**Cycle Proximity Technical Report**

Zane Unverferth

Student

Otago Polytechnic

Dunedin

unverzp1@student.op.ac.nz

Cameron Hill

Student

Otago Polytechnic

Dunedin

hillcg2@student.op.ac.nz

**Abstract**

This paper looks at the development of an Arduino device, Android application and website for the Dunedin City Council (DCC). The purpose of this system is to help the issue of vehicles passing cyclists too close by identifying streets that this occurs in.

As a solution for this problem we first developed an Arduino with a sonar sensor on it that was capable of measuring the distances of passing vehicles. This then communicated through Bluetooth to an Android application that could log the distances the Arduino read, along with time and location. This collected data can be viewed on the application and sent to a server. While on the server the data is parsed by a Python script into a MySQL database. Then a website we developed can read from the database to display all of the users close encounters on a map.

We used the Agile Development Framework. With this we kept in constant communication in the group to identify issues and keep up to date with progress, this allowed us to implement changes and helped us progress. We kept in communication with the Dunedin City Council to keep them update of our progress.

**1 Introduction**

**1.1 The problem**

Cycling is a means of transportation that can prove to be cost effective, fun and good for your health. Having dangerous roadways or not have safe routes throughout the city can make it hard for people to get into cycling in Dunedin.

We have developed a system a system that will help map out the dangerous areas for cyclists. The way the system maps out dangerous areas is by how close vehicles are passing cyclists with our system attached. Kylie Huard is the client for our project. She is the senior transportation planner for the DCC. We hope that we can gather enough data for the DCC to show them dangerous areas in Dunedin that may need to be acted upon.

This paper will show the research undertaken by Cameron Hill and Zane Unverferth in solving the problem and the development used in implementing the solution. A detailed look of the implemented system is accompanied by a description of the development process and the testing that was required.

**1.2 DCC**

The Dunedin City Council is the local government authority for Dunedin. The Dunedin City Council has this to say about cycling on their website:

“Cycling is growing in popularity in Dunedin and is increasingly becoming an area that needs our attention. We have strategies in place to encourage cycling as a means of transport as well as campaigns to make it safer”[4]

**1.3 Related Research**

Related literature alongside with constant small scale experimentation has shown that following technologies as potentially viable for use in creating the system and were researched accordingly.

**1.3.1 Cycling Safety**

As cyclists are going to be the main users of our system we researched about safety when it comes to cycling to get a better understanding when it came to how cyclists are using the roads. Our device will be measuring how far away from the cyclists the vehicles are getting so we researched various factors that would be the cause behind the two getting close. Our research found that some of the reasons were that the cyclist may not have a choice and be forced out onto the road “Cyclists may ride away from the kerb or occupy a lane – not because they want to annoy drivers, but to:

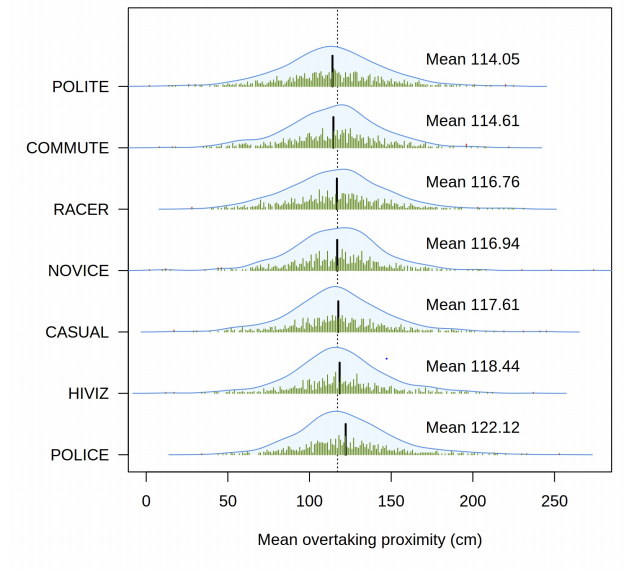
-- avoid drains, potholes or roadside rubbish

-- be seen as they come up to intersections with side roads

-- discourage drivers from squeezing past where it’s too narrow.”[1].

We are hoping that the data collected by our device will be able to be used to help improve the roads and possible add cycling lanes to areas that prove to be particularly hazardous to cyclists. It has been shown that having cycling lanes will reduce the number of crashes “The overseas research indicates that the number of crashes decreased when on-roadway cycle lanes were installed; the reduction of cyclist crashes generally varied from 35% to 50%, although one source did report an increase in cyclist crashes. Total (cycle and motor vehicle) crashes were found to decline by 6.5% to 35%.”[3].

Many different variables could come into play that could affect the readings that our device could gather, from weather to the cyclists’ competence and even the type of vehicles that pass the cyclists. Even the colour of the cyclists clothing can affect how close of a proximity a vehicle passes by them as shown from the graph down below.



**Figure 1: Mean overtaking proximities (black lines) and kernel density plots for each bicyclist outfit, with individual data points shown in rugs. The dotted vertical line is the overall mean for all data (117.50cm)**

**1.4 Options**

**1.4.1 Microprocessors**

Microprocessors such as the Arduino and Raspberry Pi have proved too useful for prototyping   many different systems. Their size makes them portable, which is important for fitting them onto a bicycle or cyclists. Both systems are capable of using analogue and digital input and output. Both systems can have also have external battery packs or power sources attached to them. We are using an Arduino for our prototyping because we already have some available to us and also we have experience in using an Arduino but we have never used a Raspberry Pi before. The Raspberry Pi is also more powerful than what we need it for currently.

