Sensing & Internet of Things - Coursework

Alberto Montemiglio

December 24, 2021

Abstract

What follows is a project aimed at tackling the issue of dehydration in people affected with dementia, one of the leading causes of acute confusional states in dementia patients [1]. What follows is the proposal of a novel, non-intrusive method to detect water intake by means of a throat microphone and a Machine Learning algorithm. Seven days of water intake data were analysed and compared to weather data to check for correlations. Then, a website with user account and profiles was built to present the raw data, the data analysis, as well as a progress bar of how much water should be drank in a day to the patient and the carer. The water intake is tailored to the age and weight of the patient using the data gathered in the login phase. Finally, as a reminder to drink more if the current water intake is not sufficient, an actuation platform consisting of a peristaltic water pump pours water in a glass.

Results show that the discrepancy between the predicted and the actual water intake is always within 30%, which can be further reduced by calibration.

This experiment proves how the use of a throat microphone is a viable and reliable method to assess a patient's water intake.

1 Introduction and Objectives

Problem Statement Dehydration in patients affected by dementia is one of the leading causes of acute confusional states, as concluded by Seymour et al. [1].

This issue can be tackled by monitoring the the hydration levels of patients or by monitoring their water intake.

However, at the present day, there is no simple, easy and inexpensive way of monitoring the hydration level in a patient [3].

As for monitoring the water intake, this would have to rely on external carers manually monitoring it. This method is not only very hard to implement in practice, but does not address the 51% of dementia people living alone, as estimated by Eichler et al. [4].

Figure 1: A commercial throat microphone. From [2]

Proposed Solution The proposed solution is a novel, non-intrusive method to detect the swallowing of water through the use of a throat microphone (Fig.1) coupled with a Machine Learning (ML) algorithm that discerns water swallows.

Objectives The objectives of the following study are trifold. Firstly, we want to investigate the effectiveness of throat microphones in detecting water intake. Secondly, we want to investigate possible correlations between water intake and atmospheric weather. Finally, we want to create an end-to-end system for monitoring and promoting correct hydration levels, aimed both at people affected by dementia living alone and carers of dementia patients.

Time Management In order to better plan the project and to deliver it in a timely manner, the Gantt chard shown in Fig.2 was compiled. A lot of time was left at the beginning of the project in order to properly scope and define the project, and to make sure that what was set to be achieved was actually achievable, time was dedicated at the beginning to learn the basics of what is shown in this report. In later stages, the knowledge acquired was deepened. The project was delivered following the schedule without major delays. The biggest problems in sticking to the schedule were faced in the first month, when the unknown unknowns were high and decisions had to be made based on little knowledge.

2 Data Sources and Sensing Set-Up

2.1 Summary of the System

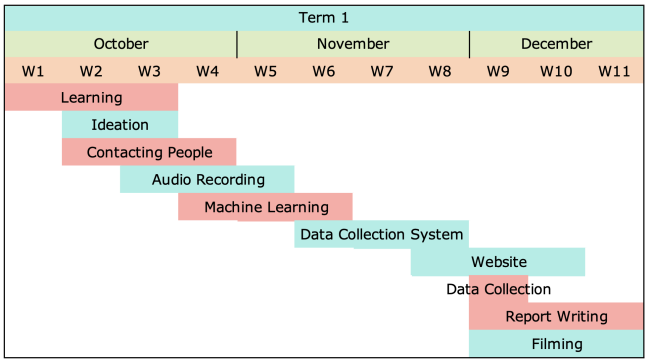


Figure 2: Project Gantt Chart

The proposed system (Fig.3) is composed of three parts: A water intake detection system, a website to show the data and monitor the water intake and a water delivery system.

Water Intake Detection System The system that detects water intake uses a throat microphone to collect the sounds of the larynx and a ML algorithm to detect water swallows. Kim et al. [5] measured one water swallow to be, on average, about 50 ml for men and 40 ml for women. Therefore, a good estimate of the amount of water drank from an individual can be extracted from their number of swallows.

The water swallow data is then uploaded to an online database for visualisation and analysis purposes.

Data Visualisation and Analysis Website A custom-made website is used to collect the data from the online database and present it in a manner that is useful both to the patient and to the eventual carer. The data is analysed and inferences are made in order to support healthy hydration levels. The website also includes a progress bar to present in a visual way the progress of water intake for the day.

Water Delivery System Following interviews with two dementia patients and three carers conducted in November, it was understood that some patients would benefit from a physical reminder to drink. To this end, we developed a water delivery system that pours water when the daily water intake up to that point of the day is below what should have been drank. To achieve this, a Raspberry Pi 3 Model B+ [6] downloads the water swallow activity from the online database and uses a peristaltic water pump to pour water from a water reservoir into a glass.

