93601

936011

S

TOP SCHOLAR

NEW ZEALAND QUALIFICATIONS AUTHORITY
MANA TOHU MĀTAURANGA O AOTEAROA

QUALIFY FOR THE FUTURE WORLD
KIA NOHO TAKATŪ KI TŌ ĀMUA AO!

Scholarship 2021
Technology

TrapApp

A Crowd-Sourced Electronic Trap Monitoring System

With thanks to

[redacted] of Predator Free 2050
Professor [redacted] of the University of Auckland
[redacted], my technology teacher
Spark NZ

[redacted]

2021

[redacted]

1 Table of Contents

2 Abstract

Aotearoa's native species are under threat from introduced mammalian predators – specifically the rat, possum, and stoat. Trapping is an effective tool to combat these species, but it is highly labour-intensive. Wireless trap monitoring systems are proven to be effective at reducing maintenance costs of these trap networks, if the sensor cost is low enough. Sprung traps in public areas could be displayed publicly via an app, allowing volunteers to reset them. This behaviour could be promoted by gamification and social media techniques. I have created such a system, which performs comparably to existing commercial options and has a BOM cost well below the threshold for economic viability. I implemented the foundations of the public, app-based volunteering system. In the future, these features could be completed and extended. The system was designed to be modular, extensible, and easy for the average consumer to set up – New Zealand is more than just wild bush, so our trapping efforts need to target the whole country.

[redacted]

3 Background

3.1 Our unique environment

Aotearoa rightly has an international reputation for incredible scenery, wildlife, and culture. Native forests, grasslands, and mountains cover 47% of our country, and are home to thousands of endemic species. This taonga earned New Zealand 41 billion dollars through tourism in 2019, which directly employed around 8.4% of our working population. Tourism's status as a cornerstone of our economy has only been highlighted by the impact of COVID – when the world can fly back here, we need to ensure our that our natural beauty has survived.

Our most famous native species are birds like the Kiwi, but we also have hundreds or thousands of unique insects and plants. Many of these are at significant risk of extinction due to introduced mammalian predators (IMPs) - namely rats, mustelids (stoats, ferrets), and possums. Already, at least 75 species of plants and animals have become extinct since New Zealand was settled. 74% of native terrestrial birds are classified as threatened with, or at risk of, extinction (Ministry for the Environment & Stats NZ, 2019).

A hit to nature is a hit to tourism – our visitors overwhelmingly come here to experience our beautiful landscapes and species. A hit to tourism is a hit to the economy, which negatively affects us all.

Economic gains certainly aren't the only reason to save our species. Aotearoa is home to many diverse and unique cultures, but one thing we all have in common is a love of the land, and a desire to maintain it.

To Māori, the land is much more than a resource – it is a connection to whakapapa, to ancestors, to gods and wairua (Timoti, Lyver, Matamua, Jones, & Tahi, 2017). Māori quickly recognised that humans have an impact on the environment and implemented conservation devices like the rāhui – a temporary restriction on the use or exploitation of an area (Royal, 2007). The Pākehā, too, learned to value the land – this even is reflected in the stereotypical 'Southern Man', who lives isolated among nature, caring for his land and sheep.

If we lose our natural flora and fauna, our many cultures are all degraded.

Pests also carry disease and cause harm to agriculture. For example, possums spread bovine tuberculosis. Infections necessitate culling of entire herds of cattle, up to around 1 million per year (Stock, 2018). The disease can also infect humans.

3.2 What we're doing

In 2016, the government committed to the Predator Free 2050 plan, with the goal of eradicating all introduced mammalian predators by the year 2050. It is generally accepted that new scientific and technological developments will be essential to achieve this goal (Norton, et al., 2016). Before discussing these novel technologies, it is important to understand the existing methods of pest elimination.

3.3 How we're doing it

Poisoning, also called baiting, is an effective and proven technique for pest *suppression*, but not full *eradication* (remember, Predator Free 2050 aims to eradicate all IMPs, not just suppress them). Poison may be deployed aerially (often from helicopters), or from ground bait stations.

The most common poison is 1080, of which New Zealand uses around 80% of the world's supply. 1080 works on all the target IMPs, though is also poisonous to dogs and other mammals. Techniques for aerially dropping 1080 have been extensively studied, meaning it is now known to be an effective and cheap tool for pest suppression across large areas. Again, this is not the full elimination which we aim to reach but is a valuable starting point. Ground application of poisons such as 1080 and brodifacoum is also used, though this is not often economically viable where aerial application could be used (Environmental Protection Agency NZ, 2006).

A single aerial drop of 1080 poison can clear 98% of possums and 90% of rats in a targeted area (Forest and Bird, 2018). Since 1080 can't be used everywhere, and doesn't fully eliminate predators, other pest control methods – primarily traps – are essential.

Many types of traps exist. The most common designs consist of a 'trap mechanism' (the conservation equivalent of a domestic rat/mouse trap) inside a wooden tunnel. The tunnel prevents unwanted species from accessing the trap and forces the target species to approach from the desired direction. Traps are usually placed in 'lines', with each trap between 50m and 200m from the last, depending on primary target species. Lines are spaced <100m - 1km apart, depending on target species. Traps lines are checked and reset every 1 to 4 weeks. This is a highly labour-intensive process, requiring a lot of time and to be repeated regularly. Many can only be accessed on foot, though all-terrain vehicles (ATVs) are also used (DOC, 2021). There is also a suggestion that a 'hot trap' effect may exist, where pests will be more attracted to a trap soon after another pest visited it. If traps were reset faster, then the hot trap effect would allow for quicker recaptures, and exponentially more pests caught.

The fundamental issue with traps is that human resetters never turn up at the right time – we always come too late, leaving the trap out of action for a while, or we waste resources visiting traps too early, before a kill has been made.

3.4 Moving towards 2050

Clearly, trapping is an essential part of New Zealand's predator elimination effort, and will continue to be – in fact, as we shift from suppression (which is achievable with only poisons) to eradication, trapping will become even more important. However, trapping is currently extremely expensive compared to other pest control methods. The bulk of this cost comes from constantly checking and resetting these traps. For this reason, technological advances concentrating on reducing trap management workload are promising and would likely have practical applications towards achieving New Zealand's Predator Free 2050 goal.