**1.4.2 Sensors**

The system would need a way to detect if vehicles are in too close of a proximity to the cyclist. There are several different types of sensors that could help us achieve our goal of distance detection.

One of the first devices we looked at was an infrared distance detector. While infrared sensors are generally a lot more accurate, they have several downfalls which make it not viable for our project. Most infrared sensors have a very short maximum range. “For instance, the SRF05 ultrasonic sensor can clearly detect objects up to 4 m away, whereas the Sharp GP2D12 infrared sensor that we’ll use in this chapter has a maximum range of 80 cm.”[2]. Our product will also be mainly used outside, which means that light will play a major factor in the accuracy of infrared detector. “It relies on light, so in bright direct sunlight, infrared sensors often won’t work well, if at all—sunlight will saturate the sensor, creating a false reading.”[2]. Because of this reason we have decided against using an infrared sensor on our device.

The other way of detecting distances we came across was sonar sensing. This method of distance measuring seems the most sensible, as the sensor can detect much greater distances, it will be easier to use outside because natural sunlight will not interfere with the readings. Problems with using sonar would be it is not as fast as infrared, therefore not as accurate. ”It works much more quickly than ultrasonic sensors because, as your intuition might tell you, light travels more quickly than sound. That means the danger of interference is much lower.”[2]. If needed we have also considered a combination of using both infrared and sonar sensors.

**2 Solution Overview**

The solution to solve the DCCs problem of knowing the dangerous areas in Dunedin was to develop a device power with a microcontroller with a sensor and a way of storing this data. Our system works by a user attaching the device to their self or the bike, the sonar sensor is constantly pinging to see if it detects an object in front of it. The device only records if an object is in front of it is between a thresholds of 30-150cms. The instant that it detects an object, the device sends the distance that it recorded via Bluetooth to an app on the user's phone. The phone app notes what time, date and GPS location that the ping was received. This data is stored on a SQLite database on the phone. When an internet connection is available the user can then send the data to a web/ftp server. This data can then be viewed on a map on a web page.

**2.1 Device**

We decided that best way to measure how close cars were passing the cyclists would to use an Arduino powered device with some kind of sensor. At the start we have had an abundance of Arduino and sonar sensors available to us. We have two current versions of our solution in use at the moment. The oldest solution that we have in current use is our first prototype. The prototype is an ATmega 328 Arduino mega microcontroller with an HCsr04 and a Bluetooth Low Energy module. Our final device is using a 5v Pro Nano Arduino with a Maxsonar M1240 sonar sensor, a Bluetooth serial module and is powered using a 9 volt battery. When an object passes in front of the sensor within the threshold of 30-150cm, the distance is put into a buffer and sent over Bluetooth to an Android app. We also included some features to try and cut down on false readings such as when the user passes a pole. We did this by not counting an incident unless the object has been in front of the sensor for 6 pings. The devices pings constantly so this allowed for us to cut out smaller objects.



**Figure 2: First prototype**



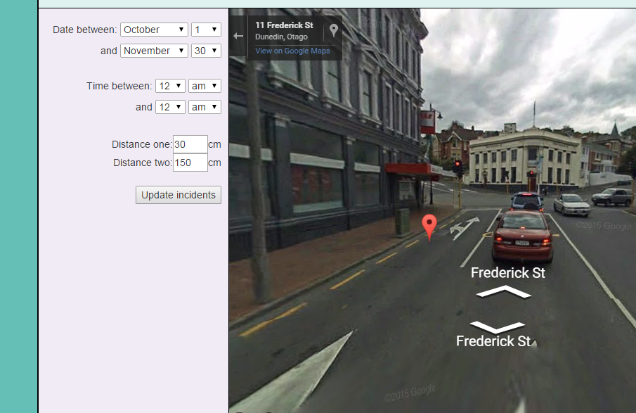
**Figure 3: Final device**

**2.2 App**

The purpose of the android application is to log, view and send the information gathered from close passes from vehicles. The application can pair up with the Arduino through Bluetooth. When it receives a distance reading of a passing object through Bluetooth from the Arduino, it gets the current GPS location and stores it with the distance and a timestamp. This information can then be viewed by the user as markers on a map. The map was implemented using the Android Google Maps API. The users can view more detailed information about the incidents by tapping on a marker which activates a dialog box which contains the gathered information. The dialog box also gives the user an option to delete the incident if they know that the incident is a false reading. An example of false information would be the cyclist passing an object closely that wasn’t a car. The information gathered can also be sent to a server as a CSV file through SFTP.

**2.3 Website/server**

The purpose of the website is to allow all of the information gathered by users to be available to the public and our client. We included details on how the project works on the website but its main purpose is to see all of the close passing incidents collected by the users. We have the information shown as markers on a Google Map using the API Google made available. The data can be sorted through date, time and passing distances. We also found that we were capable of using Google street view from the API. This is helpful for looking at how the street is laid out and how an incident could have occurred there. The markers are still shown on the street view so you can see exactly where the cyclist was at the time of the incident. The picture below shows this, the cars shown in the street view aren’t the cars that got too close to the cyclist.

**

**Figure 4: Street view with marker**

**3. Development process**

**3.1 Development process**

We followed the Agile Development Framework. The Agile Development Framework is an iterative approach to software development which embraces change and places an emphasis on client communication and feedback. Implementation consisted of three iterations, each increasing in length compared to the previous iteration. At the conclusion of each iteration a deliverable is expected to be produced. Throughout the development process constant client communication and opportunities for feedback have been utilised. We have a decent working relationship with Kylie Huard, the client at the DCC.

