TITLE GOES HERE

NPS30003 Grand Challenges in Science Report

by

Harry Causer

6164692

Word count =

Declaration

‘I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material, except where due acknowledgement is made in the manuscript. Any contribution made to the research by others, with whom I have worked at Swinburne or elsewhere, is explicitly acknowledged in the report. I also declare that the intellectual content of this report is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.’

Signed..... Date.....

Abstract

In this report ……

Acknowledgements

I would like to acknowledge the assistance and guidance provided by Dr. Francois Malherbe and PHD Student Phillip Zannon.

Table of Contents

Chapter 1

Introduction

Water is an important commodity for all of human society. The maintenance and protection of water ways, whether that be rural or urban is vital for the health and wellbeing of individuals and is an important driver in many industries (Boyd 2020). Globally 70% of fresh water is used for agriculture (Gleick 2014), in this industry maintaining the quality of water supply is vital to ensuring countries are able to feed their populations, degradation of water way health can lead to water that is not fit for use on farms.

To maintain and protect water ways there are several key factors that can be measured to provide an overview of the health of the system. These include but are not limited to; Dissolved oxygen, Temperature, Electrical conductivity, Total Dissolved Solids (TDS), pH and turbidity (Boyd 2020).

Dissolved Oxygen

Dissolved oxygen is the amount of free oxygen found in water and the most widely used measurement of water quality (Park 2007). The recommendation for general water waterway health is above 7mg/L however this is very situational and depends on the body of water that is being measured. Measurements taken need to be calibrated to the source. Dissolved oxygen can be affected by the amount of biomass in the system, the salinity or the electrical conductivity of a system (greater salt/electrical conductivity usually means lower dissolved oxygen) and temperature of the system (cooler water usually has a greater dissolved oxygen than warmer waters) (USGS 2009).

Electrical Conductivity

Electrical conductivity is used to measure the salinity of a body of water, but can also give indications of TDS, the saltier the water the higher the electrical conductivity (Slinger et al. 2007). Toepfer & Barton (1992) found that increases in salinity caused the oxygen use of fresh water fish species to increase, this over time could contribute to less dissolved oxygen in the system causing greater problems.

PH

PH is a measure of hydrogen ions in a system, or the acidity of a system, water ways should fall within a range of 6.5 – 8.5 outside this range the aquatic organisms may die which can lead to the increases in turbidity and enrichment from nutrients which can lead to algae blooms (Hancock 2016).

Total Dissolved Solids

TDS is a measurement of the concentration of dissolved minerals and organic substances found in a sample of water (Hancock 2016). Water that has a high TDS as previously stated could be high in salinity, or other minerals and is generally considered to be an issue for agriculture and irrigation (Banadkooki et al. 2020).

Turbidity

Turbidity is a measure of suspended particles within the body of water, it is measured in Nephelometric Turbidity Units (NTU), where drinking water should have a NTU of 5. Higher turbidity leads to increases in temperature and also causes algae blooms.

1.1 Aims and Objectives

Monitoring water systems can detect anomalies in the major parameters allowing problems to be solved before they become major threats to the ecosystem. The goal of this project overall is to develop a cheap and portable working prototype that can be used by anyone and can test for several of the key indicators and reports live water quality data to a web interface or app that is easy to understand and interpret, allowing for rapid detection of fluctuations in key parameters. The main objective of this report is to show proof of concept for the internet connectivity and live steaming of data.

This project is to show proof of concept for graphical user interface that is simple to understand, even for users with minimal chemistry knowledge. However this also needs to be balanced with the interface being able to satisfy the needs of more knowledgeable users. Wilson (2009) defines usability in 5 key points; learnability, efficiency, memorability, error and satisfaction. The human-computer interaction should be easy to learn, once learned an increase in production is seen, once learned it is easily remembered even after not being used for some time. The interaction should be as error free was possible and if errors occur recovery is easy and finally the interaction should be enjoyable for the user. The graphical user interface for this project will be either a web interface or a mobile app, I am more partial to web interface as it is cross platform and usable on more systems. I propose that the Water quality data could be viewed as a moving line graph for the rivers health, this of course depends on the update frequency of the data being provided, as discussed previously the frequency of reporting will have to be determined by battery capacity, if full real timing could be achieved this would be an excellent way to view the data. Another important feature would be alerts based on the water quality parameters hitting levels that are undesired.

1.2 Report Structure

Chapter 2 is a literature review of current research, Chapter 3 Methodology details the steps take to accommodate covid-19 as well as the software and coding language used for the project. Chapter 4 Results and Discussion details the important outcomes and steps taken to meet the objectives of the research project. Chapter 5 is the Conclusion and Future research and development directions.

