

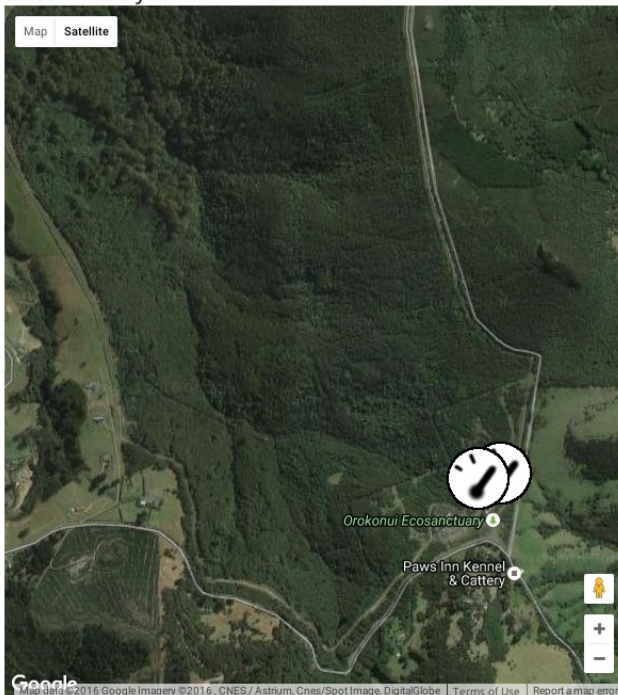
# Orokonui Monitoring

NSN: 13024979  
Digital Technology Scholarship 2016

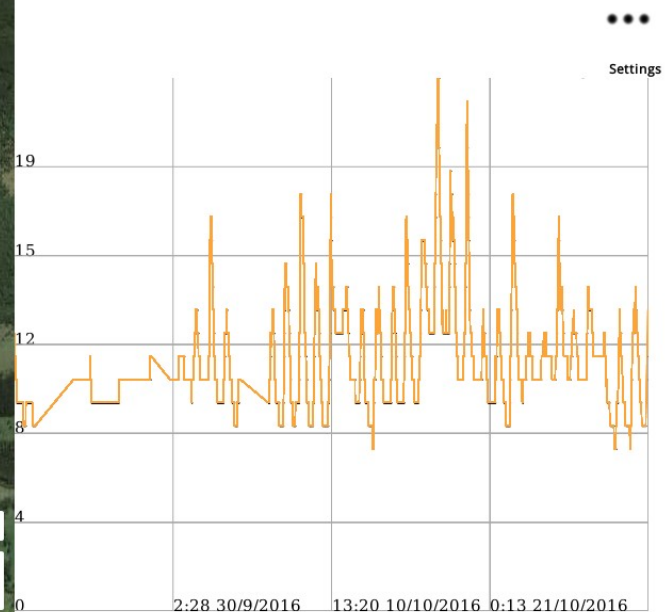
A system for data collection, presentation and action.



Select where you would like to see data from:



Temperature, vs Time, for Workshop  
Temperature (-45.7765251, 170.6048705)



## Introduction

---

### **Back story**

This project is about monitoring certain aspects of the Orokonui Ecosanctuary. The project was initially brought to my attention by my teacher, however, as of that time I did not have a lot of information about it.

I then contacted Andrew Hornblow, who is not directly involved at Orokonui, but had run an Internet of Things workshop, which Tony Stuart, a volunteer at Orokonui had discussed gate monitoring with.

From this initial communication, I did not have much specific information on what was needed, so started with designing generic solutions, and then started narrowing down ideas and specification once I had a proper meeting with Tony Stuart, who is a key stakeholder of my project.

Because of this approach, my solution is by nature modular, with certain areas adapted for Orokonui. This allowed me to develop a flexible solution that could foreseeably be deployed for other projects, making it a solution that could be used in other applications and situations such as home monitoring and automation. The final result is that my solution is remarkably multi-purpose.

### **Internet of things**

Internet of Things, or IoT, is the idea of connecting 'things' to the internet, for the purpose of data collection, remote control and automation. In this project, I connect gates – which control access to the endangered kiwis – to the internet, for the purpose of data collection, and protection from being left open. IoT is a relatively new idea and use of technology, brought about by the decreasing price, size and power requirements of digital electronics. However, since this is quite new, a lot of the protocols, infrastructure and frameworks have not yet been setup.

Due to this, I looked into many different methods of implementing IoT, and decided to make my system modular. The advantages of this are that should one of the technologies I'm using be made redundant by a newer, better alternative, it will be able to be swapped out for a new one, with all other parts (modules) still working.

### **Note**

The rest of this document was written as time progressed. For this reason, information on a particular topic is scattered throughout the document, rather than being grouped together. This way, readers find out information in the same order as I did. This will become evident throughout reading this documentation, and demonstrates the Agile development process which I deployed.

## Initial Brief

---

### **Issue:**

My client needs a system that will keep track of gates and other environmental conditions at the Orokonui Ecosanctuary. It will need to be able to upload this data to a central database, and there needs to be a user friendly way to access that data. This is important because this data allows the rangers to keep better care of the bird and wildlife at the ecosanctuary, especially the Kiwi. For example, they will be able to see that the tuataras' enclosure is at the correct temperatures and that the kiwi gates are properly closed. A central server gives single place for the data to go, without having it spread across multiple devices with possible missing information.

### **Impact:**

This project is important, because it will help Orokonui better monitor, protect, and preserve endangered New Zealand birds, including the kiwi, which they have a breeding program for. The impact will be both on conservation, due to advanced warning systems if something damaging were to occur, and research into these endangered species, by allowing analysis of the data I propose to collect.

An additional impact I plan to have through development of this project, is to develop software solutions for and within the open source world. I intend to add this product to a rich repository of free and open source software, in order to align with my ethical values.

### **Environment:**

Orokonui is an Ecosanctuary about half an hour outside of Dunedin at 600 Blueskin Rd. It is a major tourist attraction, with many native species of bird protected by a large perimeter fence.



### **Stakeholders:**

My key stakeholders are Andrew Hornblow and Tony Stewart. Andrew has connections with Orokonui and Internet of Things (IoT) knowledge, however is not based in Dunedin. He will be providing consultation over email, and will be in Dunedin at various times throughout the year. Tony is a volunteer at the Ecosanctuary who also has a background in computer science and electronics. Tony will be mediating between me and the other Ecosanctuary volunteers and staff. Through him I am able to find their needs for the project. He is based in Dunedin, and we will be consulting both in person and over email. My client overall, is the Orokonui Ecosanctuary as a whole.

### **Initial Communication:**

Andrew had several ideas for me to explore in relation to Orokonui's needs. Our initial communication is as follows:

" ...

I think there are a range of ideas you could work on so will just put a few starters out there for now:

- Pi based server for the sanctuary that can pick up the radio data from tracking tunnels, gates, traps and monitoring systems
- A system that can push this data up to a server, site, that can be used remotely from anywhere to monitor things
- Very simple to use graph to monitor just about anything... (ask for examples I have on line at present on an expiring system...)
- Systems that can raise SMS txt / E-mails / Alerts via Twitter (look up Twython.py) when things happen in the sanctuary
- Instrumentation that can monitor the traps. This is a bit more hardware, electronics and practical and in the early stages still
- Advanced radio auto forming/healing mesh to bring data back to the nearest www access

..."

From this, I decided to start by developing a website framework, which can receive data and form graphs and interfaces based on this data, as well as sending tweets or SMSs about this data, which covers points 3 and 4. I started at this end because I didn't know yet the details about the specific ecosanctuary needs, and this part would be needed regardless of what else was implemented in terms of monitoring.

## Context considerations

---

### **Environment**

I will be monitoring outdoor areas, this means all my solutions will need to be weather proof. I will need to think about casing. Additionally, I will need to consider how to get data from the sensors to a central system, in an outdoor environment where network wires are impractical, due to the distances involved. This means I will need to test radio range, frequency propagation over terrain and how weather, and/or other radio interference will impact reception. I will have to create my own wireless solution, as cellular coverage is sketchy at best, only Spark has coverage at Orokonui and using their network would charge a high usage fee.

### **Cost**

Orokonui is a non-profit organisation aimed at protecting and preserving natural New Zealand wildlife and birds. This means they do not have a large budget for IoT devices. I will be trying to find the cheapest option, which can reliably function in the environment for is desired task. This will raise some cost verses reliability questions, which will need to be considered throughout this project.

### **Usability**

My system needs to be usable by non-technical people. The people at Orokonui, and other ecosanctuaries and environmental organisations tend to be less technically orientated. For it to be of any use to them, it will need to have a user friendly interface, and be easy to maintain.

### **Ethical concerns around software development**

For true innovation to occur, development must be open. For example; if Newton were to figure out how gravity worked, but not share his discovery, how could Einstein then work on his theory of general relativity – improving on the work of Newton. The same is with software. I am standing on the shoulder of giants, and would not know the first thing about how to start with this project, if I hadn't already been not just in the software world, but specifically the open source world. I am, and will be using an array of open source software in this project. For these reasons, I would personally consider it unethical for this project to be licensed under any other type of license. Similarity, all of my non-software IP, including this document will be licensed under Creative Commons By attribution, Share-alike (CC-BY-SA 4.0) provided I remember to add a copyright notice due to the existing system of 'all rights reserved' default.

## Graphing

---

I will need to develop a way to graph the data I collect, because graphing visibly shows trends in data over time, which is a useful way for the Orokonui staff to analyse the data collected, and thus enhances the usability.

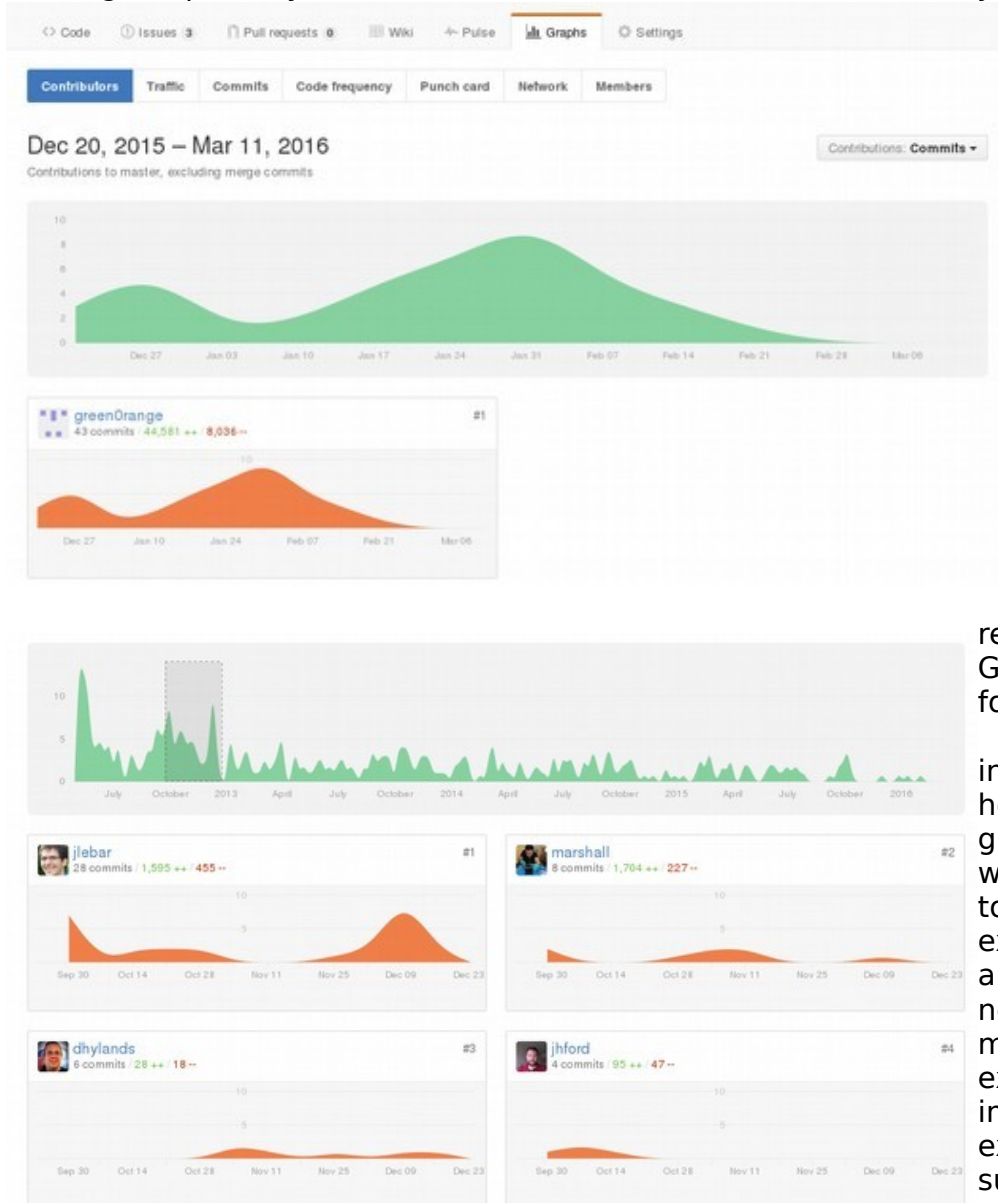
This graphing needs to be dynamic, as the data collected will be constantly updating as time progresses.

### Research into Existing Solutions:

Some examples of existing graphing/interface solutions are Github and YouTube. I researched these to see how they designed it, so that I could take inspiration.

### Github

Github has a simple interface in which the user can simply click 'graph' while viewing a repository to see the commit data set out in a visual way:



They can then click which sort of data they want to see, the upper options. (contributors, traffic, etc.)

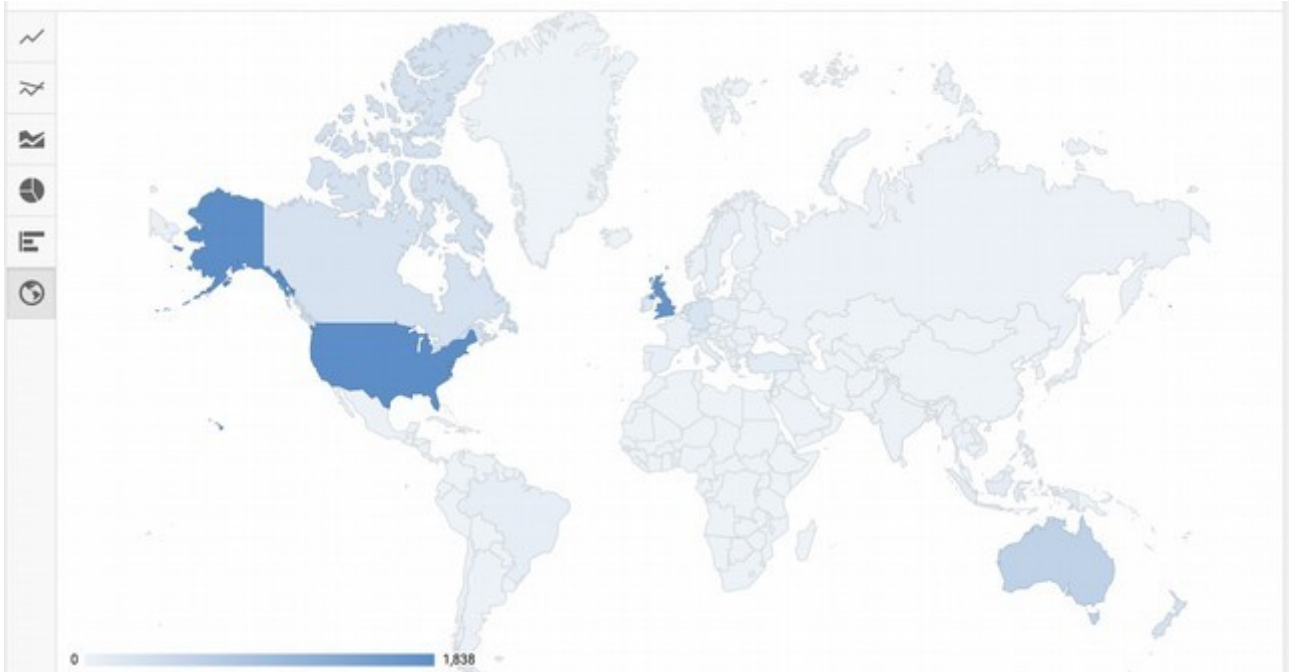
From here, you can highlight certain parts of the graph to see different people's commits during this time frame. Here's an example from Mozilla's B2G repository (Boot to Gecko, code name for Firefox OS.)

This is a very intuitive system, however it does not give the detail I would like. It is hard to select the exactitude of dates and times I would need for Orokonui monitoring. This exactitude will be important for seeing exact date ranges, such as seeing how many times gates

were opened on a certain day, or tracking down causes.

## YouTube

The other example I looked at was YouTube's analytics, which is essentially Google analytics with a different colour scheme. The thing that really interests me about this, is that it shows playback locations by country. This is something that could be implemented in this project, since each sensor has a different location, which would be interesting to compare. Here is a screen shot of how it handles locations:



Countries can then be clicked to find out more details about that country, such as views, watch times, referrers, etc. (not specific locations, that would be creepy.) It can also be shown as a bar, pie, area, and line graph, as seen in the side buttons.

## More Context Considerations

---

### Software and Server

It is important to look into software types, I will be unable to programme my own software from scratch. Software is required to transport and process data, although I will be writing custom scripts to do specific processing, I must choose a language, and base platform on which to do this.

Since I am dealing with data, and showing that data over long term cycles, I will need some place to store this. It also needs a way to be fetched and sorted quickly. Because of this, I will need a server, which can store and process data, then serve the data to on request. The best way to implement the fetch of request is to have a web server. This means it will serve up a web page, rather than have an application call a specific API in the server. Also every computational device has a web browser built in, that will be able to view the web page, whereas if I were to use a specific application, I would have to create that application for every operating system, desktop and mobile, etc. Having a web server is far less maintenance, and setup.

I will be using LAMP, (Linux Apache MySQL PHP) as it is the easiest and cheapest (free) option. Other options I have considered are Windows Server, or an Apache variant, such as mini-httpd. The problems with Windows Server are it's cost, and my lack of knowledge about Windows. The Data Centre Edition of Windows Server 2016 is \$6,155 and the Standard Edition is \$882, which is well beyond Orokonui's price range. Additionally, I am already a Linux user, and find Windows difficult at best. I get easily frustrated at it's non-unified installing system / lack of a package manager and

confused by its registry system and file tree. Therefore, it would be best if I put my server on a Linux base. Additionally, Linux is generally considered to be extremely stable, especially when run minimally. This will create a reliable base system for me to build upon. Since I am only hosting a website, there is no need to run the X11 server (windowing/ display system), which is probably the most unstable part of a Linux system.

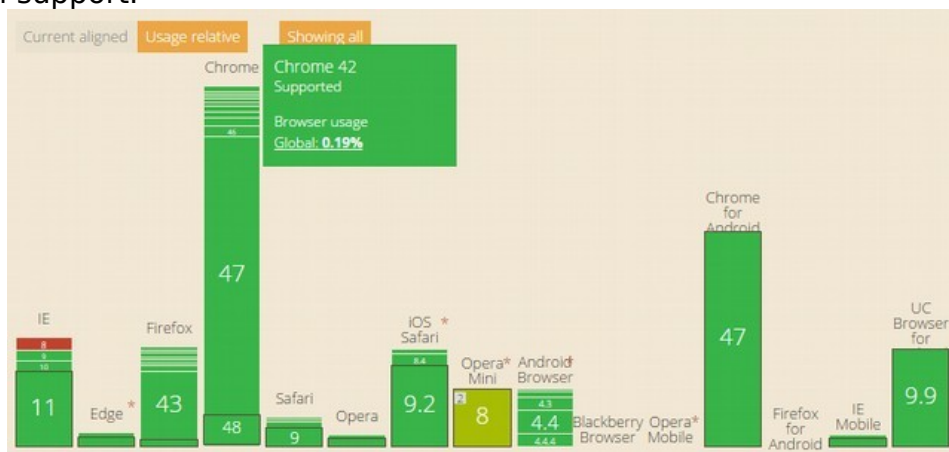
On top of this, I will need a web server. Mini-httpd, is one option, but designed for extremely small scale operations, commonly used to host internet router setting pages (e.g: 192.168.1/0.1) or the like. Apache is best large-scale web server software, that is free, Linux compatible and open source. MySQL is a database system, commonly used all over the internet for it's (relative) simplicity and freedom. It is also free. PHP is a server side scripting language, with lots of security criticisms. However it still dominates a lot of the internet and integrates well with MySQL. The security criticisms will not affect what I am trying to do, since, I do not require any login, personal, or protected information. I have used PHP before, and know how it works and what it can do.