Our working relationship within the team was an important process in order to achieve the end goal. We have had constant communication in our team to let each other where we are in terms of progress, our current problems, and our future goals we need to achieve.

The source code for the project was managed using Github. Github has been used for version controlling our code, as well as a backup for a current code and documentation. Github allowed us to branch and separate our code for testing.

The first iteration of the project we focused on the equipment needed or the ideas behind build a system that could be fitted to a bike and provide the data we need. We spent a lot of time testing out how sonar sensors worked and seeing if we could capture live data. We also tested the feasibility of the theory of using wireless communication to communicate via radio. During this time we were also testing the feasibility of storing data on Arduinos and alternative modes of storage.

For our second iteration we mainly focused on moving our project in a different direction in terms of how the data is stored, how GPS coordinates are retrieved and how the data would be retrieved off of the Arduino. After the last iteration we discovered the SD card writer/reader module conflicted with other components of the system. We came up with compromise and used a Bluetooth 4.0 module to send the data to an android phone. Instead of using a separate GPS module for the Arduino we decided to use the resources that an android phone already provides. At the end of the second iteration our system was ready for collecting real data from the city of Dunedin.

The third iteration was largely focused on getting more people to use the system and user testing. We decided that the data that the system was retrieving would more valuable if the data was easily viewable via a website or web application. Using user stories and user interaction diagrams, a website was developed and the collected data could be viewed on a map showing where all incidents took place. During this period of time we were still focused on keeping the devices in use and constantly capturing data.

**3.2 Platform/technology justification**

**3.2.1 Bluetooth/Bluetooth LE**

As part of a means of capturing when a vehicle has had a close pass with a cyclist, we decided to use Bluetooth 4.0 or otherwise known as Bluetooth Low Energy (BLE) to send the captured distanced to a phone. The name of the module is NRF8001. Our decision to use BLE first was due to the fact that its power consumption at peak power usage is 14 mA. Our second device uses serial over Bluetooth for connecting to a phone. While using BLE means lower power consumption, BLE is also a new technology that all phones do not have implemented. We made the decision to use Bluetooth serial for our second device because it is supported by a wider range of mobile devices than BLE. The module is called a Bluetooth Silver Mate, although this current model is not available anymore, there is instead the Bluetooth Gold Mate.

**3.2.2 Sonar**

We used sonar as a way of measuring how far vehicles are from the device. We did consider the use of infrared sensor, but the readings from the Sharp 2Y0A21 seemed too inconsistent and inaccurate.

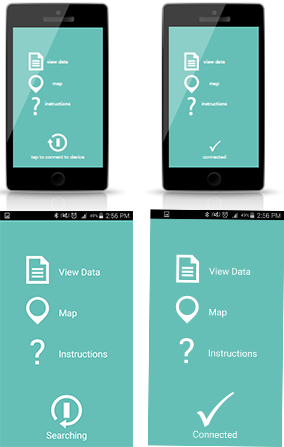
**3.2.3 Android**

We had originally intended to have only the Arduino and have everything store on there as well as a GPS chip but we ran into problems with the SD card reader. The major issue we had with the SD card reader is that it would regularly be unable to read the SD card which means that it wouldn’t be able to write our data in and we didn’t want this happening when in use. Our solution to this was to have the Arduino link up to the phone and have everything logged on a local database. We found this to be a good move because Android covers a lot of the functionality that we needed and it meant we didn’t need to include it on the Arduino so we could simplify that. The features of Android we were able to put to use were the in-built GPS, ability to have local storage, timestamps, Bluetooth functionality and its internet connection so we could upload the data once it had been collected. This solution also allowed the user to view their own information they had gathered more easily rather than having it mixed in with every other users on the website.

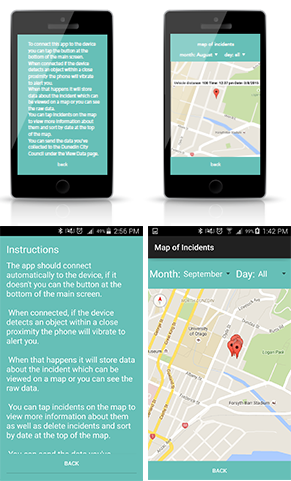
**4. Artefact description**

**4.1 interaction design**

We came up with the interface for our mobile application with help from an intern from Design Innovation Workspace, Paul Charters, who was able to come up with a slick and easy design for us to work with. The design proved to be easy for our users to understand. We wanted the main focus to be on having the app easily connect to the Arduino with little effort from the user. We did this by having the connection happen automatically on the first screen of the app when it opens. Our user feedback for this was good as they liked that they could just open the app and have it work without them having to do anything.



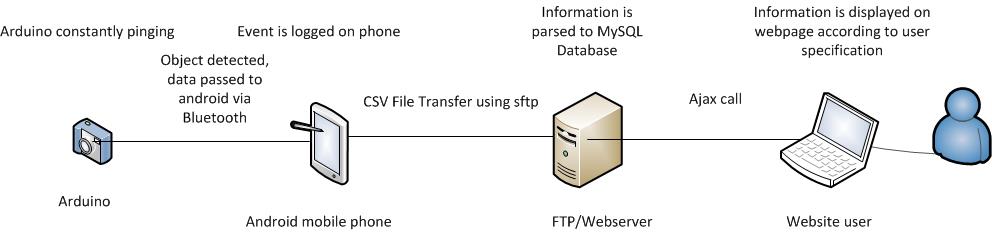
**Figure 5: Design top, implementation bottom of main screen**



**Figure 6: Design top, implementation bottom of instruction and map screen**

**4.2 Software/system architecture**

When the sonar sensor on the Arduino detects an object within the threshold of 30-150cms, the Arduino sends the distance information to the user's Android phone via Bluetooth. The distance data is then matched with the time and location that the phone received and stored on a local SQLite database. This data can then be sent off to a ftp server/web server. This data is stored as a CSV file. This is then parsed to a MYSQL database using a Python script. The information can then be viewed from a website.