2.2 Water Intake Detection System Set-Up

The body-worn system that detects swallows is composed of a Raspberry Pi 3 Model B+ [6] for computation, a lavalier microphone [7] taped to the user's neck (more comfortable solution later), a USB-to-aux adapter to connect the microphone to the Raspberry Pi and an Anker 10,000 mAh battery pack [8] to power the Raspberry Pi (Fig.6). The Raspberry Pi continuously records and analyses the audio signal from the microphone to detect swallows. Once a swallow is detected, weather data is gathered from APIs and the information is stored in an offline database that acts as a buffer. If a connection is available, swallow events are uploaded to an online database (Fig.3).

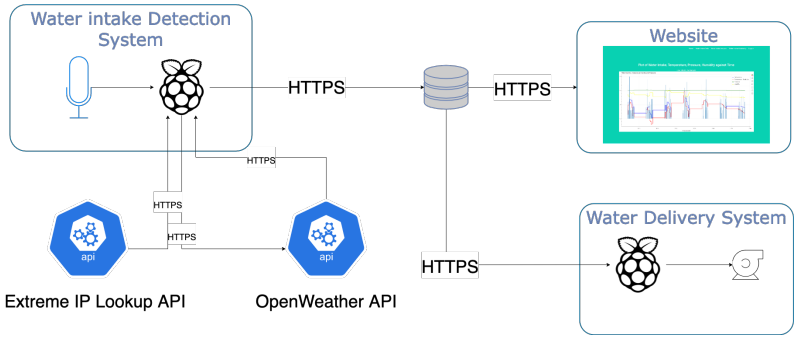


Figure 3: The system Diagram

2.3 Throat Microphone

A throat microphone (Fig.1) is a type of contact microphone that absorbs vibrations directly from the wearer's throat by way of a sensor contacting the front of the neck. Throat microphones can pick up speech even in extremely loud environments, and are therefore widely used in the military field.

We initially bought a £15 throat microphone [9], but it turned out to be inadequate for the purpose, as it was not picking up throat sounds. Following Santoso et al. [10], who used a military grade, £100 throat microphone (the *IASUS NT3 Black Ops 2* [11]), we concluded that a microphone of good quality is necessary for the system to work. As those were beyond the budget, we experimented

with the in-built microphone of Apple's wired *EarPods*, which yielded very good results. We then settled on a lavalier microphone from Amazon [7] (Fig.4), which gave the best results amongst the tested microphones, despite being the cheapest (£13).

The microphone still picked up background noise, and we hypothesize that the use of a good throat microphone would yield better results in this field, as well as being more comfortable.

The lavalier microphone was taped to the front of the neck using a piece of medical tape. Again, the use of a throat microphone would result in a much more comfortable fit.

2.4 APIs

Openweather Historical Weather API In order to explore the impact weather has on hydration, and following the coursework's requirements, the Openweather Historical Weather API [12] was used to gather weather data. The use of historical data, as will be seen in more detail later, is to accommodate for the possible lack of internet connection of the Raspberry Pi to the internet when in need to upload data to the online database. With this configuration, the Raspberry Pi is able to run for weeks without internet connection, and upload when connection is available.

Extreme IP Lookup API Because, of course, weather conditions are dependent on location, the location of the Raspberry Pi is obtained through to the Extreme IP Lookup API [13], which determines the location using the IP address. The longitude and latitude data is then input in the API call for weather conditions.

Figure 4: A lavalier microphone

2.5 Water Delivery System

The water delivery system (Fig 5) consists of a 12V power supply, a 12V peristaltic pump, a Raspberry Pi, a relay to control the pump with the Raspberry Pi, a water reservoir and a small limit switch.

The Raspberry gets from MongoDB the number of swallows since midnight, and if this number is lower than what should have been drank up to that time of the day, it activates the peristaltic pump through the relay to fill the glass with water. To detect whether the glass has been emptied, a limit switch changes the state of a binary variable when the glass is picked up and re-positioned.

The peristaltic pump has been selected because it allows for the moving of water without making the water contact anything but the tube it flows in.

3 Data Collection and Storage Process

The data was collected over a period of seven days, from December 1st to December 7th. The test subject was a 22 years old male. The data was collected for nine hours per day, from 9am to 6pm.

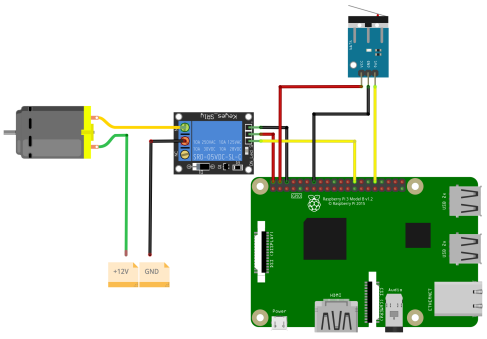


Figure 5: Water Delivery System Schematics

3.1 Sampling Rate

Swallow Sampling Human voice can extend up to 4,000 Hz. Therefore, following the Nyquist Theorem, the sampling frequency should be: $f_s \geq 8,000Hz$. This is implemented using the Sounddevice package. Within this package, it is possible to set the default sampling rate using `sounddevice.default.samplerate = 8000`.

