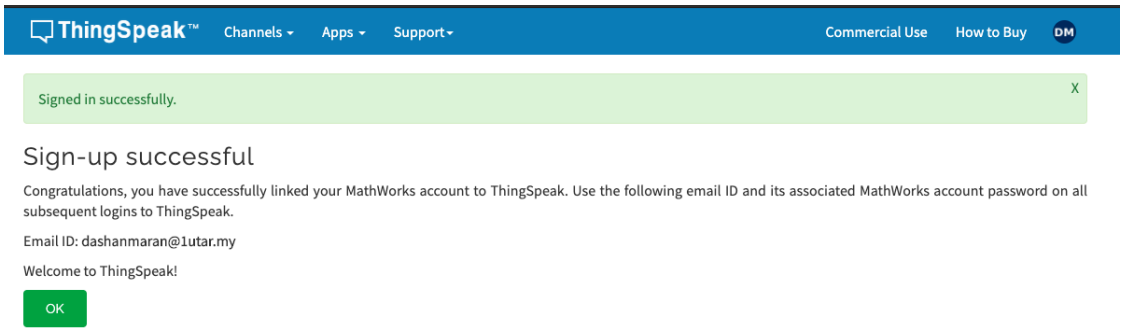


Figure 4.6.1

2. Login to your account.

3. Create a new channel by clicking on the button.

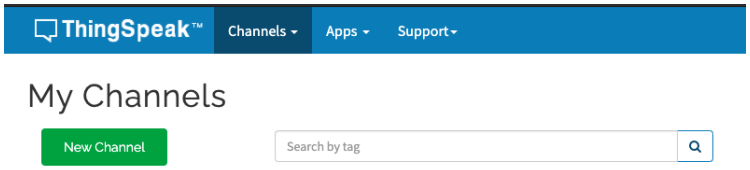


Figure 4.6.2

4. Enter basic details of the channel and save the channel

5. Channel ID is the identity of your channel.

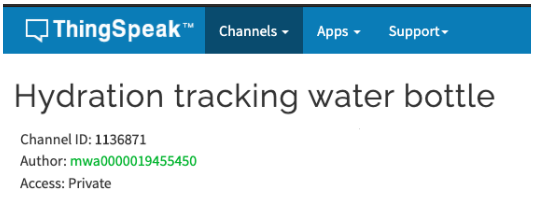


Figure 4.6.3

6. Then go to API keys, copy and paste this key into the NodeMCU code later.

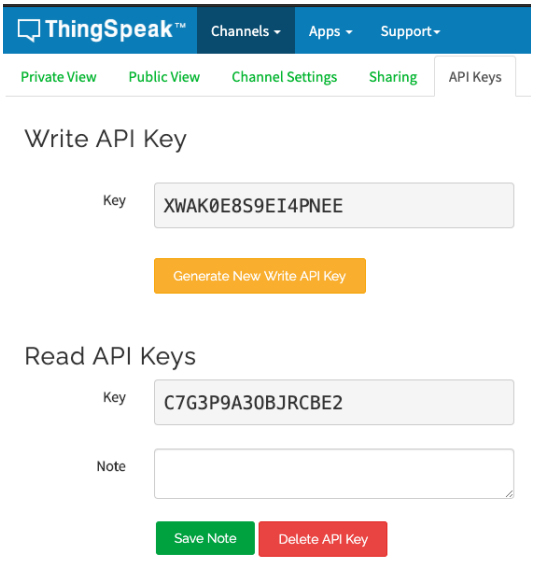


Figure 4.6.4

7. Program the API key into the NodeMCU code.

8. Open your channel at Thingspeak and output will be shown.

In this project, the data of weight of water is sent to ThingSpeak server. When the user drinks water, the remaining water inside the bottle will become different. As seen from the graph, the water level changes with respect to time. Meanwhile, by subtracting the original amount of water, the volume of water consumed by the user can be calculated.