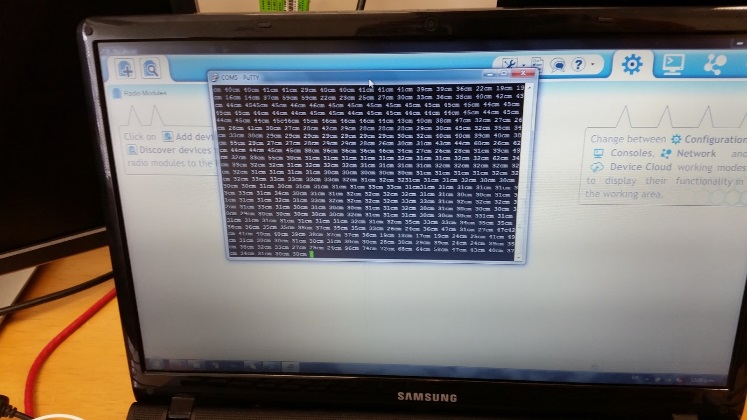


**Figure 7: System architecture diagram**

**4.3 Functional Requirements**

*The system must detect how close vehicles are passing the user*

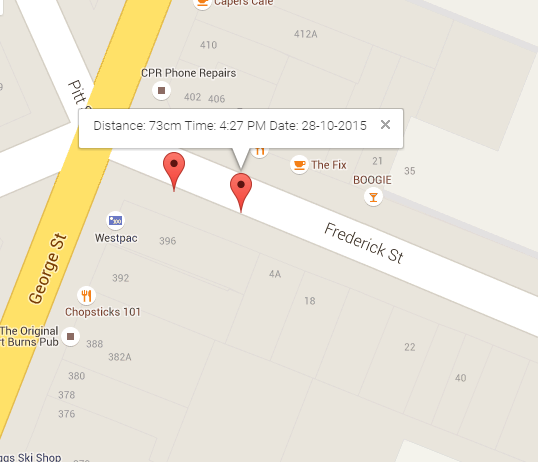
This was done by using a sonar sensor and it was accurate with measuring how far away objects were.



**Figure 8: Shows readings taken from the sonar and sent wirelessly to a computer**

*The system must record where close passes have happened*

The system is able to record the location of close passes by getting latitude and longitude coordinates from the Android devices built in GPS.



**Figure 9: User incident displayed, shows location, distance, time and date**

*The system must record when the close pass happened*

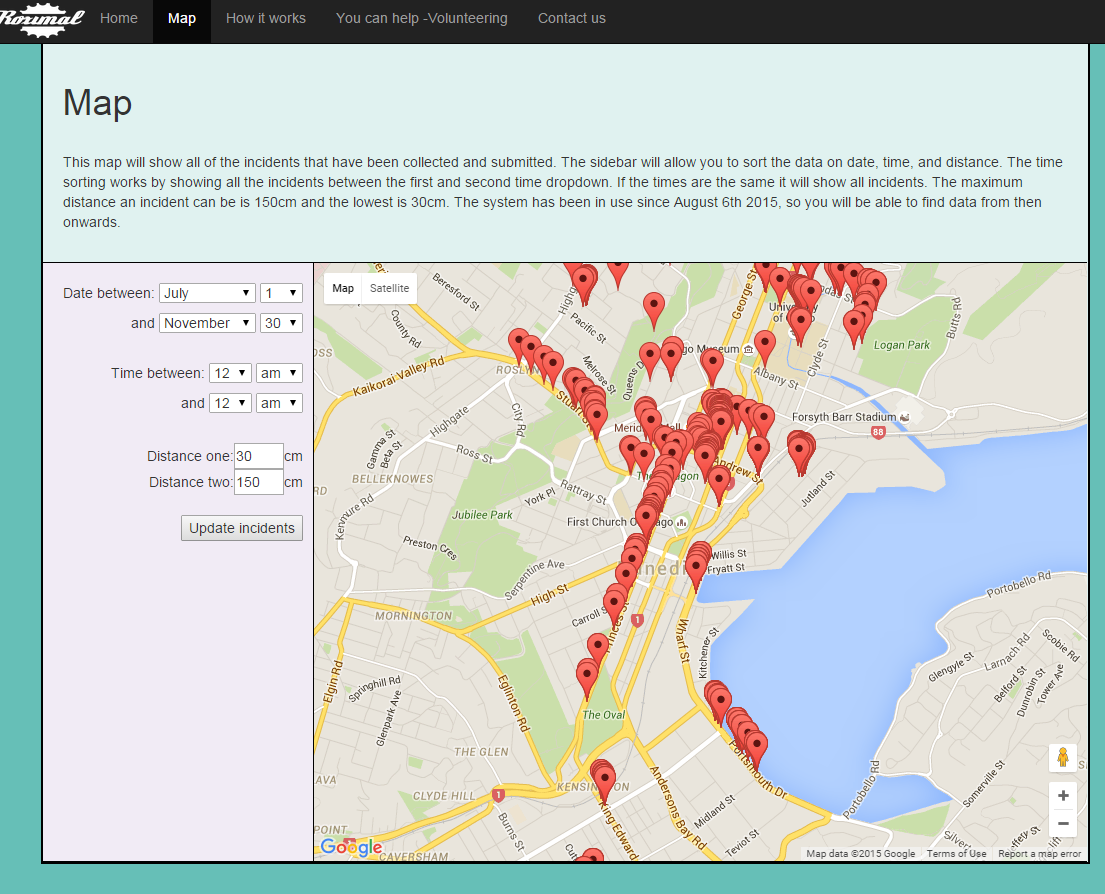
The system can do this by using the phones clock to get a timestamp of the current time.