Weather Sampling A script is automatically run every five minutes to check if there were any swallow events, and the Openweather Historical Weather API was used to sample the weather at the time of the swallow.

Glass Sampling Whether the glass has been picked up by the user to drink water is detected with a limit switch. To avoid polling and unnecessary power consumption, an interrupt system was set up and serviced by an Interrupt Service Routine (ISR).

3.2 Microphone System Data Collection

The data collected through the microphone has to be analysed by the ML model. This requires a chunk of audio in order to extract features and classify it. However, a swallow event may happen at any time. For this reason, it is necessary that the microphone records at all times, and then its data should be chunked. However, by doing this, it may be possible that the split between two chunks also splits a swallow event, resulting in the latter not being analysed. For this reason, a moving sampling window was introduced.

Figure 6: The Water Intake Detection System Set Up

Moving Sampling Window As from [10], we used a moving time window to be able to chunk the audio recordings without incurring in events not being detected. Sazonov et al. [14] found, based on 10,686 measurements, the time duration of a swallow event is be 1.15 s with a standard deviation of 0.29 s. We therefore settled on a time window of 1.4 seconds, as in [14]. In this way, the possibility of detecting a swallow twice is avoided.. Sazonov et al. [14] found 0.2 s a good duration for the step size of the moving window, as this value is smaller than the smallest time distance between two consecutive swallow events.

In order to implement in code the sampling window, the `record_swallows.py` script runs continuously in the Raspberry Pi and continuously records audio data. This is done through the `sounddevice` library. Every ten seconds, a .wav file is saved in the `audio_files/unchunked_audio_files/` directory. The name of the file is the timestamp of the file, as this will be used later to store the timestamp of a potential water swallow event.

Then, once a minute, the `ML_swallow_classifier` code uses the `AudioSegment` library to segment the unchunked audio files into 20ms chunks. The chunks are saved in the `/audio_files/audio_chunks/` directory, and the ten seconds audios are deleted.

Then, to create the 1.4 seconds-long audio files that the ML algorithm will work on. a FIFO queue is implemented: a list of 7, 0.2 s audio files is joined, again using the `AudioSegment` library, and fed into the ML model. Once this is done, the first of the 0.2 s audio files is culled from the beginning of the queue, and another 0.2 s audio file is appended.

3.3 Database Building

The database used to train the ML model was built off about a total of about 150 audio files. These were categorised in four categories: Water, Food, Empty Swallows and Noise. For people were involved in the recording of the audio files to help with model generalisation. In the future, a bigger number of people and a more heterogeneous group of subjects would further improve generalisation.

A custom script, `ML_database_build.py` was created to this end. When run, the user is prompted to select the type of swallow they wish to record. Then, the recording is placed in the correct folder.

3.4 Multi-Layer Perceptron Classifier for Swallow Classification

The ML model used is the Multi-layer Perceptron from scikit learn [15] (`sklearn.neural_network.MLPClassifier`). It uses a relu activation function, adam solver, adaptive learning rate and one hidden layer of depth 300.

It is necessary to extract features from the time-series audio files in order to feed those files to the ML model. To this end, Mel-frequency cepstral coefficients (MFCCs) were extracted, as from [14]. The mel-frequency cepstrum approximates the human auditory system's response to sounds, and is therefore useful when analysing voice and throat signals [14].

The `librosa` library was used to extract MFCC features using its `librosa.feature.mfcc` method [16].

Model Training The database entries were used for training the model. The entries were shuffled and split into training and test with an 80% / 20% split (`sklearn.model_selection.train_test_split()`), and their labels transformed into categorical data using Keras' `tensorflow.keras.utils.to_categorical()` method.

The accuracy of the model using the test data varied greatly, between 65% and 92%. This is caused by the small number of training data. After shuffling, in fact, it may happen that there is an unfavourable distribution of, say, water swallows training samples, with more of them being in the test database. This would leave the training database with only five or ten samples to train on.

Model Saving Given that the model training was done on PC, it is unfeasible to, every time the model needs to be used, train the model on the Raspberry Pi. For this reason, the ML model was saved and loaded from disk when necessary. To this end, the `joblib.dump` model was used to save the `ML_swallow_classifier_model.joblib` file in the `python_scripts/` directory, and was loaded in `ML_swallow_classifier.py` script using the `joblib.load()` method.