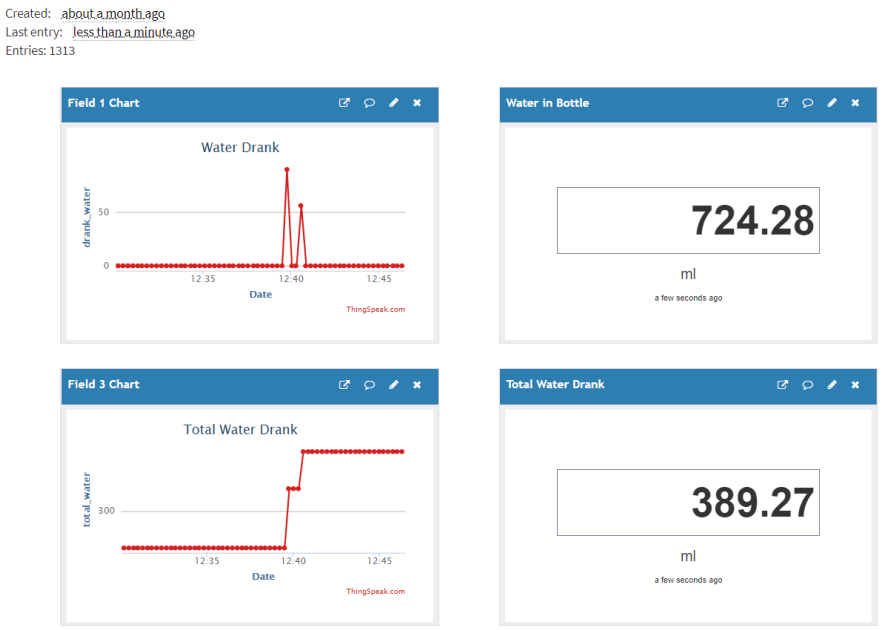


Figure 4.7: Graphical and numerical data of water consumed as seen from ThingSpeak

Using ThingSpeak, the graph and data of the water in the bottle and water consumed is generated and captured, The graph data on the top left of figure 4.7 shows the amount of water consumed at a particular instance. On the top right graph, it shows the amount of water left in the bottle with 2 decimal places accuracy. The bottom of the figure 4.7 shows the total amount of water consumed with a cumulative graph on the left.

A feature is added to remind the user to drink more water when the water they drank is below threshold. As previously discussed, the history of water that the user consumed is known. Condition 1 is set where if the water drank is less than a set amount in a time period, ThingSpeak will call the ThingTweet function to tweet out a reminder to drink more water. Not only that, different conditions can trigger different messages to be tweeted.

The reliability of ThingSpeak is reasonably good in the prototype. There is minimal connection losses or data miscommunication between the prototype and ThingSpeak server. However, the ThingTweet is not very dependable as it uses Twitter, a third party application and the Twitter account may be temporarily banned because the tweeted message is similar most of the time.

The accuracy of ThinkSpeak is very precise. This is true especially for the data, as it allows up to 5 decimal places. Not only that, when the condition is met, ThinkSpeak will trigger other functions as directed.

4.4 Mobile App (Thingstweet and ThingView)

Other than accessing the ThingSpeak website, the project provides another option for the users to check out their water consumption, which is the mobile application-ThingView developed by ThingSpeak developers. In order to access the data from ThingSpeak, users are required to key in the specific channel ID and the API key will be needed if the server is set private.

