CMP6046B Ubiquitous Computing: Final Project

James Burrell (100263300), Vaileigh Ong (100318787)

Contents

1 Agreed Contributions

James Burrell - 55%

- System hardware production and setup.
- Arduino sketches.
- Data interpretation, processing and formatting via Java program.
- Android application functionality.
- Report sections 2, 3, 4 and 7.
- Demonstration video.

Vaileigh Ong - 45%

- Firebase Database setup, structure and design.
- IDE setup for android application.
- Implementation on receiving, and output data from Firebase.
- Report sections 5, 6, and 7.

2 Aims, Motivation and Background

"This section should describe the real world problem the system aims to overcome."

The overall aim of our project is to provide a proof of concept system that can be used to show how a real-world problem can be solved through Ubiquitous Computing. Whilst the final system may not be a fully featured, ready for market product, it still aims to provide an accurate representation of what sort of components and processes are needed if the problem is to be solved effectively.

The problem we have decided to address is centered around the equestrian industry. In order for the problem to be explained, some background information is needed. Good care of horses, whatever breed, requires them to be turned-out for long periods when not in work. Turnout is the term given to the process of allowing a horse time outside of its stable to exercise, graze and socialize in a dedicated field or pasture. Turnout on a regular basis is incredibly important as it allows them to act as closely as possible to how they would in the wild, benefiting both their physical and mental well-being.

Figure 1: Horses during turnout.

Good care of horses during turnout means giving them access to fresh, uncontaminated water amongst other things. In addition, monitoring their intake of water is often undertaken by stable owners as it can be incredibly useful in assessing if a horse is unwell.

However, fields used for horse turnout can range in size from less than one acre to well over a hundred acres in some cases. In addition, turnout fields are often divided into sub-plots to allow for rotational grazing, in which a portion of the fields are "rested" to allow for its grass to grow back, whilst the remaining are used for turnout. Larger sized turnout areas are often better for the horse, but provide more logistical difficulties. One of these logistical difficulties is the supply of fresh water to turnout fields.

Stables with a large number of fields/turnout areas will require fresh water in each one. Over long distances, water pipes and automatic fillers are expensive, and once put in place cannot be moved to accommodate rotational grazing. Natural water sources such as rivers or streams are rare, unmoving, and some are not suitable for drinking from. For these reasons, many stable owners find the easiest way to supply fresh water is through refillable troughs, which are inexpensive, movable and easier to maintain and replace.

Figure 2: Top; standard trough, left; natural water way, right; automatic drinker.

However, this requires staff to identify each trough location, manually visit each one and see if it needs filling, and then transport the water to it if needed. This can be an unnecessarily long, inefficient process which has to be carried out daily. This is the problem we will be looking to address. Our proposed system aims to remove these inefficiencies by providing users on-demand access to data on the location of water troughs, the amount of water they contain, how much has been drunk over a given time period and how much water is required to fill up the troughs.

Our problem has been suggested from real-world experiences working in the equestrian industry. James has over 2 years of experience working at a livery yard and riding school, and has frequently encountered issues with water supply during horse turnout. This experience allows us to justify that our proposed system addresses a real world problem and the system's prototype could be adapted for real-world deployment.

3 Sensing

"This section should describe the sensors, the reason for these choices and the general platform of the system (e.g. Ras Pi, Arduino or Phone) and sensor system structure. The section should describe what information gain/value each sensor gives in relation to the overall aim of the project and their strengths and weaknesses."

In order for our system to meet its requirements, data must be collected on water volume and its change over time. The simplest way to get an accurate measurement of water volume is to measure its mass. The density of water varies little whilst still in liquid state allowing for it to be approximated at $1g/cm^3$ or 1g/ml. This means that for every 1g that is measured, 1ml of water is present.