Chapter 2

Literature Review and Theoretical Framework

2.1 Current state of knowledge

At present water quality data is collected manually in many countries (Chow 2019) with many tests requiring long lab turnaround times which prevents conservationists from being able to react quickly to emerging crisis (Tuna 2014). The Australian guidelines for fresh and marine water quality recommend field measurements be taken for parameters such as dissolved oxygen, redox potential and pH. Other parameters are tested in the lab, one of the many problems with sample collection is that collected samples could deteriorate while in transport causing the sample to result in data that is not an accurate representation of the site (ANZG 2018). Currently the most important factor in water sampling and testing is the people that are responsible for conducting said collection and testing (Li & Migliaccio 2011). However many of these teams are responsible for large areas (especially in rural areas) and must travel from site to site for sample collection. It is not economically viable to employee and train many people for many governments and charities.

To solve these problems in the last 13 years researchers have turned to wireless sensor networks (WSNs) that allow data from sites to be transmitted wirelessly using multiple different technologies. These include Wifi, RF, GSM and LTE. (Adu-Manu et al. 2017). WSNs are providing a way to streamline water quality monitoring by minimising the personnel and time required to collect data from multiple locations. However many of these sensors are expensive and typically only affordable in an industry based setting. As an example the Libelium Smart Water costing about $8,589.53 AUD at current exchange rates on the IoT marketplace website (2020).

Currently several countries have implemented WSN technology as part of their water quality management to varying degrees. This includes Australia, China, Peru, India, Tanzania, Portugal, Turkey, Ireland, Indonesia, Greece, Mexico and the USA (Adu-Manu et al. 2017). However many of these are in testing phases and used exclusively at reservoirs or other drinking water sites, having access to cheaper, portable devices could mean that governments could start to monitor water quality more successfully from a conservationist point of view.

There are currently many single parameter sensors and several multiple parameter sensors commercially available. As well as some newer theoretical devices.

Hall (2007) reported that no single parameter sensor tested was able to detect the full range of contaminants and that the characteristics of the water significantly affected results.

Dinh et al. (2007) investigated the deployment of a wireless sensor network in remote northern Australia to monitor water quality based on irrigation water being released in the area. Their main complication was the size of the area that needed to be covered, the system they designed only reported data on flow rate (the volume of water passing a defined point over a defined period of time e.g. litres per minute) but ran successfully for over a month. They concluded that sensor networks are a promising solution to sustainable irrigation systems.

Tuna et al. (2014) proposed a theoretical water quality management system using a wireless sensor network where the nodes in the network report independently to middleware where data is interpreted and reported to web interface, they concluded through emulated scenarios that the role-out of a practicable system would depend on transmission frequency, power and packet size, meaning that each nodes life is dependent on power consumption.

Ramya & Vijayakumar (2015) designed and developed a low cost monitor that measured the physiochemical properties of water using a raspberry pi as the core controller with data being presented on a web based interface in real time using cloud computing. They determined that raspberry pi and similar hardware are low cost and capable of processing and reporting the required data.

Chowdury et al. (2019) proposed an internet of things (IoT) monitoring system, IoT means each of the sensors in the network is internet capable, and able to report independently from each other to a main server where the data is collected and collaborated from other sensors. Due to budget limitations only small scale measurement of water quality parameters was completed but they concluded that IoT integrated with big data analytics can improve current systems making them more stable and able to communicate more quickly making real-time monitoring more practicable.

In the past 10 years smaller d.i.y electrical components such as Arduino and Raspberry Pi have become available and cheaper allowing more portable sensors and IoT devices. Using this hardware it is possible to build small, portable IoT water quality monitors that can communicate wirelessly using many different protocols, including close range Wi-Fi or Bluetooth or GSM/LTE for more remote locations.

Chapter 3

Methodology

This section will cover the coding language and software used in completing the objectives as well as changes made to accommodate Covid-19.

3.1 Covid-19

The initial goal of this project was to develop the web app interface for a prototype water quality sensor device, however due to covid-19 restrictions access to the device was not available. Dr Francois Malherbe gave access to a Netatmo home weather station device so proof of concept for a web app/interface for live steaming data can be developed.

Figure 1: Netatmo Smart Home Weather Station (Netatmo 2020)

3.2 Coding Language

The coding language used for this project was JavaScript and html. Mainly for the reason that the researcher has some basic previous experience with the language, also that JavaScript is major language used alongside html for web development. JavaScript also has many pre written open source libraries of code that can be used to add extra functionality to the language and the website or app being developed. For the most part debugging was completed with help from w3schools.com (2019) and MDN Web Docs (2019).

Table 1: Commonly used JavaScript

Code	Type	Use
const	Constant variable	Used to hold a value
async()	Async function	Enable synchronous expressions when used with await
await()	Operator	Waits for 'promise' in async function
console.log()	Method	Outputs information to the web console

3.3 Software

The first basic piece of software is a text editor which can be as simple as the basic notepad that comes pre-installed with windows. However there are other free packages available that have superior functionality when it comes to programming languages.

Initially the project was started using the Atom 1.52.0 ia32 (Figure 2) (https://atom.io/) text editor as the researcher has had previous experience with this editor. But with more experience and time spent the project was finished using Microsoft Visual Studio Code (MVSC) 1.50.0 (Figure 3)(https://code.visualstudio.com). Both editors are free to use and download and both are open source.