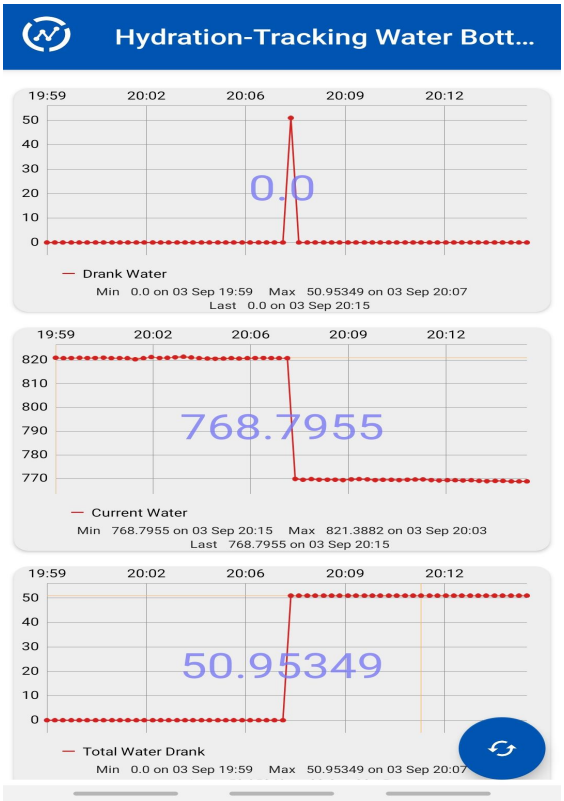


Figure 5.1: Mobile Application (ThingView) for Data Monitoring

The figure above shows the data collected from the hydration tracking water bottle. As there are three fields created in ThingSpeak: drank water, current water and the total water drank, there will be three graphs representing them in ThingView application respectively. For the first graph, it displayed the water drank by the user from 19:59 to 20:12 pm on 3 September 2020. The user drank up to 50.95349 mL of water during 20:07 pm that day. When the user did not consume water, the graph remained zero. For the second graph, it showed the current volume of water contained within the water bottle. The water bottle contained a volume of 821.3882 mL before 20:06 pm and it dropped to 768.7955 mL after the user drank a certain amount of water. The third graph displayed the total water drank by the users. The graph remained as zero and raised to 50.95349mL after the only water intake that day was consumed by 20.06. The ThingView has a satisfying feature that may bring convenience to the users where it displays the final results in numerical form behind the graphs and the providing an accurate result of up to 5 decimal places. The maximum and minimum water consumed or remained achieved during the days will be shown under the graphs as the information for users.

Besides, the users can link their Twitter account to the project's ThingSpeak account by using the ThingTweet app. React allows us to trigger a ThingHTTP request or send a tweet using ThingTweet when our ThingSpeak Channel meets a certain condition.The React will trigger a HTTP request to the ThingTweet and notifications will be updated in real-time through the users' twitter account using TwitterControl API for alert purposes. There are four conditions created to trigger actions to send tweets to users as shown in Figure 5.2.1 to Figure 5.2.4. In Figure 5.1,when the user drinks water exceeding 250 mL which is the threshold value per instance, a tweet of "Hey there, drink slowly and don't choke yourself" will be sent to the user's Twitter to advise him that drinking water too fast is discourageable. On the other hand, a message of "Please refill your bottle" will be sent as a reminder when the React detects that the water contained in the water bottle is less than 300 mL. When the user consumes a total water of less than 3000 mL which is the daily target, he will receive a notification to remind him to drink more water. The React will trigger an action to inform the user that he has successfully achieved the target of drinking 3000 mL water in a day when the user exceeds the threshold value set in Field 3, The

Name:	Water Choke React
Condition Type:	Numeric
Test Frequency:	On data insertion
Last Ran:	
Channel:	Hydration-Tracking Water Bottle
Condition:	Field 1 (Drank Water) is greater than 250

alert messages sent to the Twitter account when the user fulfills the conditions set are shown in Figure 5.5.

Figure 5.1: Notification will be sent when water drank per intake is more than 250 mL

Name:	Water Refill React
Condition Type:	Numeric
Test Frequency:	On data insertion
Last Ran:	2020-09-03 11:59
Channel:	Hydration-Tracking Water Bottle
Condition:	Field 2 (Current Water) is less than or equal to 300
ThingTweet:	HydrationT: Please refill you bottle!
Run:	Each time the condition is met
Created:	2020-09-02 10:14 am

Figure 5.2: Notification will be sent when water contained in water bottle is less than or equal to 300 mL

Name:	Water Check React
Condition Type:	Numeric
Test Frequency:	Every 60 minutes
Last Ran:	2020-09-08 14:00
Channel:	Hydration-Tracking Water Bottle
Condition:	Field 3 (Total Water Drank) is less than 3000
ThingTweet:	HydrationT: Remember to drink more water! You haven't reach daily target yet .
Run:	Each time the condition is met
Created:	2020-09-02 10:00 am

Figure 5.3: Notification will be sent when total water consumed is less or equal to 3000 mL

Name:	Water Full React
Condition Type:	Numeric
Test Frequency:	On data insertion
Last Ran:	
Channel:	Hydration-Tracking Water Bottle
Condition:	Field 3 (Total Water Drank) is greater than or equal to 3000
ThingTweet:	HydrationT: YAY! You drank over 3 Litres of water today!
Run:	Only the first time the condition is met
Created:	2020-09-02 10:01 am

Figure 5.4: Notification will be sent when total water consumed is more or equal to 3000 mL.

Figure 5.5: Notification received when the conditions set are fulfilled.