However, ice is less dense, at approximately 0.9g/ml. Therefore when the system is operating at temperatures in which ice can form in water troughs, the difference in densities would cause slight inaccuracies. Nonetheless this inaccuracy is unlikely to significantly affect the function of the system, as the volume of ice formed in water troughs is small at temperatures in which horses should still be safely turned out in, however this is still a downside of the system and so should be documented.

A load cell will be used to measure the mass of the water and its containing trough as it's accurate, readily available from online retailers, and hard-wearing, which will be useful when deployed outdoors. Load cells are also available in a range of weight capacities, allowing one to be selected that will be appropriate for the proof of concept system. The load cell we have selected has a weight capacity of 10kg. An HX711 digital-to-analogue module is required to convert the signal from the load cell.

For location data to be collected, we have chosen a Flora GPS module from Adafruit. It's ability to give location data to up to 3 metres of accuracy and at a rate of up to 1Hz makes it ideal for use in our proof of concept system. In addition, the module is fully compatible with the microcontroller we have selected for the main platform of our system.

Below shows an image of the main physical components our our system.

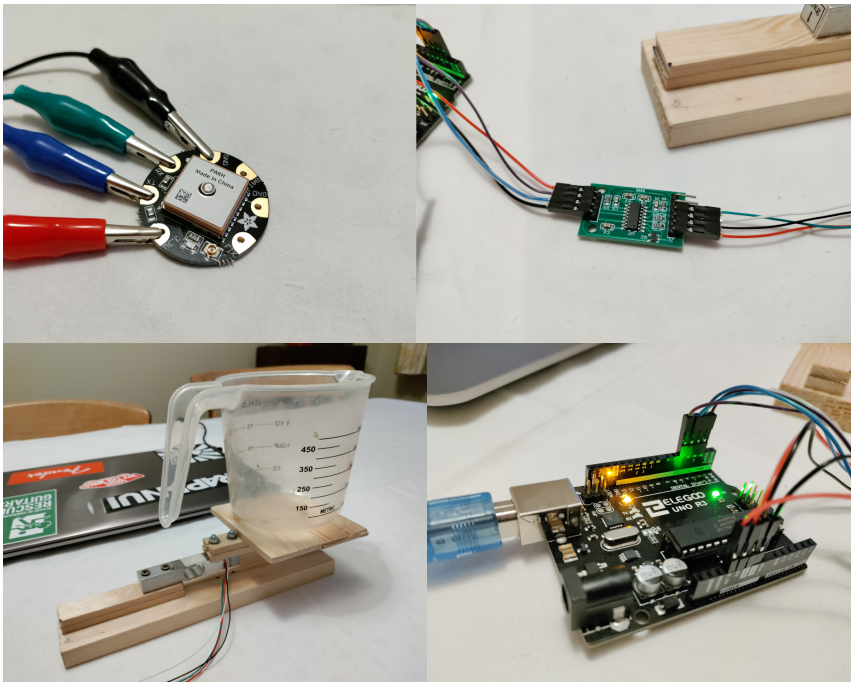

Figure 3: Top left; Adafruit GPS module, top right; HX711 DTA module, bottom left; load cell in mounting, bottom right; Arduino Uno microcontroller.

The main platforms of our system is split between an Arduino Uno microcontroller, a Java program for data processing and interpretation and a mobile application for Android. An Arduino compatible microcontroller by Elegoo has been chosen as it provides the minimum amount of functionality to meet the requirements for the prototype, and was easily sourced. The microcontroller will be used to run the Arduino sketch that takes an input from the load cell, calibrates it, and reads weighing data. It is also is used to run the sketch that receives GPS data via the GPS module.

The data is then passed to a Java program for data processing and interpretation, which then sends the processed data to a real time database via Google's Firebase platform. An Android application will then read data from the real time database and display it in an interface for the user. The application will allow the user to view information such as; the troughs ID number, it's maximum capacity, it's current volume, the volume of water required to fill the trough, and finally the amount of water drunk and the amount of water spilled within the last 5 minutes.