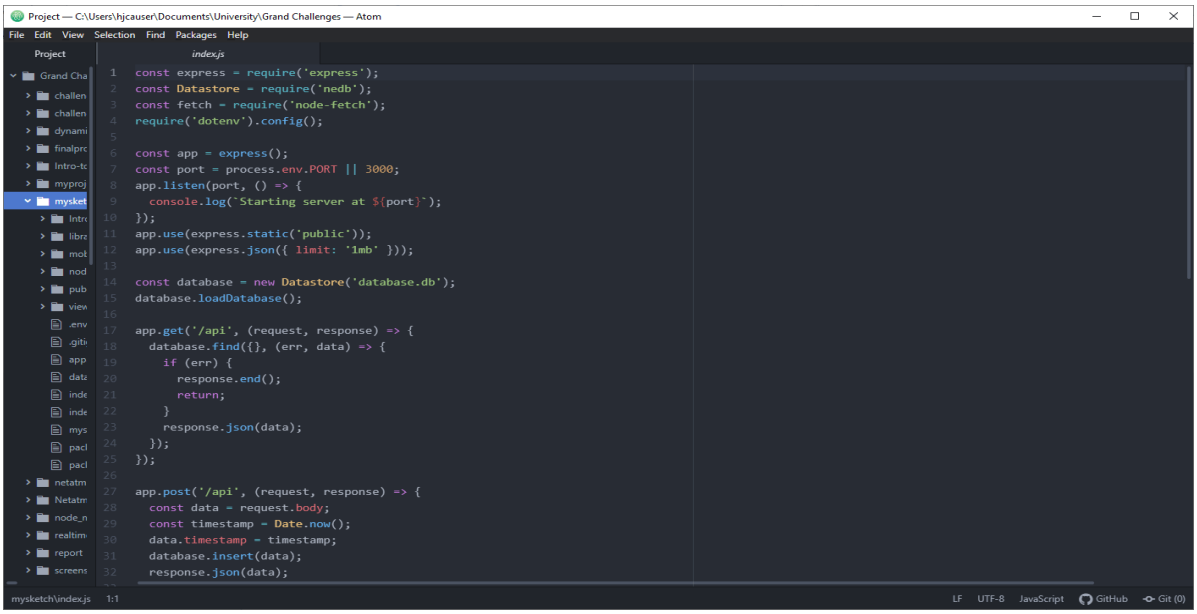

Figure 2: Atom 1.52.0 ia32

Meaning that the code can be downloaded and manipulated freely. Both work well and the choice was more aesthetic than functional, MVSC had a few visual tweaks that added some quality of life to debugging code.

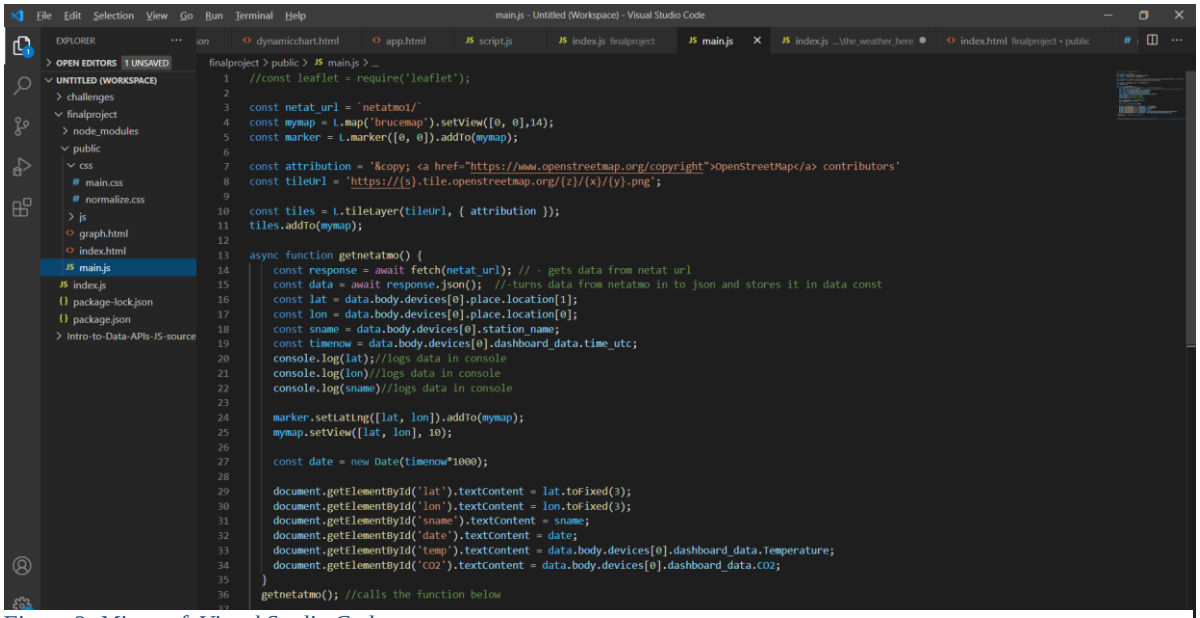

Figure 3: Microsoft Visual Studio Code

To be able to view the code in a web based environment a web browser is essential, the web browser used was Mozilla Firefox, this was used based on what the researcher had previously installed. Mozilla also has some excellent debugging tools, using the inspect element tool it is possible to bring up the console and debugger within the browser windows, however this has become fairly common place in most web browsers.