Chapter 5

Conclusion

5.1 Conclusion

For this project, Hydration Tracking Water Bottle, the group had learnt a lot of skills, stopped and overcame different obstacles throughout the project duration. One important point to take note of in a project is time management. To keep the project done in time, the group has to keep up with the planned Gantt Chart. It is very difficult to meet the objective designated in the Gantt Chart but it makes sure the project will be completed on time. The project also requires teamwork and contribution from each member, this ensures that no members are left behind and extra workloads are given to the others. With that, the project will go on even smoothly as everyone is having progress and learning together throughout the project.

To conclude this project, the objectives are achieved as a prototype has been built for measuring the water drank by the user. The prototype also satisfies the implementation of IoT for data transmission and storage of monitored data with the application of ThingSpeak and ThingView in webpage and mobile application. Besides, the prototype with ThingView application installed in smartphones allow the user to have a better understanding of their daily water consumption with notification from React that makes HTTP Requests to Twitter.

5.2 Challenges and Improvement

Throughout the term of the project, there exists some major challenges that halted our project duration. The major challenge is that physical meetings have not been done in the second semester of the project. The team is not able to meet physically to discuss the project at UTAR due to the COVID-19 pandemic that induced MCO lockdown. Without physical meeting, the team is not able to discuss everything

face-to-face efficiently. Next, to come out with a physical prototype, the team members have to learn about 3D model drawing. Not having a group member with a solid and matured knowledge on 3D drawing skill, the team has to take time to learn about 3D modelling from scratch, delaying the development of the prototype as the result.

As the completion of the prototype, various improvements can be made for the project if the conditions are met. For example, the prototype could have been added with smaller rechargeable batteries as the power source. Next, the infrared sensor method of obtaining water level data can be applied rather than the weighting method if there is a convenient way of making the hardwares water-resistance. An android app interface for the hydration tracking water bottle could have been created and designed by the team, supplementing the final product with user-friendliness on the software interface.

REFERENCES

1) Alkaline Power - AAA. (2020, August 25). Retrieved July 12, 2020, from https://www.panasonic-batteries.com/en/alkaline/alkaline-power/alkaline-power-aaa
2) Ashton, K. (2009). That ‘internet of things’ thing. RFID journal, 22(7), 97-114.
3) Arduino Nano. (n.d.). Retrieved from store.arduino:https://store.arduino.cc/usa/arduino-nano
4) Arduino vs ESP8266 vs ESP32 Microcontroller Comparison. (2020, August 03). Retrieved July 15, 2020, from https://diyi0t.com/technical-datasheet-microcontroller-comparison/
5) Arduino Vs ESP8266 Vs ESP32 Microcontroller Comparison. (2020, September 9). Retrieved fromDiyIOT: https://diyi0t.com/technical-datasheet-microcontroller-comparison/
6) Arduino Uno Rev3. (n.d.).Retrieved from store.arduino:https://store.arduino.cc/usa/arduino-uno-rev3
7) Brecher, J. (2017, May 31). What you should know about drinking water (but probably don't). Retrieved from BETTER: https://www.nbcnews.com/better/diet-fitness/down-low-h20 -n760721
8) Crosta, P. (2017, December 20). What You Should Know About Dehydration. Retrieved from MEDICALNEWSTODAY: https://www.medicalnewstoday.com/articles/153363
9) Davis, C. P., MD, PhD. (2019, August 13). Dehydration in Adults Treatment, Causes, Effects & Symptoms. Retrieved from emedicinehealth: https://www.emedicinehealth.com/ dehydration_in_ adults/article_em.htm#is_it_po ssible_to_prevent_dehydration_in_adults
10) Durbin, D and Horn, Q. (2017, April 9). BU_106: Advantages of Primary Batteries. Retrieved from BatteryUniversity: https://batteryuniversity.com/learn/article/primary_batteries
11) E. Jovanov, V. R. Nallathimmareddygari and J. E. Pryor, "SmartStuff: A case study of a smart water bottle," 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Orlando, FL, 2016, pp. 6307-6310.
12) ENC28J60 Ethernet Module. (2006). Retrieved from elecrow: https://www.elecrow.com/download/ENC28J60%20Datasheet.pdf
13) ENC28J60 Ethernet Module. (n.d.).. Retrieved from elecrow: https://www.elecrow.com/enc28j60-ethernet-module-p-587.html
14) ENC8J60 - Wiki. (2017, March 20). Retrieved from SunFounder: http://wiki.sunfounder .cc/index.php?title=ENC8J60#:~:text=The%20controller%20is%20designed%20to,maximum%20speed%20of%2010Mb%2Fs.