A lofi diagram of the entire concept system and its data flow is shown in Figure 4 and the physical system shown in Figure 5.

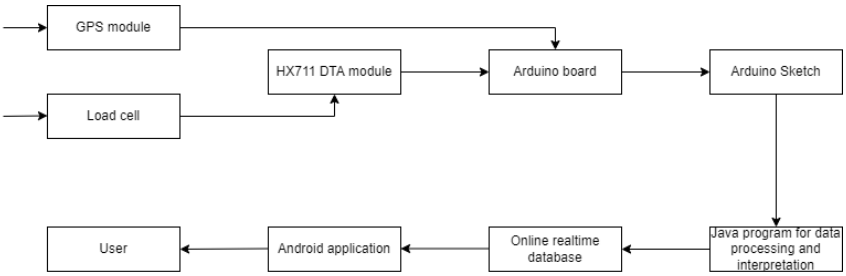

Figure 4: System concept diagram.

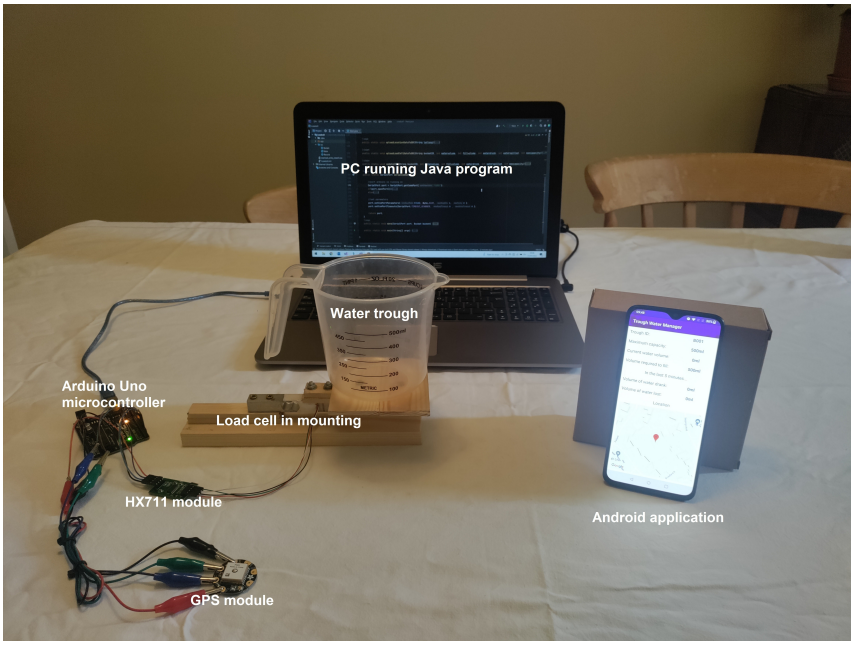

Figure 5: Implemented physical system.

4 Data Collection

"This section should describe how data has been collected, if it is from a real-world scenario, mock scenario or an approximation. There should be justification of the collection method and protocols used with strengths and weaknesses discussed. This section should also give a summary description of the final dataset collected."

Horses, like any animal, do not drink at regular, specific points during the day. Because of this, our problem means that a state, in this case the volume of water in a trough and its location, has to be continually measured. Therefore, the development of our system does not involve the sequence of collecting a large amount of data in one go, then processing the entire data set and then finally displaying it.

For this reason, in order to develop a solution to the suggested problem, data must be collected, processed, and displayed in real time. Because of the physical scale of the system that would be required in reality, we have instead created a mock scenario that allows us to collect a recreation of the data that mimics the true scenario but on a scale that is appropriate to producing a proof-of-concept system. This is illustrated below.

With mock scenario setup, we can use it in order to verify the data being collected by the load cell and GPS module are correct. In addition data processing, storage in the database, and display via the Android application can be verified too. More detailed stages to testing and

results are shown in section 6.3 of this report.