4 Proposed Solution

4.1 Automatic traps

There are already a great many types of manual trap, designed for different applications against different species. One potential improvement to traps is to make them reset themselves automatically. Such automatic traps do exist in limited capacity today.

However, they are quite expensive and new to market. This means they have nowhere near the years of testing and refinement that manual traps do. There are currently two self-resetting traps in development or on the market - the GoodNature A12/24, and the NZ Autotraps AT220. There is very limited research about these traps, and no consensus on their efficacy.

Both of these rely on relatively pricey chemical lures, compared to simple and cheap peanut butter, eggs, and waste meat, which are widely used in manual traps. The GoodNature traps also use disposable CO_2 cartridges, which cost money and generate waste.

Another issue with these traps is one of data collection. Achieving our Predator Free goal requires us to keep very close tabs on what we're catching, how often, and where. This is easily implemented with manual trapping, as maintainers can simply enter data on a mobile device. Due to their novelty, existing automatic traps have very limited collection ability. This will likely improve in coming years; however, it will still be near impossible for species information and other detailed parameters to be collected.

Options for automatic traps are very limited, meaning their use is constrained to very specific situations. For example, they all rely on gravity to clear dead animals, meaning they must be elevated. Automatic traps are clearly a promising technology, but they would require a lot more development, testing, and research to replace our existing trap designs. On top of this development time, replacing existing traps would be a very expensive, and thus lengthy, undertaking. Remember, we need to act immediately if we are to achieve our 2050 goal.

4.2 Remote trap sensors

Another promising option would be to upgrade these traps with a comparatively cheap wireless monitoring device. A chosen group of people would be alerted when a trap is triggered, allowing both the timing and routes of trap maintainers to be adjusted to maximise predators caught, and minimise resources used. Essentially, we would be able to arrive at the traps at the correct time – not too early, not too late.

These devices do already exist but, much like self-resetting traps, are in their early days. However, they don't require the extensive development and research periods that automatic traps do, since they are retrofitted on already-proven trap hardware.

The immense number of already-deployed manual traps is another key factor. Remote trap-sensor devices would be much smaller and lighter than an entirely new trap, so carrying these into the field would be less resource-intensive than replacing traps with self-resetting ones. The actual cost of remote devices would presumably be much lower than an entirely new trap. In other words, it would be far more practical (and cheaper) to upgrade current traps by installing remote monitoring systems than to replace them with automatic traps, at least until automatic trap technology is further developed.

Gamification refers to techniques applied "*to enhance systems, services, organizations, and activities in order to create similar experiences to those experienced when playing games in order to motivate and engage users*" (Hamari, 2019). In other words, how do we make people think that doing work is fun?

This may seem counterintuitive, but the technique is widely applied already – popular examples are the 'Snap Score' and 'Streaks' mechanics from Snapchat, and Pokemon Go in its entirety. The app iNaturalist gamifies the crowd-sourced documentation of plant and animal species in the environment. A similar approach could be taken for the resetting of traps, leading to a free volunteer workforce.

4.3 Chosen direction

I will design and create a system that provides remote monitoring capabilities to predator traps and makes this information available to relevant people.

Some traps, especially the less dangerous ones on public land, will be visible to the public who will provide a free labour force to reset them. App-based volunteering will raise public awareness of pest trapping and elimination, creating a positive feedback loop of volunteers and interest. Gamification and social media techniques will be applied to encourage user participation.

This system will consist of three main components. These are briefly broken down below:

1. The '**node**' is the piece of hardware mounted on each trap. It detects when the trap has been triggered and requires resetting, then sends a radio signal to a base station, or gateway. This gateway forwards received signals on to a centralised server via the internet.

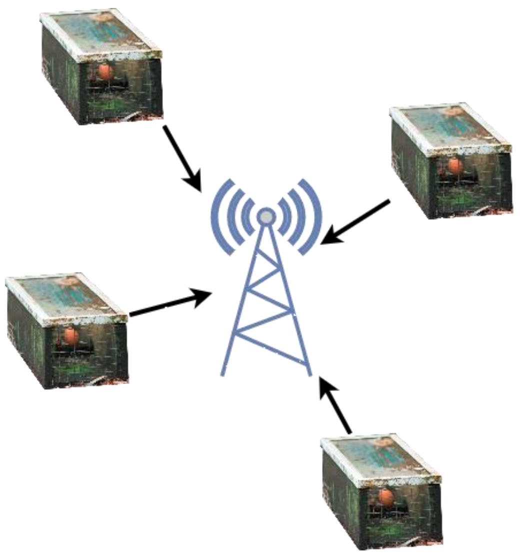


Figure 1. An illustration showing 4 traps communicating with a nearby base station (gateway).

2. The **backend server** receives, processes, and stores information from each trap's node. It provides this data to the app, and handles app-related functions such as logging in. This piece of software will run on a web server.
3. The **app** fetches relevant data from the backend server and displays it to the user in an interactive manner. This piece of software runs on the user's phone.

5 Stakeholder Analysis

Several stakeholders exist for this project, as several distinct groups have an interest in pest elimination or reduction.

My primary stakeholder group is organisations such as the Department of Conservation (DOC), regional councils, and other bodies that manage large areas of bushland or parks. In this use-case, the sensors could be used to aid management of large expanses of difficult-to-access bushland. These traps could be made public.

My secondary stakeholder group is private owners of medium-large land areas, such as farmers and lifestyle block owners. These people often have a use for pest control, but do not have the time or resources to widely implement it. New Zealand is more than just bushland – if we are to eradicate all predators, we need to consider these other types of property, and what will work for their owners. This technology may allow them to balance their trapping efforts with work.

My tertiary stakeholders are those who have an interest in backyard trapping (likely in an urban area) but do not want to invest significant time into checking traps. For this group, the system would essentially be a novelty – they could probably check their traps manually with little more effort. However, plenty of items far more useless than this are sold every day, so this stakeholder remains. Even though the use of this technology would not really be necessary, it may raise awareness of, and prompt participation in, conservation volunteering programmes.

