Andrew Sturdy – 100318044

Cameron Shipman - 100322315

Group Number - B05

Agreed Statement

Andrew wrote all the code that handled the data gathered, both real time data handling and graphing the data. Andrew wrote the Aims, Data collection and Data processing sections of the report. Andrew recorded the Problem Statement and Data Processing sections of the video. Andrew did 50% of the work.

Cameron wrote all the code that gathered the data from our sensors and the code that turned on GPIO pins. Cameron assembled the sensing circuit and gathered our data we used as example data for graphing. Cameron wrote the sections on system output, sensing and the conclusion of the report. Cameron recorded the video demo of the project working and the feedback and output section of the video. Cameron did 50% of the work.

Aims, Motivation and Background

This project aims to make a system that reminds the user to drink at regular intervals so that they meet the national recommended intake. It will measure the fluid level of a container and after a certain amount of time has passed it will issue a notification to your phone telling you to drink, it will also send a notification when the container is almost empty. The project will change the timing between warnings based on the average amount you drink when told so that by the end of the day the average recommended intake is met.

This project aims to solve the problem of dehydration when working, it is targeted at workers who spend most of their time at a desk. As people tend to forget to drink when they are focused it is likely that they suffer the effects of dehydration throughout the day, to prevent this we propose a coaster that tracks fluid level and reminds its user to drink.

There are many solutions that already exist to solve this problem, some are simple water bottles with times recorded down the side, showing the user how much they should have drunk by a certain time, while others are apps that allow you to record when you last drank and how much you drank. The issue with both is that the user requires to be aware of the problem, the water bottles need the user to look at them regularly, which won't happen when they are focused on something else, and the app requires you to track your drinking and enter the data manually.

An example of a solution that already exists is Hidrate Spark[1], they produce portable water bottles that contain scales at the base to measure fluid level. The bottle uses Bluetooth to connect to your phone, this allows you to track your consumption across a week and set yourself goals. This product does everything ours aims to do, without the downside of not being portable.

Sensing

In order to know when the user is drinking and/or not drinking, we chose the best way to do this would be to take measurements of a bottle's fluid level over a period of time. There are several sensors for measuring the fluid level of a container, some of which involve: floats, conductivity, capacitance, ultrasound, electro-optics and strain. Many of these are in widespread use for many applications, however, the majority of them are expensive, not available off the shelf (for the small format of a water bottle), bulky or unsafe for use with beverages. A load cell was selected as the most viable solution for our project due to its non-invasive, inexpensive qualities, and ease of implementation. The load cell is used to weigh the bottle and its contents which allows our system to infer its current fluid level. The drawback of using a load cell is calibration needed to keep the system accurate. *'Variation in container weight or fluid density, or even the position of the container on the load cell platform may alter what the system "sees"'* *(Gems Sensors & Controls).* This issue is manageable as the accuracy of the load cell does not wander too far to invalidate the collection of data.

The system is comprised of a load cell – mounted to a suitable platform so that a bottle can be set upon it, – a HX711 amplifier, an LED and a Raspberry Pi. An Arduino could also be used for these components, but the RPi was chosen due to availability of resources. The system is essentially a high-precision weighing scales with computing functionality.

The system is connected so that the load cell output is sent through the HX711 breakout board, which converts the analogue input to a digital output for the Raspberry Pi to read via its GPIO. The LED is also connected to the RPi via GPIO. The raw data output received from the HX711 is converted to grams on the Raspberry Pi using an available Python HX711 library *(Zak, 2017).*

As there is only one sensor used for the entire project, the information provided by the load is invaluable as it is fundamental to the system's functionality.

Data Collection

The data collected in this project was real world data that was gathered through continuous use throughout the day for a week. The data gathering required constant use of the device so the user was not allowed to drink out of any container that was not placed on the device, this was so we could collected as much data as possible, however this does not entirely mimic real world use as most users will get a drink when they need to, even if it is not the drink that is placed on the device.

As the device needed to be used constantly, the user needed to drink from it whenever they felt the need to, ignoring the alerts when necessary. They also needed to fill the container up when it was nearly empty instead of getting a different drink. The container used needed to stay consistent, so the weight was calibrated correctly, because of this requirement, the data gathered will be different to normal use data.

The data set this produces shows the drinking habits of the user across a week, it shows the amount they drink and when they drink throughout the day. This data would be very different in real world use due to the user not drinking solely from the container on the device throughout the day, this would cause there to be chunks of the data where the fluid level didn't change. The user is also likely to change the container they drink out of throughout the day, which would mean the load cell would need to be recalibrated.

Data Processing

The data gathered is used in 3 ways, it is used to alert the user when they need to drink and when the container is almost empty, to change the time between warnings so they drink the national average recommended within the day, and it is used to graph their drinking habits so the user can see how much they drink across a day.

The device warns the user when the weight of the container reaches a certain point, so they know they need to fill it up. Users are also alerted after a certain amount of time has passed since they last had a drink, this alert happens on an hourly basis by default as the NHS recommends 1.2L of water to be drank a day and the average mouthful of water is 30ml meaning you would need to drink every 40 minutes. The timing between alerts is then updated once every 2 hours based on how much you drink each time and how much you have drunk in total, the timing is adjusted so that you meet the 1.2L recommended.