3.5 Physical Storing process

If one of the 1.4 seconds audio chunks is classified as having a water swallow, its timestamp should be uploaded to a database. However, connection may not always be available, and there is the need to create a buffer in case data cannot be uploaded. For this reason, when an audio chunk is classified as containing a water swallow, its timestamp is saved into `database/unuploaded_database.csv`.

3.6 Online Storing

The swallow events, as well as the weather data, are stored in an AWS database, and the chosen database program is MongoDB. The `pymongo` library is used to connect to the database using python.

3.7 Online Storing Process

A cron job-triggered script, `python_scripts/upload_to_online_database.py`, checks if there is an available connection to the online database. If there is, it sends a get request to the *Extreme IP Lookup API* to get the location of the Raspberry Pi, and then, using the location data and using the filename of the 1.4 seconds audio chunks, which is its timestamp, gets the weather information from the *Openweather Historical Weather API*. All of this information is organised into a dictionary and then uploaded to the online database. The requested data information is temperature, feels like temperature, humidity and pressure.

4 Basic Characteristics of the end-to-end Systems Set-Up and Data

4.1 Open Source Set Up

The eventuality of an open-source distribution is not to be excluded. To aid in this direction, the source codes are available on Github, and they are written in such a way as to be installed and ready to use with a maximum of two (provided) bash instructions.

Installation of the Codes The library requirements for the the water intake detection systems are saved in the `requirements.txt` file, and can be installed by running: `$ pip install -r requirements.txt`.

Then, to run the codes automatically, one must run `$ sudo crontab -e` and add the provided three lines to the end of the file to generate cron jobs.

The process is the same for the water delivery system, although, of course, with different requirements and different cron jobs.

Hardware All the required parts are listed in this report, and can be bought by the user. The video and the present report show the connections to be made, and no soldering is required.

4.2 Website Set-Up

The user creates an account on the website and, along with their username, they input their height, weight, age and gender. This data is then used to calculate the required water intake. Possible connection to health apps, such as Apple Health, Samsung Health, Google Fit, Garmin Coach or even Strava will be explored in the future. This would result in a better estimation of health by the mentioned apps, as their water intake monitoring is quite poor and almost never used. After the sign in, the users can access the website in a secure way by using their credentials. It is also possible to remember the password to speed up the sign in process.

4.3 Online Database Set Up

The set up of the online database is automatic. Once a user signs in, a new database collection is created for them, and will store their data without requiring any set-up.

5 Data Interaction/Visualisation/Actuation Platform

5.1 Data Interaction / Visualisation - Website

Following interviews with the above mentioned subjects, it was understood that showing the data in a clear way would be helpful both to the patient and to the eventual carer in order to promote a better control of the water intake. To this end, The data coming from the water Intake system is shown on a custom-built website.

5.1.1 Website Features

The website show each user their information. It is capable to provide session management, access control and multiple users. It has sign-up and login forms for the users to create accounts and log in. There is the possibility to save the password for faster access. Message errors are flashed in case of mistakes

The user's passwords are hashed so that they are not saved in their unencrypted form in the database. The website has pages accessible to non-logged users, and once the user has logged in, protected pages are shown. A user model stores the user's data.

The website shows water intake data relative to the user that has logged in, and information can be stored for multiple users. The user can perform various actions on the graph, such as zoom in and out, window zoom and save.

5.1.2 Website UI

The website is designed to be used by the target audience. Therefore, various features were implemented in order to make it intuitive and easy to use. In particular, the colours create a good contrast, key for readability.

5.1.3 Website Structure

The website's index page shows information about the system and provides links to the Github page and the video.

There is a sign up page and a log in page. Once the user decides to sign in, a form is loaded and prompts the user to insert user name, password, email, weight, height, and age. There are examples to guide in the use of proper units of measure.

The user can then log in and can choose to save the password for future log ins.

Once the user is logged in, three more pages appear:

- Water intake, which shows the raw data of water intake, as well as data on temperature, feels-like temperature, humidity and atmospheric pressure. The graph includes features such as zoom in and out, window zoom and save.
- Water Intake Analysis, which shows trend, seasonality and residual components, as well as autocorrelation and correlation of the data.
- Water Intake Monitor, which shows a progress bar of the day's water intake over the calculated required daily water intake. The daily water intake calculation is tailored on the user and based on the information the user has provided in the log in phase. This page also calculates whether or not the user, at the current time of the day, is above or below the amount of water they should have drank so far in the day. A message is shown to present this information.