15) Griffith, H., & Biswas, S. (2019). Container Type and Fill Level Classification Using a Bottle-Attachable IMU Sensor. 2019 IEEE 23rd International Symposium on Consumer Technologies (ISCT). doi:10.1109/isce.2019.8900999
16) HC-05- Bluetooth to Serial Port Module. (2010, June 18). Retrieved from components101: https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf>
17) HC-05 - Bluetooth Module. (2018, March 10). Retrieved from components101: https://components101.com/wireless/hc-05-bluetooth-module
18) Hunger vs. thirst: tips to tell the difference. (n.d.). Retrieved from PKD FOUNDATION BLOG: https://pkdcure.org/blog/hunger-vs-thirst/
19) Industrial Devices & Solutions. (n.d.). Retrieved August 18, 2020, from https://industrial.panasonic.com/ww/products/batteries/primary-batteries/lithium-batteries/models/CR3032
20) Instructables. (2017, September 21). Arduino Powered Water Bottle. Retrieved from https://www.instructables.com/id/Arduino-Powered-Water-Bottle/
21) Instructables. (2017, October 10). Measuring Water Level With Ultrasonic Sensor. Retrieved from https://www.instructables.com/id/Measuring-water-level-with-ultrasonic-sensor/
22) Kaner, Gül & Genç, Uğur & Dinçer, Salih Berk & Erdogan, Deniz & Coskun, Aykut. (2018). GROW: A Smart Bottle that Uses its Surface as an Ambient Display to Motivate Daily Water Intake. 10.1145/3170427.3188521.
23) Khatri, M. (2019, May 30). WebMD. Retrieved from What is Dehydration?What Causes It?: https://www.webmd.com/a-to-z-guides/dehydration-adults#1
24) Load Cell Straight Bar 0-50kg Weight Sensor for Arduino IoT. (n.d.).. Retrieved from shopee: https://shopee.com.my/Load-Cell-Straight-Bar-0-50kg-Weight-Sensor-for-Arduino-IoT-i.40459773.789073370
25) Load Sensor. (n.d.). Retrieved from Sparkfun: https://www.sparkfun.com/datasheets/Sensors/loadsensor.pdf
26) Nodemcu ESP8266. (2020, April 22). Retrieved from components101:https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet
27) React - Thingspeak Iot. (n.d.). Retrieved from ThingSpeak: https://thingspeak.com/apps/reacts
28) TAL220B Miniature Load Cell [PDF]. (n.d.). HTC-Sensor.
29) Tommy, R., Raja, V., & Krishna, A. S. (2018). Interactive Water Bottle Application on a Smartphone to Enhance User Water Consumption. International Conference on Current Trends in

Computer, Electrical, Electronics and Communication, CTCEEC 2017, 1072–1073. https://doi.org/10.1109/CTCEEC.2017.8455048

30) What Is The Difference Between The Arduino Nano And The Arduino Uno? (n.d.). Retrieved from Quora:https://www.quora.com/What-is-the-difference-between-the-Arduino-Nano-and-the-Arduino-Uno#:~:text=One%20key%20difference%20between%20the,when%20you%20need%20breadboard%20compatibility

31) 24-Bit Analog-To-Digital Converter (ADC) For Weigh Scales. (n.d.). Retrieved from sgbotic: https://www.sgbotic.com/products/datasheets/sensors/hx711_english.pdf

APPENDIX

SURVEY

WE MAKE WHAT YOU WANT

SURVEY ON HYDRATION TRACKING WATER BOTTLE

Please tick the boxes with the options that you prefer.