I will also be using HTML, since it is the language of the internet, and any other mark up or mark down languages will encounter issues with most, if not all, browsers.

I will be using HTML5, rather than Flash player to create the graphs, because Flash player is out dated, slow, resource intensive and insecure. At one point, Firefox blocked flash by default, because of security risks for an entire month before Adobe fixed the risks Mozilla was concerned about. I don't want to build something holding on to end of life technology, as this decreases the life cycle of my project and I am aiming to future proof it as much as possible. As well as these security risks Adobe Flash has a 3rd party add-on, which decreases performance, whereas HTML5 canvases and JavaScript are built in to all modern browsers. Flash is also only supported on desktop, whereas modern smart phones have HTML5 enabled browsers.

However, Internet Explorer 8, which has a 1.02 global usage share, does not support HTML5 canvases. Additionally, Opera Mini, only partially supports it, unable to play complex animations. This is not relevant as I will not be using these complex features. Also, this is not very relevant due to the low usage my site will have, and the Orokonui staff all use browses which do support HTML5. It is unlikely future staff/users would use outdated browsers.

The following chart shows browser by global usage, and whether they support HTML5 canvases, represented by colour, red being no support and bright green being full support.



(Image credit: <http://caniuse.com/#feat=canvas>)

Flash player is installed on 98% of computers, according to Adobe, so take that with a [large] grain of salt. Therefore if I use flash, 2% of users will be unable to view graphs, however, if use canvases, 1.02% of users will be unable to use it. Additionally, when showing IE8 usage from only New Zealand, the usage decrease to 0.36%. Good on us for not using bad browsers!

Another way to show data is rendering the graphs on the server, and sending an image. However, this causes more stress on the server, and cannot be interactive, neither can it be scaled to different screen sizes without producing artefacts, although the browser could pass on screen size information to the server for specific rendering dimensions. However, this then becomes a privacy issue.

Therefore, to ensure 100% of users can use my site, I will be using HTML5 canvases. However, if their browser does not support this, I can render a server side version, and send an image. This will be accompanied with a message saying that they need to upgrade their browser, as a public service announcement to anyone still using outdated technology.

However server-side rendering may not be needed. As part of the testing phase, I will research what web browser the Orokonui staff are using, and if they are using that which supports canvases, I need not bother with server-side rendering.

### **HTTP vs HTTPS**

These are basically the same protocol, however, https includes additional security features. These features are

- TCL/SSL. Transport layer security/ Secure Socket Layer
- Certificate/ Identification Proof

These features effectively stop man-in-the-middle attacks. However they produce some extra requirements of my server. Firstly it must have a certificate, which can cost, and must be renewed yearly. Secondly, a greater amount of setup is required. An additional advantage is the Google rank search results with https prioritised over http, so there is a greater SEO advantage.

For now I have decided on http, to keep it simpler. There is always room to upgrade to https in future. SEO is not a big deal as the people using the site will all know the server's URL, and have no need to search for it.

### **Privacy Concerns**

Since I am making a website for the data display, I will have to take ethics into account. The graphing process will have to know the screen size of the device, which could be used to track that device across the internet. All sizing will be handled on device, via JavaScript and CSS. If an unsupported browser is used, a rendered graph image will be request of a pre-set set of sizes. Also, GPS, if supported will be used to find the user's location on a map of sensors. I will pass on locational data to Google maps, from within the browser, which it entering my server. It will then be under Google's privacy policy. My site will not have any user accounts, so I do not need to store user data. For analytics, I do not plan to collect detailed information, and will only collect page views, bounce rate, and any feedback they provide. All analytics will have personal or traceable data (i.e: name, ip address etc, system information) removed. If a user volunteers personal data in a feedback form, that will be kept.

### **Specifications so far**

---

After researching the context considerations and having initial discussions with Andrew and Tony, I have come up with some initial requirements and specifications.



- Interface scales to screen size.  
This is important as access needs to be convenient, and viewable by everyone. It needs to look good on a smart phone, yet also be able to show a full sized version on desktop. Since I am using HTML5 Canvases, drawn to with javascript, and can get the screen size, and scale the drawing process from with the JavaScript script.
- Simple interface, usable regardless of technical skill  
Anyone who want to view bird statistics, should be able to do so. Anyone who visits the sanctuary should be able to view this information, this includes people from small children to elderly people.
- Provide details  
Despite being simple, my interface also need options, it needs to be able to provide advanced statistics and data for those that require more detail.
- Database  
I will need a database to store all sensor data, since sensor implementation. This means the size of the database will expand of time and I will also have to consider this. Maybe a feature that warns the user to add more storage, removes old data, or exports old data somewhere before removing it.
- User Interface integrates with a database  
I will need a database for my interface and graphing program to get it's data from. MySQL is a standard web database that displays and integrates with PHP.
- Twitter Integration/ notifiers  
Andrew would like my system to send tweets, sms and or emails about the data it is collecting.

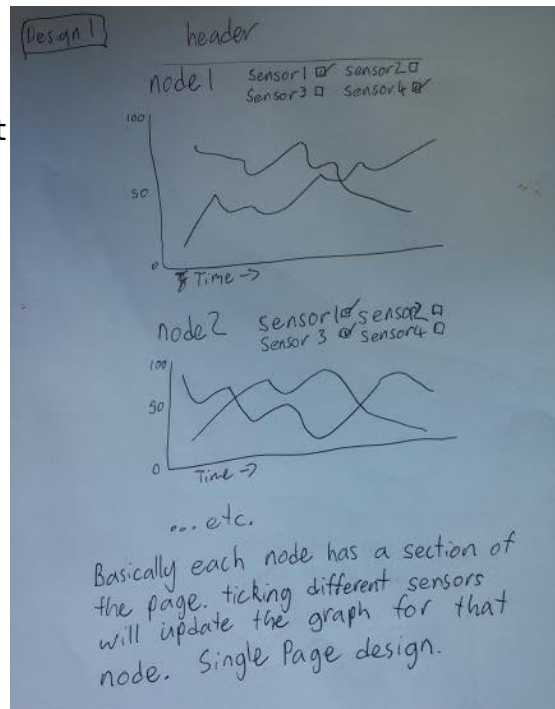


Illustration 1: First Design Proposal

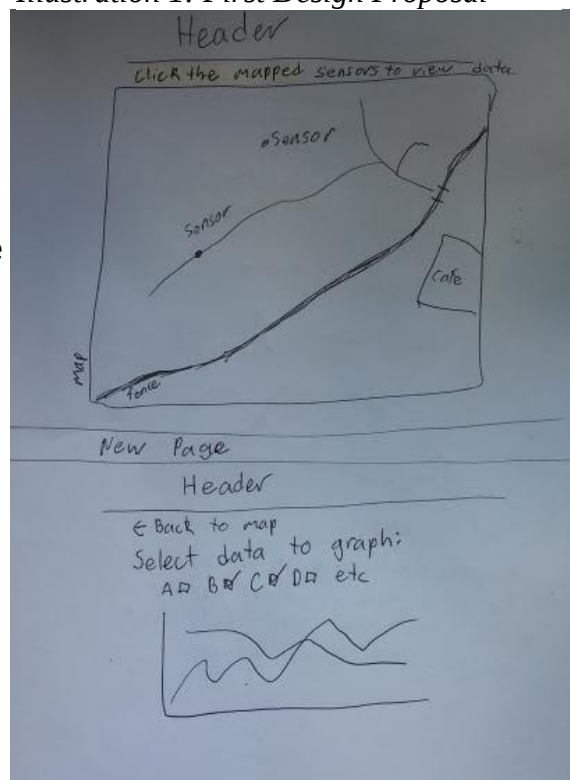


Illustration 2: Second Interface Proposal

Investigating into the hardware and backend components of my system would take time and on-site meetings. The best use of my time at this stage was to develop the user interface. This is the advantage of a modular system.

### User Interface (UI)

I proposed 2 interface design ideas to Andrew. His feedback is as follows, my original

## email starts with a greater than sign:

> Here's a few ideas I came up with:

> 1) Single page view, with each node having it's own section. Check boxes mark

> which sensors are graphed, and the graph is update as options are selected: (see Illustration 1)

Good. Whatever is smooth and works will be good. The type of GUI (smartphone versus mega pixel home / desk top monitor where you can pull a plot across to the side of the screen while you bring up another plot... This sort of thing + being able to maximise the graph plot on say a small hand held device so be able to see anything. You go for what ever you are using and probably the smaller screen needs to be thought about. Cayenne uses a default 1/4 / 1/6th size plot that you can 'enlarge' with an enlarge button on it. See screen shots. Only problem I have with them is it is only recent history with variable resolution so you can not easy look at say 3 hours a week ago

> 2) First, the user will be brought to a map of the sanctuary, with nodes marked.

> They can then click the node they want and view data from that node:

This has to be the best, most intuitive 'one shot' application for even an old person or young kid to understand

Esp if it has a friendly "You are Here" if your phone GPS can locate you within the map.

> Thoughts? In both design I forgot about the time frame, this would be added

> along with the sensor checkboxes in both cases. Or I could use an interface

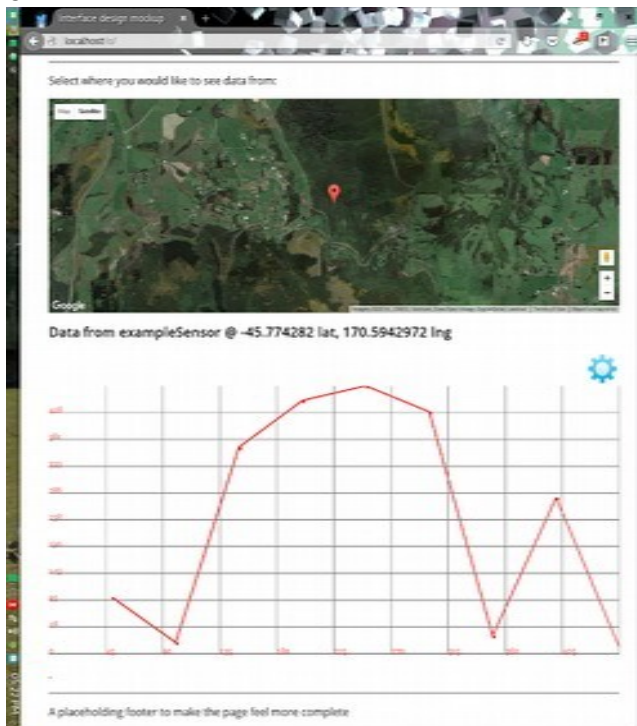
> similar to the existing solution if preferred.

Fine tuning. Best to remove this sort of 'clutter' to a behind the page (right click / options etc)

See the cayenne method

The default settings can be adjusted or nominally pre set by the person installing the sensor for the page of interest.

From this information, I created a prototype interface using HTML, CSS and JavaScript. I also included some PHP in order to easily connect to a SQL database later on.



Above is my prototype. I used Google's map API to embed a map onto the page.

This means I can drop click-able pins onto the map, marking sensors. These will update the graph.

The graph itself sets its data from an array, which is passed to the JavaScript graphing function through PHP. It then scales the data according to the canvas size, which itself is defined by the screen size.

The options button sets the size of the options div to 100 percent and 300px, revealing options for the graph. Its default size is 0x0, with overflow contents set to hidden, effectively hiding the options. Clicking the button again restores this state.

Incoming transmissions are passed to a python function. Args are Sensor location, Sensor name, Sensor type, Sensor value and time of reading.

```
sloc = (lat, lng)
sName = "human readable sensor identification name"
sType = "type of sensor, eg: temperature"
sValue = float # Whatever value the sensor detects, units is described by
sensor type.
Time = int # Unix time of sensor reading, eg, this was written at
1457920053
# This can also be made a float if more than a seconds
accuracy is required.
processIncomingData(sLoc, sName, sType, sValue, time) # This is how the
function will # look in
python.
```

This function will add the new information to a MySQL table, will have different tables for each different sensor location.

I am using Python to do this because it is a great general purpose scripting language, its tag line is 'batteries included' since it has libraries for basically anything, including databasing.

I will use the library 'MySQLdb' with `import MySQLdb`. This is the MySQL library I could find the most documentation for online, and the one in which looks the simplest.

I also created a tweeting system in which tweet configuration can be done for a config file with human readable syntax. This works by having an indication mark in the text that tells my python script where to substitute in variables, what sentence to use for each data type, and what to ignore as a comment, it uses a series of for loops to examine the config file and decide what to tweet.

I tested this system, and found that it worked well, it may or may not be implemented, based on whether stakeholders use Twitter or not.

Here is a progress update email thread:

```
Subject: Re: Orokonui project update
Date: Sun, 20 Mar 2016 18:13:38 +1200
From: Andrew Hornblow <Picaxe@clear.net.nz>
To: William Satterthwaite <william.satterthwaite1008@gmail.com>
```

Good stuff

Do you have this running on a Pi or a setup where I could start to send you a radio signal from a simulated or real model of a gate or trap line? What sort of input signal are you expecting or is it set up to take? Is it like Craig's setup or would you like me to help with the serial

radio data input signal?

This can take a little bit of fiddling about to get all the parts working correctly

What would be the best way to meet up with you officially with your mentor / minder in April so we could give this system a test on site? I will be doing some site visits and more range tests with the advanced 'LoRa' technology in the week of 18th to 22nd April.

I can bring with me demo kit that you could loan / borrow or purchase that you could use to act as a real setup or at least a simulation at home or school while you carry on developing the code.

I hope this comes together for you for a great school project or competition entry some time in 2016!

~ Andrew

On 19/03/16 17:26, William Satterthwaite wrote:

Project update 19 March 2016:

- Database is now fully setup, and integrated into the grapher.
  - Graphing options, user can now click different dropped pins to select data from different areas. Dropped Pins are dynamically created based in the latitude and longitude stored in database. User can also select different data types, however there not yet a GUI way to do this, it must current be typed into the end of the url
  - Different graph options can be saved/shared and recreated the same way, due to all options data being added to the end of the url.
  - Python script will insert new data into the database, as well as tweeting the new data.
  - Tweets can be configured using a configuration file, with human-readable syntax.
- (<https://github.com/green0range/schol-Orokonui-data-handler/blob/master/twitter.config>)

I've also put the project files in two github repositories;

Python script to insert data to database and send tweets:

<https://github.com/green0range/schol-Orokonui-data-handler>

Website frontend: <https://github.com/green0range/schol-Orokonui-grapher>

Cheers, William

## **Options Submission**

After testing my prototype, I ran into an issue with the way I was handling options. All options were submitted as a GET method form, this means that they are appended to the end of the URL, for example:

<http://localhost/index.PHP?linecolour=000000&gridcolour=111111&keycolour=566666&xlbl=1>

My PHP script can then easily pick the variables off the end of the url with `$_GET["linecolour"]` where linecolour is the option I want the value of. (000000.) I am using this method over POST, which would hide the variables, so that users can created customised graphs, and then share them simply by copying the URL. However, when I submit a new options form, with a previous submission already appended to the url, it deletes the previous.

To fix this issue, I created a new PHP script called optionsHandler.PHP. I used POST to get the options form data to this script while using GET to submit the previously submitted data. The script then ran through the submissions, using the new POST submissions where available but keeping the GET one where they are not. It then appended all the non-blank submissions, to the index URL and redirected back to that, this solved my issue, the script runs almost instantly and causes no notable delay, although I was testing on localhost, and will need to consider internet speeds in a later test.

## **Usability Customising Graphed Data**

It is important to be able to customise the graphed data, so that a user can more easily interpret the data, i.e. by selected different data types, line colours for the types, a line weight that they can easily see it, a good grid size to show them relevant information, etc, etc. I want this customisation to be shareable, and the easiest way to do this is by making all the customisation options part of the URL, so that a user can simply copy the URL of the customised graph. This is why I am using GET rather than POST, and why I needed to make a data handler PHP script, so that get options from one area could be stored, and new ones added, rather than deleting all the old options.

### **Multi-series Data**

Next I started working on getting multiple y series data points (x is time and therefore constant) on the one graph. I decided to use the keyword type in the GET part of the URL to store which options the user chooses. However the type needs to be any array to store multiple data choices, so I separated choices with commas and added a part of the script which as GET type options separated by commas to the \$selectedTypes array. It shows in the address bar like this: index.php?type=temp,athing. This again allows easy sharing of different graphs, as copying the URL will store user options, including selected data types.

#### **Variables**

Tick the variables you want displayed.

temp :

athing :

I also made a new section in my options div, to contain the different data types. I used check boxes, which are dynamically created according to the data available in the database, for the selected location. I also made it so that the script automatically ticks boxes that were already selected.

### **Why Dynamic**

I am dynamically creating the interface in order to display live data. A static website can only be updated manually, this is not practical, as Orokonui staff, need to see and get notified of any issues that arise, as soon as they arrive, this is why I am creating a dynamic website interface that updates automatically.

### **Dynamic Considerations**

However, with data that can be displayed automatically, I need to ensure that reliable data is displayed. This calls for checksums on radio transmissions, and anti-tampering measures when sending data across the internet. I should also have a checking algorithm to make sure the sensors are not being tampered with, for example a rainfall sensor reporting high rainfall when someone has poured water on it. This could look at previous data and detect when it is physically possible for such amount of rain to fall in such short time.

### **Testing**

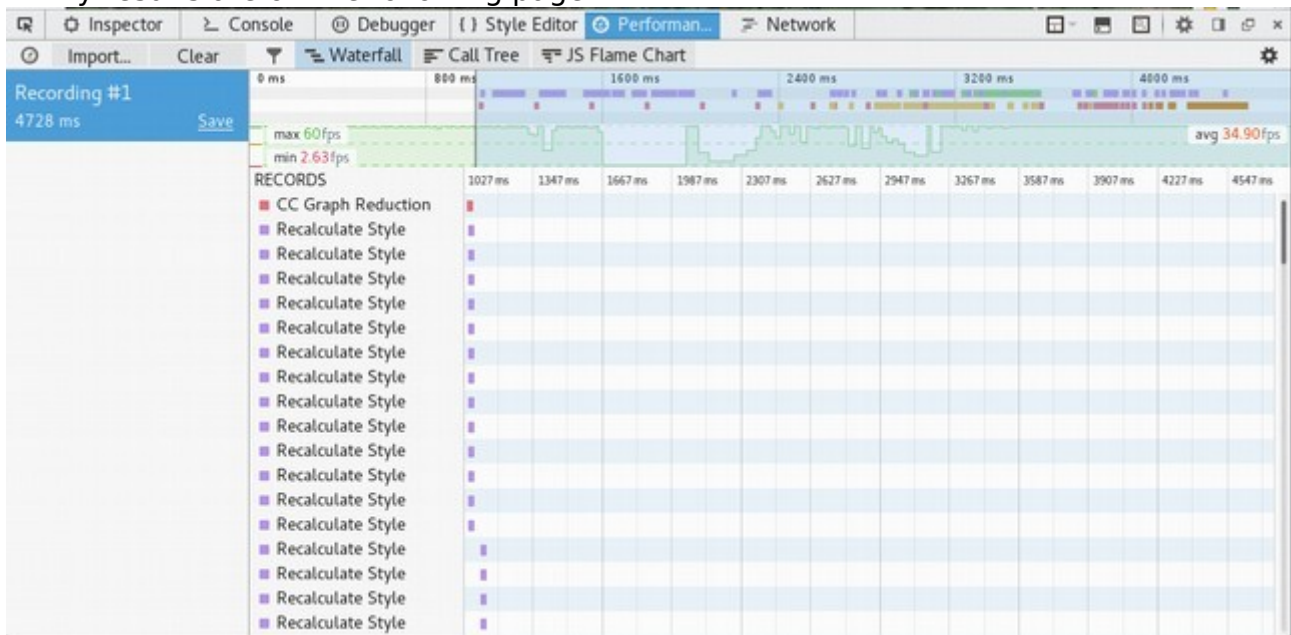
As well as testing that it works, I also wanted to test that everything ran quickly and efficiently. This is important as most people give up on using a technology if it is too slow. Perceived delay causes users to wonder what is happening, if the application is working and retry clicking buttons that don't seem to have immediate action. If the button is already pressed and currently processing, this restarts the processing and makes it even more delayed, causing a lesser user experience and often frustration. I setup a performance test using Firefox's built in performance test feature. I started the

test, and the reloaded the page, showing a single series graph with 3 data points. All customisation options were set to default. I tested this on a computer with an Intel i5 Quad Core 3.5GHz CPU and 16GB of RAM. My network speed is about 2.15mbps to the internet (for downloading Google maps) and negligibly fast to my server, which was on the same LAN. According to the March 2016 Steam hardware survey, this is slightly higher than most, with the average cores being 2-4, speed being 2.3GHz - 2.69GHz and RAM being 8GB. Of course this survey is skewed towards the high end as gaming computers require better hardware. I will need to test on lower-spec hardware later on.