5.1.4 Website Code

The website is built with python using the `Flask` library. The pages are developed in HTML and use custom CSS. Following is the folder and files structure.

```
flask_auth_app
    project
        __init__.py                     # setup the app
        auth.py                         # the auth routes for the app
        db.sqlite                       # the database
        main.py                         # the non-auth routes for the app
    models.py                           # the user model
        README.md                       # Readme file
        requirements.txt                #Required libraries
        templates
            base.html                   # contains common layout and links
            index.html                  # shows the home page
            login.html                  # shows the login form
            signup.html                 # shows the signup form
            waterIntake.html            # shows the water intake raw data
            waterIntakeAnalysis.html    # show the water intake data
            waterIntakeMonitor.html     # shows the water intake progress bar
        water_intake_analysis_page.py   # data analysis, graph generation
         water_intake_monitor_page.py   # data analysis, graph generation
        water_intake_page.py            # data analysis, graph generation
```

Blueprints The website is built around two Blueprints [17]. One Blueprint handles the regular routes, such as the index and the protected profile pages. Everything related to authorisation is handled by another blueprint. The blueprints are initialised in **__init__.py**.

Routes Each blueprint handles different routes. A route takes the form:

```
@auth.route('/signup')
def signup():
return 'Signup'
```

main.py routes the `index.html`, `waterIntake.html`, `waterIntakeAnalysis.html` and `waterIntakeMonitor.html`. **auth.py** handles the authorisation-related pages, so `login.html`, `logout.html` and `signup.html`. In addition to that, `auth.py` also handles the POST form data, both from `signup.html` and from `signin.html`

HTML Templates Each route retrieve custom-built templates written in HTML and enhanced with CSS. A base layout (`base.html`) is written as a template, and all the other pages inherit from this base layout.

User Models User models are necessary for session management. They are represented by classes that are then translated to tables in a database using `Flask-SQLAlchemy`. **models.py** defines the User class with its attributes, which in this case are email, password, name, heigh, weight, age.

The Database The database used for this app is an SQLite database. The `Flask-SQLAlchemy` library is used to create the database using the Python REPL.

Sign-up In the sign-up, the data that the user submits is added to the database.

It is important to make sure a user with the same email does not exist in the database. To this end, the database is queried with the email, and if a user is found, an 'Email address already exists' message is flashed and the current user is redirected back to the sign-up page to try again. If a user is not found, then the `generate_password_hash` from the `werkzeug` library is used to hash the password and all the data is stored in the database. If this is not the case, the chosen password must be saved in hashed form in the database for security reasons.

Login In the login phase, the email address inserted is checked for a correspondence in the database. If so, the inserted password is hashed and compared to the one in the database.

User Sessions Once the user has logged in, `the user_loader` method from the `flask_login.LoginManager` package is used to create a session for the user that will persist as the user stays logged in, which will allow the user to view protected pages.

Protected Pages To create protected pages, the `Flask_Login`'s `@login_required` decorator is placed between the route and the function that calls the page. This prevents users that are not logged in from seeing the route. An example of this is shown below:

```
@main.route('/profile')
@login_required
def profile():
return render_template('profile.html', name=current_user.name)
```

As seen above, it is possible to pass users attribute, such as the name, to the HTML page:

```
...
<h1 class="title">
  Welcome, {{ name }}!
</h1>
```

`if` statements are added to the navigation bar on the top of the `templates/base.html` page to show protected pages only if a user has logged in.

5.2 Actuation Platform - Water Delivery System

As explained above, the water delivery system provides a point of contact with the system that fosters a better hydration for the patient. The details of the system are explained above.

6 Data Analytics, Inferences and Insights

6.1 Data Processing

The data was sampled over seven days, from December 1st to December 7th. In total, 351 water swallows were collected. The data, once uploaded following the procedure outlined in Section 3, is downloaded from the AWS online database using the `pymongo` library. This is done, depending on the use case, via different queries. Processing is carried out in the `water_intake_page`, `water_intake_analysis_page` and in the `water_intake_monitor_page`.

The data from the online database is organised in a dictionary, and transformed in to a `pandas.DataFrame` object.

Since the water swallows come at different times, in order to make sense of the data, the water swallows are binned into 20 minutes intervals using `(dataFrame.dt.floor(""))` and `DataFrame.groupby()[""].count()`.

Water intake Calculation The website automatically calculates water intake based off the user's attributes. The formula is according to [18].

Bokeh Plots The library used to plot the data is the `Bokeh` visualisation library, which has the ability to generate plots that can be published in web pages. The data is embedded into the web page using the `components()` methods, which returns a `<script>` containing the plot data and a `<div>` that contains the plot.

6.2 Water Intake Page

The water intake page displays the raw data, as shown in Fig.7. A Bokeh plot shows the water swallows, binned in 20 minutes intervals, as well as temperature data, feels like temperature, pressure and humidity. The graphs are interactive, so they allow for zoom in/out, box zoom, restoree and saveMoreover, it is possible to hide some of the lines by clicking on the correspondent legend entry.

6.3 Water Analysis Page

The water analysis page analyses the raw data shown in the water intake page.