1. What is the size of water bottle that you use?

500mL	
1L	
2L	

2. Do you have a smartphone?

YES	
NO	

3. Have ever used a hydration tracking device before?

YES	
NO	

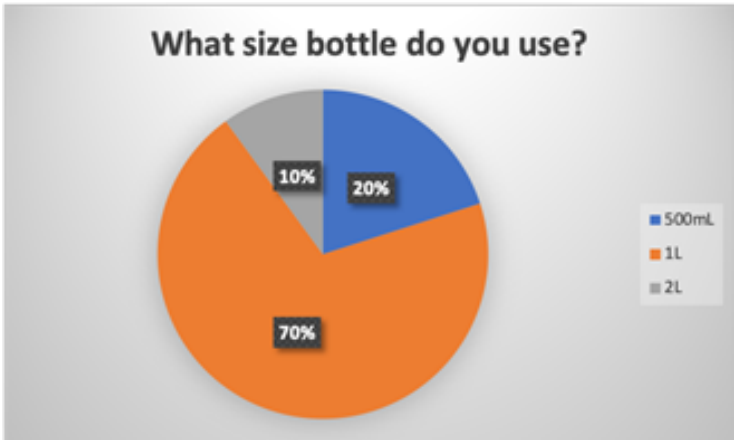
4. Which OS do you prefer?

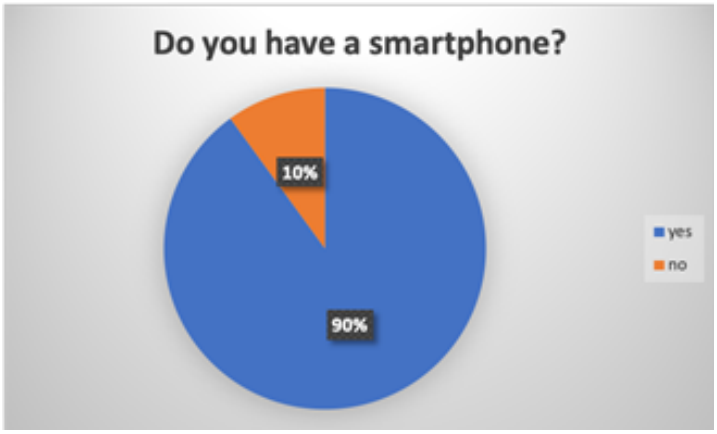
ANDROID	
IOS	

5. What are some features that you expect in a hydration tracking device? (you are allowed to choose more than 1)

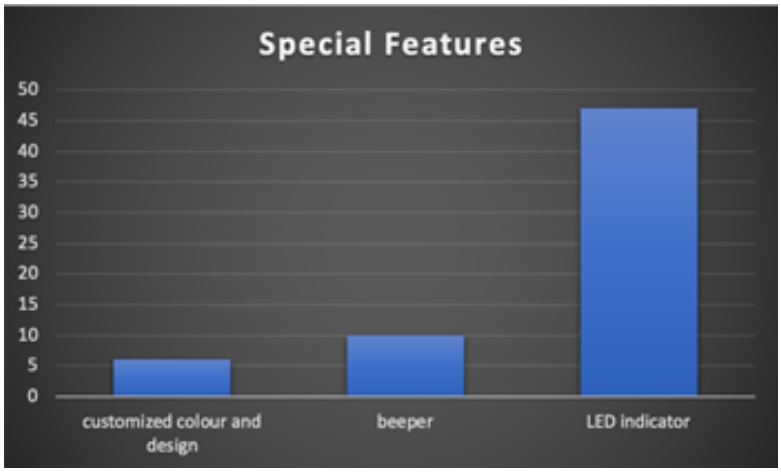
Customized colours and design	
Beeper to notify when you are behind track	
LED indicator to indicate water intake	

RESULT OF SURVEY






Which OS do you prefer?
40%
60%
android
ios


Special Features
50
45
40
35
30
25
20
15
10
5
0
customized colour and design
beeper
LED indicator

