Water
Action
Initiative
New Zealand

WaiNZ Chemical Analysis Android App

Technical Documentation

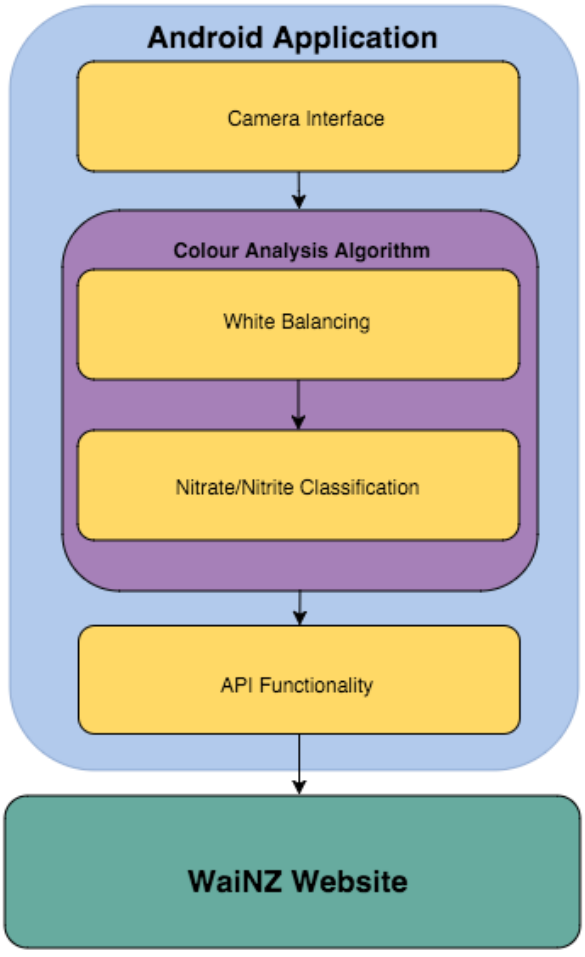
Introduction

This document focuses on the technical aspects of a new WAINZ Android application that measures water quality and content by analyzing an industry proven chemical card to measure the level of nitrites and nitrates in a body of water in order to further improve the monitoring of New Zealand's rivers and streams. We discuss our achievements & how the application works, the testing that has been done to date, and possible future work.

The Project

The application is an extension of *WaiApp*, an android application that was developed by in the year prior to our course. The project was divided into two main components: the *Android Application* and the *Colour Analysis Algorithm*.

The Github repository can be found at: http://github.com/PsukheDelos/Riverwatch



Android Application

WaiApp's purpose was to submit photos of pollution events to the WaiNZ website's database with geolocation and description information. We essentially needed to do the same thing, albeit after analysing a few colours from the test strip. *WaiApp* already had the ability to take pictures, enter incident description information, geolocate the event, submit the incident, and cache incidents to send later when out of range. So, we set out to repurpose *WaiApp* to fit our needs. There were four major tasks that needed to be done: upgrade the development environment, clean up the existing code base, develop a customised camera interface, and integrate the algorithm.

Upgrading Development Environment

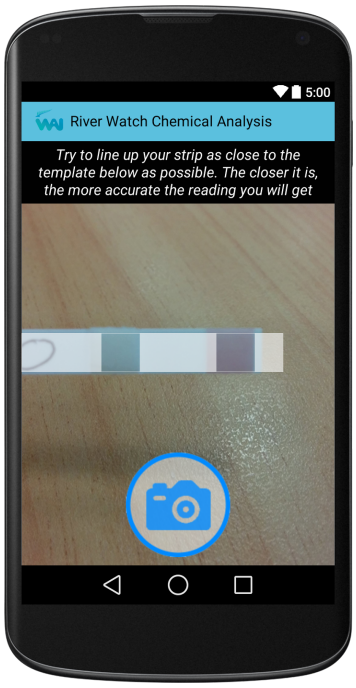
WaiApp was used as a starting point for the Android Development. As it turns out, it was developed in Eclipse. However, this is no longer the preferred method of Android development. So, we successfully migrated the existing codebase into Android Studio. Future developers will be able to pick up here by simply opening the project folder in Android Studio.