I used Firefox because of personal preference, I personally like to customise every detail of my experience, and Firefox, with the use of themes, add-ons, and about:config, is excellent for this. It also looks nice by default, where as the Linux version of Chrome tries to use it's own custom title bar design, instead of the system one, which looks truly awful. As seen below, Firefox also has excellent debugging system, as well as syncing nicely with my tablet and phone, the latter of which runs Firefox OS.

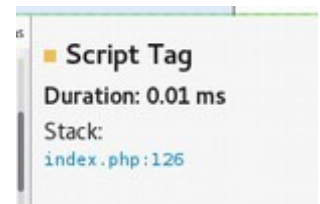
However Chrome is the most widely used browser, so it must work on Chrome, which is relatively easy, as Chrome also has the most features and is the highest scoring in almost every web test ever, so if it works at all, it probably works on Chrome. Of course I will manually check and verify this, but don't expect any problems here.

My results are on the following page:



Overall it took 3360ms - 3.4 seconds - to load. Most of this was recalculating styles, as shown by purple. I suspect most of this load time was caused by Google maps, as I have a relatively simple styling sheet, which should not use much load time. Also the map has to download images across the internet, rather across a local network.

After looking through the data, this appears true, selecting different script tags in the waterfall tree, shows their origin. I am pleased to say that the script from index.PHP at line number 126, took 0.01ms to complete. Line 126 is the beginning of the graphing script tag in the compiled PHP script. This means my graphing script runs quickly and efficiently.



However, even though I have been using Firefox as my test bed, not everyone

does. It is currently about third in user share. In the following I test my site on other browsers:

### On Internet Explorer (IE)

Tests on Internet Explorer 11 showed that clicking the sensor marker was unresponsive. Closer inspection of the error console showed that Object doesn't support property or method 'includes'

This is referring to the following JavaScript statement

```
if (!window.location.href.includes("?")) {  
    // Do stuff  
}
```



This statement is used to detect whether a GET statement already exists. Window.location.href is the url and includes tests if the url contains a question mark somewhere within it's string. According to this testing, Internet Explorer doesn't support the include function. Because Internet explorer is still (unfortunately) a widely used browser, and version 11 is the latest, before it was re-branded as edge, I will need to find another way to detect if a character is contained in a string. Closer inspection of the include function showed that it is a new addition to JavaScript, with very little support, especially in mobile. (image: Mozilla dev network)

Feature	Android	Chrome for Android	Firefox Mobile (Gecko)	IE Mobile	Opera Mobile	Safari Mobile
Basic support	No support	No support	40.0 (40)	No support	No support	No support

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	41	40 (40)	No support	No support	9

The includes function is a newer version of the contains() function, which was

changed after a bug reported that sites using MooTools, which had it's own contain function, broke Firefox 17.

Trialling the older method of contains, yielded the same result in Internet Explorer, it appears that this feature was never included in Microsoft's flagship browser, even in the older form.

I found a solution to this problem by using the indexOf function, which is supported by all browsers, including Internet Explorer. The index of function finds the first occurrence of a substring and returns its position, returning -1 if it does not exist. This means that if the string "includes" the substring, it returns >-1, if not, -1, effectively imitating the include function.

I have updated my webpage to use this technique, and IE 11 renders it perfectly fine.

It is also necessary to test earlier versions of IE, since IE tends to be used by non-technical people, who don't know how to install a better browser, or even upgrade IE.

Testing on IE 6, shows that the browser is unable to even render transparency. However this is not a big deal as IE 6 is the default browser of Windows XP, which is

official dead, (lost support from Microsoft) as of last year.

Testing on IE 9, reveals that the system is unable to support the Google maps API which I may be using. However, there are some alternative maps I can look into, one in which was specifically designed for Orokonui.

### Chrome (Latest)

I then tested the system using Google Chrome, and it works perfectly fine, although I must make sure every part of my site has fonts styled through CSS as default Chrome fonts are ugly. This will also ensure the usability and design is consistent across browsers.

I do not feel the need to test earlier versions of the browser as almost all Chrome users are on the latest version due to it's automatic updated system, and increase technical understanding of the majority of it's users. Also, Chromium is basically the same, so different testing shouldn't be required.

### Opera (Latest)

I also tested on Opera, and found that it worked perfectly fine.

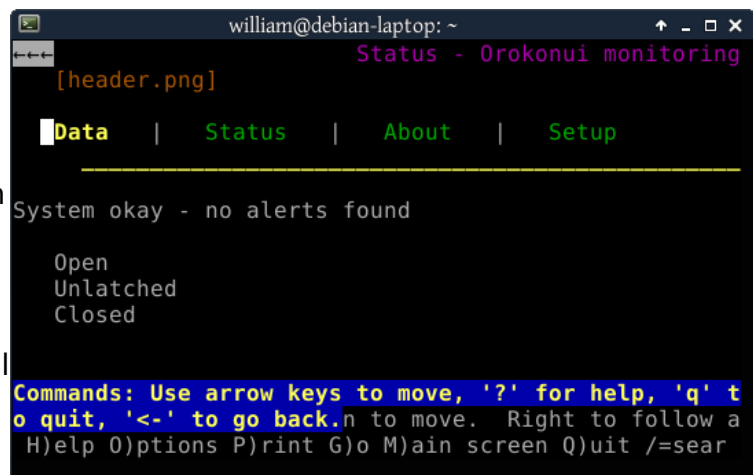
I do not need to test on older versions of Opera, as again Opera users are usually technically inclined and keep their browser updated. Opera also has a very small user share.

### Smaller Browsers

Other smaller browsers, such as Middori, Ice Wessle, Brave or the many others, are usually based on an existing open source browsers, such as Firefox/Gecko or Chromium, and therefore use the same rendering engines. Both Gecko/Firefox and Chromium work perfectly fine, so these are of no concern.

### Terminal Browsers

Terminal browsers, such as Lnyx, are by nature text only. They do not support any JavaScript or Image features. They have a very small use case, navigating documentation on a non-graphical Linux installs (such as setting up Arch) or people interested in replicating the early Internet. Therefore they are not of any consideration to me. Lnyx is still able to show some information.



### Client Communication

My initial project development was based on email communication with Andrew, then, on the 6<sup>th</sup> April, 2016. I discussed the project with Tony, the afore mentioned volunteer from Orokonui. He agreed that the system needed to be modular. Such as having the graphing/interface part, completely separate, and working independently of the both the data transport (radio link) and sensor parts. Luckily, I had already designed it this way, based on communication with Andrew and my own research. I



will need to keep doing this when adding in new features.

Tony clarified that the most important need of the ecosanctuary was the monitoring of kiwi gates. The data that needs to be gathered was whether or not the four gates around the Ecosanctuary are open or not. It then needs to alert someone if the gates are left open for a predefined time.

The reasoning being that there are "...four gates within the Ecosanctuary which separated the juvenile kiwi from the adult kiwi. They were kept separate to stop the adults from picking on the juveniles.

Sometimes the gates were accidentally left open by visitors, and the Ecosanctuary was in need of a system which would alert staff if the gates were left open" - Me quoting John Lewis, Otago Daily Times reporter quoting me. A scan of the full newspaper article can be found in the scholarship\_docs directory.

However, this was the baseline project, and from here other ideas were developed. The reasoning behind these other ideas was 'if we have the gates setup, why not add other sensors, and more of them in other places.' This makes my job almost infinitely expandable, and is the reason the system needs to be entirely modular. It also needs to use standardised systems for different modules to interact.

So far, I have almost finished one of the modules - the graphing module, from my previous research and testing. This needs a standard way to get and store information. The standard way I'm using is MySQL. Additionally, I will need to create documentation of how this system works, so that it may be taken over in future by anyone looking at expanding the system. It must all be open source, so that any future maintainer can maintain the system without my involvement. I will attempt to make it as low maintenance as possible, but IoT is a very new concept, and as such standards are constantly changing.

The other modules that are required are:

- A data transport/ radio link module - to move data between sensors and server/interface/display.
- A sensor collection module. - to collect the data.

After this base system is created, and setup with the 4 gates, other proposals from the Orokonui staff were:

- Other sensors throughout the Ecosanctuary, this would require a longer range radio link. Each sensory computer could have inputs for
  - Temperature
  - Sound (recording bird calls.)
  - Trap status (sensors in traps to see if traps have closed)
  - Wind speed (Used to close walking tracks if too high.)
  - Humidity - This would be put through a 'Fire Likelihood' prediction algorithm.
    - This should also be expandable and modular, so that future sensors can be added.
- Cameras
  - These would be placed by waterways, and show if stoats are getting in through to water gates.
- A logging system that collates logs from each sensor, integrated with alerts.

Another thing we discussed was the data transport/radio link module. From what I understand, Andrew is currently making the transmitters broadcast multiple times, hoping that one of the broadcasts is received. However, Tony is not happy with this system, and wants an a way to ensure that data always gets to it's destination.

We will also need to test which frequencies provide the best coverage. Frequency hopping was suggested as an option to overcome frequencies that are being used by

different devices (we will be using a public part of the electromagnetic spectrum).

We also arranged that on 13<sup>th</sup> of April, a week after the meeting, I would visit Orokonui to look at the area in more detail.

I will also need to make an official proposal to Orokonui.

## Updated brief

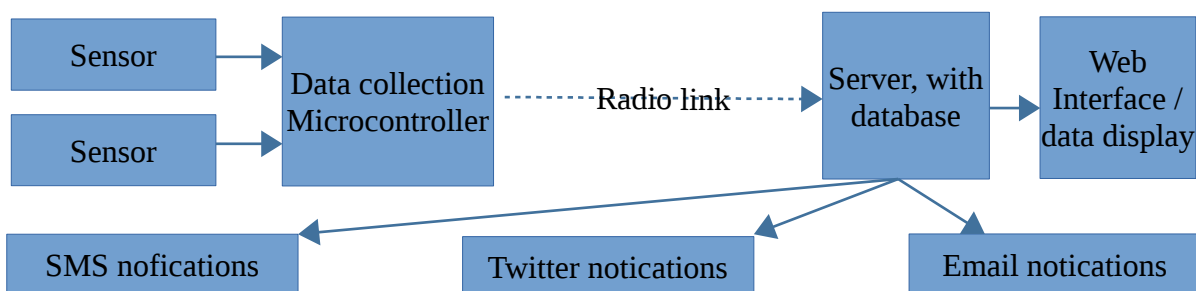
---

From this meeting, I developed my updated is brief as follows:

I will be creating a system to record, and inform staff and public with interest, as to the live status of the Orokonui Ecosanctuary. This system will be designed as a modular open source system. As such, it will have five basic components.

- Data collection
  - This will by small microcontroller, with connected sensors, such as switches or wind monitors to detect environmental changes.
- Sensors
  - These will be connected to the afore mentioned data collection micro-controllers. Each sensor type will be able to be connected to each microcontroller. Microcontroller can support multiple sensors.
- Data transport (radio)
  - This will transmit the data from the data collection microcontroller to a central database and server (both as one device.)
- Interface
  - This will be a web interface served from the afore mentioned database and server combination.
- Notifiers
  - This will notify an Orokonui staff member or group of members if something is wrong, for example a gate is left open. It can use SMS, email, or twitter notifications. Each of these different notification methods will also be modular.

Due to the modular nature of the system, if any part of the system needs to be upgraded, or in the case of hardware parts, malfunctions, it can be replaced without disrupting the rest of the system. The open source nature means that anyone will be able to continue development of the system if required, without rewriting the entire codebase. I will provide extensive documentation on how the system works so that this can be done easily if required.



### Wide-area Identification Verified Protocol (WIVP)

After the meeting with Tony, I started working on a protocol which we could use to send sensor data back to the server. Although it had not been described as a big issue in the meeting, I decided security was a lot more important than we had discussed.

For example; a malicious sensor node could report fake readings to a database node, or a malicious node could imitate a database node and steal sensor information. Sensor information is not top secret, and will be publicly released through the web

interface, however such a malicious act could stop the real database node from ever receiving the data.

So I wrote a document about the concept of an identification verified protocol, and circulated it around my Orokonui contacts. The document can be found attached as reference material.

As an overview, the sensor node sends a connection request, to which the server replies with it's public key. The sensor sends a random one-time connection key to the server encrypted with the public key, to which the server sends it's identification certificate encrypted with the connections one-time key. If happy that the server is not an imitation, the sensor sends the data, encrypted with the servers public key, and the server send an acknowledgement back. Repeat until all data is sent. The sensor then sends a terminate command, along with it's certificate, and the server either keeps or dumps the data, depending on whether the certificate checks out. It then sends an acknowledgement. This is the last transmission for that connection.

## Power

---

Power is a key issue in the development of this project because sensors are in remote locations, and must be powered 24-7. It is impractical to replace batteries every few days, and it is being implemented in an ecosanctuary, which has a 'clean green' image.

**Andrew's reply to this proposed protocol with some concerns about power consumption.**

...downside of back and forth systems is power, complexity and cost.

The Energy budget for the above ideas is very dependant on how active, how fast data (actually needs) to be sent forward. Things like RF transmission that uses most of the energy. Think about 'Duty Ratio' I.e. how much time Transmitting versus how much time waiting and actively listening. When you start to get the numbers for this then battery design and power supply becomes the biggest cost and consideration in any project. Ideas and issues such as...

- mA power consumption when T, Rx, CPU active versus Sleeping saving power
- Required duty cycle of these three states
- mAh on average milli Ampere Hours (fuel economy)
- Battery capacity (mAh or even Ah) (how big is the fuel tank) - How deeply the battery can be discharged without flattening or damaging it...
- What is the average worst case scenario if there was say no sun or snow on the solar panel for 'X' number of days
- Cost, capacity and expense of solar panels

Most people will fall off their chairs when they see how big an issue / cost this is.

**He also gave me a list of thing to research:**

Some things to look up and or add to your general knowledge:

- Data Channel versus a Data Circuit (just ask this is not easy to understand...)
- Advantages / disadvantages of Both !
- Power needed to keep a radio receiver / transceiver going
- Power consumption of a simple ASK RF Tx when idle
- Power consumption of various micros (you compare them)
- - Pi
- - Arduino
- - Picaxe

-- AVR etc

Before looking into this problem in any great detail, I think transmissions could be divided into critical, and non-critical, where all non-critical transmissions are instead stored, and sent at the top of every hour, to save on power consumption. This could be a system level feature so that software using the transmission API think they are transmitting as normal, and indeed are if the feature is disabled in a config file. There could also be battery checks, which stop all transmissions when lower than 25% and put the entire device, including sensor input and recording systems into sleep mode at 20% and complete shutdown at 5%. If it hasn't reached complete shutdown, and the solar panels suddenly start providing input again (eg; snow melts) it could automatically come out of sleep mode, transmit a status update, and continue as normal. Complete shutdown would require manual restart.

### In depth power research

After doing some more research into components and their power consumption, I have arrived at the following result.

Component	specific model	Used		Produced		Stored	
		Voltage (V)	Amperage (mA)	Voltage (V)	Amperage (mA)	Voltage (V)	mAh
Picaxe	18M2		1				
Transmitter	MICRF113YM6-TR	3.6	12.5				
Receiver	MICRF213AYQS	3.6	3.9				
Solar Panel	Sanyo AM-8701CAR			6	31.6666666667		
LiPo Battery						3.7	1000
<b>Total</b>		7.2	17.4	6	31.6666666667	3.7	1000
		Note, components will be in series					

Run time on battery alone (h)	Run time on battery alone (d)
57.4712643678	2.3946360153
Run time with 8h solar charging (h)	Run time with 8h solar charging (d)
310.8045977011	Infinite, it will charge enough per day to replace any battery usage.

You can see the full document in my programs docs folder under *Power Budgeting.ods*

### Transmitter Types

All my cost analysis and power budgeting so far hinges on a lower powered short range transmitter. This type of transmitter would work in short range cases, such as the four gates. However, would be unable to cover the entire Ecosanctuary. The entire Ecosanctuary can be covered by more expensive, long range transmitters, however, this would increase costs. My current plan is to place a transmitter and receiver on every device, for data verification and handshaking. This is feasible due to the low cost of transmitters and receivers, but may not be with the long range system.

A possible way to get around this is to create a mesh network from the smaller transmitters, where each sensor node would listen for other sensor node

transmissions, and then re-transmit them until it reaches its intended destination. However, this involves each node constantly listening for transmissions, which will use more power.

### Proper processes

To start implementing this project I had to fill out a research request form for the ecosanctuary, and sign the proper paper work, this is an important legal process I had to consider.

### Meeting at Orokonui

On the 13<sup>th</sup> of April, I went to Orokonui to have a closer look at the physical requirements and environment.

This is one of the gates I would need to monitor:



With this gate design, it is possible to use a magnetic switch. The magnetic part would be on the opening part of the gate, and when it reaches proximity to the frame (i.e.: the gate is closed) it would trip the magnetic switch.

Most of the gates use one of two latch designs. This first one is shown above, and is the best design in use. The other is as follows:



This design does not auto close so readily, and is the one Tony is concerned about, since it can close without properly latching, and then be blown open in a strong wind. This means my sensor has to detect when it is latched, not just partially closed, and therefore send an alert when it is only partially closed.

## **Tuatara Temperature**

During the meeting at Orokonui, another aspect of the brief was brought up. It would be useful, to use the system to monitor the temperature of the Tuatara borrows, which cannot drop below 5°C. If it does, or gets close, it would send an alert.

Also, all alerts would need a configuration menu, in which they can be disable/enable, threshold values changed, and new rules added.

## **Distances**

The distances involved in line of sight transmission are:

Longest distance to Kiwi gate from reception:	349m
Longest distance to Kiwi gate from workshop:	410m
Longest distance across Ecosanctuary:	3000m

## **Status: OK**

Another point brought up at the Orokonui meeting was that a manager or duty staff don't want see exact details from the sensors, however just want to know if everything is OK. This brought up the idea of a 'Status: OK' web page, which would simply tell the visitor that everything is working, or any issues that have arisen, but only show critical data and not bother them with stats. This could also be sent as a text at the end of each day.

## **Heartbeat**

Another issue brought up was that some gates would not be opened daily, and if transmissions were made on each opening, the server would not hear from them for an extended period of time. This could be remedied by a daily system check and status update. This could send information such as battery status, uptime, CPU temperature, etc. If I were to use the system which sends gate opening numbers at the top of the hour, this could also act as a heartbeat.