The data can also be graphed in the form on a line chart that plots fluid level of the container over time. This would allow the user to see when they fill up their container, when they drink and how much they drink at a time. This visual representation is meant to allow the user to better plan their refills so that it does not disrupt daily activities like meetings.

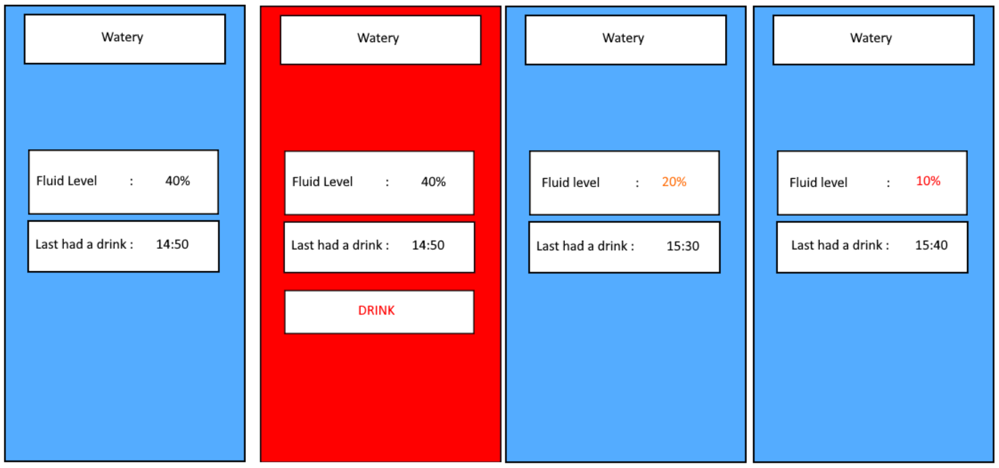
A HX711 board is used to amplify the output of the load cell allowing it to be read by our device, this can cause some noise to occur within the weight measurement. To handle this noise all of the weight calculations and logic operate within certain boundaries so that small inaccuracies do not affect the program.

The data gathered is in the form of "[Hour of the day, Minute of the day], fluid level", this data needs to be converted into change in fluid level and a standard unit of time like minutes. To convert the fluid level, we record the previous fluid level every loop and perform a subtraction each time to work out the change. We convert the time to minutes so that it is easier to work with as hours require more logic to work with than minutes, but it does make the graphed data harder to read.

As the data is processed in real time no accuracy checks can be performed as it would slow down the rate at which data is handled also as the data has no way of being validated without the user manually checking how much fluid is in the container. In a controlled environment, these variables could be tested, however the data cannot be validated as it is produced.

System output and feedback

When the system alerts the user, it does so via an app. We chose to use an app as most people will always have one on them and it does not cause any additional sounds. If a buzzer/LED was to be used, the device would become a lot harder for some people to use as it could disrupt activities like meetings or calls. If a more subtle approach was used, then the chances of the user not noticing would go up.


Watery

Fluid Level : 40%

Last had a drink : 14:50

Watery

Fluid Level : 40%

Last had a drink : 14:50

DRINK

Watery

Fluid level : 20%

Last had a drink : 15:30

Watery

Fluid level : 10%

Last had a drink : 15:40

The app's background colour changes based on if they need to drink or not, if the user has been alerted to drink it becomes red, otherwise it stays light blue. This colour change was used to make it obvious to the user that an alert has been issued. The app shows the fluid level of the container and when the user last had a drink. If the fluid in the container goes to 20% or below, the font colour changes to orange and when its below 10% it changes to red. This was done so that the user would notice that they need to refill the container, however we don't think this needs to be as big a change as the alert causes, this is because the user is likely to notice that the container needs to be filled when they drink from it.

The downside to using an app is that people who don't have their phone out/are not allowed to won't be able to use the device. The user is also not likely to notice if they have their phone on silent as they won't get the notification sound.

The device also outputs a line graph, this is stored as a png file for the users to access at any time. We chose to store it as a png file instead of putting it in the app because it would take up a lot of screen space, making the alerts harder to notice. A line graph was used as we are displaying continuous data. This graph can be used to show spikes in consumption, which can inform the user of when they need a drink at hand in the future.

The downside of storing the graph instead of displaying it on the app is that some users won't be able to find the graph and some people do not have a lot of space available on their phones to store these graphs.

Conclusion

The system produced is a coaster like object that uses a load cell to find the weight of a fluid container. The weight is used to alert the user when it needs refilling and to keep track of how much they are drinking. The advantage of this system is that it can be used for any container such as a bottle or mug, as long as it is calibrated for it. A downside to this method is that the load cell requires calibration every time the device is turned on, which makes it hard to use the device repeatedly. As the device is not portable and relies on being the only source of hydration it is not suitable for most people, as most people move around during the day and just get a drink when they need it from whatever source is closest.

Overall, the project met the aims of alerting the user when to drink and when the container needs refilling, however due to the sensor used its audience became more focused than originally intended as it is only applicable for people who spend all day at a desk. Although the audience shifted from what was originally intended, the project meets the aims set out for that select group as they do not move around a lot making the detriments of the project not as prevalent.

References

[1] - https://hidratespark.com/ Hidrate Spark