Cleaned Up Code Base

WaiApp was a fantastic starting point for our application. However, there was quite a bit of unused code left over from both the previous team and the migration process. We endeavoured to clean up as much of this as possible and add better documentation throughout the code.

Customised Camera Interface

WaiApp was designed for the purposes of submitting plain images. For our purposes, we were designing an application that needed to be able to read in the colours off a chemical test trip. We needed to a camera interface that would guide users in taking a photo that would be of use in the *Colour Analysis Algorithm*. To do this, we replaced their Camera (which was using a simple Android method of getting the system camera) with a Custom Camera. This is important as later on we only take sections of the image to feed to the algorithm for it to process.



Algorithm Integration

Finally, we needed to integrate the algorithm into the application. This involved being able to send the correct sections of the image to the algorithm for processing and reconfiguring the information sent to the WaiNZ website to accommodate the values we produced. Next, we will discuss the algorithm.

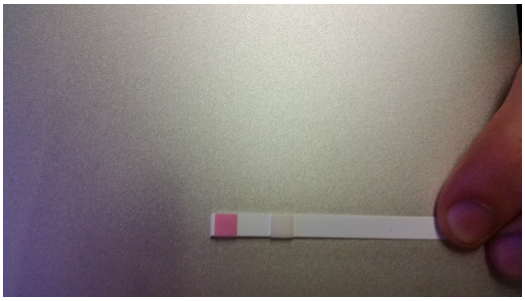
Colour Analysis Algorithm

White Balancing

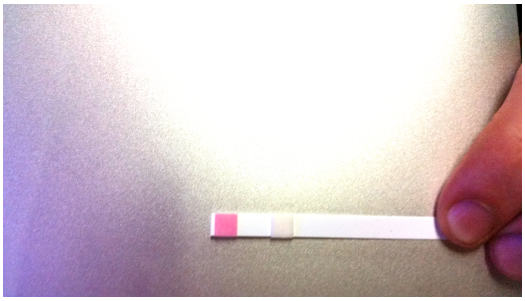
Our white balancer accepts the two images as subsections of the full photo. These are:

- ❏ The coloured section of the strip to be white balanced
- ❏ The section of the strip which should be treated as white (the area between the coloured section)

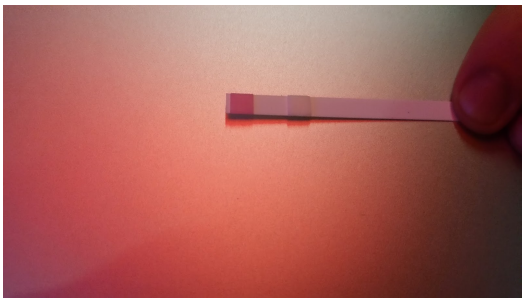
1. The white balancer then extracts the red, green and blue values of every pixel to separate lists.
2. We sort the red, green and blue lists in ascending order.
3. We remove the upper and lower 25 percent of all red, green and blue values then average the remaining values to find our estimated “white point.”
4. We then find the multipliers in each of the red, green and blue channels which, when applied to our “white point” will bring the white point to true white.
5. Next, we apply this multiplication to every pixel in the coloured section of the strip.