## **Agile**

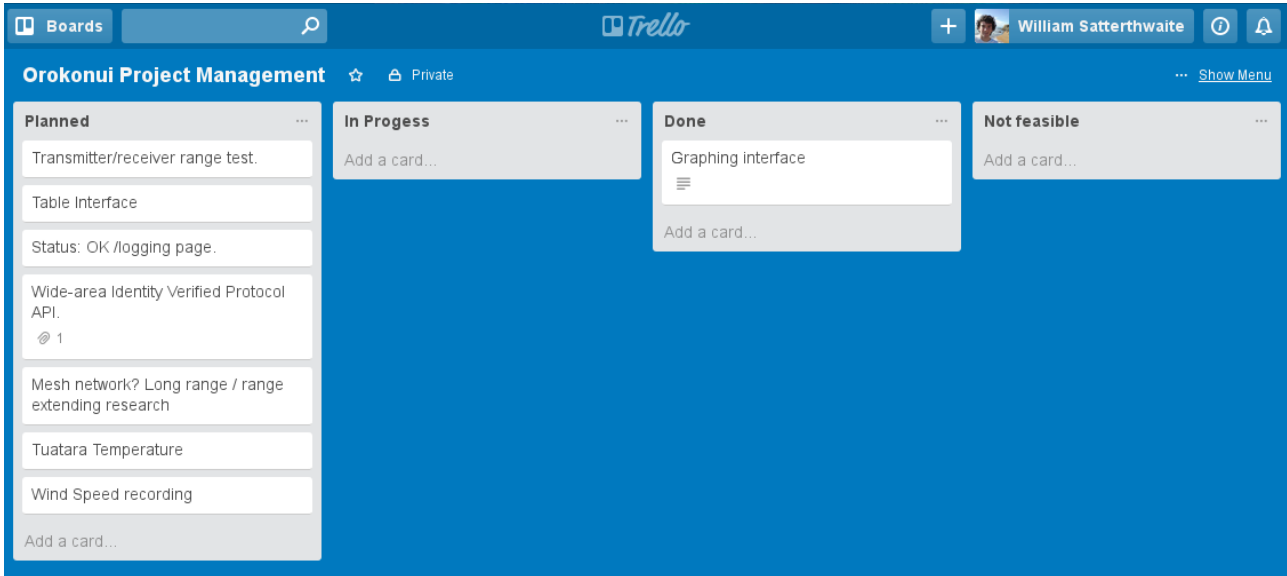
I am using the 'Agile' development process. In this development process I go back and forth, developing a small part (module) of the project, and then getting feedback on that part. Without knowing anything about Agile, earlier in the project, I was doing this naturally, so it is a style that seems to work well and come naturally for me.

Additionally, I cannot 'waterfall' for this project, as the client needs are constantly evolving. With Agile I am to let the project evolve in it's best, most natural, and required direction, whereas with 'waterfall' it would be locked into an initial direction. Without iterative and modular development as in the Agile approach, there is a high risk that his project would end up not meeting the needs of the stakeholders and inevitably fail at producing the required outcomes.

## **Project Management**

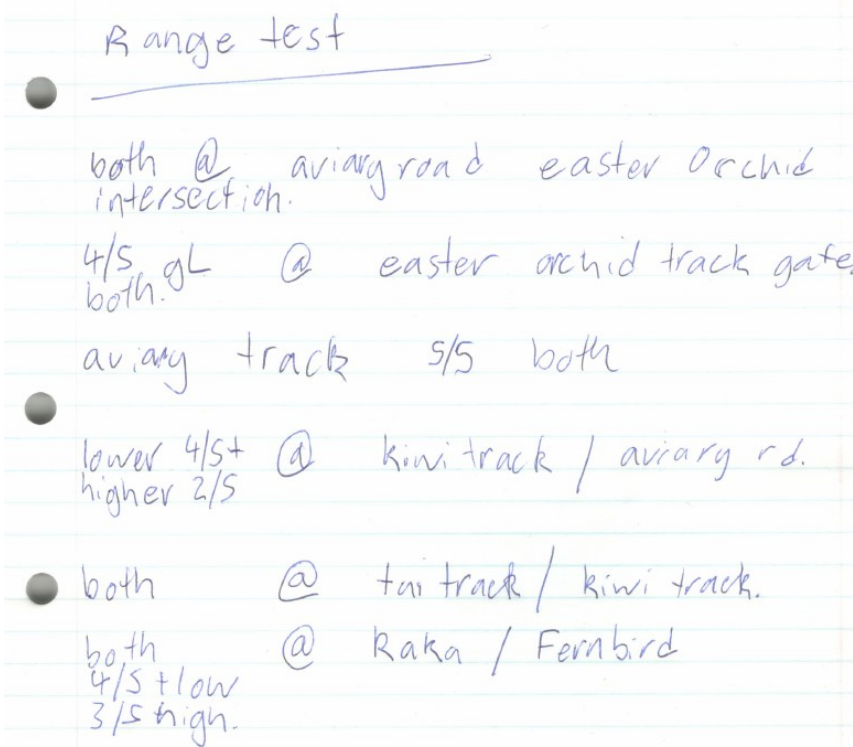
I decided that a Kanban Board would be the best way to manage my project, since it will let me visibility see what needs to be done, focus on that and track my progress. However, I do not have the wall space to create a physical one, and would want to be able to view it at both home a school. For this reason, I'll be using the web interface version Trello.

## Project Management - 15 April 2016



### Range testing

On Monday 18 April, I meet with Andrew and Tony at Orokonui. During this time we trialled some transmitters for range. The following is a scan of my raw data.



The results were fairly conclusive: Using these transmitters, the entirety of the kiwi gate area could be serviced.

This was also a time for me to have a first look at the circuitry and physical solution, as up until this point I had mainly been working with high level software (python data processing, MySQL, PHP, HTML/JS web interfaces.) and had only looked up component specs online.

### LoRaWAN

LoRaWAN is another transmission technology I am looking into. It is a standard designed for long distance IoT devices, used by The Things Network. The Things Network is an International initiative to setup an open LoRaWAN network in cities across the world, the test city being Amsterdam, which had the Things Network successfully setup in six weeks. The Dunedin branch of The Things Network is being setup at Otago Polytech, with the project lead by Tom Clark. I met with Tom to find out more about this.

### Handshakes

One of the context considerations of my brief is data integrity. I need a way to

insure this is for a receiving node to send an acknowledgement byte. The field node would then see the acknowledgement and know that it's message got through. If it doesn't see an acknowledgement it knows that it either wasn't received or got scrambled in transmission, and therefore knows to resend the message.

However a single byte wouldn't work as it could be a number reading from a sensor. To uniquely identify the signal is as an acknowledgement, it would need to be at least 2 bytes. I decided on 0110000101101011, which is 97,107, in base 10, or "ak" for acknowledgement in ASCII.

I set up a two nodes, one which transmitted a 1 (00000001) and another which received the 1, checked that it was a 1, and transmitted an "ak". The second node flashed a green light when while sending the "ak", and the first flashed an orange light when it received the "ak" confirmation.

I then took this on a range test to find it worked fine over ~300m line of sight. (NZQA: Please blank this image if used in exemplar as it contains locational information.)



### **Short range transmitter vs LoRa**

The main issue with LoRa is that it is not fully implemented around Dunedin. If this project was done in a year or two's time, LoRa would be the obvious solution. However, at the moment, the Things Network is still being setup, currently with one test gateway at the Polytech, well out of range of Orokonui.

The advantage with LoRa and The Things Network, is that this is being setup all over the world, and could become as prevalent as the cell phone network is today. This means, using The Thing Network LoRa gateways, my solution could be moved anywhere, and work.

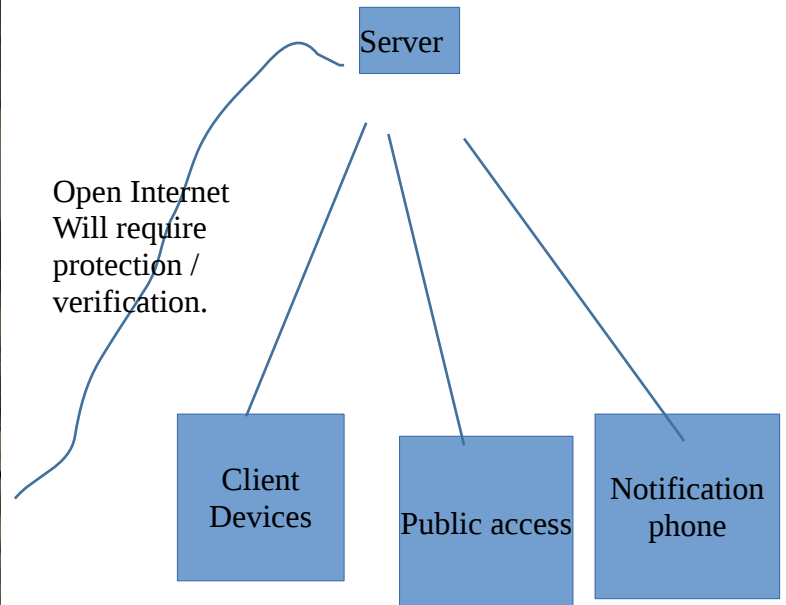


However, that is a possible future, and at the moment, the infrastructure is not there. It is currently better to use a one off radio transmitter solution. Due to my system being modular, after the infrastructure is built, it will be possible to switch.

For Orokonui, there is an internet connection at the visitor centre, so data transport between Orokonui and server is not an issue. This is a place where LoRa would be really useful, however due to the internet connection, it is not needed.

Overall, I will be using to short range transmitters, as they cover the area I need, and LoRaWAN is not yet at a usable stage.

### Plan overview, 3 May



### Getting data from collector to server

One of the issues (and greatest advantage) of sending data over the internet is that anyone can do it. This means that a person of malicious intent could send false data to my server, imitating my collector. My server needs a way to ensure that the data it is getting is from the collector. To do this, I am making a passcode constant which must be sent by the collector to prove that it has the authority. The issue with this is that the connection between collector and server is open. Anyone could watch for an update, record the passcode, then use it to send there own false data. To overcome this issue, I will turn to the RSA cryptosystem.

In this system, the server generates two keys, a public key, and private key. It gives out the public key to anyone who asks, but keeps the private key private. The public key can only lock, but not unlock, the private key is only able to unlock.

Through this system, my collector can ask for a public key, look the passcode with it, then send the locked version to the server. The server unlocks it, and checks it is correct. A middle-man attacker is unable to see what the passcode, since it is encrypted ("locked") with the public key.

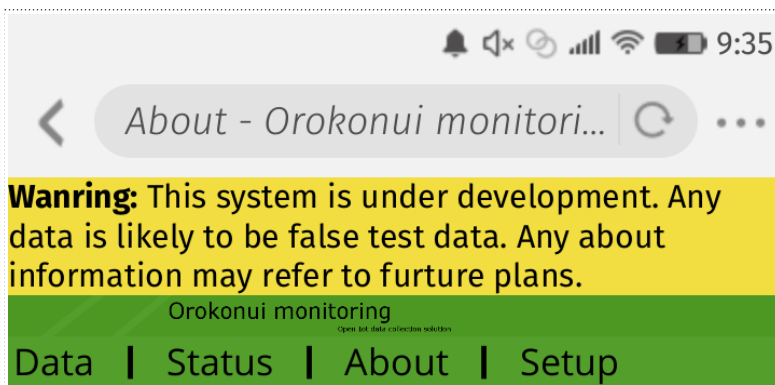
One difficulty of this is that RSA uses the totient of 2 primes. For security, these primes must be random. Since primes occur unpredictably on the number line generating large random primes is difficult. After eliminating even numbers, there is not much more to do than trial and error/brute force, which takes a lot of computational time. As a solution, I will be using relatively small keys, at around

12bits, and a timeout system.

A 12bit key is relatively easy to brute force, but if I can make a timeout system, so that the key is only valid for a few seconds, it should still provide plenty of security, and the collector will be able to request the public key, then encrypt and send the data within that time.

I then decided to abandon WIPV, as it has more security than is feasible or required for a Picaxe sensor to handle. It would add far too much computational time for the Picaxe to calculate encryption keys. I will instead be using a much simpler data package, that instead has security built into the checksum. This uses two key numbers in calculating the checksum, that are never broadcast, only there products are.

I have also decided that using CHIP is not feasible, as it is a very new device (just off kickstarter), it does not have enough documentation or software libraries for my to get serial data from it. This means I will have to go with my other option: the Raspberry Pi.



Orokonui monitor is a project to monitor some aspects of the Orokonui ecosanctuary though Internet of things (IOT) devices. This doubles in warning the Orokonui staff if issues arise, with an early warning SMS and status page, as well as letting the public and researchers know about the status of the ecosanctuary.

### What's Monitored?

- Kiwi gates

These are 4 gates supportate the kiwi's, the openings and closings of these are tracks, as well as a warning system for when they are left open too long.

- Tuatara temperature

The temperature inside Tuatara borrows.

### Data table

Not all the data I am dealing with is best graphed, and although graphs give great overview, they are not exact. For this reason, below the graph, I will be adding a data table, which is the graphed data laid out in table form.

### CSV

Additionally, it is sometimes useful to download data, to analyse in custom programmes etc. For this, I will create a PHP script that automatically puts the relevant data from MySQL and puts it in a comma separated variable file. (CSV)

### Mobile testing

Getting the interface to work well on mobile is very important because mobile provides quick and easy on the go access to data. Initial testing is positive, see left. The system I am using to do this is having two different Cascade Styling Sheets (style.CSS and mobile.CSS) for each of the different screen sizes, as detected by media queries. This is far easier than having 2 different websites, (for example, youtube and m.youtube.) and means I can manage the site from one

codebase. Since I have designed my site to be split into divisions (divs) I can reposition and resize them for mobile and desktop, without changing the HTML, and PHP scripts.

To implement my different CSS files, I am testing the screen size using the media option.

```
<!-- Import Styles CSS -->
<link rel="stylesheet" type="text/css" href="style.css" media="screen and (min-device-width: 600px)"/>
<link rel="stylesheet" type="text/css" href="mobile.css" media="screen and (max-device-width: 599px)"/>
<!-- Font Import -->
```

One issue I had with this implementation is that javascript graphs are not styled by CSS, they are drawn as a script, and therefore do not contain the larger font size I was using in mobile.CSS for the rest of the page. However, this was easily remedied as I can dynamically detect the screen size and scale the graph accordingly from within the script itself. I used the following:

```
// Scale font for mobile
if(window.screen.availWidth < 600){
    cd.font = 'italic 40pt Calibri';
}
```

Where cd is the context drawer.

The result is shown right.

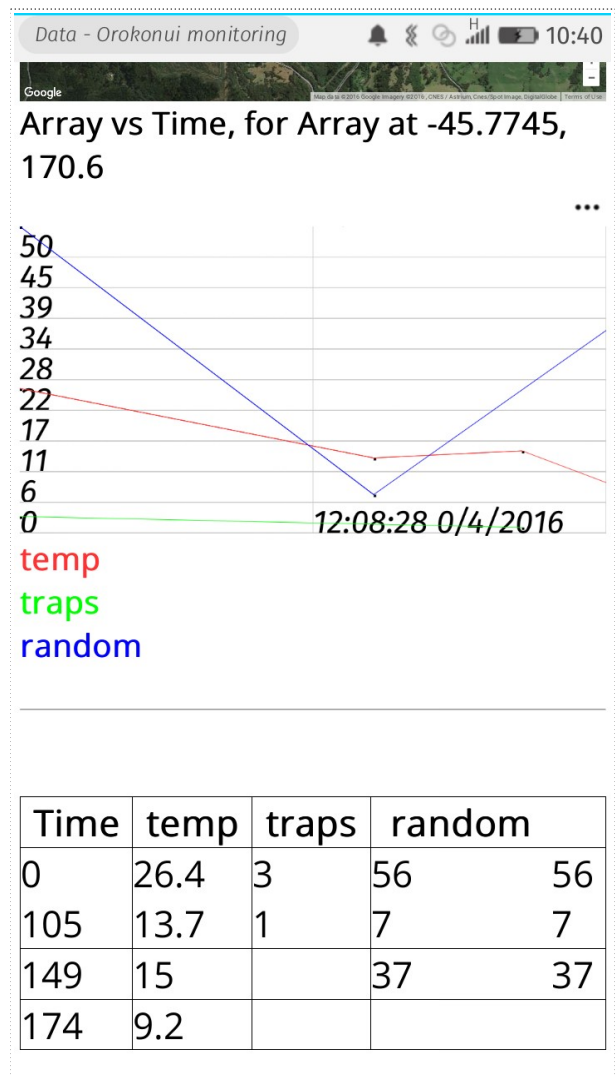
**Flaws**

Possible issues with the detection system is that mobile screen dPi tends to be far higher than that of a computer monitor, so I should employ a system to find dPi and resolution, then calculate the overall size.

Also, the map still uses the desktop style, I am unable to style it since it is provided by Google, so I will have to look into whether they provide a mobile aPi, or a different maping solution.

Additionally, testing will need to be done on other phones, since mine has a rather unique operating system, which may render things differently. I can find my old Android and borrow an iOS device to test this. On the desktop side I will also have to borrow a Mac to test Safari, and use Virtual Machines to test Internet Explorer.

The default number of markers also proven an issue with the size of the screen. By default it shows 10 dates along the bottom, which all draw on top of each other on mobile. This can be changed in options, but I should include a function that detects how many can be drawn based on screen size.



## Bug fixing

### Importance

Bug Fixing is an important part of development. As I near completion, I need to test and ensure that it full works, if not, these issues - 'bugs' - must be fixed.

### Graphing

As mentioned before, there are a few flaws in my current graphing solution, mainly around data being drawn on top of other data, and therefore making it confusing and hard to read.

9:12:06 405724928/373385159/206255201.603430446/20623/342010653/2511633/381.082011.55/10306/3/0.0475.133

I fixed this by instead of giving the markers a default of 10, I dynamically created to default based on screen size with a maximum number of 10. This can be overridden through the options menu, but I don't see this as an issue.

```
for (var i=10;window.screen.availWidth < i*300;i--){
    xlabel = i;}

```

This successfully fixed the issue on both mobile and small-screened desktops.

## Help

### Main Page Instructions

Having instructions on how to use the web interface is important in teaching new users (such as future staff, or when the system is first deployed) how to use the web interface. It needs to be informative, but simple. I don't want to a 400 page instruction manual, as this would be hard to read and potential users would simply give up.

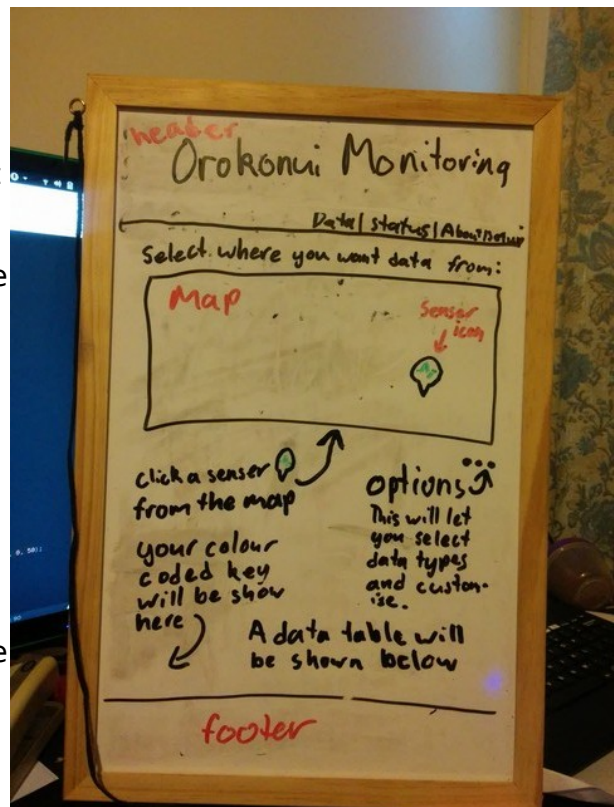
Instead, I think it would be best to have diagram style instructions, such as arrows pointing at all the on screen buttons, and explaining what they do.