The `statsmodels.tsa.seasonal` package was imported, and its `seasonal_decompose()` method was used to analyse the water intake data. The `seasonal_decompose()` method takes an array as an input, as well as a period over which to perform the calculations. The selected period was one day. The function returns a `DecomposeResult` object, which contains seasonal, trend, and residual values.

`statsmodels.tsa.seasonal` integrates well with `pandas`, therefore the three components of the `DecomposeResult` object can be transformed into a `pandas.DataFrame` object and plotted using Bokeh (Fig.8).

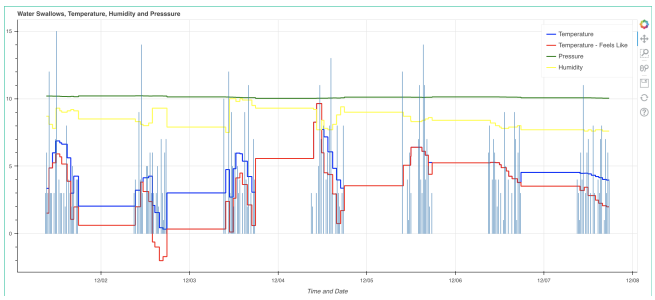
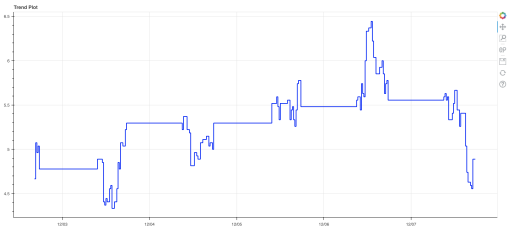


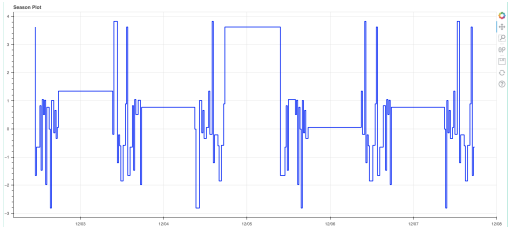
Figure 7: The Water Intake Page

Inferences from the Data Correlation As it can be seen from Fig.8, there is no apparent seasonality in the data, nor trends, nor apparent dependencies between temperature and water intake. This was to be expected, as we noticed water intake, if analysed swallow per swallow, is usually quite random. There is no consistent increase in water intake at lunchtime, nor afterwards, nor before. Based on this observation, we analysed the total water intake per day versus the average temperatures. The results are shown in Table 1

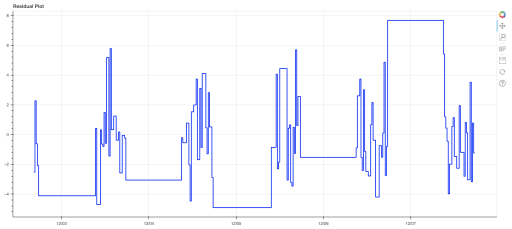
As it can be seen from the table, there is a slight correlation between the mL drank per day and the average day temperature. However, the test subject performs daily physical activity (outside the time bounds of this study). As it can be seen from the data on December 3rd, the water intake is lower than the other days, although the temperature is not significantly lower. This day corresponds to the subject's rest day, and the water intake is lower in accordance to [19].



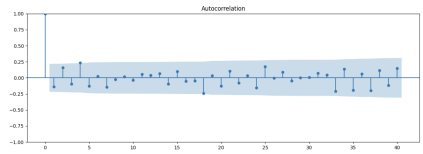
(a) Trend Plot



(b) Season Plot



(c) Residual Plot



(d) Autocorrelation Plot

Figure 8: Data Analysis

Table 1: Total Daily Water Intake vs. Daily Average Temperature

	Dec 1st	Dec 2nd	Dec 3rd	Dec 4th	Dec 5th	Dec 6th	Dec 7th
Water Intake [mL]	2044	1911	1704	2007	2051	2023	1996
Average Temperature [°C]	4.9	2.2	4.0	5.9	5.2	4.5	4.1

7 Discussion on the Important Aspects of the Project

Results The water intake detection system was compared to the amount of water that was actually consumed, measured by refilling a water bottle of known volume and weighting the remaining water at 6pm. Table 2 shows the results.

As it can be seen from the results, the estimations are within 30% of the real value. There is a constant offset from the results, which probably indicates that the amount of water per swallow of the test subject is different from the mean value found by Kim et al. [5]. Therefore, it would be a good idea to calibrate the system at set up by drinking a known amount of water.

It is interesting to see that also in this case the rest day affects the data, bringing the estimated value closer to the actual value. This may be because of water swallows of smaller average volume.

It must be said that these results were achieved in optimal conditions, with the test subject who has performed most of the database collection, in a silent room and with the test subject in a sitting position. Tests should be performed outside of these ideal condition to test generalisation properties of the system.