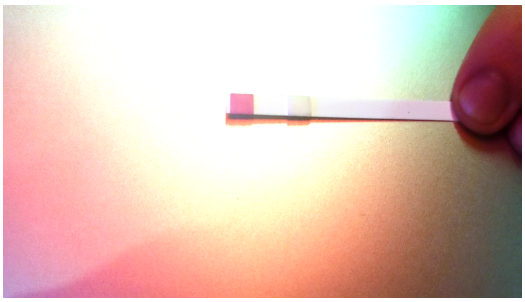
Normal Light



Nitrate Analysis: 13.9



Same Strip Under Red Light



Nitrate Analysis: 14.5

~7% error

Nitrate/Nitrite Classification

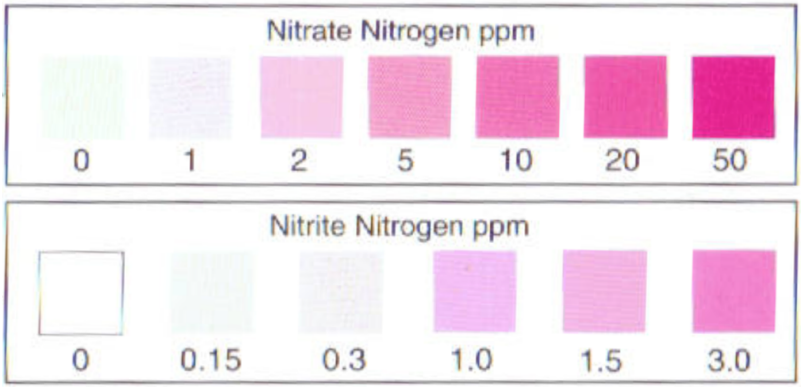
The classification part of the algorithm is responsible for taking the white balanced image and converting it into actual nitrite/nitrate levels. It is a variation on the K-NN Algorithm. The input to the classification part of the algorithm is in two parts:

- ❏ The white balanced coloured segment of the strip to analyse
- ❏ A map, from (Double) nitrate/nitrite (depending on the analysis to be done) to a Color which represents that level (From the side of the bottle)

1. The maps from class label to colour are constructed.
2. The algorithm extracts the median, HSB Color from the original white balanced image (input 1 above).
3. The distance to each colour in HSB space is calculated.
4. We find the best two nitrate/nitrite classes.
5. Finally, the nitrate/nitrite levels are calculated.

Original Image

Converted Image



Reference Colours

Further documentation of the algorithm can be found in the code comments.

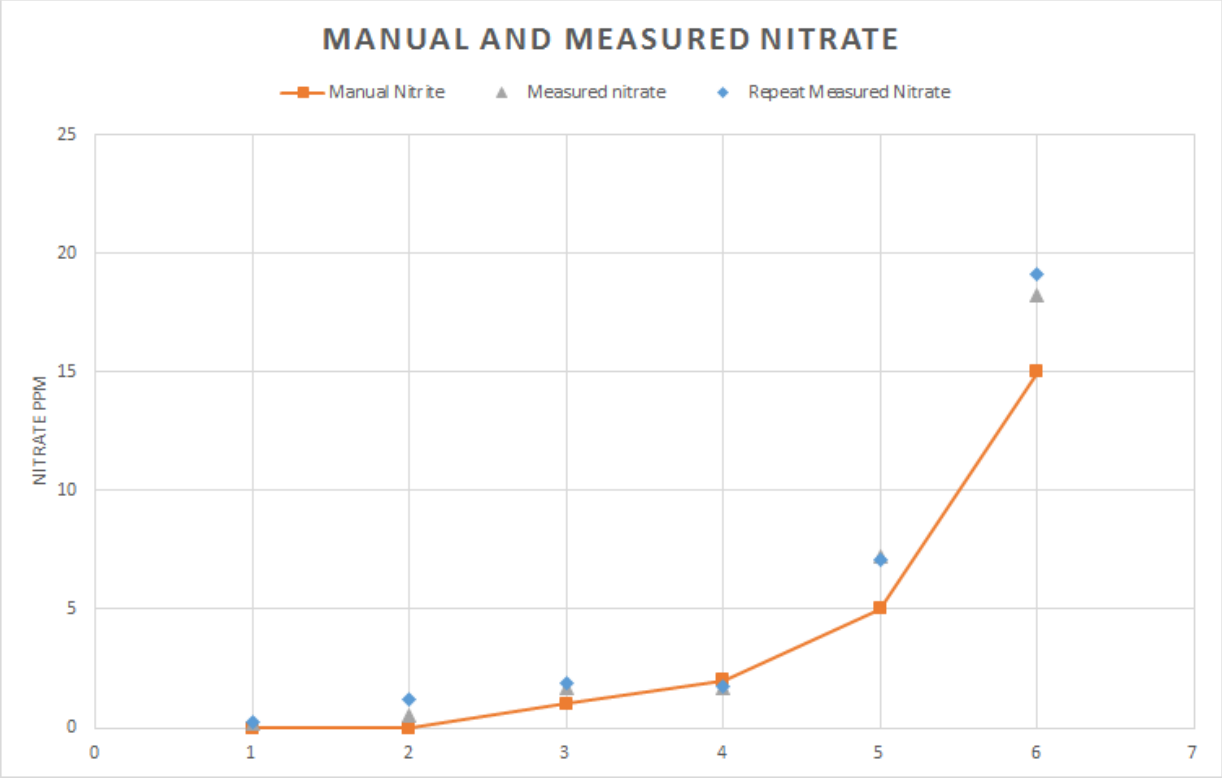
With the project opened in Android Studio, navigate to:
/app/src/main/java/nz/co/android/riverwatch/colour_algorithm

Testing

Manual vs. Measured

We tested our implementation by first using the test strips as a normal person would. We'd write down what we thought the reading was by comparing it to the side of the bottle. Then, we'd run it through the algorithm and see how it compared.