Code for Calibration

```
#include <HX711_ADC.h>
#include <EEPROM.h>

//pins:
#define HX711_dout D4
#define HX711_sck D5

//HX711 constructor:
HX711_ADC LoadCell(HX711_dout, HX711_sck);

const int calVal_eepromAdress = 0;
long t;

void setup() {
  Serial.begin(57600); delay(10);
  Serial.println();
  Serial.println("Starting...");

  LoadCell.begin();
  long stabilizingtime = 2000; // preciscion right after power-up can be improved by
adding a few seconds of stabilizing time
  boolean _tare = true; //set this to false if you don't want tare to be performed in the
next step
  LoadCell.start(stabilizingtime, _tare);
  if (LoadCell.getTareTimeoutFlag() || LoadCell.getSignalTimeoutFlag()) {
    Serial.println("Timeout, check MCU>HX711 wiring and pin designations");
    while (1);
  }
  else {
    LoadCell.setCalFactor(1.0); // user set calibration value (float), initial value 1.0
may be used for this sketch
    Serial.println("Startup is complete");
  }
  while (!LoadCell.update());
  calibrate(); //start calibration procedure
}

void loop() {
  static boolean newDataReady = 0;
  const int serialPrintInterval = 0; //increase value to slow down serial print activity

  // check for new data/start next conversion:
  if (LoadCell.update()) newDataReady = true;

  // get smoothed value from the dataset:
  if (newDataReady) {
    if (millis() > t + serialPrintInterval) {
      float i = LoadCell.getData();
      Serial.print("Load_cell output val: ");
      Serial.println(i);
      newDataReady = 0;
```

```
      t = millis();
    }
  }

  // receive command from serial terminal
  if (Serial.available() > 0) {
    float i;
    char inByte = Serial.read();
    if (inByte == 't') LoadCell.tareNoDelay(); //tare
    else if (inByte == 'r') calibrate(); //calibrate
    else if (inByte == 'c') changeSavedCalFactor(); //edit calibration value manually
  }

  // check if last tare operation is complete
  if (LoadCell.getTareStatus() == true) {
    Serial.println("Tare complete");
  }

}

void calibrate() {
  Serial.println("***");
  Serial.println("Start calibration:");
  Serial.println("Place the load cell an a level stable surface.");
  Serial.println("Remove any load applied to the load cell.");
  Serial.println("Send 't' from serial monitor to set the tare offset.");

  boolean _resume = false;
  while (_resume == false) {
    LoadCell.update();
    if (Serial.available() > 0) {
      if (Serial.available() > 0) {
        float i;
        char inByte = Serial.read();
        if (inByte == 't') LoadCell.tareNoDelay();
      }
    }
    if (LoadCell.getTareStatus() == true) {
      Serial.println("Tare complete");
      _resume = true;
    }
  }

  Serial.println("Now, place your known mass on the loadcell.");
  Serial.println("Then send the weight of this mass (i.e. 100.0) from serial monitor.");

  float known_mass = 0;
  _resume = false;
  while (_resume == false) {
    LoadCell.update();
    if (Serial.available() > 0) {
      known_mass = Serial.parseFloat();
      if (known_mass != 0) {
```

```
        Serial.print("Known mass is: ");
        Serial.println(known_mass);
        _resume = true;
      }
    }
  }

  LoadCell.refreshDataSet(); //refresh the dataset to be sure that the known mass is
measured correct
  float newCalibrationValue = LoadCell.getNewCalibration(known_mass); //get the new
calibration value

  Serial.print("New calibration value has been set to: ");
  Serial.print(newCalibrationValue);
  Serial.println(", use this as calibration value (calFactor) in your project sketch.");
  Serial.print("Save this value to EEPROM adress ");
  Serial.print(calVal_eepromAdress);
  Serial.println("? y/n");

  _resume = false;
  while (_resume == false) {
    if (Serial.available() > 0) {
      char inByte = Serial.read();
      if (inByte == 'y') {
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.begin(512);
#endif
        EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.commit();
#endif
        EEPROM.get(calVal_eepromAdress, newCalibrationValue);
        Serial.print("Value ");
        Serial.print(newCalibrationValue);
        Serial.print(" saved to EEPROM address: ");
        Serial.println(calVal_eepromAdress);
        _resume = true;

      }
      else if (inByte == 'n') {
        Serial.println("Value not saved to EEPROM");
        _resume = true;
      }
    }
  }

  Serial.println("End calibration");
  Serial.println("***");
  Serial.println("To re-calibrate, send 'r' from serial monitor.");
  Serial.println("For manual edit of the calibration value, send 'c' from serial
monitor.");
  Serial.println("***");
}
```

```
void changeSavedCalFactor() {
  float oldCalibrationValue = LoadCell.getCalFactor();
  boolean _resume = false;
  Serial.println("***");
  Serial.print("Current value is: ");
  Serial.println(oldCalibrationValue);
  Serial.println("Now, send the new value from serial monitor, i.e. 696.0");
  float newCalibrationValue;
  while (_resume == false) {
    if (Serial.available() > 0) {
      newCalibrationValue = Serial.parseFloat();
      if (newCalibrationValue != 0) {
        Serial.print("New calibration value is: ");
        Serial.println(newCalibrationValue);
        LoadCell.setCalFactor(newCalibrationValue);
        _resume = true;
      }
    }
  }
  _resume = false;
  Serial.print("Save this value to EEPROM adress ");
  Serial.print(calVal_eepromAdress);
  Serial.println("? y/n");
  while (_resume == false) {
    if (Serial.available() > 0) {
      char inByte = Serial.read();
      if (inByte == 'y') {
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.begin(512);
#endif
        EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.commit();
#endif
        EEPROM.get(calVal_eepromAdress, newCalibrationValue);
        Serial.print("Value ");
        Serial.print(newCalibrationValue);
        Serial.print(" saved to EEPROM address: ");
        Serial.println(calVal_eepromAdress);
        _resume = true;
      }
      else if (inByte == 'n') {
        Serial.println("Value not saved to EEPROM");
        _resume = true;
      }
    }
  }
  Serial.println("End change calibration value");
  Serial.println("***");
}
```