Trade-offs The already mentioned trade-off that led to the selection of a cheaper lavalier microphone instead of a throat microphone did not prevent the experiment from succeednig. However, a throat microphone might become necessary outside test conditions.

Cost The total cost of the system is presented in Table 3. As it can be seen, the price is just above £200. This is not a prohibitive price when considering that half of that amount is spent on the throat microphone. As it can be seen from the results, good data can be

Table 2: Calculated Water Intake vs. Actual Water Intake

	Dec 1st	Dec 2nd	Dec 3rd	Dec 4th	Dec 5th	Dec 6th	Dec 7th
Predicted Water Intake [mL]	2044	1911	1704	2007	2051	2023	1996
Actual Water Intake	1770	1528	1448	1465	1538	1598	1536
% Wrong	24%	20%	15%	27%	25%	21%	23%

l6cm

Table 3: Cost of the System

Part	**Cost**
Raspberry Pi 3B+	£35
Throat Microphone	£100
10,000 mAh Power Bank	£15
Raspberry Pi 3B+	£35
Peristaltic Pump	£15
Limit Switch	£1
Relay	£4
Wires	£1
Power Supply	£10
Total:	**£216**

obtained from a very cheap microphone, although the performance is unknown outside of the test environment. Moreover, a 3D printed case can be built to fit around the neck instead of taping the microphone.

Local vs. Cloud Processing The ML algorithm currently runs on the Raspberry Pi, which is worn on the body. This option was chosen to accommodate for some houses that have little access to wireless internet connection. In this way, the user would still be able to get some off-line water intake data without relying on online computation. However, this approach has a drawback: the body-worn microcontroller must be able to run a ML model, and is therefore bigger and more expensive. In the future, if wireless internet access on dementia patients' houses stops being a concern, the microcontroller could analyse the total energy of the audio chunks, and if they are above a certain threshold, they are sent to be processed online. The online server would then classify the audio chunks candidates to identify water swallows. This system would result in a lower part cost, and a possible delay in computation if no connection is available.

Battery Life The Raspberry Pi was run on a 10,000 mAh power bank, and the battery lasted the whole day, in accordance to [20]. As there was no way to find the battery level of the power bank system in a precise manner, further tests must be performed in order to evaluate with more precision the power consumption of the system.

Security The implementation of password access to the website guarantees the user with a good level of security to protect their personal data. Further measures may include two-factor authentication.

8 Avenues for Future Work and Potential Impact

UKDRI Collaboration The device could be further tested through the UK dementia Research Institute (UKDRI, [21]). UKDRI is running a project called "UKDRI Smart Home", which aims at creating dementia-friendly 'Healthy Homes' and provide insights into how dementia develops. This project would give further insights into hydration-related altered mental states and would couple well with the already existent EEG, sleep and behaviour sensing [22]. However, from interviews with two researchers working on this project, it is clear that the process to test within the UKDRI Smart Home project may be long.

Business Model Once perfected, the system would be sold as a subscription model; this benefits the company, which is guaranteed a constant cash flow, and the user, who would get a replacement product in case of malfunctioning.

The water delivery system is optional, and can be added if the user feels like it would bring them an advantage.

Open Source As a cheaper alternative, the system could be provided under an open source license. In this way, everyone would have access to it at the cost of hardware only. This may foster a collaboration between schools, who would learn to use a Raspberry Pi, and charities active in the field of Dementia, such as the Alzheimer's Society [23].

References

[1] D. Seymour, P. Henschke, R. Cape, and A. Campbell, "Acute confusional states and dementia in the elderly: the role of dehydration/volume depletion, physical illness and age," *Age and ageing*, vol. 9, no. 3, pp. 137–146, 1980.

[2] "Retevis r-151 1pin 3.5mm ptt throat mic earpiece covert air tube headset with slow rebound earbud for mobile phone c9038a sale - banggood.com." https://www.banggood.com/Retevis-R-151-1Pin-3_5mm-PTT-Throat-Mic-Earpiece-Covert-Air-Tube-Headset-With-Slow-rebound-Earbud-p-1210902.html?cur_warehouse=CN. (Accessed on 12/03/2021).

[3] R. A. Oppliger and C. Bartok, "Hydration testing of athletes," *Sports Medicine*, vol. 32, no. 15, pp. 959–971, 2002.

[4] T. Eichler, W. Hoffmann, J. Hertel, S. Richter, D. Wucherer, B. Michalowsky, A. Dreier, and J. R. Thyrian, "Living alone with dementia: prevalence, correlates and the utilization of health and nursing care services," *Journal of Alzheimer's Disease*, vol. 52, no. 2, pp. 619–629, 2016.