Node.js (https://nodejs.org/en/) is a JavaScript runtime it was used in this project as the web server environment, meaning it provided the server side structure for the web app/interface, server side refers

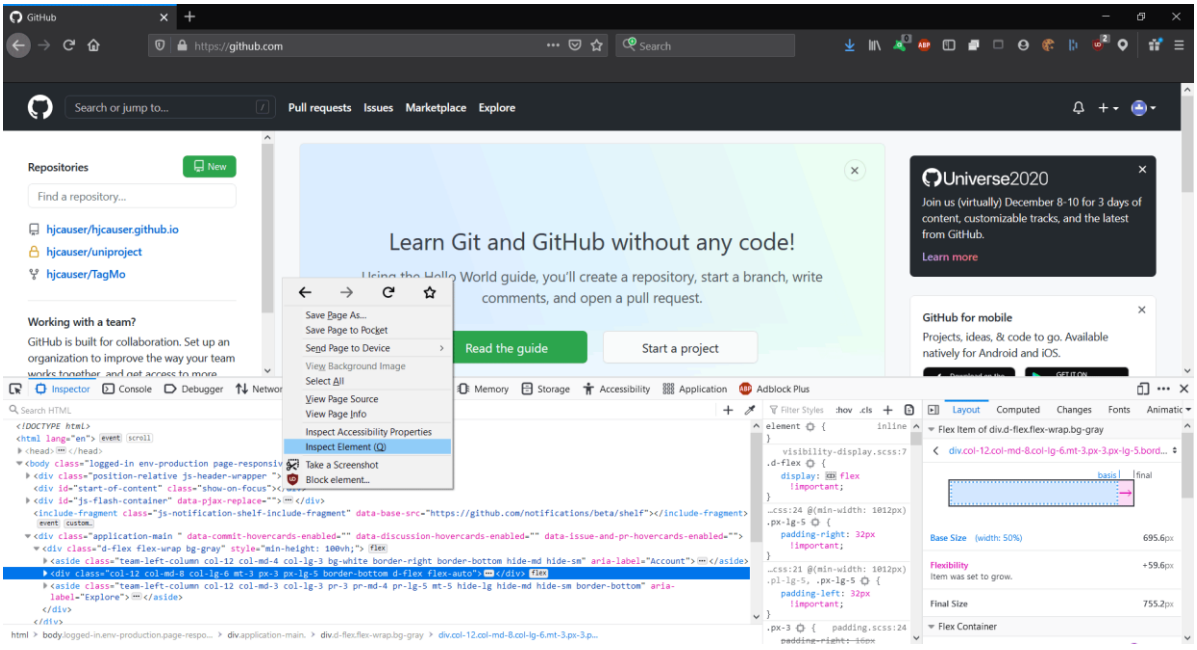

Figure 4: Mozilla Fire Fox - Inspect Element

to the part of the program or site that is not visible to the user of the webpage and is responsible for dealing with authentication and storing and retrieving data from a database.

NPM (https://www.npmjs.com/get-npm) is a package manager for Node.js it allows access to thousands of open source packages.

The next piece of essential software for this project was CMDER (https://cmder.net/) which is a console emulator, it replaces the default Windows command prompt app adding more functionality and ease of use. The console is used to initiate Node.js and install packages using node package manager.

Figure 5: CMDER with Node.JS initialised

3.4 Packages

The packages used in this project can be seen in Table 1 below;

Table 2: JavaScript Packages

Package	Description
chart.js(https://www.chartjs.org/)	Charting and graphing package
express(https://expressjs.com/)	Web app framework
fetch(https://www.npmjs.com/package/fetch)	Fetches url content, used for API connection
html5-boilerplate(https://html5boilerplate.com/)	Front-end template
leaflet(https://leafletjs.com/)	Addition of interactive maps
nedb (https://github.com/louischatriot/nedb)	Database management
node-fetch (https://github.com/node-fetch/node-fetch)	Fetches url content, used for API connection
plotly.js (https://plotly.com/javascript/)	Charting and graphing package

Chapter 4

Results and Discussion(Not completed)