6 Further Research of Context

6.1 Scientific consensus on wireless trap monitoring

Several studies have analysed the feasibility of wireless sensor networks for trapping. A Manaaki Whenua report (Warburton, Jones, & Ekanayake, 2015) conducted a cost-benefit analysis and found that, "Significant benefit-to-cost ratios can be obtained, but these depend particularly on the price of the technology but also on other parameter values used" – for example, the time before the trap must be rebaited, the lifespan of the node, etc.

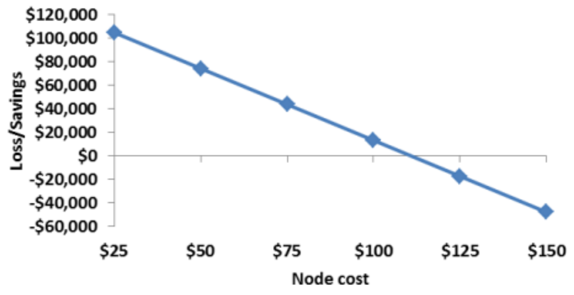


Figure 2. Loss/Savings of implementing a wireless sensor network system by node cost (Warburton, Jones, & Ekanayake, 2015)

Clearly (Figure 2), the cost of each sensor is of chief importance. For my device to be economical, the price must be less than $100. Of course, the exact figures may have changed slightly since publication, but the point remains – this technology is extremely price sensitive.

Another related New Zealand report again found that this technology was economically justifiable, and that, "in a modelled example, we estimated that operational cost savings of up to 70% could accrue from use of wireless sensor networks". They also found that this technology could be helpful for "increasing the quantity and quality of data from wildlife monitoring studies" (Jones, Warburton, Carver, & Carver, 2015).

An Australian study experimented with this technology for dingo-trapping (Meek, et al., 2020). They found it:

> provides a solution to checking traps daily when the distance to and between traps cannot be covered within an appropriate time frame. Although trap alerts can never replace the value of daily trap checking by the trapper, they provide a solution to a management problem, namely, one of accessibility to sites. (Meek et al., 2015)

This proves that the technology works in the real world. (The NZ studies were based on simulations and mathematical models, so it is helpful to have some real-world evidence.)

Overall, there is good modelled and experimental support for the use of this technology. These studies show that it would provide cost savings, and therefore will help us to achieve our Predator Free 2050 goal as hoped. This indicates that it is worth pursuing further.

6.2 Environment

Traps are usually deployed in areas of native bush, tussock-lands, etc. The challenges associated with such environments are as follows.

6.2.1 Weather

It could feasibly rain, snow, hail, or flood. The node must therefore be waterproof, and must not falsely trigger if the detector switch gets wet. The node must not rust, so must be made of plastic or galvanised/painted metal.

6.2.2 Groundwater

If the trap is on or near the ground, it will become wetter than if raised higher. If flooding occurs – even minor surface flooding – the trap may be partially submerged. This could be mitigated by careful installation, and thorough waterproofing by design.

6.2.3 Vandalism and theft

This is especially likely if installed in a city, and even more so if the location is visible publicly, via the app. The node should be unobtrusive so as to not draw the eye, sturdy, and possibly attached with one-way fasteners or locked to the trap. Ensuring the node is not reprogrammable without a special password would help to reduce the desirability of stealing nodes. Users should only be able to view public traps that are nearby to them and require resetting. Other public traps will be hidden to make finding and intentionally damaging them difficult.

6.2.4 Animals and other mechanical stresses

When deployed, nodes will not have an easy life in terms of mechanical stress - they will likely be shaken while being transported and knocked around whilst deployed in the field. The node must therefore be sturdy and able to resist crushing and vibration, including around the antenna and detector switch connections.

6.3 Trap dimensions and details

The node will be mounted to, or nearby to, a trap. Traps generally consist of a wooden tunnel with steel mesh on each end, and one or more trapping mechanisms inside. These traps vary in size and internal trapping mechanism depending on target species and trap model. It is therefore important that my node fits on a broad range of traps. Practically, this means that it must be small enough to fit on the smallest trap possible. The switch that detects when the trap mechanism has been triggered ('detector switch') must be swappable, and a version must be produced for each trap mechanism type.

The largest trap commonly available is the DOC250, with dimensions 400mm long x 300mm wide x 250mm tall.

The smallest tunnel-based trap commonly available is the "Victor Professional trap and tunnel", with dimensions approximately 550mm long x 145mm wide x 175mm tall. This means that the node should be no larger than 145mm wide if it is to be mounted on the top of the trap, or no larger than 145mm x 175mm in footprint if it is to be mounted on the side. If the node is mounted on the side of the trap, it should be raised above the ground to prevent water ingress, so must be smaller. I think the ideal location for the node would therefore be on top of the tunnel, but flexibility is desirable.

6.4 Deployment of traps and nodes

The trap has several requirements to make it suitable for transportation to the field. Traps are often carried in a backpack to their final destination, so nodes would have to be transportable in the same manner, preferably nested inside the trap itself to reduce volume.

- The node should fit inside the trap's tunnel for easy transport to its deployment location.
- The node must be fairly light, to ensure it can be carried. Batteries and any metal housing will likely be the heaviest part of the node.
- As discussed before, the node must be sturdy for both deployment and its general use.

7 Brief

I will create a system which notifies relevant people, and, in limited circumstances, the public, when a pest trap has been triggered and requires resetting.

7.1 System overall

The product must be designed to be used and managed in large quantities by organisations such as DOC and Predator Free NZ, as well as in smaller quantities by farmers and other medium-sized landowners.

The product must allow easy remote management and receipt of notifications for large groups of traps.

The product must be as flexible as possible, allowing for different use types such as large-scale public (e.g., urban volunteering via the app), large-scale private (e.g., in a national park or predator-free island), and small-scale private (e.g., on a farm).

7.2 Hardware (node)

The product should be sustainable, repairable, designed to last, and made of materials safe for nature.

The node must cost less than $100 in order for its use to make economic sense.

The node should easily be attachable to a pest trap, as well as other similar mounting points such as a tree or fence post. This installation must be able to be completed in situ (i.e., on a trap that is already deployed in the bush) easily with tools that can be carried on a person, or no tools. Practically, this means that screwdrivers, a small hammer, and a power drill are probably allowable.

The node should be able to wireless transmit to a base station which may be up to several kilometres away.

It must be sturdy, weatherproof, and fit for the environment.