*The system must have a way of getting the data to a meaningful place*

This is achieved by using the JSch library for Android. It allowed us to SFTP a CSV containing all of the incidents information to the server.

*The system must read the data*

The system can read the data by using a web page. It reads the data using an AJAX call to a PHP file that reads the database so we can easily update the information. The information is displayed as markers on a webpage to show where all of the incidents for all the users have been taking place.



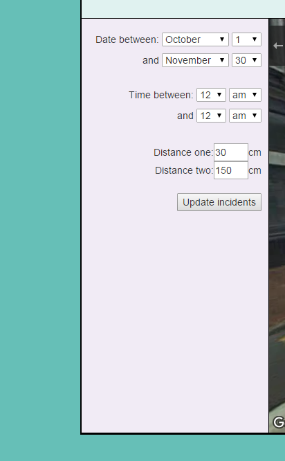
**Figure 10: Displaying the collected data on a webpage**

*The system must present the data in a useful way*

This was not achieved, because our client did not provide feedback on layouts that we previously presented to her, but then later commented that they found that the way we had presented the data was not useful to them. She said it wasn’t useful because of the amount of junk data. We didn’t have enough time left at that point to try and find better ways of cutting out the junk data.

*The presentation of data must be changeable*

This was done by allowing the data to be sorted by the side bar on the side of the map, as shown in the figure X. The data is sortable by date, time and distance.



**Figure 11: Sidebar for sorting**

**5. Technical Highlights**

1. The android application we built was capable of using Bluetooth Low Energy and normal Bluetooth connection so that it was accessible to a wider range of devices as older devices don’t support Low Energy.
2. Getting successful communication between the Arduino and phone via Bluetooth
3. Being able to have the location of the close encounters logged on the phone and visible using the Google Maps Android API.
4. Successful parsing of phone data to MySQL database

**6. Testing**

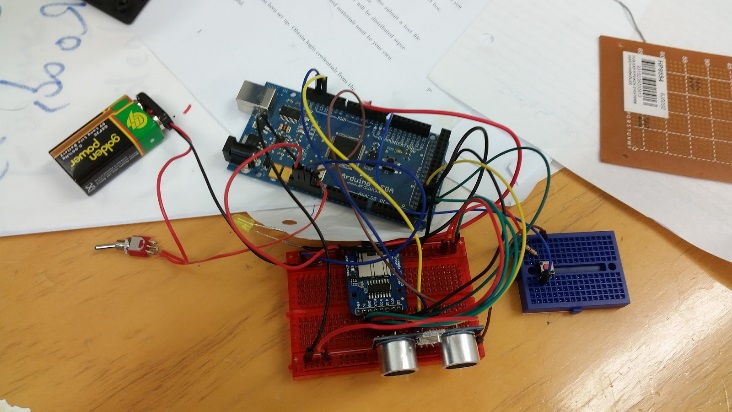
**6.1 Development Testing**

**6.1.1 Sensor testing**

Early on our testing was focused on determining the best method of getting distance readings and storing the data we were going to be collecting. We started with testing a sonar sensor and found it was fairly accurate for what we needed. We tested an infrared sensor as well but found that it had too many issues. The infrared sensor was accurate up to 50 cm, distances over this proved to provide wildly inaccurate results. This was due what we believe to be refraction and reflections depending on what the surface of the material was like that was used to test. Acute or obtuse angles proved to create inaccurate readings for all of the sensors because the sonar ping did not reach back to the transceiver.

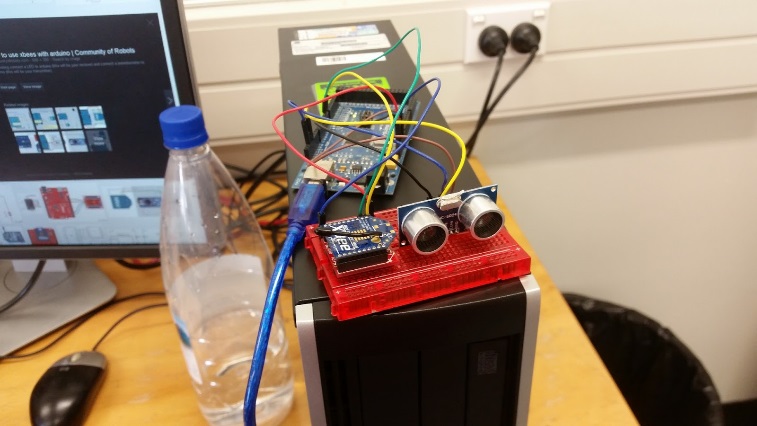
**6.1.2 Data storage testing**

For storage we had everything stored as a text file on an SD card that was attached onto the Arduino. We were able to store data on this but it had its limitations. We weren’t able to get it to log more than one file at a time so we had to append onto files we had already created. We also had issues with getting the SD card reader to read the card which made us look for alternative methods of storage which led us to using an Android phone to pair with the device.