This setup is advantageous as it allows the real scenario from our problem be mimicked on a smaller scale, allowing us to develop the system to prove its functionality without having to produce a system that could be actually used in the true scenario.

However, the mock scenario we have created does not simulate all the factors that may be present in the true scenario. For instance, weather conditions that affect the reading from the load cell, such as changes in temperature, wind speed and rainfall, cannot be accurately emulated at this scale, and so does not allow for testing our proof of concept system fully.

5 Data Processing/Interpretation

"This section should describe how the raw collected data has been processed in terms of data cleaning, feature engineering and use of modelling (e.g. machine learning), labelling and domain knowledge where appropriate. It should also describe how it has been evaluated and the output of the processing tested for reliability and accuracy."

The data collected from Arduino were processed has two types: data from the load cell and data from the Global Positioning System (GPS) module. Data from load cell returns the current weight as the data to allow the project to observe the volume of water in the trough. The weighing process may be effected by multiple factors. One of the concerns of this project previously mentioned is the impact of weather conditions such as wind speed and rainfall, with the weight of the water in the trough potentially being effected by this. The GPS module provides data in National Marine Electronics Association (NMEA) format. It will provide sufficient data to track the location of where the trough is situated. The following report discussed the methods to normalise the data from the load cell.

Every weight data received from the load cell will undergo a preliminary cleaning process. The cleaning process aims to normalise the data from the load cell. In the Arduino sketch, multiple readings from the load cell are collected over a period of time. The weighing data is normalised by calculating the average value from the weight readings collected. The average value from the load cell is then outputted at a rate specified in the Arduino sketch. GPS data is passed directly to the Java program.

The Java program is used to read the data from the Arduino programs, process it and finally format the data and upload it to a cloud database. When there is no weight on the load cell, fluctuations can cause weight data to read as small negative values due to the condition of the environment. Therefore the weight data's absolute value is taken and rounded to the nearest whole number in order to be more consistent and therefore more accurate when processed. The GPS data is received in NMEA format, which provides multiple bits of information in one sentence. The Java program will extract the information on latitude and longitude by splitting the sentence at the appropriate points.

The Java program has declared array lists to store every record of a drink or spill within a 5 minute duration, and the with the array lists being will be cleared of any records older than 5 minutes after each new reading. More details of this are explained in the demo video.

Firebase from Google is the is the real time cloud database we have used in this project. It stores the data in the cloud to ensure that the front end of this project can retrieve the data when the data has been updated. The Real-Time database is one of the most used products available from Firebase, and it also supports offline and online network states (Yahiaoui, 2017).

The Java program uses the Curl networking tool to send HTTP requests from the Java program to the Firebase Real-Time Database with the updated trough data. Curl is a command-line tool that uses Uniform Resource Locator (URL) to request and update data (Kalin, 2013). Therefore, the Java program allows performing any action through the server to Firebase Real-

Time Database. It updates the data received from the Arduino program to Firebase Real-Time Database.

The cloud database has the unique number of each water trough, the volume of water drank, the volume required to fill, the latitude and longitude of the water trough, maximum water capacity, the volume of water split and current water volume. The Java program provides the data for all fields and updates it within a specific duration. For the mock scenario, the maximum capacity of the water trough is set as 500 millilitres.

The latitude and longitude data is stored as a string separated by a comma, and the GPS data is extracted in the Java program. The array lists declared in the Java program is used to compute the information for the database. The current water volume used the value of weighing data. The total volume drunk and spilt are calculated from the entries in the the array lists. The Java program considers any decrease in water volume of less than 5% as a spill, and any decrease greater than 5% a drink.

The front-end of this program provides an interface via an android application. It will show the water trough ID information, maximum capacity of the water trough, current water volume, the volume required to fill the trough, volume of water drank, volume of water lost, and a location marker on a interactive map. This project uses the android studio as IDE to develop the application.