The node should require maintenance only very infrequently (other than resetting and rebaiting the trap). This includes recharging and changing the batteries.

The node must wirelessly report when the trap attached to it has been triggered or reset.

7.3 Software

The software should consist of a backend server that stores and processes trap data and requests, and an app that allows the user to interact with this data.

The server must store all important parameters about the trap.

The server must have enough storage space to store many catches, and, if a social media element is implemented, photos and data related to this.

The server must be powerful and efficient enough to supply clients with requested information and process trap updates quickly.

The app should run on both Android and IOS systems. Being able to run in a browser or as a desktop app (e.g., on a laptop) would be ideal as well.

The software system must allow certain traps to be viewed and reset by public volunteers, in a manner similar to iNaturalist and Pokemon Go. Other traps must only be visible to select members.

The software system must be designed in such a way that a malicious public user cannot disrupt a significant number of traps. Offending users must be kept track of and banned.

The software system should encourage volunteers to continue volunteering through the use of gamification and social media techniques.

The software system should allow for statistical analysis and/or export of trap data, likely to external software such as CatchIT.

8 Analysis of Existing Devices

A number of trap monitoring systems already exist and are in use in New Zealand. The details of each of these are summarised in the table.

8.1 Table of existing wireless trap monitoring devices

Name	System architecture, radio type	Cost per trap (node)	Cost per gateway	Purported range from trap to gateway. Line-of-sight.	Battery	Notes
Celium	Proprietary RF system. Node -> central gateway -> web via satellite (iridium) or cellular.			50km	AA batteries. "Several years", 8 possible.	
Econode	LoRaWAN	$105+GST	$450 (indoor), more for outdoor	15km	5 years. 4x alkaline AA	Has temp, humidity, pH (?!), sound pressure, movement, orientation sensors. Nodes 'phone home' every few hours. Gateways are rebadged Jaycar ones. Integrates with arcGIS and trap.nz. Requires drilling into metal for mounting on DOC200.
MinkPolice	2g moving to NB-IOT so probably CAT-M1	$262.00 plus GST	Not for sale	Relies on Vodafone NB-IOT network	1 year. 4x lithium AA	Initially Dutch, NZ re-design working on Vodafone cellular networks.
Xtrap	Sigfox				10+ years	Gateway on pole with 40km transmission range. Solar and wind powered.

8.2 How we can be better

There are obviously many trap sensor options already available. My design will improve upon the current options in these ways:

8.2.1 Price

None of these companies display a price for each unit on their website, already implying that they are expensive – and suggesting that these are not intended for the average person. After inquiring, I found that Econode costs $105 + GST per node, and MinkPolice costs $262 + GST per node. Already, Econode only just breaks even, according to the Manaaki Whenua modelling. MinkPolice certainly does not. This doesn't mean it is useless – there are likely some places where it is economical, but this cost severely limits its usefulness.

A reduction in price is quite feasible: the system can be constructed from off-the-shelf electronic parts, which are as ubiquitous as they are cheap. It could be produced in New Zealand, minimising expenditure managing an offshore factory and dealing with 'invention companies'. It could even be produced by volunteers, if the system is overseen by a charitable group. More robust competition in this emerging industry will also drive down cost. Broadening the market appeal of this product, such as by targeting urban individuals as well as traditional large stakeholders, could make a business more feasible.

8.2.2 Useability and access

None of these systems are available for purchase online easily - a system must be inquired about, quoted on, and, presumably, installed by an expert. As discussed in Stakeholder Analysis (5 above), managers of large areas of bushland are but one of three stakeholders. Again, New Zealand is more than bushland, and our trapping must reflect this. Targeting only large purchasers, like these existing products do, is only a partial solution.

As discussed in the Brief (7 above), the system will be designed in such a way that allows non-professionals to install and configure their own sensors easily and without specialised tools.

9 Research and Design, Pre-Concept Stage

9.1 Node hardware

The node consists of several subsystems, each corresponding to a function that the node must perform. Some functions may require multiple linked subsystems. Each subsystem will roughly correspond to a discrete electronic component. First, I will list the functions/subsystems, then discuss each subsystem in detail.

Function	**Subsystem name(s)**
Wirelessly communicate with the backend server.	Communication
Provide the rest of the node with steady supply of electricity.	Power
Process changes in trap state and command the radio to send data. "Brains of the operation".	Microcontroller
Locate the trap in the world, as they may move – especially if people steal or accidentally move them.	GPS
Physically and electrically connect to other components nearby to the node, or inside the node – for example, the detector switch, any devices used to configure the node, and the battery pack.	Connectors
House and protect the node's delicate insides. This must be watertight.	Housing
Detect when the trap mechanism has been triggered. This is external to the node, though is connected via the Connectors subsystem.	Detector switch

9.1.1 Communication

9.1.1.1 About the subsystem

Obviously, the node (hardware on the trap) must communicate with the internet. There are a number of ways that this can be done. I will go over these options and decide on the best one.

There are several requirements for this:

- Low power consumption. The brief states that we must minimise trap maintenance. One part of this is maximising battery life, which can be achieved with low power radio communication.
- High range. Lower range means that either traps will be constrained to existing in specific locations around the base station or will necessitate installing extra base stations. Neither are ideal.
- A radio system must either have very good existing network coverage, or the ability to deploy your own network, or a mixture of both.

9.1.1.2 To Mesh, or not to Mesh

In a normal network, several nodes communicate directly with one gateway. This has the disadvantage of requiring all nodes to be in range of the gateway, requiring more gateways. This leads to more cost.

Instead, a mesh network design could be used. In a mesh network, nodes can communicate with each other, as well as the gateway. This means that nodes out of the gateway's range could pass the message from trap to trap until it found a gateway.

Also, we must consider how traps are arranged in the field. They tend to be in discrete lines, with 50m-200m spacing, as discussed in 3.3 How we're doing it above. This means that a gateway could be placed at the end of one of these lines, and trap-to-trap mesh communication could be used to pass messages down the line. If this mesh technology wasn't used, a line would need several gateways, which may be less cost-effective.