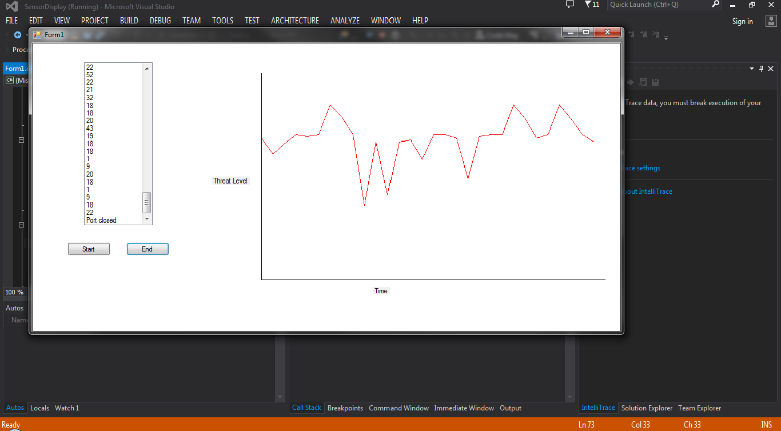


**Figure 12: Arduino with SD card reader**

We also set out to test if we could find a way of wirelessly transmitting data from the Arduino, as we didn’t originally plan to use an Android we used XBees as a method of transmitting data wirelessly. We had the distances the sonar was reading in transmitted straight to a computer with a program to read it in and chart it out shown in figure 13 and 14.

****

**Figure 13: XBee connected to Arduino**

****

**Figure 14: Graph of wireless data received**

For this testing session we wanted to tighten up the accuracy of our current physical prototype. We wanted to see if the device could accurately count the amount of people that passed in front of it. This is a proof of concept, if the device can count people accurately, then if should be able to count larger objects such as cars accurately too.

The test consisted of people walking in front of the device as many times as they liked, and us counting how many times someone crossed in front and seeing if the device had the same count as us.

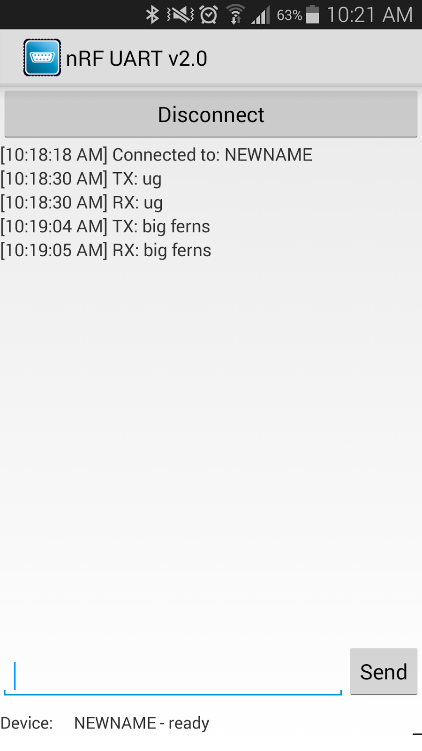
When we initially started testing, we noticed the device was counting people, this was due to us switching libraries that the Arduino was using for the sonar sensor. The device was giving a different timeout default than we expected, which lead to a bug in the code.

After we fixed this bug, we then started testing again. We counted that 62 people had crossed in front of the device, but the device counted 213 people.



**Figure 15: People passing in front of the sensor**

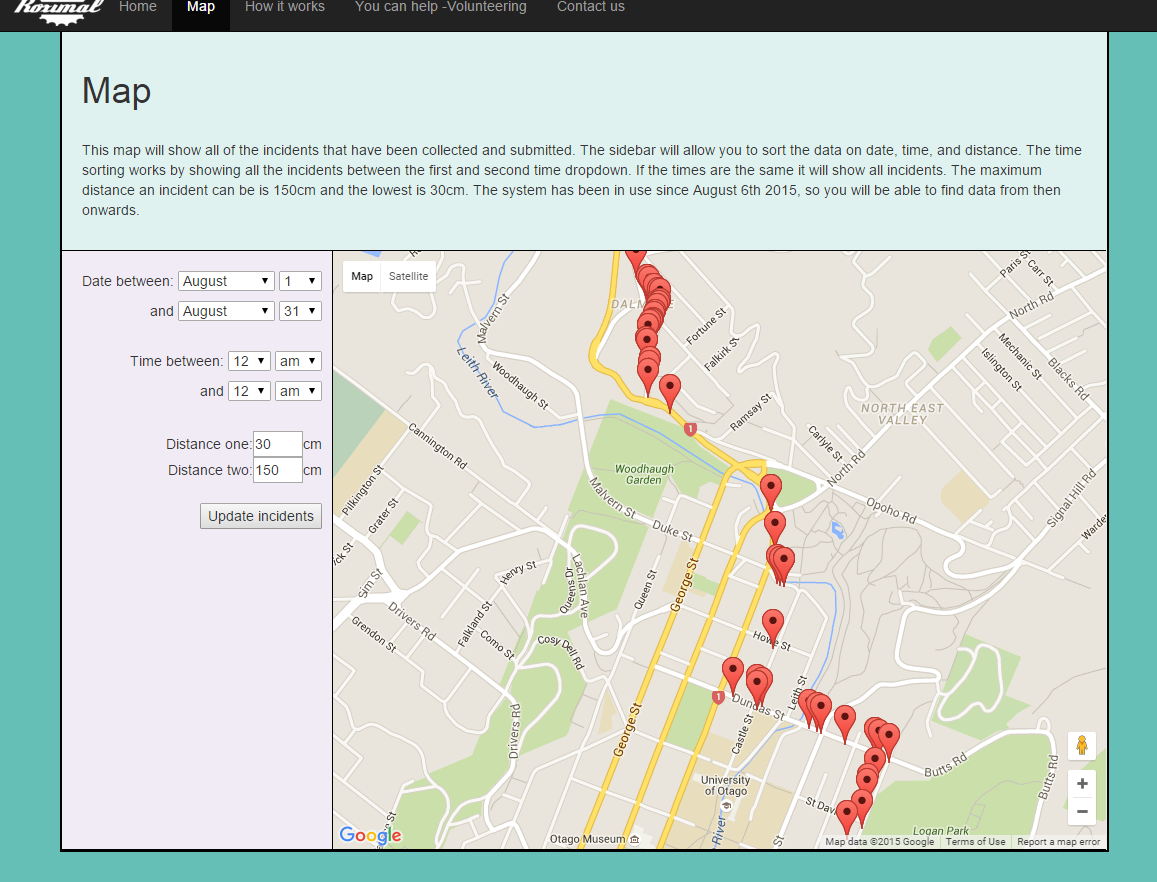
We tested the feasibility of using Bluetooth as a method of connecting the phone to the Arduino. We did this by wiring in a Bluetooth module into the Arduino and using an app that already existed called nRF UART. And set up a simple test by having the connection work as an instant messaging system. The testing was successful as we could send messages back and forth between the two devices so this allowed us to continue forward with developing for Android because we knew that it was possible to achieve.