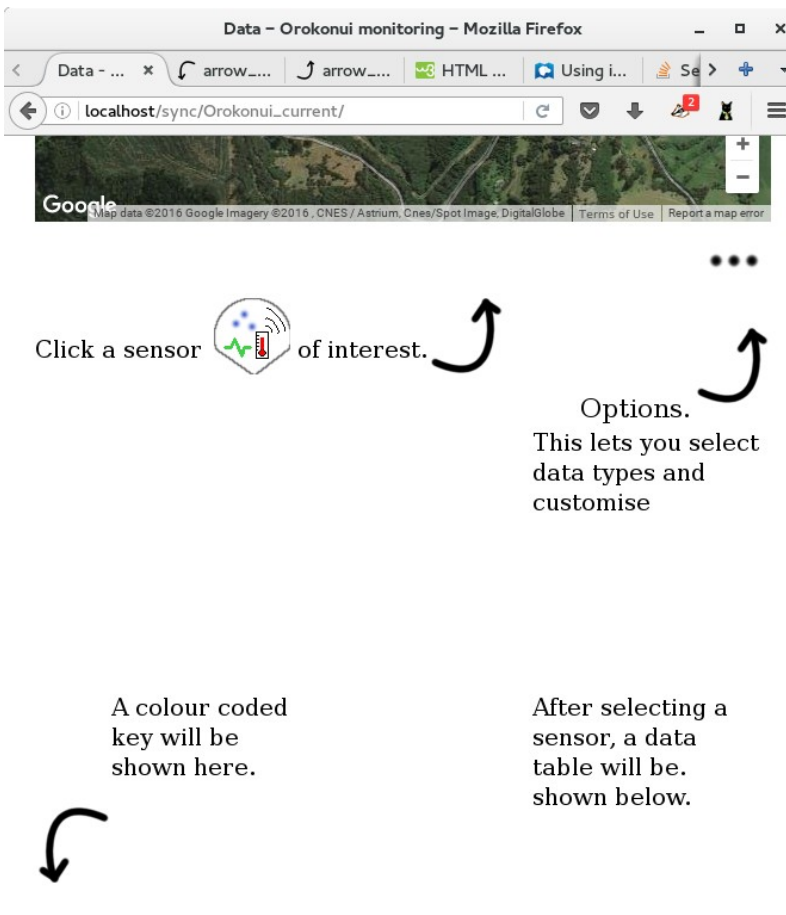
This diagram could be drawn to the canvas, when on the data page when it is not in use. This is the only page that would need these style instruction.

With a canvas, I can draw the text with the filltext() function, and dynamically change text size according to screen space, while adding in icons and arrows with the drawImage() function.

My design can be seen right, red is an annotation and not part of the design. Yes, I know I need to clean my whiteboard.

I set about creating a set of arrow icons in Gimp, which is an image editor comparable to Photoshop. It stands for Gnu Image Manipulation program. (Gnu being a recursive acronym for Gnu's Not Unix - Foss really love acronyms.)





I am using Gimp because it has an excellent selection of painting tools and effects, from lightning shaped brushes, motion blur effects to a things that turns any image into a 16<sup>th</sup> century style oil painting. It can export as almost any image format, including ico, which is super useful for packaging Windows applications. I will be using the PNG export option for all my web resources since it is specially designed for web stuff (PNG is Portable Network Graphic.) I created a 128x128px arrow using the paths tool for smooth curves, some antialiasing and a transparent background. I then scripted the formation of the diagram help in JavaScript, the result is shown left.

Click a sensor of interest.

Options.  
This lets you select data types and customise

A colour coded key will be shown here.

After selecting a sensor, a data table will be shown below.

For mobile; I used different font sizing, as well as positional change factors. (I.e: width as changed by a factor of 2 for mobile, whereas height was changed by a constant amount.

```
fillText("text", 50*width_factor, 50+height_change);
```

For image resources, I also created a resources div, which is hidden from view, but contains all the img tags with ids needed for canvas drawing.

I pushed it up to my server and tested it on mobile, this time also comparing different phone types:



As seen, it works on all bar the Galaxy, which seems to show the desktop version of the instruction diagram. I suspect this is due to the Pixel density, all of the phones have a much higher Pixel density than a standard desktop monitor, and most laptops.

However, I suspect the Galaxy is reporting real Pixels, whereas the others have their browsers setup to report Pixels of equivalent size to a standard desktop for compatibility with websites like mine that judge based on screen size. (Only the Huawei has a width lower than my threshold.)

From this test, I need a better way to detect if a device is mobile or not, and cannot do it based on Pixel widths alone. A way to do this is by getting the user agent, with `$_SERVER['HTTP_USER_AGENT']` on PHP. I can then look for the substring 'mobile' and echo out the appropriate style sheet link/JavaScript variable.

For example, the user agent of my phone is:

```
Mozilla/5.0 (Mobile; rv:34.0) Gecko/34.0 Firefox/34.0
```

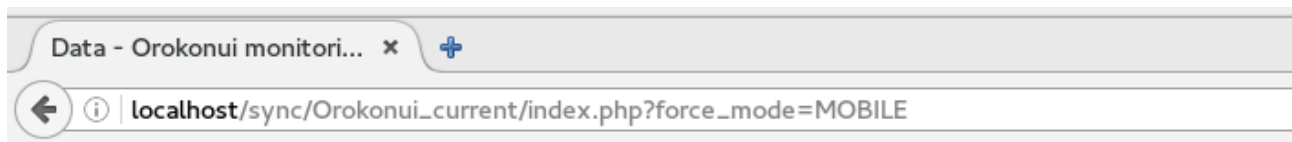
And my laptop's is:

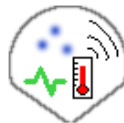
```
Mozilla/5.0 (X11; Fedora; Linux i686; rv:45.0) Gecko/20100101 Firefox/45.0
```

From this test, at least in the Firefox user agent, mobile is clearly identified. Therefore this is a system I could pursue for finding if mobile or not.

However, user agents are whatever the browser's creators make it, and therefore if a new browser was made, post the creation of my interface, it may not have a recognisable user agent, and not follow the convention of having the word 'Mobile' if mobile. For this reason, I will keep the size detection as a secondary system.

```
If !('mobile' in userAgent){if(size < threshold){echo "mobile";}else{echo "desktop";}}else{echo "mobile";}
```



*Click a sensor  of interest. ↻*

I then implemented this system, and it works well. I also added a `force_mode` option through the GET method, so that I could test the mobile site on desktop, and visa versa. This was all then made redundant, when my teacher pointed me towards a new development in the HTML5 standard which I had not heard of called 'viewports' (yes me, missing tech news!) This lets me define virtual Pixels as a fraction of the browsers physical width or height, and then use these as units when defining sizes.

### **PrototyPing**

To see what it would be like have this interface on the internet, I did just that. I already have my own personal website, hosted on a CHIP micro-computer from my house. My router is setup to port-forward to it when receiving requests to port 80 - for web browsing, 22-for ssh, and 4200 - for shell in a box (ssh through web browser). It runs Apache2 with PHP and MySQL on a Debian Linux base, which is the same software as I plan to use for the main server, meaning that it is a fair test. I setup a subfolder and placed my interface so far into it.

This is an important part of prototyPing because it lets me see how the site will run on a real web server, rather than just using localhost.

### **Not using home hosting**

It also highlights a few hosting issues. I am unable have my server up 24/7/365, as would be needed for the Orokonui system. An example of this is when my server went

down on Friday the 20<sup>th</sup> of May because of a power cut. I cannot afford to have an automatic backup generator, so when the DCC fails at power gridding, everything goes down. Additionally, I'm on a slow residential ADSL internet connection, with a dynamic ip address, I counter this with a simple script that finds my ip every 12 hours and updates the DNS, but this is not a very reliable system. It works for a personal website, but not for a critical thing like Orokonui.

## Collector devices

---

I will be needing a small, cheap, low-powered computational device, with internet connection and serial input capabilities to collect data. This will sit inside the Orokonui visitor centre building, and be connected by serial to a Picaxe radio receiver. It's job will be to collate the data from all sensors, which it receives over radio, and then pass it on to the server through the internet.

### **CHIP**

The CHIP is a small credit card sized ARM computer. It is a very new device, which was funded through kickstarter. The company behind it - Next Thing Co. - is still fulfilling kickstarter backer orders. I was one of these backers, and already have a CHIP device. It runs a full on Debian Linux based distribution, and can be booted into a shell system to save on resources and power. However, being a new device lacks the documentation needed to effectively use the General Purpose Input Output (GPIO) Pins. These Pins would be needed to input serial data.

### **Raspberry Pi**

The Raspberry Pi is similar to the CHIP, but has been around for much longer and has far more documentation. It is slightly more expensive, but has the ability to display through HDMI, rather than composite that the CHIP uses. It also has less brown-out problems, which frequent the CHIP due to it's low current power controller. For this reason, I have ordered a Raspberry Pi Model A+. I ordered the A+, this is not necessarily as the final solution that I will use for Orokonui, but as a product version that I will be able to use for other projects later, and test Orokonui code on. It has the full 40 GPIO Pins, over the model A's 26, and has a smaller footprint and power usage requirements. However, the A+ doesn't have an Ethernet port, which would be useful for Orokonui. For testing, my router has a super useful USB1B port for wired USB internet connections. Again this is not for Orokonui but I am thinking ahead to other personal projects. I will look into getting Orokonui a different model (one that has Ethernet) as the project nears implementation.

## More Help

---

### **Importance**

This is hopefully unnecessary, if I have designed my system intuitively enough, a help page should be unneeded. However, I do have a few advanced features in the option menu that need more explaining than I can do with an instruction diagram. For this reason I created a small instruction page.

In writing instructions, rather than going over every single feature, research shows it is better to be task orientated.

Having help fits in with the 12 Usability Heuristics, which I am using as guidelines to build my site around.

## Development Software

In my development I am using text based coding. For this I need a variety of text editing software.

For development, I am using the following text editors:

- Atom. This nicely highlights syntax for many languages, including PHP, HTML, JavaScript, and CSS. It has an excellent colour scheme for readability and looks nice. It is able to edit multiple files at once with the use of tabs, and has auto-completion of known programming languages. Unfortunately, I only works on 64bit computers, so I am only using it on my home desktop.
- Komodo Edit. This is the next best thing to Atom for a 32bit computer. I am using it only tablet/laptop hybrid computer, while away from home. It has a slightly less nice colour scheme and interface design.
- Gedit. This is a simple text edit, that lacks features such as auto-completion, but is much less resource intensive, and starts more quickly. I use it for quick edits.
- Nano. This is a command line interface text edit. It lacks a graphical features, however is useful for quick edits to the copy on my server through an ssh terminal.
- PyCharm Community edition. This is a full IDE, rather than a text editor, specifically designed for python. It includes debugging systems, advanced syntax highlighting, prediction, and error detection, inbuilt version control systems (VCS) and heaps of other feature's I've never needed to use. It has great workflow, with folder views, and tabs files, with lower console and output screens, or, for extra productivity, a distraction free mode, which removes any clutter and allows concentration on coding. I have used it for all my python development thus far and cannot recommend it enough.

Additional tools I am using are:

- Bash. This is a terminal/ Command Line Interface I use to control my server.
  - Scp. Copy command over ssh, works like cp but over a network. Allows uploading/download to/from server.
  - Ssh, allows remote access to my server.
- Git. Version control system that I use in the later half of development once I have a viable product. This is used to put source in a public repository to for fill my open source requirement and also allow easier syncing between my computer(s) and server.

Please note that I have a very fast usage cycle, and change software many times a year, based on changing preference or just further exploration of the FOSS world. For example, I started this project on a Manjaro Linux Base system, and have now switched to Debian. I would argue that this ability and freedom to change is fundamental to my understanding of a wide variety of software and technology in general, and for it's education merits alone, free open source software is far superior to all else. This is another reason I am morally and ethically compelled to release this system under a copyleft license.

### Interface meeting / review + next steps.

---

On the 24<sup>th</sup> of May, I met with Tony to discuss my prototype interface and next steps.

The interface was overall good, but I should consider these things:

- A status of each gate, rather than one overall status log system
- Changing the icon for the sensors, as my current one has different sensor types integrated into the one icon and therefore is confusing where to click. (It looks like click different parts selects sensor types when it is all one generic image.)



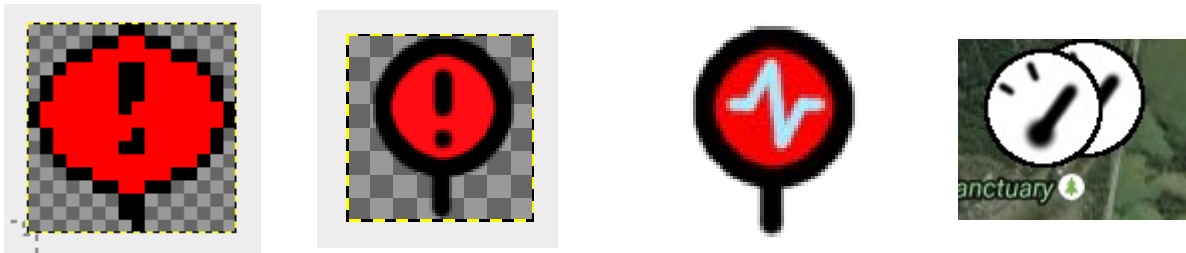
Next, I need to work on further testing of the radios for data corruption. Tony had been testing this himself and was worried about possible corruption.

### Data Corruption Test

To test the amount of corruption, I setup a python script that reads in data from serial, and checks it against a check sum, for an n number of times, logging each time, then logging the losses, corruptions, losses + corruptions, etc, as both numerical values and as percentages. I packaged it with a command line interface (CLI), and it has be found in the resources folder of my project.

### Better Sensor Icon

To make a better icon for the on-map sensors, I first tried to create an icon with 16x16 Pixels, to try and keep it as simple as possible. However, this did not fit with my websites high detail, antialiasing fonts and other images. I then redraw to design on a larger canvas, rather than scaling up which just made it Pixelated. I decided that a graph would best fit a generic sensor icon, and used this style image on a sign background. Initially I used an exclamation point (!) however decided this did not fit well. The final is below, right.



I then abandoned this idea since I didn't think it would fit with the colour scheme of my website, and went with a 'car speedometer' approach. This shows a lot more clearly what it is, as speedometer are already associated with the monitor of data, usually speed, but translates better to generic sensor than an abstract graph line.

### Parity Bits

In early transmission tests, I transmitted 2 bytes, followed by a checksum byte. The checksum was calculated as the average of the 2 bytes previous.

Over 100m, this resulted in a 44.2 percent error rate, and 3 percent error rate over 10m. This is unacceptable, so I used my knowledge from last year's external, about error control methods, which I learnt from the Computer Science Field guide.

One of the error control methods, as described in the CS Field Guide, is to turn the data into a binary grid. To the grid, each column and row has added either a 1 or 0, so that all rows and column are even. In this example I used 43:00101011

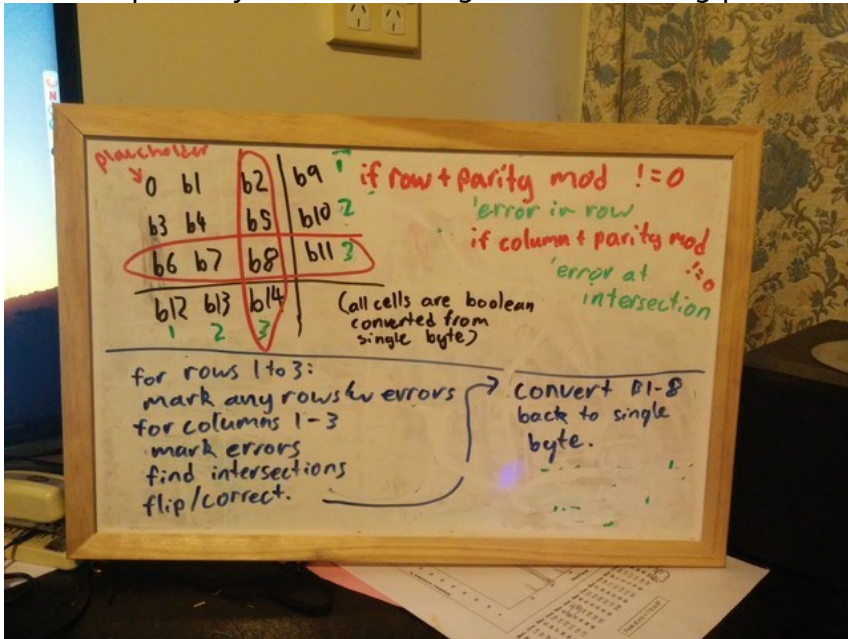
```
0 0 0      0 0 0 | 0
1 0 1      → 1 0 1 | 0
0 1 1      0 1 1 | 0
          -----
          1 1 0
```

I then turned the parity bits into a byte, by adding in placeholder 0s, then transmitted them, along with the original data. On receiving, the Picaxe turns the data stream back into a bit array, and then tests whether the rows and columns are even, if they are, the data is valid. If they are not, it finds the intersection between the row and

column, then assuming that bit was flipped during transmission, it flips it back. If one of the parity bits gets flipped it doesn't matter because there will be no intersection, and therefore it won't flip the wrong bits.

The system has a massive advantage over a simple checksum, because assuming enough data gets through, it can correct small errors, whereas the checksum only alerts that there is a problem. However, theoretically, the parity bits and the data could be flipped, so that they are still even, this would not alert the receiving node that a problem exists, and is why I am still using the checksum as well. I set the checksum to be sent with parity bits so that it too could be fixed if corrupted.

I setup the system according to the following plans and pseudo code.



To test this system, I used the afore mentioned Python script. I set it up to take a sample of 100 transmissions, and set the transmitter at 100m, then 300m.

In both tests I got 0 corrupted packets, and only between 5 to 10 percent lost transmissions, which is acceptable and impressive. A video of my testing can be found at the following link, or in the scholarship docs folder as 'transmission test 28 May 2016.mp4'

There is also an email

thread regarding my tests in [scholarship\\_docs/emails/transmission\\_tests.eml](mailto:scholarship_docs/emails/transmission_tests.eml)

Of course I was testing at my house, which has a different, lower, amount of interference to Orokonui. The only interference I have is my own WiFi router, whereas Orokonui has WiFi, Walky-talky radio system, remote car keys, etc. I will need to complete testing at Orokonui to see how this effects the error rate.

### Necessity of Error Control

Minimising the amount of errors I get in transmission is crucial for my project. Orokonui staff need to know if the gates are left open, as the two kiwi groups must be kept separate otherwise they fight each other. However, if the system gets a false transmission - a transmission error causes it to think the gate is open - it will cause a panic for no reason, and annoy staff. Visa-versa, if the gate is open, but a transmission error causes it to say it is closed, the kiwis will fight, and kill each other.

Either way, it is very important the errors are minimised, and detected. For this reason I will be using both parity bits and a check sum.

### Prototyping the Receiver

For further testing, I need to have a receiver device at Orokonui, that can be remotely updated, and viewed. This will let me place prototype sensor devices at the gates, view their radio transmissions remotely. Remotely, I should be able to write code to interpret the data and update the collector system. The requirements of a deployed collector is:

- Remote access, this can be achieved through weaved ssh.
- Updates to Picaxe receiver, the Picaxe will be connected to the RX serial on the

Pi, and ground. To update code on the Picaxe, I need to additionally connect it to the TX serial Pin on the Pi. Additionally, I can power from the Raspberry by connecting to a 3.3V power rail. This means that the collector will not need have it's own batteries. The Raspberry Pi will be connection to mains power, so the collector system will never need battery replacement.

- Receiver and Transmitter modules attached to the Picaxe.

The Picaxe needs to be placed on the roof, because this is the best place to get radio transmissions. One way this would be overcome is to repurpose an Ethernet cable, as these are designed to carry data over huge distances using a system of twisted pairs.

Tony tested the use of an Ethernet cable to download data to a Picaxe. His email is as follows:

HapPily downloading programs and looking at serial data through a 20m ethernet cable to the Picaxe.

Tony

This means I will be able to have the Picaxe radio part of the collector on the roof, while the Raspberry Pi is inside with internet connection.

Due to Pin layout difficulties I will not be using the same Pin connections for both programming the Picaxe and getting radio transmissions relayed back to the Raspberry Pi, however, I attached two ethernet sockets to the deployment board, one of which was for reprogramming. This means although a change of sockets is needed to reprogram the Picaxe, there is no need to bring a laptop to the rooftop, which would be dangerous.

### **Security**

For security, I decided that the collector should have 2 passcodes. These passcodes are used for check sum calculation, in a way that the correct checksum cannot be produced with both of them and the passcodes cannot be found by observing the checksum. It is important because it protects against false data being sent by a fake sensor. I am calculating checksums as the average of the sent data. With 2 passcodes, it would now be calculated as the average of sent data plus the passcodes. I am using 2 passcodes as the data is sent in plain text, and only one passcode would therefore only require simple algebra to calculate, whereas 2 is a lot harder.

Instead of hardcoding the passcodes, I made the Pi read them in from a .pass file. This means it is easy to change them in future. However, the Picaxe also needs to know the passcodes. For this, I made it send a config request packet over serial to the Pi on startup, the Pi then sends the passcodes, and it starts normal receiver operation.

The sensor Picaxes will be hard to update the passcode on, however this is necessary for security.