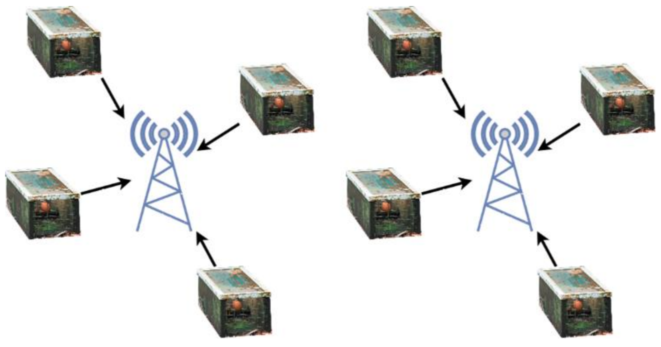


Figure 3. A non-mesh "hub and spoke" network topology.

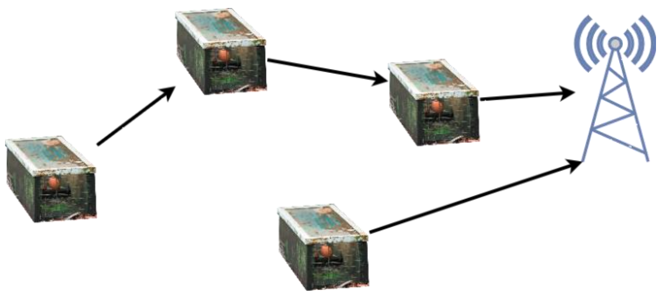


Figure 4. A mesh network. See how fewer gateways are used, yet the effective range remains the same.

I will likely not implement a mesh network due to its complexity. However, the capability to implement a mesh network will be a factor in deciding what radio system to use, since it is a potential future improvement.

I will now analyse the options for this hardware subsystem.

9.1.1.3 NB-IOT (specifically CAT-M1)
NB-IOT is a subset of the familiar 4G LTE standard (commonly used in mobile phones). NB-IOT stands for **N**arrow-**B**and **I**nternet **o**f **T**hings. Narrow-band refers to the range of frequencies over which a single signal is carried. The wider the bandwidth, the more data can be carried per unit of time, but the more power used to transmit or receive. For example, a rough phone call will have a bandwidth of around 3KHz (3000 hertz), AM radio has a bandwidth of about 20KHz, and FM radio can have up to 92KHz. Normal 4G LTE has a bandwidth of up to 20MHz (20 million hertz), normal WiFi has a bandwidth of 22MHz, and NB-IOT (specifically CAT-M1) has a bandwidth of 1.4MHz.

Ultimately, this means that it is a fairly low-power, medium data-rate option. It would not be possible to install base stations in areas where they do not already exist in an ISP's network, as this is a proprietary radio standard relying on restricted frequencies and closely guarded trade secrets. Essentially, we are stuck with whatever coverage exists already, unless we get an ISP to improve it. This would be highly expensive.

While the range of Spark's network is fairly good, there are large unconnected regions in national parks such as the Hunua Ranges, and most of the Coromandel. These are the regions where trapping is likely most wanted, so it does not seem like a good idea to pick a communication method that likely will never work in many areas where traps will be deployed. The MinkPolice sensor listed above uses Vodafone's NB-IOT network. This system does not allow for the creation of mesh networks.

9.1.1.4 LoRaWAN
LoRa is a long-range, low-power radio protocol invented by SemTech with a range in perfect line-of-sight conditions of >10km. LoRaWAN is a protocol built on top of LoRa which allows LoRa devices to connect to the internet via LoRaWAN base stations ("gateways"). In New Zealand it operates in the free-to-use 915MHz frequency band, which means that anyone can set up a LoRaWAN gateway without special permission, unlike NB-IOT.

Spark operates a LoRaWAN network, though it is primarily concentrated in urban areas – providing coverage in bushland is probably not economical for them. There is an international collection of hobbyists who operate a shared network of free-to-use LoRaWAN gateways called The Things Network (TTN). The coverage of TTN is quite poor in New Zealand, however it shows that a LoRaWAN network is easy enough to set up for the average technically-minded person. This means that a similar network could be established to provide coverage to traps in regions not already serviced by Spark's LoRaWAN.

LoRaWAN gateways are relatively cheap: A very basic type that can only receive a signal from one device at a time is less than $100 (excluding solar power, internet connection, etc). A more advanced fully-LoRaWAN-compliant type is upwards of $500, again excluding utilities. Gateways deployed could access the internet via LTE, 3G, satellite, or wired connection.

Since LoRaWAN is built on the point-to-point protocol LoRa, the hardware used for these protocols is identical. This means a LoRaWAN radio could be used to send normal LoRa packets from node to node, implementing a mesh network.

In the city, where LoRaWAN networks are already established by bodies such as Spark, establishing a network of connected traps, such as for the publicly-accessible social media component, would be as easy as distributing nodes and traps then connecting these to the local network. The Econode trap already uses LoRaWAN, though it is not clear what connectivity they use – when I enquired, it was suggested that I try TTN, though this isn't really a good option coverage-wise. I suspect that they use a similar approach of utilising existing networks where available and building their own in other places.

The bandwidth of LoRaWAN is more complicated than CAT-M1 as it varies on-the-fly. It is on the order of a few hundred kilohertz, much lower than that of CAT-M1. LoRa achieves such high range at such low power by reducing the data rate (and therefore bandwidth) to almost nothing. This is fine, however, as a node only needs to send small amounts of information.

9.1.1.5 Sigfox
Sigfox is a radio system similar to LoRaWAN, with a fixed bandwidth of 200KHz and an ideal range of 30-50km in perfect rural conditions. Sigfox, the company, maintains a global Sigfox network which covers more of New Zealand than Spark's LoRaWAN network, but cannot be extended with private base stations without the cooperation of Sigfox. This means that connectivity is limited to whatever is already provided by Sigfox, like with NB-IOT. The Xtrap system uses Sigfox and "Sigfox-on-a-Pole" gateways, meaning that they pay Sigfox for the rights to deploy a Sigfox network. This is better than NB-IOT, however still not an ideal solution.

9.1.1.6 Custom radio
There is no inherent requirement for me to use an existing radio system. The company Celium took this approach. Their proprietary radio protocol can reach up to 50km, and still run off batteries for "several years". However, this approach would realistically require the expertise of an electrical engineer (or many) and would take far longer than I have to complete this project. For this reason, I will opt to use an existing radio solution.