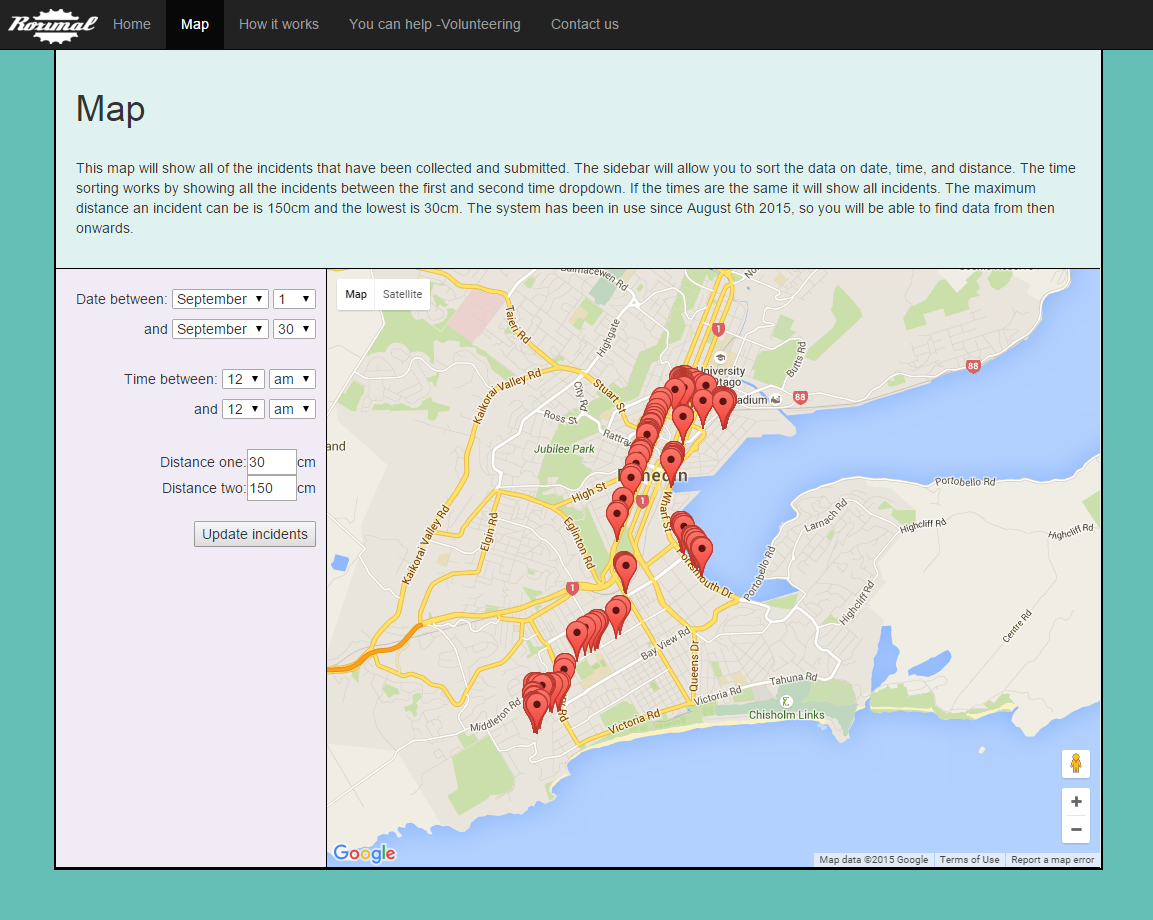
**Figure 16: Messages sent between the Arduino and Android**