The results below are of six such tests. The line/squares represent what a human perceived the nitrate level to be. And the triangles and diamonds represent what the application determined the nitrate level to be. You can see in the higher ranges the perceived value of human vs algorithm begin to differ. However, one thing to note is that the algorithm is fairly accurate with itself. The diamond and square represent two readings of the same strip. Therefore, when they are close together, that means the algorithm's ability to repeatedly get the same result is high and that is good. In future work, we talk a bit more about what could be done to better test this application and some of reasoning behind why our results may be off.



Coverage

The main application work was built off of a previous teams WaiNZ application, *WaiApp*. Because of this, all the italicised phones above were ones used by the previous team. Since our code uses the same API, we are certain that our changes would work with theirs up to and including the newer Android 5.0 Lollipop.

Device	**Version**
LG P500h	*Gingerbread (2.3.3)*
Samsung Galaxy Y	*Gingerbread (2.3.6)*
Galaxy Nexus	*JellyBean (4.3)*
Samsung Galaxy Note 3	*KitKat (4.4)*
Samsung Galaxy S3	*KitKat (4.4.2)*
Samsung Galaxy S5	Lollipop (5.0)
Nexus 7 (2013)	Lollipop (5.1)
Nexus 5	Lollipop (5.1)

Future Work

Android *Camera2* API

As you may have noticed, the pictures come out blurry. This is because the older camera api does not allow one to manually adjust the focus of the camera. Instead, you have to use targeted autofocus, which we found does not work particularly well when taking a photo of something very close to the camera. This does not affect the ability to perform a colour analysis, however, it does look worrying from a user's perspective. We chose to carry on with the original Camera API as we were targeting a broad range of Android Devices. However, you may wish to target less devices for the sake of overall better user experience. You could implement a fixed focal length from the camera. Having done this, you could even design a device that clips to the camera to hold a test strip that exact distance. Just a few ideas.

Add other testing strips (pH?)

Before our nitrate/nitrite test strips arrive, we developed the algorithm using pH test strips. This may be something your client is interested in. Or perhaps there are other types of tests available. Head on over to the Hach website and see what they've got to offer!

Porting (Android -> iOS/Windows)

This is pretty straight forward. Wrap your head about the algorithm and slap it into an iOS or Windows app!

Further Long-Term Testing

We did some testing of the algorithm in the real world scenario. However, we were limited by both time and resource in what we could do. We acquired solutions that had known measurements of Nitrate and Nitrite in them from the Chemistry Department. However, over time these solutions begin to break down. By the time we were formally testing, we suspect our solutions may have been compromised. We'd recommend sourcing known concentrations of Nitrate and Nitrite and testing immediately. We'd even go as far as recommending doing this in a controlled lab environment with the assistance of a lab tech to ensure that proper concentrations are being used, and perhaps being able to use a more advanced device (The chemistry department has a spectrometer) to again measure the concentration levels. You could also test various mobile devices. There is no guarantee that all camera hardware is capable of producing reliable results. Have a think about how you'd ensure that. Maybe test phones with the worst camera hardware. Further testing would go a long way in proving the actual validity of the results we are getting!

Contacts

For further information regarding this project feel free to contact us! Glen & Jack are your best points of contact regarding the *Android Application.* Colin, Luke, & Nick are best for any questions you may have about the *Colour Analysis Algorithm*. We're happy to help!

Team Opera

Colin Douch	quirliom@gmail.com
Nicholas Garrett	nickgarrett86@gmail.com
Luke Inkster	luke.inkster@gmail.com
Glen Peek	glen.peek@gmail.com
Jack Robinson	jackrob1994@gmail.com