9.1.1.7 Deciding on a radio system
To ensure that the system works anywhere in New Zealand and can be set up by a layperson, I require a radio system that can be extended freely and easily. This eliminates Sigfox and NB-IOT, because these rely on proprietary infrastructure set up by another company. Therefore, the only good, long-range, low-power option open to me is LoRaWAN. This is proven to be capable by Econode's use of it.

9.1.1.8 Choosing a hardware module

There are a huge variety of LoRaWAN 'modules' - premade circuit boards that implement all of the hardware required for LoRaWAN and can easily be connected to a microcontroller. It is important that the LoRaWAN module I choose is powerful enough to connect to a base station at a decent range. For this, I must discuss the physics behind radio transmission power.

LoRaWAN in New Zealand operates at the AU915 frequency standard, meaning the transmission frequency is around 915MHz. There are legal limits to transmit power, defined as the maximum EIRP (**E**ffective **I**sotropic **R**adiated **P**ower, measure of the actual power radiated from an antenna). For the 915MHz range, the maximum legal EIRP is 0.0 dBW.

The dBW (decibel-watt) is a logarithmic scale of power, where a lower number is less power. dBm (decibel-milliwatt) is a similar scale which measures the same quantity but is more often used for radio calculations. 0.0 dBW = 30.0 dBm, so this is my maximum EIRP.

A 3 dBm increase approximately corresponds to a doubling of range as this, too, is a logarithmic scale.

To calculate EIRP, the following formula can be used: $EIRP = P_t + L_c + G_a$, where P_t is the transmitting power of the radio, L_c is the signal loss from cables and connections, and G_a is the antenna gain - essentially, how much the antenna amplifies the signal, which is measured in dBi. For the following calculations, I will assume ideal conditions, therefore Lc = 0.

The initial options I found for radios are shown below:

Name	Radio transmit power (dBm)	Claimed line-of-sight range	EIRP (5dBi antenna)	Legal?
Ebyte E32-915T30D	21-30	8Km	35	Not without turning power down
Ebyte E32-915T20D	10-20	3Km	25	Yes, 5 dBm under

If I was to choose the 30 dBm model, I would have to turn down the power to ensure the system is legal. This is better than buying a radio which is 5 dBm lower than it can be, given that 3 dBm is double the distance. In watts, a linear scale, 30 dBm = 1W but 25 dBm = 0.31W. This is clearly a huge difference.

Therefore, I planned to use the Ebyte E32-915T30D 915MHz LoRa radio. This radio connects to the microcontroller through UART serial, a simple 2-wire, bi-directional wired communication protocol. The antenna connects via a SMA-K plug. The radio runs on 5V and draws 660mA when transmitting.

Unfortunately, I discovered that this option would not work. There is a plethora of LoRa radio modules (such as the EBYTE range), though they are all based on one of three integrated circuits made by SemTech (the inventor of LoRa): the SX1272/6/8.

The Ebyte LoRaWAN modules listed above are indeed based on the SemTech chips, however they connect to the main microcontroller via UART instead of SPI as is standard for SX127X chips, indicating that they have a custom built-in microcontroller interface. They are therefore not compatible with any of the existing LoRa software libraries, so cannot easily be used for LoRaWAN as is required for this product. These two topologies are described in Figure 5 below.

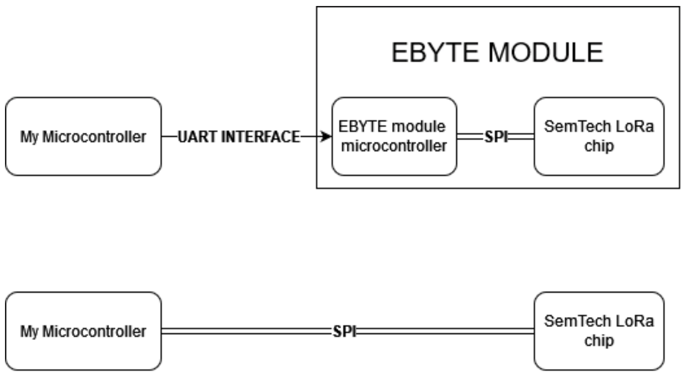


Figure 5. Interfaces between devices in EBYTE module (top), and a SX127X (bottom). Finesse and control are lost in the UART interface, making LoRaWAN impossible.

Therefore, I will need to use a LoRaWAN module which does not have a microcontroller interface – I must communicate directly with the SX127X chip.

Only a minority of these LoRaWAN modules exist in 915MHz versions, which are required to connect to existing New Zealand infrastructure.

Figure 6. An RFM95W, based on the SX1276

Two promising modules are the RFM95W or RFM95C by HopeRF, as pictured in Figure 6. Both are based on the SX1276, with the only difference being a metal 'can' which shields the C version from RF interference. Both output 20 dBm of transmit power and run off 3.3V.

RFM95 modules are very small, require an antenna to be soldered on directly, and have non-standard spacings between the pin connections. This necessitates a breakout board - a small circuit board which converts the pins to standard spacing and adds an SMA connector for the antenna. These are not readily able to be purchased fully made, but the board, connector, and pins can be bought or made separately, then assembled.

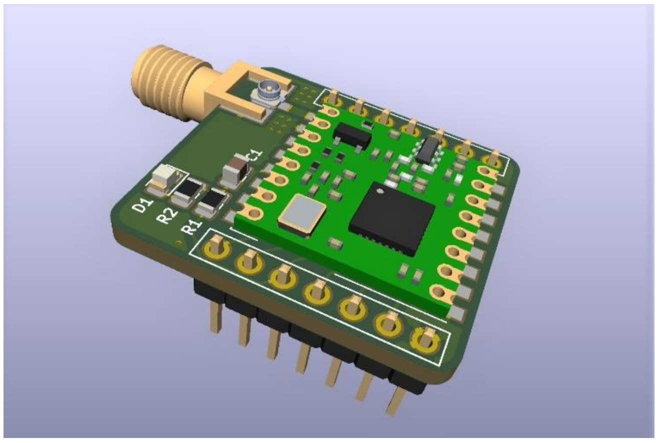


Figure 7. An RFM9x breakout board, designed by GitHub user attexxx. I will use his open-source design.

I will use an RFM95W, with attexx's breakout board design.