The android application has implemented Firebase as a dependency of the application. It receives the data from Firebase Real-Time Database to display the information. Every time the application views a change in the database, the application will display the new data if the device has an internet connection.

Source code for Arduino sketches, Java program and Android application are available in the ZIP file alongside the PDF of this report.

6 System Output and Feedback

"This section should describe the choice and design of all system output that are feedback to the user(s) and any lo-fi diagrams of images. This section should give reasoning for these choices and discuss the strengths and weaknesses. If any user testing was done, this should be discussed here."

This project has developed with a combination of hardware and android application. The hardware will receive the data, and the android application will show the data on the front end. The hardware output provides the weight of the water trough so that the program can update the data in the database. The android application displays the information by receiving data from the database. The following report will interpret all of the outputs from the project. In other words, the android application is for the user to monitor the status of the water trough. The seven key information points that are shown on the android application are the following:

1. Trough ID. The Trough ID is the unique identifier for the water trough. This value is special to every water trough. A unique identifier makes the user more manageable on all the water troughs, and it can bypass unneeded confusion between water troughs. In addition, it is easier to add or modify details on a specific water trough.

2. Maximum capacity. This is the maximum capacity of the water trough. This number can let the user know the capacity of the water trough. The size of every water trough might be different, so this can be set for each one.

3. Current water volume. The current water volume in the water trough. This is made for the users to observe when required to fill the water trough. It will show how much volume

of water current has without checking it in person. The user can know when to refill the water trough from the android application.

4. Volume required to fill. This explains the water volume that needed to be filled to full. The users can bring enough water to fill the water trough. The user can bring exactly or sufficient amount of water to refill the water trough.

5. Volume of water drank. This shows the volume of water drank by horses in the past 5 minutes, but the duration can be varied to anything. This can be used to observe drinking habits of the horses, which is useful in monitoring their health.

6. Volume of water lost. The volume of water lost by other factors. The factors can be water being spilled or through evaporation.

7. Google Map. The Google map shows the location of the water trough on an interactive map. The user can spot the location of the water trough to refill the water.

6.1 Testing

The testing is to ensure all functionality of the project system is working correctly. The following method has been used and the results are displayed in Table 1.

6.1.1 Testing the Load Cell

1. A container of water of known mass is placed on the scale. The load cell is then calibrated using the known weight. This represents a water trough in a turnout field.

2. We can then verify that the correct weight is detected and displayed.

3. To simulate a horse drinking from the trough, an amount of water is taken out of the container. In this case, we use a syringe so we know the true amount of water that has been removed.

4. We can then verify that the correct amount of water has been 'drunk' and displayed in the application.

5. We can then repeat this process for what would be considered a spill.

6. The same can then be done for simulating filling the trough, with a known volume of water added and the value then displayed on the application should be accurate.

6.1.2 Testing the GPS Module

1. A simple test can be used to give a somewhat accurate verification. The system is first placed in a room on one side of the building it's in. The location on the map in the application is checked.

2. The system is then moved to a room on the opposite side of the building, and the location on the map in the application is checked.

6.2 Test Cases and Results

No.	Testing Case	Expected Outcome	Actual Outcome	Pass/Fail
1	Accuracy of data collection	The hardware system returned the correct value of the water.	The hardware system returned 50.00 for 50 milliliters of water	Pass
2	Data sending to Cloud real time database	The hardware system stored all the information in the right field and with the correct values	The project system has stored 'bucketid' with B001, 'drunkvolume' with 0, 'fillvolume' with 498, 'location' as 52 39.1103,001 22.0994, 'maxvolume' as 500, 'spillvolume' as 4 and 'watervolume' as 2 into the database.	Pass
3	Receiving data	The android system received the correct data from database.	The android system has displayed 'Trough ID' with B001, 'Volume of water drank' with 0, 'Volume required to fill' with 498, 'Maximum capacity' as 500, 'Volume of water lost' as 4, 'Current water volume' as 2 and showing correct location on maps.	Pass
4	Updating data	The hardware system updated all information within specific duration.	The project system updated the data every 3 minutes if got changes.	Pass
5	Data synchronisation	The android system is synchronising the data from database. It changed the data immediately when the data is modified on the database.	The android system changed and displayed all information when the project system has updated the data in database.	Pass

Table 1: Test cases and results.