### **Tight Server Security Vs Weak Radio Security**

With the radio communication, tight security is not necessary. For a person to 'hack' the system, i.e.: inject false data, they would have to have a transmitter within Orokonui's radio range. They are confined by physical constraints. The only people potentially near Orokonui are farmers, and those visiting Orokonui tend to be tourists. Both of these people groups have a low likelihood of wanting to inject false data, therefore super tight security across radio transmissions is not needed.

However when the collector sends data to the server, I cannot use the same weak security techniques. On the Internet there are no physical constraints; anyone, from anywhere can send false data to my server. For this reason my server needs to be able

to verify data comes from the collector, through a much strong encryption technique. This is why I am using RSA encryption as afore mentioned for the server, but not over radio.

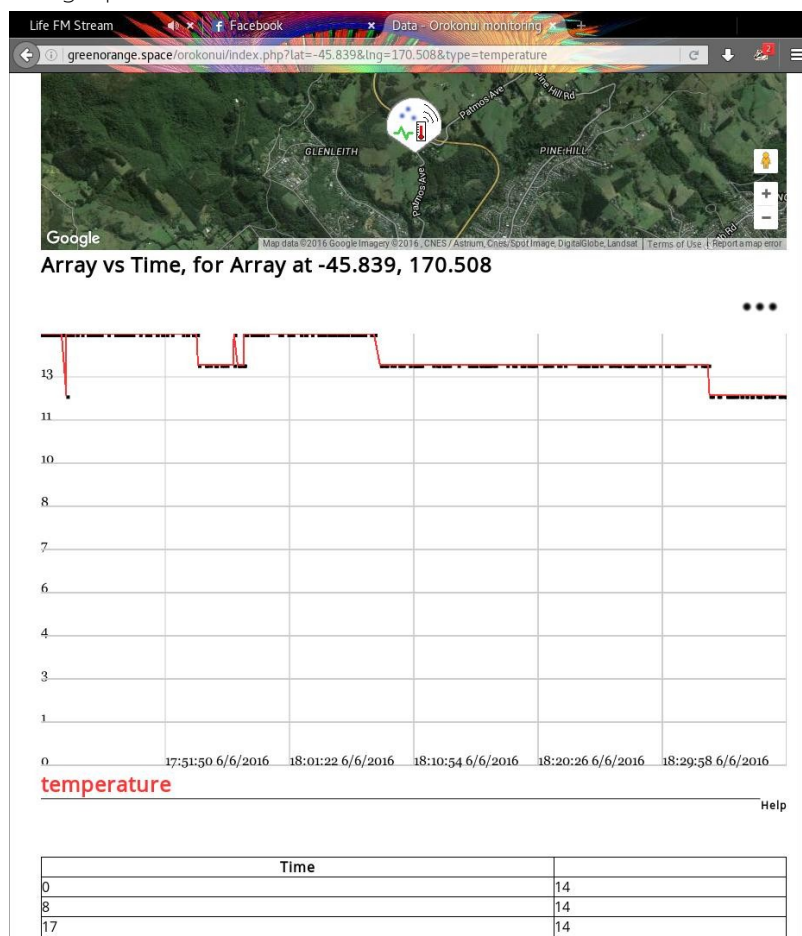
### A full test

On Saturday the 4<sup>th</sup> of June, I did a full test of the sensor, collector, and server systems, minus alerts. Here is my email describing this test:

Subject: Full tests (sensor > collector > server)  
Date: Sun, 5 Jun 2016 15:52:57 +1200  
From: William Satterthwaite <william.satterthwaite1008@gmail.com>  
To: Andrew Eee <[Picaxe@clear.net.nz](mailto:Picaxe@clear.net.nz)>, Julie McMahon <[jm@kingshigh.school.nz](mailto:jm@kingshigh.school.nz)>, Mary.Seelye Tony.Stewart <[mail@mesajs.com](mailto:mail@mesajs.com)>  
Attachments: Docs.zip (6.7MB)

Hi all,

Yesterday I completed the first full test. I had a sensor detect the temperature, and send it to my collector node. The Picaxe part of the collector, verified the data, sent an acknowledgement broadcast and forwarded the data on to the Raspberry Pi through serial. The Pi then timestamped the data, matched up the latitude and sensor types through a config file and sent the data on the server over my LAN. My server then stored the data in a database and showed the results as a graph. The test was successful:



However, there I quite I few thing I will look into improving.

- The collector should cache non-critical data, and sent it to the server ever hour or so.

Currently, it contacts the server after even transmission, which means it spend most of it's time calculating  $11205101^{123} \times 2349701$ , due to my tight encryption system, this is a rather large workload for the Raspberry Pi, and takes ~half a minute of calculation, I had to extend the timeout on the server. Of course stuff like gate openings will be sent immediately, I may also have to weaken the key strength.

- Related to the last point, I've tried to multi-thread the Raspberry Pi system, so it checks serial in on one thread, and adds any data to a buffer array, while contacting the server and sending whatever is next on the buffer in another thread. However, this doesn't seem to be work, as I'm not getting serial data in often as I'm expecting.

- Tables show all data ever, they can be miles long, I will need to implement some sort of 'next page' or 'load more' system.

- Currently, the collector sends an 'a' as acknowledgement, but doesn't say what it acknowledges. So if 2 sensors were to send at the same time, the collector would only be able to process one, when finished it would sent an 'a' and they would both see that as an accepted package, the one that was missed would never resent. I will need to include an id in the acknowledgement system.

I also have some specs on how the system works, some light reading :) They are attached.

Cheers,  
William.

Attached to this was full system documentation, explaining how different components (modules) worked and interacted together.

Specificity, I outlined the Id and Naming system, Radio Interface, Serial Interface, and Server Interface. To see the full documentation on these systems, see the docs folder on finished project. Remember, this is not designed for NZQA, but a requirement of my project. It is written very differently from what you find here.

This documentation is important for my project, as the project must be maintained, added to, and reconfigured after I am gone. To make this as easy as possible I must provide strong documentation as to how the system works, and how modules connect.

As an overview:

- Id's and Naming
  - Used in Radio communication. Firstly, a type id, this is either “:01” for a collect node, or “:02” for a sensor node. No further id is needed for the collector. The first byte of sensor node transmission is both the node id, and sensor type, with the node's id in the tens column and sensor type in the one's column of a base ten byte.
- Radio Interface
  - I am using baud 600 for the radio communication. This is a very slow data rate, but allows for greater stability. All transmissions must start with the type id. A transmission for sensor to collector must also identify itself with node id and sensor id. The collector must respond with either an “a” for accepted or an “r” for rejected. This must be followed by the node id of the sensor, encase two transmit at similar times. All sensors must include a checksum and parity bits, the validity of these cosponsored to whether a collector will accept or reject. The checksum must be calculated using the afore mentioned 2 passcode.
- Serial Interface
  - This is about getting data from a radio receiver Picaxe, to a Raspberry Pi, over a serial data connection (wired). To get data being sent over serial, a user similarly needs to import a provided serial interface lib to create a connection object and call the get\_input function. The workings behind this

library is described in the docs.

- Server Interface
  - This is how the collector get data to the server. A server library is provided, and this can simply be done calling the send function. Note, a server.config file is needed.

### **Radio Time, And Respecting Other's Usage**

Since I am using a publicly available frequency (433MHz), which is free to use for scientific, research and medical purposes, I need to be respectful of others using this frequency. This means minimising the number, length and frequency of transmissions. I also need to wary of other potential devices using the frequency and transmissions collisions. For my resend policies, I should also implement a random delay, in case other devices have the same resend delay, and therefore collide repeatedly, until both devices give up.

### **Antenna Design**

The following is a discussion with Andrew and Tony about antenna design:

Subject: Re: Antenna configurations.  
Date: Wed, 08 Jun 2016 12:57:12 +1200  
From: Andrew Hornblow <Picaxe@clear.net.nz>  
William Satterthwaite  
To: <william.satterthwaite1008@gmail.com>  
CC: Mary.Seelye Tony.Stewart <mail@mesaJS.com>

Hi  
Have a look at what I have done for the Pingwing monitoring antennae...  
Just use the various ideas and ham something together with the 'sticky  
out bit' approx 200mm  
167mm is the 'technical' correct wavelength but I found slightly longer  
the system to be hapPier

I would be very interested in any comparative range tests  
(error rate versus Horizontal / vertical / axial rotation of the  
following general idea.  
Also you could 'optimise' the antennae length and may get as much as 2x  
the range if you hit the 'sweet spot' where radio antennae are 'tuned'  
to the RF Tx and RF Rx

Suggest 'robust' PVC stub (might even get FREE offcut waste from  
plumbers etc) with GOOD caps on them?

Suggest you could make each stub based on a simple gutter bolt through  
the opposite sides of a vertical PVC Pipe?  
So vertical Pipe (easiest to mount, use a Pipe clip etc and poke  
vertically up in the air) with cross TEE of antennae sticking out  
opposite each other near the top.  
This will make 'almost something called a simple Di-Pole antennae but  
with one side Tx and the other Rx. This will 'semi-help the directive  
and RF properties

See I use ONE directional RF only so a single simple whip on the top is  
good.

Suggested technique....  
'Plated' simple gutter bolts are cheap and easy to solder a wire onto

the top of the screw part and this wire goes to the RF Tx or RF Rx Pin directly with shortest route possible.

A bit of ANY metal but I used fencing wire and made a loop that fits under a washer and is BOLTED with the square nut from the outside. 'GUNK' the whole thing mechanically together with no more gaps or silicone rubber / glue.

Make SURE you make a loop or place an eye protective ball, (plastic bead?) on the ends of the wires  
(This will give the kea something to play with also rather than eating the CAT cable:)

~ Andrew

Subject: Re: Outdoor units  
Date: Thu, 9 Jun 2016 20:41:49 +1200  
From: William Satterthwaite <william.satterthwaite1008@gmail.com>  
To: Mary.Seelye Tony.Stewart <mail@mesajS.com>  
CC: Andrew Eee <Picaxe@clear.net.nz>, Julie McMahan <jm@kingshigh.school.nz>, John and Jan Satterthwaite <jjSatt48@clear.net.nz>

Tony,  
Hold off buying anything for now, I've been talking to my Dad about the casing, and he has a few ideas similar to that of Andrew's. He has a few Pipe offcuts, as well as necessary tools, glues, etc, however, we may need to buy the caps. I will discuss this further with him and try get something knocked up this weekend. You have his email now anyway.  
In regards to the antenna, I think there will be enough range with two separate antennas for the gates, but maybe not for further distance/expansion. The only way I can see to get around this is by using a digitally controlled switch. There would be 2 of these, one for transmit, the other receiver. When receiving I turn off the enable Pin for transmissions, and turn on the enable for the receiver. Visa versa for transmitting. The output/input antenna Pin for transmitter/receiver would first have to go through their own switch, and then combine into one antenna. I just have to be super careful in the code to make sure only one switch can be on at a time. Or would it be possible to build an OR gate? Do the transmitters/receivers have any startup delay, because I can use the enable output on the Picaxe to power these during transmission/ receiving, instead of having them powered all the time. Not important on the collector, but would be needed on sensor devices.  
Cheers,  
William.

On 09/06/16 19:34, Mary.Seelye Tony.Stewart wrote:  
William

Andrew's approach of a (reasonably) waterproof box inside a Pipe, with the aerial coming from a cap over the end of the Pipe, looks good. Would you like me to obtain some at the weekend? Or obtain it yourself, and I'll pay you back.

Then we have to figure how to attach the Pipe to the metal pole over the Visitor Centre (VC), well enough to withstand a hurricane!

Further to our discussion on Tuesday, I see all sorts of issues relating to the separate transmitter and receiver modules and the associated two aerials:

- Cumbersome to mount.

- If the aerials are vertical, over the VC one is likely to be close to the metal Pipe which could drastically change its radiation pattern. Possibly for the best(?), but another variable to test.
- If mounted horizontally, transmission/reception will be directional, strongest perpendicular to the aerials. Not a problem at the gates – the aerials can be aligned appropriately. But at least two of the gates will be in the weaker reception pattern of the VC collector, and these are the gates that are furthest away.
- Hand shaking issues we discussed Tuesday concerning how long the sensor waited for acknowledgement, randomising re-transmissions so that sensors did not interfere with each other, etc.

I've talked this over with Andrew, and we feel that for now it would be best to keep it simple and follow his transmit only method, get it up and running, and run live tests on site. Handshaking could be added later on, but you may find other issues take priority, such as user modifiable rules/timeouts for sending SMS messages, sending the messages etc.

Tony

PS You need to keep track of costs so I can reimburse you. Costing no doubt forms part of you write up anyway.

### **Changes To The Collector - Server Transport**

After my full test, I needed to change the way data is transported over the internet from the collector to the server. During my test, my secure encryption delayed this process considerably. Additionally, it was sending data to the server as it came, which was every few seconds, put a lot of extra stress on both server and collector. It would be far better if non-time-depended data was cached at the collector, and then sent to the server in one larger packet, less frequently. Of course the gate opening would be considered time-depended/critical and sent immediately.

However to have this flexibility, I need to change the way data is sent. Currently, it is done with GET variables for sensor type, value, etc. Instead, to send multiple, I can use a GET variable called data, which receives a CSV string, so it can process lots of variables and data rows within the one.

### **Rules (.PHP)**

Reconfiguring is required, since times change. For example, the Orokonui staff do not want to receive alerts that the gate is left open, if the Ecosanctuary is open – people could be legitimately using the gate. For this reason, I will have to set acceptable times where alerts are not sent.

However, if the opening /closing time were to change, these would need to be changed. Or, the Ecosanctuary could be closed to the public, but still have staff working on the tracks – and therefore using the gates. For this reason reconfiguring needs to be a possibility.

This could be done through config files on the Raspberry Pi, however, the Pi will be running headless, and therefore require some ssh skill to reconfigure. Even worse, static programming, would require re-programming and re-compiling for every change.

However if I use the Raspberry Pi to send the data to the server in a very raw format, I could reconfigure almost everything server side. This could be done through rewriting PHP scripts, or through using the same idea as config files, but instead storing the configuration in a MySQL database.



If it were in a database, it could be updated/changed through an admin webpage, with friendly GUI Orokonui staff can understand.

From this page, they could set 'rules', such as

```
if <gate number> <open/close> for <greater than/less than> <time> between  
<time> and <time> on <day> <send/don't send> alert to <ph number>
```

For example:

```
if gate 1 open for greater than 2 minutes between 4:30pm and 8:00am on  
weekdays send alert to 0221838769
```

Multiple rules could be created, with customised alert messages and properties. This gives the Orokonui staff complete control over the system, without writing PHP scripts. All options would be available in a drop down box or HTML entry tag.

And thus 'rules.PHP' was born.

However, with a system like this, security is an issue. The website will be public, so I need some sort of login system to block random strangers from changing the settings. I do not want to worry about trying to get a CA to sign a certificate, as these cost money and/or must be constantly updated. They are only valid for a certain time - usually a year, however free CA's provide certificates that can expire in a matter of months. This is not something I want the Orokonui staff to have to worry about.

For this reason I am using HTTP, rather than HTTPS, however this means server - client communication is not encrypted. This is not a big deal for most of the website, or even the rules page, as it doesn't matter if some knows that an alert will be sent out if the gate is left open at midnight, they can guess that anyway. However, the password/access code to change the midnight alert settings do need to be encrypted, otherwise anyone could change the midnight alert settings.

To send it encrypted without using HTTPS, requires it to be encrypted client side manually using JavaScript. I can do this using, borrowing the key system from data handling.

To do this, I put a PHP script at the top that detected if the browser had been redirected through the backend. If it had not, it did so. If it had, it used the keys given to it by the backend to dynamically create a javascript that encrypted the entered password with those keys. Here the user is show an entry box in which the input an access code. When submitted, a JavaScript function is called which uses the keys to encrypt the string and then redirects to submit.PHP again with the encrypted access code. Submit.PHP then gives the browser an access token for the session, which is made of a hash of the client's IP address. This means it can rehash the IP to see it is the same before, accepting the access token. The IP is also salted.

### **Rules, Design Considerations**

I decided to keep the entire rules configuration area to a single page, or rather a single PHP script, which echoes out different pages based on various inputs.

This means I can wrap the same header and footer around the page, and simply change the page contents variable that this echoed out between them, which makes the server a lot cleaner as it doesn't have a different HTML file for every configuration option.

It also allows me to store common objects and addresses, for example, every link must refer the same token and session id otherwise the user will be logged out as soon as they click. I can store this link string the variable and simply append it with the content needed rather than rewriting all the tokens.

## **More Security Considerations**

Once logged in, a user is given a token, and session id. On the server the token is never stored, however a hash of the client's IP, token and a server stored salt are stored on a database with the session id for reference.

Any time the user tries to update something, download a log file, etc, the token they provide is re-hashed with their ip and salt, according to the relevant session id, then compared to the original hash, if different, a 418 error is given indicating that the server has been turned into a teapot because of an authentication error.

If the hashes match, access is granted, this keeps the system secure, without requiring that a user re-enter their password.

This security is important, because otherwise anyone could potentially edit the sending of alerts, annoying the Orokonui staff, or possibly disabling important alerts, with negative effects on kiwi and other wildlife.

## **Alerts System**

After beginning my development of the rules page, I decided to investigate API's that would let me send text messages from within a PHP script.

The API I found is called Twilio, their pricing depends on carrier:

Vodafone: \$0.10617/sms  
Telecom: \$0.1213/sms  
2 Degrees: \$0.14083/sms

This seems a little pricey, since for me, prepay is \$0.09, and because of my plan, this becomes unlimited (plus minutes and data) for \$19/mo. Twilio do not seem to have a per/month payment system, meaning the cost to Orokonui will depend on how many times they leave the gate open. Twilio supports inbound sms, therefore I could setup a system of querying the server through text message.

Plivo's pricing is better:

Vodafone: \$0.068/sms  
Telecom: \$0.078/sms  
2Degrees: \$0.098/ms

This is much closer to my 9c expectation, however doesn't support inbound texts, no reply queries can be made. This shouldn't be a problem unless some of the Orokonui staff use non-smart phones, as my website should suffice. However all the afore mentioned prices are USD, meaning they will fluctuate with the exchange rate. Currently, 0.09\$US works out about 0.13\$NZ, which is still really high!

Because of this, I will investigate local carriers, although there is no obvious API advertised on their websites.

Mtext.co.nz 250/24h

Of course if a cheap option comes along later, I will be keeping all the text sending stuff within a single function, as part of my modular design specification. This will be able to be swapped out for another API, with very little, if any changes to the rest of the codebase.

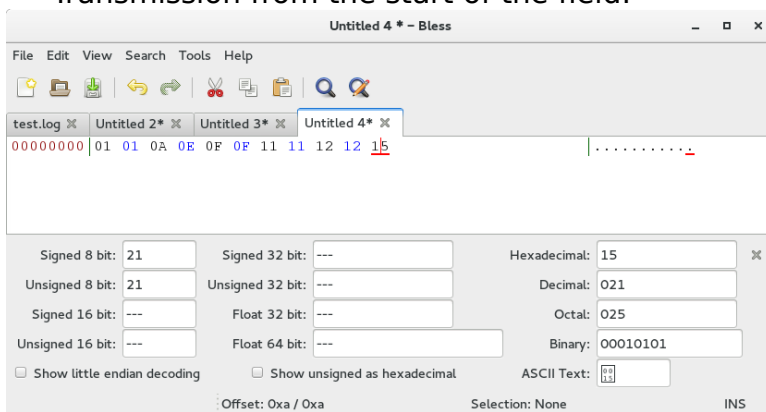
For now I will be using plivo, as receiving texts is not needed, and I am trying to keep the price as low as possible (and a text is a text, there is no 'better text').

### Transmission Tests With Soldiered Board

After soldiering a collector, I did some further transmission tests. To test with noise, I did a test at school. Around school are several school based WiFi networks, (main KHS, BYOD, the iPad's network, etc.) Additionally there are about 1000 students, each of whom have phones creating noise in the cellular range, bluetooth, etc. To test this, I modified my serialdisplay script - which I had made earlier to output whatever data the Picaxe presented to the screen - To output that display to a log file. I then used a Picaxe programmed to send a counter variable across radio link and walked around the school grounds, stopping to send from various locations. I then analysed the log file:

Since my transmissions were raw binary numbers, rather than being formatted as ASCII, I used Bless hex editor to view the data.