9.1.1.9 Antenna

All radio systems require antennae, whether to transmit or to receive. Antennae are designed for a specific radio frequency. In New Zealand, LoRaWAN operates around 915MHz (or 923MHz), so I will need an antenna tuned to these frequencies. As discussed earlier in EIRP calculations, the antenna provides an apparent gain. This is not free energy. Instead, the antenna concentrates the already-existing energy away from some areas in space, and towards others. My node and gateway are likely to be at roughly the same elevation – certainly not vertically above one other. This means that energy can be diverted from the vertical axis, and towards the horizontal, providing an apparent gain.

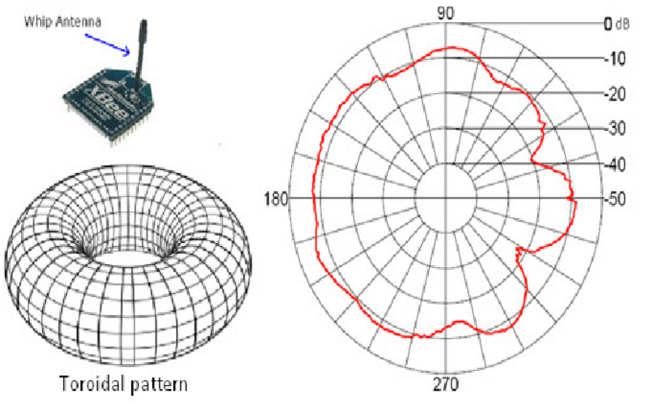


Figure 8. A diagram showing radiated power of a whip antenna mounted vertically.

As shown in Figure 8 above, a whip antenna has the desired radiation characteristics. They are commonly used in LoRaWAN for this reason.

A prevalent and cheap option is a 915MHz 5dBi (gain) whip antenna with an SMA connector. I will use this.

Figure 9. The chosen whip antenna.

9.1.2 Power

All existing remote monitoring systems, as well as automatic traps such as the AT220 and A12/24, run on batteries which must be replaced at an interval of months or years.

9.1.2.1 Solar panels?

On the surface, solar power appears to be a promising option – they could allow the trap to run unmaintained for longer.

However, these nodes are primarily going to be deployed on the forest floor where only a small amount of light reaches this point, so the panel must be able to charge the battery on nearly nothing.

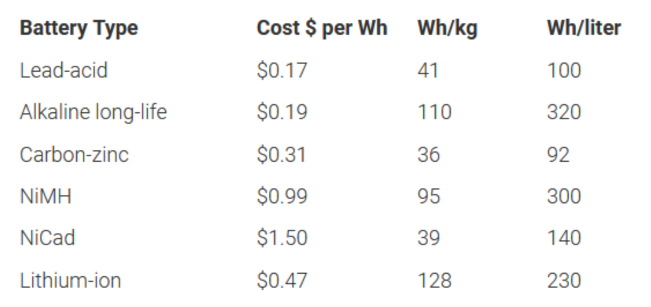
Unfortunately, all solar power controllers at a price point compatible with this project do not function well in low light. Electrical engineer Andreas Spiess experiments with several solar charge controllers in a YouTube video (Spiess, 2017). None of the affordable options tested by him would be useful in my low-light scenario.

Therefore, I will not use solar power.

9.1.2.2 Power storage

Since I do not need the capability of recharging the batteries while deployed in the trap, I will conform to the established practice (in existing trap monitoring systems) of using packs of rechargeable or replaceable NiMH or alkaline batteries, which can easily be swapped.

These batteries also have the advantage of a higher power density over Lithium-ion, which I would likely use if the goal was to use solar power (see Figure 10).



Battery Type	Cost $ per Wh	Wh/kg	Wh/liter
Lead-acid	$0.17	41	100
Alkaline long-life	$0.19	110	320
Carbon-zinc	$0.31	36	92
NiMH	$0.99	95	300
NiCad	$1.50	39	140
Lithium-ion	$0.47	128	230

Figure 10. Energy density by battery type

9.1.2.3 Power regulation

9.1.2.3.1 Why regulate?

The microcontroller, radio, and other electronic components all require an accurate and constant voltage. If this voltage is too low, they will not function correctly. If it is too high, they will be permanently damaged.

The chosen components require a 3.3v supply. Each battery cell typically has a voltage of 1.5v charged, and 1v discharged. To keep the battery voltage above the required 3.3v even when flat, a minimum of 4 battery cells must be used. This means a fully charged voltage of 6v, and a flat voltage of 4v.

Therefore, I will need a voltage regulator to reduce the 'raw' voltage of the battery to 3.3v. There are two types of DC-DC voltage regulators: Linear and switching.

9.1.2.3.2 Linear regulators

Linear regulators are simple and cheap but are quite inefficient: the proportion of power corresponding to the dropped voltage is lost as heat. The efficiency of these regulators η can be approximated by the following formula:

$$\eta = \frac{V_{out}}{V_{in}}$$

In this case, V_{in} is at most 6V, and V_{out} is always 3.3V. Therefore, the worst-case efficiency of a linear regulator is in this application $\eta = \frac{V_{out}}{V_{in}} = \frac{3.3}{6} = 0.55$.

This is not good. While the efficiency would increase as the battery discharges (V_{in} decreases, so the directly proportional η increases), this is still a terrible efficiency for a system that is meant to operate for as long as possible on a set of batteries.

9.1.2.3.3 Switching regulators

The other type of voltage regulator is a switching regulator. These have a high constant efficiency – it doesn't change based on the difference between the input and output voltages like a linear regulator does. This regulator type also draws an extra, constant amount of current. I am down-regulating voltage – the output is lower than the input. This means I need a 'buck' switching regulator, or buck converter.

9.1.2.3.4 Choosing a regulator

All the components in the design are intended to use a very small amount of power. Once I have a final bill of materials I will calculate the expected battery life, however I estimate it will be over 1 year.

I chose to use a small switching (buck) regulator purchased on AliExpress. This has a claimed efficiency of 97.5%, and a static current of 0.85 mA. Using these parameters, Figure 11 shows power loss by power draw, for each type of regulator.