In this chapter the

4.1 Graphs and graphing live data

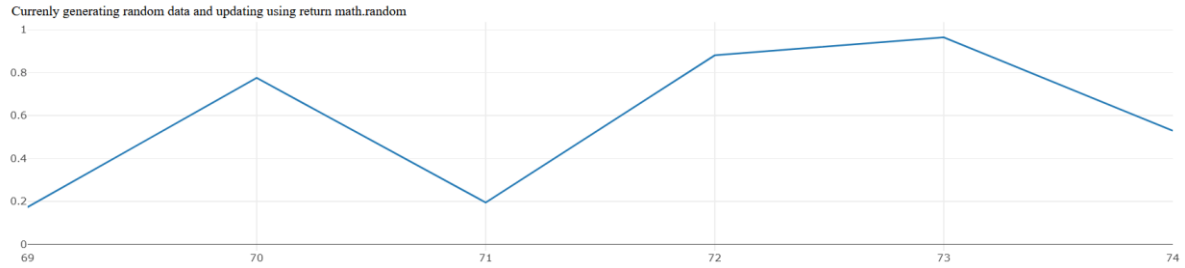

Figure 6: Live Updating Graph

Figure 6 shows the a graph that has been made to live update using random math function, updating in this way is a proof of concept that can be used to display live water quality parameter data. The

```
12          <script>
13              function getData() {
14                  return Math.random();
15              }
16              Plotly.plot('chart',[{
17                  y:[getData()],
18                  type:'scatter'
19              }]);
20
21              var cnt = 0;
22              setInterval(function(){
23                  Plotly.extendTraces('chart',{ y:[[getData()]]}, [0]);
24                  cnt++;
25                  if(cnt > 5) {
26                      Plotly.relayout('chart',{
27                          xaxis: {
28                              range: [cnt-5,cnt]
29                          }
30                      });
31                  }
32              },1000); //speed of calling next data point in milli seconds
33          </script>
```

Figure 7: Script for Live Updating Graph

update speed can be set at line 32 in figure 7.

In writing the JavaScript for the live updating graph Figure7, the Math.random() in function getData() is used for generating the data for the y-axis this can be replaced with a variable that is storing data

that is has been fetched from a live data source using an API, this graph uses the plotly js package, which can be seen being referenced in the plotly.plot command.

Dr Francois Malherbe asked if it was possible to create a graph from an excel file or csv file. Figure 8 shows the proof of concept of this. This graph was generated using a csv file downloaded from the NASA website. Part of the problem in doing this is script has to be written so that the data taken from

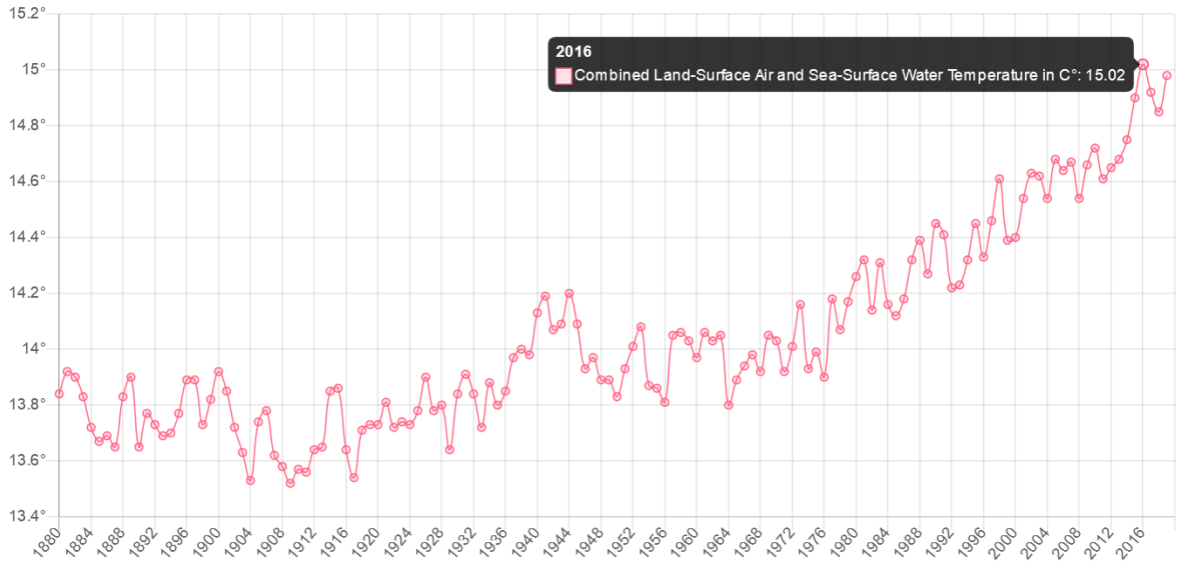

Figure 8: Graph created from external csv/excel file

the excel file can be recognised or 'parsed' by the JavaScript code. This can be seen in figure 7. The package used to make this graph was chart.js, this chart package has more functionality and support than the previously used plotly.js.

```
56      async function getData() {
57        const xs = [];
58        const ys = [];
59
60
61        const response = await fetch('ZonAnn.Ts+dSST.csv'); //where to get data from refers to the csv in local directory
62        const data = await response.text();
63
64        const table = data.split('\n').slice(1); // prepares the data so that it can be parsed as javascript
65        table.forEach(row => {
66            const columns = row.split(',');
67            const year = columns[0];
68            xs.push(year);
69            const temp = columns[1];
70            ys.push(parseFloat(temp) + 14);
71            console.log(year, temp);
72        });
73        return {xs, ys}; // adds the data to its respectuve array/variable
74      }
```