**6.1.4 Prototype testing**

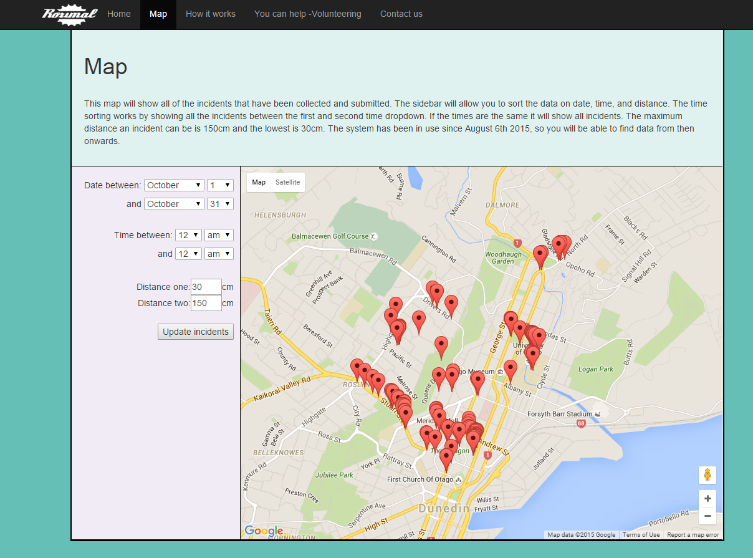
We first started our prototype testing at the deployment of our MVP. The testing consisted of having the device and app in real world use as much as possible, with as many different users as possible. Four different users have used our system for different lengths of time.



**Figure 17: First user testing**



**Figure 18: Second user testing**



**Figure 19: Third and fourth user testing**

We wanted to prove that our device was actually capturing real evidence, that it could actually detect vehicles from varying distances in different scenarios. We tested our prototype by taking a car and a bike to an empty parking lot and testing various distances of passing to see if the device would behave in the way we thought it would. The maximum distance set on our device is 1.5 meters. The app was modified so that it would be when an object was in the threshold, this proved that the device detected the car on video.  We tested passing distances of 0.5, 1, and 1.5 meters and beyond 1.5 meters. The device captures incidents how we had planned and was able to capture all incidents in the 0.5 and 1 meter range. The 1.5 meter testing results varied but it could be due to the fact that we may have gone further than 1.5 meters on some passes as it can be hard to maintain a straight line on a bicycle.



**Figure 20: Setting up the lanes**

****

**Figure 21: Test video 50cm pass parked car**

We are still yet to do user interface testing for the app. We are going to come up with a list of tasks for users to complete for the app and see how easily the users are able to complete the tasks.

**7. Deployment**



**Figure 22: User on a bike with the device**

The Dunedin City Council will be unable to continue with this project once we’ve finished as they don’t have the funds or the staff to work on this further. The Dunedin City Council is still interested in the results however, so we have we have a representative associated with Spokes who is willing to take on this project. We are still yet to meet with him but we will be able to hand over our code base, and devices that we’ve created. We will provide documentation and instructions on how it all works and what would be required to continue on with the project.

**8. Analysis**

A big problem for Dunedin is how dangerous cycling is. We have helped address this problem by mapping out the dangerous areas in Dunedin by using sonar distance measuring device linked to a GPS logging android application. From the limited amount of use our system has had, we were still able to see some trends in roads that are dangerous to cyclists.

From the data that has been provided by our user testing, we have been able to come to conclusions about different areas of Dunedin. The most common road that has had multiple close calls in has been on George St/Princess St. Most

We have developed a device that is capable of measuring distances of passing objects from cyclists. An Android application that can communicate with the device through Bluetooth that can also log information of incidents and send them to a server. We also have a webpage that is used to display all of the user collected incidents to identify dangerous roads, and can also be used to help the public identify which roads could be avoided until further improvement.

The main driving factor behind making this project successful and useful is having more and more people actively using the system. The more people that are using the system means that there will be a wider and more diverse set of data that is added to the system and a better coverage of Dunedin in terms of landmass.

**9. Future work**

We are striving to keep our system in use and development.  If we had more time given to work on the project we would like to implement more into the system.

**9.1 Device**

* Integrate a rechargeable battery system
* Fully design a robust case that is waterproof and resistant to damage
* Have a fully designed attachment system that could allow for easy hot swapping of device onto different bikes

**9.2 Mobile App**

* Integrate more features to clean out junk data
* Support for IOS devices

**9.3 Webpage**

* Have better solutions of eliminating junk data
* Have ways of analysing the data to find the dangerous areas

**9.4 Server**

* Further tightening of security
* A fluent and dynamic way of adding cycling data to the MySQL rather relying solely on a cron job

**10. Conclusion**

We have developed a mobile system and a backend required to address the need of gathering evidence if the dangerous roadways in Dunedin for the DCC. We have addressed this by providing: a system that can detect the passing distances of cars and record when and where this happens, a place that stores this information and a way to review the incident data.

Following the Agile Development framework methodology has allowed us to embrace change and allow for feedback and constant communication with their client. Testing and documentation has proved as useful tools for helping direct the project to provide a helpful and useful tool for finding dangerous roads in Dunedin for cyclists.

**11. References**

[1] Safety, C., Draft, P., & Document, C. (2014). Safer Journeys for people who cycle.

[2]Arbenz, M. (2011). In Action, (November 2010).

[3]Walker, I., Garrard, I. and Jowitt, F. (2014) The influence of a bicycle commuter's appearance on drivers’ overtaking proximities: an on-road test of bicyclist stereotypes, high- visibility clothing and safety aids in the United Kingdom. Accident Analysis and Prevention

[4]Dunedin City Council. *Cycling*. Retrieved from http://www.dunedin.govt.nz/services/cycling