Transmission from the start of the field:



Collating this data, I formed mapped diagrams. The full logs are in the logs directory. A summary of all this:

This block contains a collage of images related to the transmission tests. At the top left is a photo of a Picaxe board on a concrete surface. In the center is a map of King's High School and surrounding area, with a red pin marking the school. To the right is a photo of a hand holding a wire attached to a metal fence. Below the map is a photo of a hand holding a wire attached to a green poster. At the bottom right is a photo of a computer monitor displaying a map. Text annotations include '90% of transmissions got through' pointing to the map, '43% of transmissions got through' pointing to the fence photo, and 'Collector, antenna use: wire hung around C# poster.' pointing to the poster photo. A label 'T6' is also present near the computer monitor.

From this, it seems that interference, even on completely difference frequencies is a big factor. Notice how big groups of numbers are missed, i.e: 1-7, during this time I suspect someone did a big download over WiFi or something. The smaller misses may be a small caused by a small handshakes over WiFi, etc. I doubt there will ever be this much interference at Orokonui, except for when walkie-talkies are in use, but it gives an idea of worst case scenario radio environment.

I then did the same tests at home, in a less radio noisy environment. In this test I put the circuitry inside the casing, which also tested it's weather-proofness. I also did tests in varying directions.

The weather test went well, I left it running overnight, during rainy weather, with no negative effects.

This is very important as the roof will have some extreme weather, which my system must be able to withstand.

The spread of tests also let me test the directional aspect.

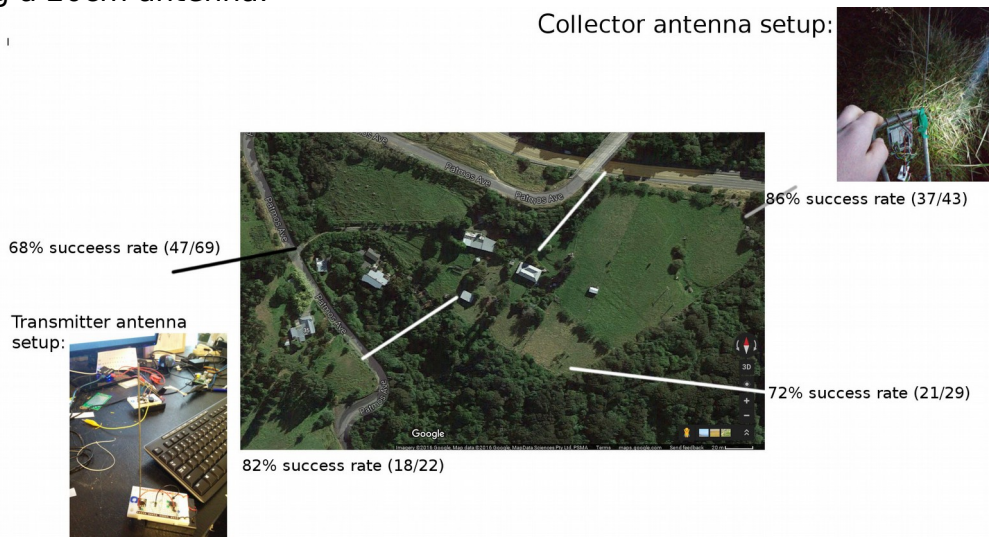


I was quite surprised by the letterbox result, the letterbox is in a valley, far below the collector, so the radio waves where able to propagate around the hill.

The antennae I was using during the tests was metre-ish length of wire hanging down from the pipe casing. This is not the optimal length of pipe or type of antennae I will be using at Orokonui.

Because of this I repeated the test with both a 20cm long wire antenna in vertical and horizontal direction. I equipped the transmitting nodes with the same type of transmitter antenna. This will show which orientation propagates best. After this I also tested the two orientations with handshaking to see which worked best in two way transmission.

Using a 20cm antenna:



## Manchester Encoding

To try to get better transmission rates Tony suggested using Manchester encoding, this involves constantly switching between 1 and 0, which stops the receiver's base line from wandering. After each bit, the opposite bit is sent, this is then discarded.

The radios I am using transmit on 433.92MHz using Amplitude Modulation (AM), so a 1 is a high amplitude wave and a 0 is a low amplitude wave. However the receiver doesn't know where the mid point is so it sample all inputs and takes an average to find the dividing line between 1 and 0. This mean if lots of 0's are transmitted, then the mid point will be lowed and any noise that constructively interferes will be enough to turn the next 0 into a 1 in the eyes of the receiver.

This is where Manchester encoding is so useful, is keeps the average constant.

## Testing System

It is important to test parts of the system while building it, however my modules all rely on each other. For example, my data handling module must be triggered by the collector uploading data. If a want to test it, I have to also test the collector, and radio transmissions and sensors all at the same time. Although it is important to do these large scale tests, it is impractical to do to many of them. For this reason, I decided to make a library of test systems, which emulate module by feeding test data in the format other module would expect.

The first of these test systems is a basic Python powered collector emulator. It allows user input data, with formatting aids, and then uses the same Python library I made for the actual collector to send this to the server.

## Id Systems

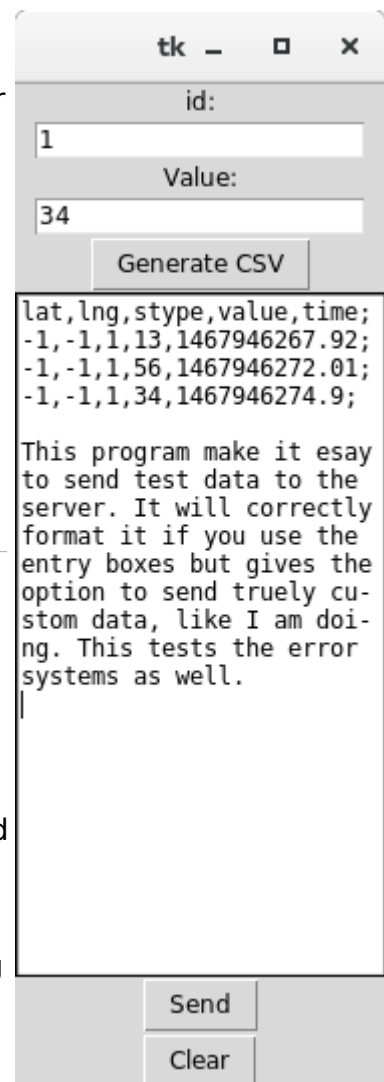
I need to be able to identify which node is sending what data, to do this I am giving each sensor and node an id number. However to save bandwidth on transmission I am containing the two ids into one byte, as two 4bit numbers, giving a total of 16 possibility.

For example, node 10, with sensor 13, would send the id byte 10101101, which splits into 1010 and 1101, which read back 10, 13.

On the server-side I set an integer of \$ids=173, or whatever else the collector sent up, and loop 8 times, putting the modulo two of \$ids in an array and then dividing \$ids by two.

This creates the array (1,0,1,0,1,1,0,1) It then splits it into the half byte and returns 10,13

In writing this code I needed to see what was happening, so I used an echo statement to show me variable manipulation as it was happening (using 255 in screenshot).



The screenshot shows a Tkinter window titled "tk" with a close button (X) and a maximize button (square). It contains two input fields: "id:" with the value "1" and "Value:" with the value "34". Below these is a "Generate CSV" button. The output area shows a CSV list of data points: "lat, lng, stype, value, time;" followed by four rows of coordinates and values. Below the output is a "Send" button and a "Clear" button. A text area below the CSV output contains the following text: "This program make it esay to send test data to the server. It will correctly format it if you use the entry boxes but gives the option to send truely custom data, like I am doing. This tests the error systems as well."

thepppArray255

```
127.563.7531.87515.93757.968753.9843751.99218750.996  
( [0] => 1 [1] => 1 [2] => 1 [3] => 1 [4] => 1 [5] => 1  
[6] => 1 [7] => 1 ) 15.15success
```

The importance of saving bandwidth is that for every extra byte an extra parity byte is needed, the amount of bytes send across radio is doubled. The more sent across radio the longer the transmission time and the higher the chance of it being scrambled or interfered with, this is why I am combining the two ids into one byte. The limitation is that there can only be 16 nodes and sensor types. This is why I am making a modular, open source system. This way an extra byte can be added, if needed, however I doubt this will be needed, as 16 should be enough.

### Alert Storage And Execution

The alert system takes in fields from a html form, the form looks like this:

#### Alerts

Here you can setup alerts to be sent out if certain things happen within the ecosanctuary.

Current Alerts:

Add new:

\*If 's  is   Then send an alert to

Only between these times?  :  and  :

Delay before sending alert (alert will only be sent if the statement remains true after the time delay.)

\*The alert should say:

**Send All clear?** (if problem resolves itself - the opposite condision is meet - an 'all clear' text will be sent.)

From here, a script converts the times into seconds but leaves everything else as is before sticking it in a database called rules.

I am storing everything as is because I cannot execute the rule until I have data, and there is not much point in further processing until then.

### Orokonui Tests

On Thursday the 14<sup>th</sup> of July, I when to Orokonui to test the collector with Tony. We found out that firstly, the 5 volt power supply rail on the Raspberry Pi is unstable.

This power come directly from micro USB powering the Pi, which takes it's share then leaves the rest on the 5V GPIO pin. Since this is coming directly from a wall to USB supply, it fluctuates slightly around 5V, which cause the Picaxe receiver to not get such clear signal.

To combat this, we moved the Picaxe supply to a the Pi 3V3 output pin. This gives a constant 3.3V supply as regulated by the Pi. Since the collector is only listening for incoming signals, it doesn't need as must power as the transmitter will, and is able to work on 3.3V.

Secondly, the board I soldiered was faulty, we found this when it wouldn't work, even with the transmitter in the same room. We instead tested with a breadboard, using the same program. We then sat the breadboard in the window of the office workshop, and ran the transmission tests. From this we found that the two gates

nearest the workshop are in perfect range, but the far two are out of range of the office window. This could be because of the shallow angle the transmitter must transmit. This barely skims the ground, and goes through a lot of thick bush, effectively blocking transmission.

A solution to this would be to either put the collector on a high mast, or put it in the visitor centre, were it is more central.

The importance of these tests is to see what works, and what needs adjusting, before the final implementation, by which time it is difficult to make changes.

I was using Manchester encoding in all these test.

I left the breadboard transmitter, and Raspberry Pi with Tony for a week while I was away. He was doing more tests in this time, as we need tests in all weather conditions. For example, if the bush is wet, it will likely block even more reception.

### **New Checksum Calculation**

I made a new checksum calculation formula to give better security. The formula is as follows:

```
let c be checksum, let I be id, let v be sensor value, let x, y be security codes.
```

```
C = (i+vx)/y
```

```
Note this is calculated on the Picaxe so any overflows (larger than 255) will go to 0.
```

### **Aesthetics**

It is important to make sure the boxes either look pleasing, or are hidden from sight, as Orokonui staff do not want the Ecosanctuary looking ugly. This will make it both a worse place to work, and detract from their marketing and overall viability.

The transmitters also need to be both child and bird proof, as children visiting the Ecosanctuary, may see an antenna poking out of a box and want to pull on it or bend it down. Birds similarly are attracted to bright colours, and may peck at the antenna or casing. Additionally, they will have access to the collector, and may want to peck at the Ethernet cable I am using for power and data.

### **Updated Documentation**

As before; it is important document how each part/module of the project connects so that if future modifications are needed, they can interface with existing parts. I updated the documentation to the new checksum calculation, and included the Manchester encoding.

### **Recreating Status**

After working with the inputs for some time, I went back to improving output, that is the processed data my system gives. One of these is the status page, which shows a log of all the current issues at Orokonui, dynamically updating to show new issues and remove old one.

Before, I had been giving it placeholder data, and had the ability to remove issues from within the web page. This is problematic, as problem could be dismissed without being properly fixed, instead the system should auto-remove fixed issues, but only as it detects they are fixed.

There were also formatting issues with the old version, for this I created a formal documentation of how the log file should be formatted. For this reason I decided it would be better to start over, rather than try to fix the old version. This shows the importance of having a clear formatting spec and plan.

Other considerations for the status page is it would be good to have an 'at a glance' status of all the gates, such as indicator lights across the top of the page, and it must be mobile friendly, as staff will want to check it on the go. With my new design I can

also implement extra features, such as a sorting system, highlighting entries, by there type, etc.

I made the new status display the data in a table, rather than simply line by line as the old did. This lets me display meta data, such as time and sensor types, in a much easier to read way. I also took away the ability to remove critical alerts, as this could result in issue being removed without being fixed.

So I made a test program that toggled Pin 4 on and off, then while Pin 4 was on, tested the voltage of Pin 3. I connected a jumper lead between the 2 Pins, and made it output the Pin status on screen. This gave mixed results, it did not seem to work at first, but did when I instead connected Pin 3 to the supply voltage. After being first connected to supply, it then seemed to work with Pin 4, but only after having a supply connection.

I took my multimeter to this and found that Pin 4 gives 0.6V, whereas the supply is 4.5. This is most probably causing the issue. The problem with this is that to save power I need to be able to toggle the power, to save it while in sleep mode.

### **Urgency and repeated data transmission**

Due to the complications of having two antennae and both transmitter and receiver on every device. I have decided to go with one way transmission only. However to ensure that the data gets through I will have to send it multiple times. If there is a 50% change of the data not getting through, and I send it 10 times, then  $0.5^{10}$  is 0.09765625% chance of the data missing, so sending multiple times significantly increases the chance of data being received.

Of course this isn't straight probability, something like snow or rain would significantly reduce chances of successful transmission. Making the more times it transmits the better.

### **Side By Side Web Interface**

During class in Week 1 Term 3, I got two Digital Technology teachers to critique my web design. Their feedback is as follows:

Mr. Smith:

- Options need to be identified more clearly
- Variables need to be identified more clearly by name
- Needs less ambiguity and user options eg Pick from several Colours rather than entire hex range, Pick from several line weights rather than a number
- Time/Date could be made more user friendly
- Time in output needs to be readable, not a number
- Sensor icon needs to be smaller
- Latitude Longitude also needs a name (keep L/L as well)

Mr. Greenfield:

- Have the graph and map side by side.
- preset option for common use.
- change setup to admin.
- In rule setup have select location rather than select node.
- Header way to big.
- Keep help to one page in settings, rather than on each page.
- Dividing line needs to go to bottom.

Based on this, I first set the map and graph to be beside each other, rather than underneath. This means that the user doesn't need to scroll, as this detracts from the browsing experience of some people. This is the most major change, the others I will get on to as I get time.



## **Text Delays**

There are times in which a delay is needed, and sending the text message shouldn't be instant. For example, if a gate is open, there should be a delay before sending the alert, rather than immediately. If the text were sent instantly, the alerts phone would get a text every time a visitor opened the gate, which is it too much. However, if that was delayed by a minute or two, then rechecked, it would be a useful indicator of if the gate were open.

When the system gets new data it checks the rules, and sends messages, to add the functionality of delays, I set it to also check if there was a delay in the rules, and then if so insert the to send alert into another database table. It added a time to send entry, as the current time plus the alert value.

Now I needed something to send the delayed alerts. For this I used Cron, which is a scheduling program. I created a PHP script that re-evaluated the alerts and sent them if they were still valid, and deleted them otherwise, first checking that the time had expired. This is a script I would run at a short regular interval with Cron.

However Cron can only schedule by minute, hour, day, month, or day of week, but not second. So, I made a shell script that runs the PHP script 4 times, at 15 second intervals, then set this to run at once each minute.

I used PHP for the script, even though I did not have it displaying a web page because it allowed me to import by main script and reuse the checking and texting functions. Additionally, I already know how to access the database in PHP, rather than looking up how to do it in some other language.

I used shell script because it is the easiest and simplest to use to launch other programmes from within.

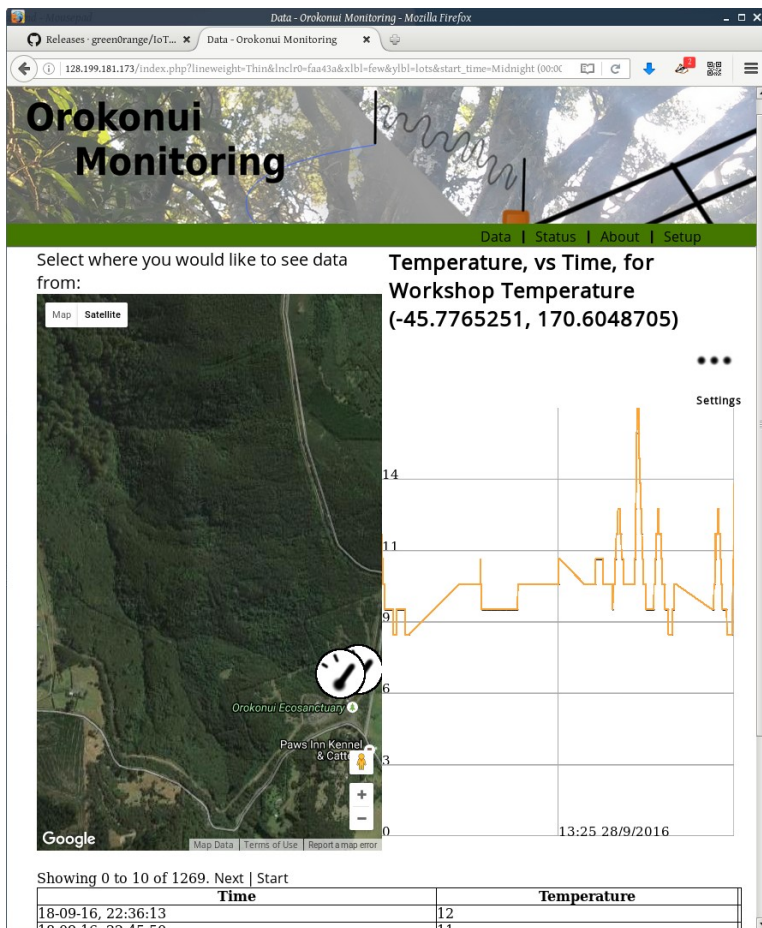
## **Creating a Colour Picker**

I decided to create my own colour picker as a response to the feedback I got. I decided to code it myself rather than find a colour picker library because it is a relatively simple project and an opportunity to practice my JavaScript skills.

I created colourPicker.js as a stand alone file, rather than a function in index.PHP. This means it can be changed and reused independently of the index page. I placed this in the resource/lib directory. For the implementation API, I designed it to place a colour picker inside a div, and output hex code to an input. This gives the option for someone to input their own hex code if they want, or could be hidden in a 0 width hidden overflow div, like my resource images are.

To make a colour picker, simply execute the loadColourPicker function and pass it the div to place into, and the input to output to. When I initially implemented this, it it only worked for one of the inputs, this was because it was using a preset id on all implementations of the colour picker canvas and was then unable to differentiate between them.

I fixed this by giving each instance an id, which was handled internally by colourPicker.js so that I didn't have to change my API.



## Tidying Up the Web Interface

After breaking the web interface by trying to implement the feedback I got from Mr Smith and Mr Greenfield, I had a Bug Fixing Party and tidied it up.

Left is showing the temperature graph. It works as expected, with the option page much more refined, but with less options. (drop downs rather than input numbers, colour picker, etc).

I personally prefer to have all the power; define exact numbers and hex codes, but that's why I'm in the 1% who use Linux, and shouldn't design any 'control panels' as they will be far 'too complicated.'

This shows that unless you are making an interface for yourself, it is important to get testing from the people who are going to use it, as you have different expectations and expertise than the end-users.

## Gate Devices

The gate devices need to be relatively easy to setup and configure. Since I am using a Hall Effect sensor, they need to be calibrated to the magnetic field around each gate.

To do this, I am writing a setup program that will run on the Picaxe before the main program executes. The setup uses a button to know when to take readings of the magnetic field, and then stores several readings in memory as reference ranges for the main program. Here are the readings and ranges it takes:

Closed range = closed reading +/- 5

Unlatched range = unlatched reading +/- 5

Open range = everything else

However, I then decided against this method. In this, all calculations happen on the gate device. This means I am limited by memory and processing of the Picaxe, and am unable to design a program that tracks magnetic drift and accounts for it.