[5] S. I. Kim, J. H. Kang, D. I. Lee, J. R. Jo, H. J. Kim, J. B. Lee, Y. H. Jin, T. O. Jeong, and J. C. Yoon, "Measurement of volume of a swallow for liquid swallowing in healthy young adults," *Journal of the Korean Society of Clinical Toxicology*, vol. 11, no. 2, pp. 114–118, 2013.

[6] "Buy a raspberry pi 3 model b+ – raspberry pi." https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/. (Accessed on 12/21/2021).

[7] "Lavalier microphone, agptek clip-on lapel omnidirectional condenser mic with wind muff type c 3.5mm jack for phone pc laptop dslr camera recording interview, podcast, voice dictation: Amazon.co.uk: Musical instruments & dj." https://www.amazon.co.uk/AGPTEK-Microphone-Professional-Omnidirectional-Condenser/dp/B07SHSHW6H/ref=sr_1_4?crid=218WKCVSWT7KB&keywords=lavalier+microphone&qid=1640021780&sprefix=lavalier+microphone%2Cmi%2C132&sr=8-4. (Accessed on 12/20/2021).

[8] "Anker power bank, powercore slim 10000 portable charger, ultra slim battery pack, compact 10000mah external battery, high-speed poweriq charging technology power bank (usb-c input only) : Amazon.co.uk: Electronics & photo." https://www.amazon.co.uk/Anker-Ultra-Compact-High-Speed-VoltageBoost-Technology/dp/B07QXV6N1B/ref=sr_1_2?crid=HYBK1PJCR9RY&keywords=anker+10000&qid=1640085887&sprefix=anker+10000%2Caps%2C163&sr=8-2. (Accessed on 12/21/2021).

[9] "Frezen flexible throat mic microphone covert acoustic tube earpiece headset with finger ptt for kenwood pro-talk xls tk two way radio for baofeng uv-5r uv5r uv-82 bf-888s walkie talkie 2pin: Amazon.co.uk: Electronics & photo." https://www.amazon.co.uk/gp/product/B083K9HDX6/ref=ppx_yo_dt_b_asin_title_o06_s00?ie=UTF8&psc=1. (Accessed on 12/24/2021).

[10] L. F. Santoso, F. Baqai, M. Gwozdz, J. Lange, M. G. Rosenberger, J. Sulzer, and D. Paydarfar, "Applying machine learning algorithms for automatic detection of swallowing from sound," in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 2584–2588, IEEE, 2019.

[11] "Nt3 black ops 2 throat mic headset - iasus concepts official online store." https://shop.iasus-concepts.com/product/nt3-black-ops-2-throat-mic-headset-2/. (Accessed on 12/24/2021).

[12] "Historical weather api - openweathermap." https://openweathermap.org/history. (Accessed on 12/24/2021).

[13] "Ip lookup geolocation api - extreme-ip-lookup.com." https://extreme-ip-lookup.com/. (Accessed on 12/24/2021).

[14] E. S. Sazonov, O. Makeyev, S. Schuckers, P. Lopez-Meyer, E. L. Melanson, and M. R. Neuman, "Automatic detection of swallowing events by acoustical means for applications of monitoring of ingestive behavior," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 3, pp. 626–633, 2009.

[15] "sklearn.neural_network.mlpclassifier — scikit-learn 1.0.1 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. (Accessed on 12/20/2021).

[16] "librosa.feature.mfcc — librosa 0.8.1 documentation." https://librosa.org/doc/main/generated/librosa.feature.mfcc.html. (Accessed on 12/20/2021).

[17] "Modular applications with blueprints — flask documentation (2.0.x)." https://flask.palletsprojects.com/en/2.0.x/blueprints/. (Accessed on 12/23/2021).

[18] "How to figure out how much water you should drink plus an easy way to track it." https://www.mynetdiary.com/how-much-should-you-drink.html. (Accessed on 12/19/2021).

[19] M. N. Sawka, S. N. Cheuvront, and R. Carter, "Human water needs," *Nutrition reviews*, vol. 63, no. suppl_1, pp. S30–S39, 2005.

[20] "Power consumption benchmarks — raspberry pi dramble." https://www.pidramble.com/wiki/benchmarks/power-consumption. (Accessed on 12/24/2021).

[21] "Uk dri: Uk dementia research... — uk dri: Uk dementia research institute." https://ukdri.ac.uk/. (Accessed on 12/22/2021).

[22] "Ukdri smart home — research groups — imperial college london." https://www.imperial.ac.uk/systems-algorithms-design-lab/research/ukdri-smart-home/. (Accessed on 12/22/2021).

[23] "Alzheimer's society - united against dementia." https://www.alzheimers.org.uk/?gclid=Cj0KCQiA_JWOBhDRARIsANymNOYeaUQGf3g-i7i9NyzAqSmwzWnnyc5RjW82F8Dsk6fFF2xjZB2m27QaAp8gEALw_wcB&gclsrc=aw.ds. (Accessed on 12/24/2021).