7 Conclusion

"This section should give a concluding overview of the whole system. It should summarize the systems merits and shortcomings in relation to any testing/evaluation that was done in each of the sections as well as in relation to the original aims and the real-world problem to be solved."

7.1 System Merits

The project provides an automatic approach compared to traditional water supply during turnout. This project is suitable for use on any water trough that needs to be filled, and it is suitable to place the water trough both indoors and outdoors. Instead of walking around the turnout area to check if the water needs to be filled, this project provides an insight into all information needed on the android application.

This project has improved the efficiency of filling the water and reduced the time to inspect the

condition of each water trough. Moreover, the project provided information on how much water is needed to be filled for each trough. It reduces the human resources requires to get the right amount of water rather for filling a trough instead of bringing extra or less water. One of the highlights is the Google Maps features on the android application. The Google Maps features provide the exact location of the water trough to an accuracy of 3 metres, so time is not wasted searching for it in a turnout area.

7.2 Limitations and Potential Improvements

A limitations of our system is that monitoring the water trough requires a network and GPS signal. The connection between the hardware and the android application is a cloud database. The android application and hardware are needed to access the internet to receive and update the data. In summary the main downside of the project was that the hardware system was not wireless. The Arduino must be plugged into a computer for data to be transferred from the Arduino to the Java program, and also to act as its power supply.

The first improvement that should be made to the system is to to address this issue. This would require replacing the physical connection from the Arduino to the PC with a wireless connection. This could be done by implementing a long range communication module to the system, such as the HC-12 module pictured below. This would allow for weight and GPS data to be sent to the PC at a maximum range of 1000 metres. The Arduino could then be powered by an external battery. This would allow for our proof-of-concept system to show that it could be implemented at a trough at any point in a turnout field, which would allow users to track information about troughs location and contents fully remotely.

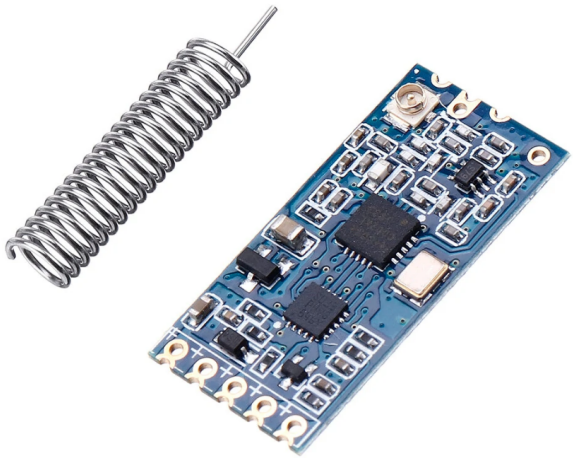

Figure 6: HC12 long distance communication module for Arduino microcontrollers.

7.3 Summary

In summary, we believe we have produced a system that effectively addresses a potential solution to the problem posed in Section 2. The system accurately provides a proof of concept system that could be scaled up and modified for implementation in the real world.

Whilst the system that we have designed is not full proof to cover for every eventuality that it could be posed with in reality, we believe it delivers an effective solution given the time and resource constraints we are posed with during its development.

References

Kalin, M. (2013). *"Java Web Services : Up and Running"*. Accessed 01/05/2022.

Yahiaoui, H. (2017). *"Firebase Cookbook"*. Packt Publishing. Accessed 01/05/2022.