Figure 9: Parsing excel data in to JavaScript

Chart.js package is accessed by including code highlighted in Figure 9 in the header section of the html code.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content ="width=device-width, initial-scale=1.0"/>
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Grand Challenges: Importing Data from a CSV</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js@2.9.3/dist/Chart.min.js"></script>
  </head>
```

Figure 10: Accessing Chart.js package

The graph and graphing settings are edited by changing the code in Figure 11. Being able to access

```
    async function chartit(){
    const data = await getData();
    const ctx = document.getElementById('chart').getContext('2d');
        const myChart = new Chart(ctx, {
      type: 'line',
      data: {
        labels: data.xs,
        datasets: [{
          label: 'Combined Land-Surface Air and Sea-Surface Water Temperature in C°',
          data: data.ys,
          fill: false,
          backgroundColor: 'rgba(255, 99, 132, 0.2)',
          borderColor: 'rgba(255, 99, 132, 1)',
          borderWidth: 1
        }]
      },
      options: {
        scales:{
          yAxes:[
            {
              ticks: {
                //include degress sign in ylabels
                callback: function(value, index, values) {
                  return value + '°';
                }
              }
            }
          ]

        }}
    });
```

Figure 11: Chartjs Settings

excel files shows that historical data or data saved to a database can be graphed statically.

Ultimately data from the prototype device would have been more useful here to display but was still inaccessible at this point.

4.2 Accessing live data (not completed)

From a programming perspective accessing live data has many hurdles, the first of which is how to do it so that it is useable by server that is receiving it. The method used with the netatmo device was Representational state transfer also known as REST. This is a code on demand web architecture that allows connections between servers and clients (Massé 2012). The developers the netatmo provided a REST application programming interface (API) which unlocks certain data and makes it accessible to developers that may require that data. Many companies and government services today provide REST APIs for data collected, many are free while others are a paid service. Figure 12 shows a basic set of API data from that updates live and gives the location of the international space station (ISS).

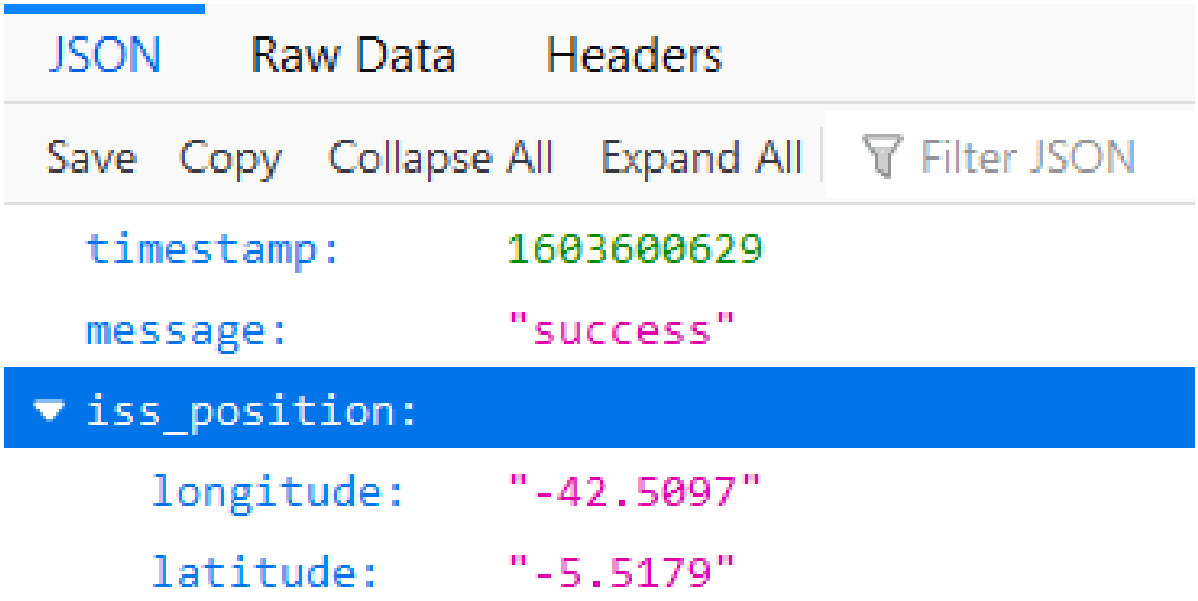

Figure 2: API data for ISS (acessiable http://api.open-notify.org/iss-now.json)

The ISS live data is free and doesn't require an access code or authentication, which makes it simple to access. Other APIs although free require developers to sign up for a unique code that can be used as an identifier and track data requests, as many of the free services allow a certain number of requests per day. Furthermore netatmo being a proprietary and commercial venture requires the user to authenticate using their netatmo account only allowing access the data generated by their particular device.

Authentication using OAuth (used to grant access to APIs without giving users passwords or names to the server) was problematic and difficult to debug and tom date not functional, so to access the netatmo data, code from github user JuYoungAhn(2018) was used allowing an OAuth access code to be generated that could be hardcoded into the server side programming. This is not recommended for a live version as it is not secure or good practice, however it allowed progress to be made and is fine security wise for a local version, which only required the OAuth access code to be updated in the code when it expired. Figure 13 also shows the basic code used for the server side programming, this code is not viewable from the client side in the browser window, but is responsible for delivering the

other files in the public directory (and also connecting to the API to access station data). The console.log(json.body.devices[0]); is a debugging function that shows the API is connecting by