Code of Main Program

```
#include <ESP8266WiFi.h>
#include <HX711_ADC.h>
#include <EEPROM.h>
#include <ThingSpeak.h>

#define HX711_dout D4
#define HX711_sck D5
#define CHANNEL_API_KEY "S1XEBNHKL0LWB3FQ"
const char* ssid     = "@home_2.4G@unifi";//Replace with your Wifi Name
const char* password = "notyourwifi1998";// Replace with your wifi Password
unsigned long channel = 1095893;//channel number in Thingspeaks(without brackets)

HX711_ADC scale(HX711_dout, HX711_sck);
WiFiClient client;//Creates a client that can connect to a specified internet IP address
and port as defined in client.connect().
//Change this calibration factor as per your load cell once it is found you many need to
vary it in thousands
//float calibration_factor = 369.69;
float calibration_factor;
const int calVal_eepromAdress = 0;
long t;
long stabletime = 2000;
float current_water2, drank_water, total_water;
float current_water1 = 0;
static uint32_t prevTime;
uint32_t curTime = millis();
boolean _tare = true;

void setup() {
  Serial.begin(57600); delay(10);
  scale.begin();
#if defined(ESP8266)|| defined(ESP32)
  EEPROM.begin(512); // uncomment this if you use ESP8266/ESP32 and want to fetch the
calibration value from eeprom
#endif
  EEPROM.get(calVal_eepromAdress, calibration_factor); // uncomment this if you want to
fetch the calibration value from eeprom
  scale.start(stabletime, _tare);
  scale.setCalFactor(calibration_factor); //Calibration Factor obtained from first
sketch, start scale
  WiFi.begin(ssid, password);
  ThingSpeak.begin(client);
  scale.tare();//Reset the scale to 0
}

void loop() {
  float recorded_water;
  static boolean newDataReady = 0;
```

```
  const int serialPrintInterval = 1000; //increase value to slow down serial print
activity

  if (scale.update()) newDataReady = true;

  // get smoothed value from the dataset:
  if (newDataReady) {
    if (millis() > t + serialPrintInterval && (scale.getData() >= 5)) {
      current_water2 = scale.getData();
      Serial.print("Water in Bottle: ");
      Serial.print(current_water2); //Up to 2 decimal points
      Serial.println(" ml"); //Change this to kg and re-adjust the calibration factor if
follow ml
      newDataReady = 0;
      t = millis();
    }

    drank_water = cal_water(current_water1, current_water2);
    current_water1 = current_water2;
    total_water += drank_water;
    Serial.println(drank_water);
    delay(500);
    Serial.println(total_water);
    if ( curTime - prevTime >= 24 * 60 * 60 * 1000UL ) //This if-statement resets the
timer and variable of total water drank to 0
    {
      prevTime = curTime;
      total_water = 0;
    }
    ThingSpeak.setField(1, drank_water);
    ThingSpeak.setField(2, current_water2);
    ThingSpeak.setField(3, total_water);

    ThingSpeak.writeFields(channel, CHANNEL_API_KEY);
    delay(15000);
  }
}

float cal_water(float current_water1, float current_water2) {
  float drank_water;
  drank_water = current_water1 - current_water2;
  if ( ((current_water1 - current_water2) <= 10) || (current_water2 > current_water1) ) {
    drank_water = 0;
  }
  return drank_water;
}
```