So, to combat this, I decided to make a system that sends the threshold values, and the raw data to the server, where ranges are instead tracked and processed server side.

There are both advantages and disadvantages to these approaches; here I have created a table of them:

Using Picaxe		Using Server	
Advantages	Disadvantages	Advantages	Disadvantages
Less power usage, the Picaxe is not required to send every magnetic change, but only the significant ones.	Low processing power and storage, therefore it is unable to track changes over time. This is limited due to not being able to store full history, and not having enough processing power to dynamically track changes. Also unable to be updated easily if new tracking techniques are needed.	Able to track and store data basically indefinitely. Able to run much more complex algorithm to achieve much more accurate tracking. Able to be updated easily if better tracking is needed or new systems are developed.	Picaxe uses more power to send data, most of which will not be of importance in the immediate term. Although the server will still log it and use for generating new thresholds long term.

### **Fitness for Purpose - Beyond Orokonui**

As I have progressed through this project, and realised it's potential, an increasing key consideration of mine is how this system could be used beyond Orokonui, in other ecosanctuaries, organisations or even home monitoring. As the developer, I would like to see my work reach it's fullest potential. Part of developing for other unknown areas is to make it extremely flexible. The other consideration is to ensure that others have the freedoms to take it as far and to whatever application they want. To do this I have made it free (as in freedom) open source software, (FOSS) as well as modular. This helps everybody, including Orokonui, as if their needs change in future, they will be free to adjust the software, rather than being locked down by a proprietary license.

Therefore, I am using the GNU General Public License, version 3 (GPL3), as this has strict enforcement of freedom, rather than being permissive, as licenses such as MIT are. This means that if it is combined with other software, or has changes made to it, those changes of their added software must be licensed, or re-licensed under the GPL3 as well. This makes sure my vision of everyone being able to use this system is enforced, and my code cannot be stolen for use in proprietary products (at least not with out the FSF pushing a law suit).

As I have explained above, only with an open source license can my project be fit for purpose in the broadest sense. Another step in this has been to make it modular. Now I have decided to make it even more modular than originally planned. This means that it can be used as a frame work for implementing any kind of sensor (module) and not only designed for gates.

Tony would like an indicator on the status page for each gate, say green, orange and red for closed, unlatched, open respectively. However, I envision this being used by some who are not monitoring gates, so am creating a system in which custom HTML and JavaScript can be written and implemented into the status page, so that by default it will show the log and nothing else, but gate indicators can be written a loaded in from the admin/ setting page.

Of course for this, I will write the HTML and JavaScript to do the gates, but it will not

be part of the source code, however stored as a setting in the database.

To do this, I need to have a API which I can lookup gate values from, here are the necessary variable I came up with.

GET\_NODES → returns an array of all nodes. (HR name)

GET\_SENSORS → returns an array of all sensor

NODE.<node\_hr\_name>.<sensor>.VALUE → returns the last recorded raw value

NODE.<node\_hr\_name>.<sensor>.STATE → returns the last state

The state is only for sensors that have a different state form value. For example a magnetic sensor might read 128, which is in the closed threshold. The value would be 128, but the state would be 'closed'. If a sensor does not have an interrupted state, the value and state will be the same.

The API will work by replacing these key words with their returns at runtime.

I have created documentation for the is the docs repository.

The purpose of this is that it will put the power with the users, if Orokonui later decide they want a wind gauge on the status page, they can code that in without even looking at the source. Likewise, other users, can add their own modules, and not be forced to deal with a hard code gate module when they are not monitoring gates.

Likewise for data interpretation, i.e. the thing that fills out the state variable, will not be hard coded into the source but added in settings. I will install the system for Orokonui with these settings already added, but leave them out of the source as others may want do use a system different from hall effect sensors, or may not have gates at all.

## **Server**

To start with, I was given a product key to use Microsoft Azure, for hosting. When I went to set this up, I was given a Windows 8 style of hard to use interface that gave me very little control of the server.

I was unable to install Cron or the like for scheduling, and was unable to manage file permissions, for writing user data and the like. This I would have been able to do with simple command line access, and command line access should be made a usability heuristic. This puts me further off Microsoft, and as I Linux user, that's hard to do.

For these reasons I instead opted for Digitalocean instead, which let's me have a full virtual server, with ssh access, running my choice of operating system and web server stack. This costs 5USD per month. I have the first two months free, as I used a referral link from the Linux Luddites podcast, of which I am a listener. This server, or 'droplet' within their 'digital ocean' as they like to call it has:

1 Core

512 MB memory

20 GB Server grade SSD (very fast)

Runs Debian 8.5 64bit (Jessie) Same as all my computers minus Xorg. - I can use it quickly and efficiently.

On top of which I installed a LAMP stack, with Apache, MySQL and PHP.

This setup is perfectly equipped for what I will be using it for. The server is physically located in Singapore, which gives me a ~200ms Ping. It's not ideal, but not noticeable for webpage loading. SSH is a little laggy, but nothing unusable.

## **Testing the Install Scripts**

For an easier install of the collector, I made an install script, called install.sh. This

runs interactively, asking the user how they would like to configure the installer, such as what passcode they want to use and what the server's address it. This then automatically generates a config file, and handles the startup script. Since Raspberrian is based on Debian, I aimed the script at Debian, using the update-rc.d command, to create an initialisation script. To test the script, I made a Debian virtual machine with Virtual Box and installed and ran the installer, I then restarted the VM to make sure it ran on startup.

This is important because it's one step closer to my project being able to be used by other organisations, and by people of more limited technical know-how. It makes it more useful in the broadest sense.

## Prototypes

---

### **Temperature Testing**

As a prototype to sensing the temperature of the tuatara enclosure, and to test the radio, uploading, server data display and modularity of my system, I decided to make a temperature sensor, as a proof of concept prototype for the transmission, server, collector and temperature sensing. It also proves the modularity of my system, as I am able to use the existing framework to install the temperature sensor with no extra effort. This is described in an email thread with my teacher below.

### **Proof of Modular Concept**

Well with a modular system, the framework's all there. Simply clone the sensor template file, (which has all the Manchester encoding and transmission code) change the id's to represent temperature, and enter the new id's into the server's settings. Then make a temperature checking circuit and change the input data to an analogue digital converter of temperature resister. Test.

The hard part is getting the framework in place. And the gates need an extended framework, like the magnetic drift calculator and testing modules, temperature is easy. (And fingers crossed, it will not all break when at Orokonui.)

I'll put this in documentation as proof of modular concept.

William

On 17/09/16 21:56, Julie McMahon wrote:  
That's excellent William – you never cease to amaze :-)

On Saturday, 17 September 2016, William Satterthwaite  
<[william.satterthwaite1008@gmail.com](mailto:william.satterthwaite1008@gmail.com)> wrote:  
Hi,

I've just made up a temperature board and tested that the collector and uploading process work properly, we have a go! What interval should I wait between sending temperature data, I suggest every 10 minute would be enough. Also, would you like me to solder the temperature sensor up properly or is it fine on breadboard? What date is good for transfer?

Cheers,  
William.

I sent this to Tony, who then installed it at Orokonui the next Monday. The

temperature started rolling into the server during my digital technology class, 11 degrees Celsius.

### **Using Github's Issue Tracker**

There are a lot of things that are not critical to the success of my project, but are improvements of small non-critical bugs that I would like to do or fix. Since this project has evolved into something more than just useful for getting Scholarship of just being for Orokonui (for example, I would even like to do some monitoring in my own house). I will do and fix these things after I have submitted this document, as I have limited time during the year. After I have submitted, I will also be able to accept pull requests, and have more community development. For this potential afterward use, and further development, I am keeping track of these things using the GitHub issue tracker. So that I don't forget any future idea, or non-critical issues.

### **Implementing the Gate**

To test the ranges of the hall effect sensor, I made a model gate at home out of two pieces of wood and a hinge. This allowed me to test what the best position of the hall effect was, as well as verify that the device was working correctly. These tests can be found in the videos folder of my additional resources.

On Wednesday 28 September, Tony and I implemented a magnetic switch on the gate. We first tried to implement a Hall Effect Sensor, as I have mentioned. However, a pin broke off the HES we were using during implementation, and no spare was available. This raises some concern over the delicacy of such devices and how to secure them to a gate.

Our method was to have it poking through a thin piece of wood, with the 3 pins bent backwards in a tape wrapping then being soldered to a 3 core cable. The bending backwards while fitting the sensor into the wood is what most likely broke the pin.

However, we were still able to implement a gate sensor node, since I also had a magnetic switch sensor on me at the time. This can only give a digital reading, so it is not as useful as the HES, but is enough to tell if the gate is opened or closed. It does not give the finer readings such as 'closed but not latched'.

We decided that this would be 'good enough', for a prototype, and implemented this anyway. Here is what it looks like:

The magnet was drilled and embedded into the gate, while we used the same sensor poking through wood approach for the sensor. The Picaxe is protected in a plastic container attached to the side of the fence post. This serves the purpose of being both decent looking (aesthetic) and weather protection. A cable comes out the bottom through a rubber seal, the goes into the sensor. Another wire leaves the rubber seal, acting as an antenna. This would not work for all the gates but since this was



implemented on a gate in relatively near the collector, it works fine.

This will be all I will implement for now (until NCEA exams finish) as I do not have the time for more. It does work as a successful prototype, and is able to correctly tell when the gate is open and closed.

As I have said, this is a long term project, and I will be continuing it through the summer and beyond (I am studying Computer Science at Otago, so will have easy access to Orokonui, at least for 2017). In this time I hope to expand the project with the following:

- Sensor on all gates
- More temperature sensors, to get a better idea of the area, even if some are in shade. Some separate temperature sensor for the Tuatara's
- Rainfall sensors
- Humidity sensors
- Collating this data for an automated fire warning system

As well as expanding the software and frame work of the system.

### **Prototype Fixes and final adjustments**

When we installed the prototype it seemed to be working, but this didn't last for long, as shown in this these emails:

Subject: The gate  
Date: Mon, 10 Oct 2016 11:46:48 +1300  
From: Mary.Seelye Tony.Stewart <mail@mesajs.com>  
To: William Satterthwaite  
<william.satterthwaite1008@gmail.com>  
Peculiar.

Battery voltage was about zero (is the picaxe napping????). Replaced the batteries, and immediately received 4 'gate open' texts on my phone. Opened the gate again a minute later - no more texts.

Looked at sensor\_data. Only 2 gate open records, 1min apart.

Turned power off and on again, and about a minute later opened and closed the gate. No more texts, no more entries ins sensor\_data. LED goes on when gate open.

Tony

Subject: Battery voltage  
Date: Mon, 10 Oct 2016 14:03:56 +1300  
From: Mary.Seelye Tony.Stewart <mail@mesajs.com>  
To: William Satterthwaite  
<william.satterthwaite1008@gmail.com>

Even weirder! All 3 batteries were below 1 volt. But one had reverse polarity???? After few hours, it was still reverse polarity, but only about 0.1v.

Later that week, I discussed this with Tony and Andrew. One of the things Andrew pointed out is that I needed a heartbeat, so that we would know if it was working even if the gate was not being opened.

Then on Labour day weekend, I visited Orokonui, and found that the picaxe seemed to be drawing about 20mA all the time. It should only draw about 0.5 at most. I ended up replacing the picaxe, and making programmatic adjustments to ensure it sent a heartbeat. I then ensured it worked by setting it to text me, and opening and closing the gate, watching the texts come in. I then walked around the ecosanctuary, to let the heartbeats collect up. I was surprised that every 100 seconds I got a text. Luckily, the wildlife didn't seem to be concerned about my text alert sound. I guess they are used to it from all the visitors to the ecosanctuary.

This was because I had accidentally set the texts to happen when the gate was closed. The heartbeat was done by sending out the current gate status, so it was sending 'closed' and I was getting the text for every heartbeat. This was 'accidentally' a much better way of testing the heartbeat. Afterwards, I set it back to text Tony and made sure it only sent texts on open. We then had an email conversation about the regularity of heartbeats:

Subject: Re: Heartbeats  
Date: Sun, 23 Oct 2016 21:54:24 +1300  
From: William Satterthwaite <william.satterthwaite1008@gmail.com>  
To: Mary.Seelye Tony.Stewart <mail@mesajs.com>  
CC: Julie McMahon <jm@kingshigh.school.nz>, Andrew Hornblow  
<picaxe@clear.net.nz>

It is still on the 'testing' setting, where it pulses every 100 seconds. (after 5 minutes of inactivity) I think we should keep this high rate for a week or 2 just to see if and when it is dropping packets. It can be dropped down with a simple change to the program. (if (time % interval) == 0)

Discussion around what the best interval is, is our next step. I think the temperature at 10m is to regular, and a heartbeat should be even less than that of the temperature.  
... Some information about my server API – Tony was interesting in automatic texting...

Hold off calling out the ambulance for now  
William  
--

William Satterthwaite

You can [verify](#) I sent this using my attached PGP signature and [public key](#).

On 23/10/16 20:59, Mary.Seelye Tony.Stewart wrote:

William, the log shows the gate reporting in every couple of minutes. Racing pulse, send ambulance?



## Final feedback

---

Here is the feedback I got from stakeholders after implementing the prototype.

### **Tony's feedback**

“This project had to address a number of difficult issues. Such as unreliable radio links, detailed user rules for when alerts should be issued, hardware construction, not to mention three different computer operating environments.

William is clearly very competent technically, thorough, and determined to have things working properly. Initially unfamiliar with much of the componentry, it has been a great learning experience for him. He explored alternatives, and spent time devising and testing solutions for the trickier issues. Of particular note was his attempt to use Hall Effect sensors to try and distinguish between gates that were closed (kiwi can't escape) and those that were closed and securely latched (can't blow open in strong winds) - that impressed sanctuary staff. While time and hardware issues meant he did not complete this component and reverted to a reed switch, it's something worth pursuing.”

### **Andrew's Feedback**

“The project William has been working on in 2016 is a comprehensive IoT remote monitoring system. He has built hardware, modeled, tested and debugged starter concepts and developed code working on a diverse range of IT systems interconnecting PIC micro controllers, Through Raspberry Pi to the end point GUI interface. The end result is a practical Remote Telemetry Instrumentation system. This project is 'deceptively complex' covering well, almost every aspects of design of a model IoT system, in places to very low levels of detail.

- Programming small battery powered sensor instrument interfacing to PIC based ultra low powered micro controllers
- Efficient interfacing of digital and analogue sensors: maximising power, signal levels, modes, ranges of the sensor physics and practical electronics
- Design and practical site testing of a range of discrete radio devices and diverse antennae systems for instruments and the receiver
- Design of an efficient original low level radio serial radio data protocol appropriate for use in the 433MHz ISM radio spectrum
- Optimisation of time, power and all aspects of the data packet: Rate, Preamble, Serial:ID, Multiple Data Channels, Check Sum and Error detection.
- Interfacing the local radio mesh serial data protocol to TCPIP, www and end server Orokonui Raspberry Pi data concentrator, filter, interface.
- Development of a range of protocols for secure passing and handling of alarm and low level data messages passing between remote systems Aspects of how the server and user and GUI were worked on I have had less own personal experience or knowledge but information. The project has been well described and formally documented in:

<https://github.com/greenOrange/loT-ecosanctuary-web-monitoring/releases/tag/0.1-beta>

William has worked hard and developed his own approach ideas almost entirely independently. He has taken base concepts as explained and adapted them coming up with some great original approaches and ideas. Located in Taranaki I have only been able to mentor this project very occasionally, in conversation when visiting the Dunedin area. I was impressed from the start with images of home / shed based radio

telemetry range test results within days of handing over some base radio devices and materials .

William has managed all this and handled competing time, school, weather and access to a controlled remote site. At times there have been real hold ups and frustrations with time frames, site access, and replacement materials. He recently managed an on the fly code and circuit modification towards the end when parts broke. I am sure it will be a most memorable year. I have every confidence that a lot or really deep learning has taken place. Hardware + Software and an inside, practical working knowledge of IoT + can do attitude will take him a long way. William will be a valuable asset to any learning institution, team, workplace that this line of work takes him.

~ Andrew”

### **Teacher Feedback**

“As William’s teacher, my role was to help him develop and apply his technological knowledge to solving a real-world problem. William’s ability to synthesise a broad range of technical, social, environmental and ethical considerations into the development of his solution has been outstanding. He is a natural technologist with an inherent ability to look at a problem from many different angles and delve deeply into possible strategies for implementing solutions. Not only is William’s technical expertise is beyond his years, but so are his maturity and ethical practices as demonstrated by his commitment to developing open source, modular solutions that will provide benefit to the wider community. He truly believes and understands how knowledge is progressed through sharing of ideas and collaboration. William works independently to research, model, and prototype possible solutions with excellent project management skills. However, he is also able focus on the fact that the end goal is to benefit key and wider stakeholders. Therefore, despite his ability to work with independence, he also communicates his ideas frequently with his stakeholders and seeks to take on board and implement their feedback. William’s approach to development of this project truly epitomises the goal of producing outcomes with a focus on fitness for purpose in the broadest sense.”

### **Final Evaluation**

---

As seen above, I have worked with, and satisfied the needs of my stakeholders. Here is my final evaluation of the project.

### **Context considerations**

For my final outcome, I have had to consider many broad ranging context considerations. Some of these have put constraints on the function of my product, such as the low power usage requirement, meaning I am unable to constantly shine a laser across the path and detect traffic - it would use too much power. However, I have managed to counter these limiting factors by synthesising a modular system, so that if I weren't concerned about power, I would be able to add in a traffic counter with lasers and have it work with the rest of the system. Similarly, I have considered the fact that other people's needs will be different from that of Orokonui's and made the project open source so that other users can modify it as they see fit for their individual needs.

Another example of one of my considerations and challenges for me was the need to detect and handle data corruption. I overcame this with Manchester encoding which is something that I learnt a lot with on the project, as I had not looked into data transmission before. If the transmissions didn't need to be reliable, I would not have ever done this.

Again I had to overcome another challenge; APIs and documentation. I had to look into the documentation of various other projects and libraries. This helped me understand how good documentation was important, and how to most effectively use other's - and make my own - APIs. An example of this was implementing the texting function, and trying to use the Plivo texting API.

The security concerns have been a big part of this project. Implementing the RSA cryptosystem, has been a challenge and highlight of the project. During this project, I have had to consider where people could potentially break in to the system to change settings or inject false data, and how likely these cases are. For example the radio transmission while using security checks, are much weaker than that used on the internet, because potential hacker are constrained by the physical world, whereas people online are not.

A big challenge for me was designing the interface. This is because I tend to design highly configurable and customisable interfaces, which is what I like. However, others find the many options and presentation of options confusing. This is why it was critical to test my web interface with Mr Smith and Mr Greenfield, who are Digital Technology teachers with an expertise in interface design.

Through considering all of these broad ranging factors, I have managed to create a usable prototype, which I plan to develop further over the coming new year and beyond.

### **0.1 released**

To signify that my software was ready for basic use, I made a 0.1 release on Github. You can find the release document on by following the link below.

<https://github.com/green0range/loT-Ecosanctuary-web-monitoring/releases/tag/0.1-beta>

In it I included notes on how to download and use the software, and where to find documentation. Documentation is very important because it lets other people download and use the software instead of downloading and deleting when they can't figure out how it works.

I also include I final features rundown which is shown here.

- Collect and graph data from mapped nodes:
- Select data between certain time ranges
- Shows usage instructions if nothing is selected.
- Customisation options for graphed data includes colour, weight and markers.
- Data in table form to show exact values.
- Downloadable CSV for own analytics.
- Alerts system that sends SMS messages if programmable bounds are broken.
- Status page shows rundown of all alerts
- Status page can be customised to show detailed data displays according to JavaScript API.
- Login system for making system changes
- Easily deploy new nodes/sensor by assigning id numbers to human readable names.
- Setup custom data interpretation scripts
- Stores data from any authorised collector using the correct protocol.

Alongside this I also credited software libraries I had used, such as GMP and Pikaday. All of these libraries are open source under GPL compatible licenses. It is important to credit where I have used this, and what licensed software I am legally able to use with mine.

Also included in the release document, was my own copyright notice and GPL license header.

## Live version

The live version of my project is at: <http://128.199.181.173/>  
This is a final summary diagram of the project, as a general framework.