```
const express = require ('express');
const app = express();
const fetch = require('node-fetch');
//require('dotenv').config();

app.listen(3000, () =>console.log('Listening @ 3000'));
app.use(express.static('public'));


app.get('/netatmo1', async (request, response) => {
const netatmo_api1 = `https://api.netatmo.com/api/getstationsdata?access_token=
const netat_response = await fetch(netatmo_api1);
const json = await netat_response.json();
response.json(json);
console.log(json.body.devices[0]);
});
```

Figure 3: Server side Netatmo

importing the data into the developer console in CMDER, once the data is imported you can start working with it to create the user interface.

```
C:\Users\hjcauser\Documents\University\Grand Challenges\finalproject  (finalproject@1.0.0)
λ node index.js
Listening @ 3000
{
  _id: '70:ee:50:03:b7:9c',
  station_name: 'BruceFM',
  date_setup: 1412663879,
  last_setup: 1600330611,
  type: 'NAMain',
  last_status_store: 1603603389,
  module_name: '1BruceFM',
  firmware: 177,
  last_upgrade: 1552794327,
  wifi_status: 36,
  reachable: true,
  co2_calibrating: false,
  data_type: [ 'Temperature', 'CO2', 'Humidity', 'Noise', 'Pressure' ]
  place: {
    altitude: 162,
    country: 'AU',
    timezone: 'Australia/Melbourne',
    location: [ [redacted] ]
  },
  home_id: '5f631b73dd16a7697b2c5218',
  home_name: 'NetBruce',
  dashboard_data: {
    time_utc: 1603603377,
    Temperature: 16.2,
    CO2: 589,
    Humidity: 57,
    Noise: 39,
    Pressure: 1022,
    AbsolutePressure: 1002.6,
    min_temp: 14.3,
    max_temp: 28.1,
    date_max_temp: 1603547378,
    date_min_temp: 1603583454,
    temp_trend: 'stable',
    pressure_trend: 'up'
  },
```

Figure 4: Console logged netatmo data (location blurred)

4.3 User Interface (Not yet completed)

Chapter 5

Conclusions and Future Directions

5.1 Future Research and Development

While currently there are many ways to monitor water quality there are is a hole in the current market for a device that is cheap, portable, able to report data wirelessly and easy to use.

Development of a device that meets these parameters would make water conservation and monitoring more mainstream and accessible not just for governments and industry but for individuals who want to be accountable for their environment.

To continue improving the existing prototype to allow for live data a REST API would have to be developed, this would not only allow for water quality data to be live steamed but also would allow data from other water quality meters to be collected and manipulated at the server point to provide a bigger picture of water way health.

Other interesting features that could be added with integrated with data from other API's including weather data, air quality data and satellite mapping data allowing for a more comprehensive view of the ecosystem.

The water quality meter would also have to include a server side database that would record current data and allow it to be recalled at a later date. Similar to the static graph from excel data in Figure 8.

The current project was creating a web interface that would run across most devices, however for mobile and tablet users, apps are a much more refined option, while it is possible to build a web based app using JavaScript, android and apple devices require a different programming languages Java(different from JavaScript) and Swift respectively.

5.2 Conclusion (incomplete)

While the prototype is in working order, there are still many steps to make it internet connected and accessible, with many of these steps require an IT professional or student at the minimum to develop the essential online security required of a commercially available device. The graphs and web app developed during this project can at least provided proof of concept but the developer has minimal IT training and experience, as can be seen with the OAuth problems.

Bibliography*

Abbasi, T 2012, Water quality indices, Oxford, U.K. : Elsevier, Oxford, U.K.

Adu-Manu, K, Tapparello, C, Heinzelman, W, Katsriku, F & Abdulai, J-D 2017, 'Water Quality Monitoring Using Wireless Sensor Networks: Current Trends and Future Research Directions', ACM Transactions on Sensor Networks (TOSN), vol. 13, no. 1, pp. 1-41.

Ahn, JY 2018, "JuYoungAhn/Netatmo-Nodejs," GitHub, viewed 30 September, 2020, <https://github.com/JuYoungAhn/Netatmo-Nodejs>.

Andrews, W 2017, Arduino Playground: Geeky Projects for the Experienced Maker, Place of publication not identified: No Starch Press.

ANZG 2018, Field Sampling Program, Australian and New Zealand Governments and Australian state and territory governments, viewed 21/08/2020, <https://www.waterquality.gov.au/anz-guidelines/monitoring/field-sampling-program>.

Anon 2019, "MDN Web Docs," MDN Web Docs, viewed <https://developer.mozilla.org/en-US/>.

Anon 2019, "W3Schools Online Web Tutorials," W3schools.com, viewed <https://www.w3schools.com/>.

Banadkooki, FB, Ehteram, M, Panahi, F, Sh. Sammen, S, Othman, FB & El-Shafie, A 2020, 'Estimation of total dissolved solids (TDS) using new hybrid machine learning models', Journal of hydrology (Amsterdam), vol. 587,

Bhargava, SK 2009, Practical methods for water and air pollution monitoring, New Delhi.

Biswas, AK & Tortajada, C 2019, 'Water quality management: a globally neglected issue', International Journal of Water Resources Development, vol. 35, no. 6, pp. 913-916.

Boyd, CE 2020, Water Quality An Introduction, 3rd ed. 2020.. edn., Cham : Springer International Publishing : Imprint: Springer.

* Swinburne Harvard referencing style has been used.

Chowdury, MSU, Emran, TB, Ghosh, S, Pathak, A, Alam, MM, Absar, N, Andersson, K & Hossain, MS 2019, 'IoT Based Real-time River Water Quality Monitoring System', Procedia Computer Science, vol. 155, pp. 161-168.

Glasgow, HB, Burkholder, JM, Reed, RE, Lewitus, AJ & Kleinman, JE 2004, 'Real-time remote monitoring of water quality: a review of current applications, and advancements in sensor, telemetry, and computing technologies', Journal of Experimental Marine Biology and Ecology, vol. 300, no. 1, pp. 409-448.

Gleick, PH 2014, The World's Water, Washington, Dc Island Press/Center For Resource Economics.

Hall, J, Zaffiro, AD, Marx, RB, Kefauver, PC, Krishnan, ER, Haught, RC & Herrmann, JG 2007, 'On-line water quality parameters as indicators of distribution system contamination', Journal (American Water Works Association), vol. 99, no. 1, pp. 66-77.

Hancock, N 2016, TDS and pH, Safe Drinking Water Foundation, viewed 22/08/2020, <https://www.safewater.org/fact-sheets-1/2017/1/23/tds-and-ph>.

Massé, M 2012, *REST API design rulebook [designing consistent RESTful web service interfaces]*, Beijing [U.A.] O'reilly.

Netatmo 2020, "Smart Weather Station Indoor Outdoor," Netatmo, viewed 25 September, 2020, <https://www.netatmo.com/en-eu/weather/weatherstation>.

Hongpin, L, Guanglin, L, Weifeng, P, Jie, S & Qiuwei, B 2015, 'Real-time remote monitoring system for aquaculture water quality', International Journal of Agricultural and Biological Engineering, vol. 8, no. 6, pp. 136-143.

Dinh, TL, Hu, W, Sikka, P, Corke, P, Overs, L & Brosnan, S 2007, 'Design and Deployment of a Remote Robust Sensor Network: Experiences from an Outdoor Water Quality Monitoring Network,' 32nd IEEE Conference on Local Computer Networks (LCN 2007).

Li, SY, Tao, JH & Yu, L 2013, 'Research and Application of Real-Time Remote Monitoring and Early Warning System of Source Water Quality', Advanced Materials Research, vol. 779, pp. 1408-1413.

Li, Y & Migliaccio, KW 2011, Water quality concepts, sampling, and analyses, Boca Raton, FL.

Mukhopadhyay, SCe & Mason, Ae 2013, Smart Sensors for Real-Time Water Quality Monitoring, 1st ed. 2013.. edn., Berlin, Heidelberg : Springer Berlin Heidelberg : Imprint: Springer.

Papadopoulos, NJ & Jannakoudakis, A 2016, 'A chemical instrumentation course on microcontrollers and op amps. Construction of a pH meter. (operational amplifiers)(Report)', Journal of Chemical Education, vol. 93, no. 7, pp. 1323-1325.

Silva, S, Hoang Nghia Nguyen, Tiporlini, V & Alameh, K 2011, "Web based water quality monitoring with sensor network: Employing ZigBee and WiMax technologies," 8th International Conference on High-capacity Optical Networks and Emerging Technologies.

Toepfer, C & Barton, M 1992, 'Influence of salinity on the rates of oxygen consumption in two species of freshwater fishes, Phoxinus erythrogaster (family Cyprinidae), and Fundulus catenatus (family Fundulidae)', The International Journal of Aquatic Sciences, vol. 242, no. 3, pp. 149-154.

Tuna, G, Nefzi, B, Arkoc, O & Potirakis, SM 2014, "Wireless Sensor Network-Based Water Quality Monitoring System," Key Engineering Materials, vol. 605, pp. 47–50.

Vijayakumar, N & Ramya, R 2015, "The real time monitoring of water quality in IoT environment," 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015].

Wilson, C 2009, User experience re-mastered your guide to getting the right design, Amsterdam; Boston.