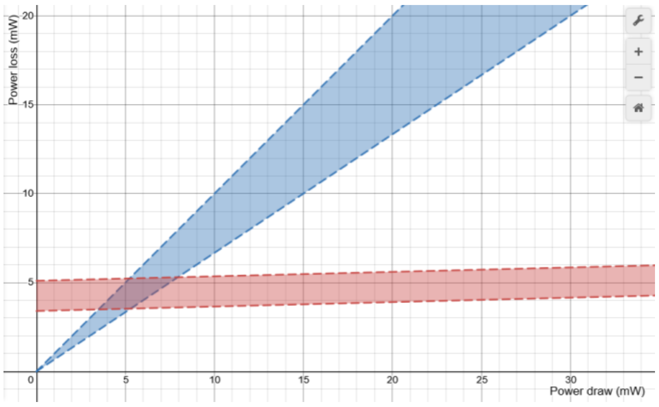


Figure 11. Power loss (mW) by power draw (mW) for switching (red) and linear (blue) regulators, across voltage range 4-6v.

It is worth considering that AliExpress claimed characteristics are notoriously unreliable – real-world data is required if we are to be certain. For my initial prototypes, I will use a switching regulator as these tend to be more efficient over a larger range. However, it may be that a linear regulator is more efficient in the long run. This can only be determined by experimentation with a working system.

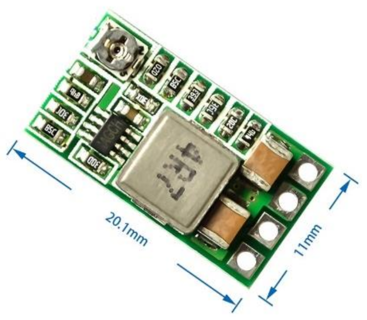


Figure 12. Chosen switching regulator.

9.1.3 GPS

To locate the trap, which may move by natural or anthropogenic means (i.e., stealing, vandalism), a GPS unit will be used. There is not much variety in the GPS module market at this price range, so I will be using the ubiquitous GY-NEO6MV2 (or similar equivalent) module, with a NEO-6M GPS chip. This communicates with the microcontroller via UART. It requires a voltage between 3.3V and 5V. This is easily provided by the power system discussed in Power (9.1.2 above).

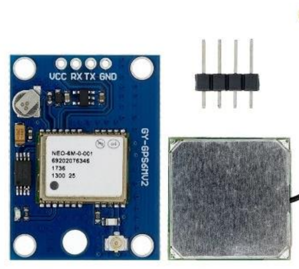


Figure 13. Chosen GPS module.

9.1.4 Microcontroller

9.1.4.1 What's a microcontroller?

A microcontroller is a small, low-power computer. This is the component that reads the state of the detector switch, checks the GPS, and sends the message over radio. It is therefore a very important component, and must be carefully matched with all other components in order to work well.

9.1.4.2 Microcontroller options

Low power consumption. The microcontroller must be able to enter a low-power 'sleep' mode. It should be able to exit when the detector switch is triggered (via an interrupt, see 9.4.3 Interrupts below), and periodically.

Supply voltage: The microcontroller should run on 3.3v so the same regulator as the radio can be used.

UART ports: Has at least 2 UART ports: one for the GPS, and one to connect to a configuring external device (e.g., laptop). If this isn't possible, it must be able to multiplex between them, which would require extra hardware.

Size: The microcontroller and development board must be small enough to fit inside the housing.

Low cost, high availability: The microcontroller must be cheap enough to fit within the price range.

9.1.4.3 Microcontroller options

Following is a list of potentially relevant microcontrollers accessible to me:

Name of dev-board	Actual microcontroller chip	Voltage	Flash, RAM	Sleeping current draw	UART ports	Required UART arrangement	Notes
Arduino Nano	ATMEGA328p	3.3V if oscillator <12MHz	32Kb, 2Kb	~10 µA	1 hardware + 2 software (emulated)	Debug/setup on hardware, GPS software emulated.	Oscillator <=12MHz is difficult to find. Has UART-->USB for debugging already onboard. Not really necessary but nice to have.
Arduino Pro Mini	ATMEGA328p	3.3V on 8MHz version	32Kb, 2Kb	~10 µA	1 hardware + 2 software (emulated)	Debug/setup on hardware, GPS software emulated.	Easy to find 8MHz versions. Doesn't have UART->USB so less power consumption but slightly less convenient.
NodeMCU, Wemos D1 mini	Esp8266	3.3V	4Mb, 45Kb	400 µA in light sleep, 20 µA in deep sleep	2 hardware + 1 software	Debug/setup and GPS on hardware.	The microcontroller cannot wake from deep sleep by a normal interrupt from the detector switch. External circuitry could be used to reset the microcontroller, but this would add extra complexity. ESP8266 has WiFi which I would not use, hence is quite power-hungry.
MSP430 Launchpad	MSP430G2XXX	3.3V	32Kb, 512B	< 10 µA	1 Hardware + 1 Software + 1 USB	GPS on hardware, debug/config on USB.	Much less software support than Arduino. I have no experience with this platform.
STM32 Bluepill	STM32F103C8T6 (ARM Cortex M3)	3.3V	64Kb, 20Kb	< 10 µA	3 Hardware	Both on hardware (ideal).	Should have similar software support to Arduino, uses Arduino framework.

9.1.4.4 Choosing a microcontroller
There are two promising options: the Arduino Pro Mini, and the STM32 bluepill. The STM32 bluepill is better as it has 3 hardware UARTs, uses slightly less power, and has much more RAM and flash memory. Therefore, I will use the STM32 unless I run into issues with software support (e.g., a library for the radio). In that case, I would switch to the Arduino Pro Mini.

9.1.5 Connectors
This hardware system consists of multiple interconnecting parts - the node, the detector switch, and the battery pack. There must also be a way to connect a laptop or mobile device to the unit to configure and debug it. All of these connections require robust, easy-to-use and, in the case of the external ones, waterproof connectors. One commonly available connector is the SP13, an IP68-rated waterproof 13mm plug. This is easy to mount in a face, so would be ideal for the detector switch and configuration port which need to be mounted externally. The male side (connected to the switch and configuration laptop) is quite long, which is not ideal for the battery which must be mounted inside the unit, as it will take up too much space. This plug does not need to be waterproof as it is protected by the outside casing, so a much smaller DC barrel jack can be used.

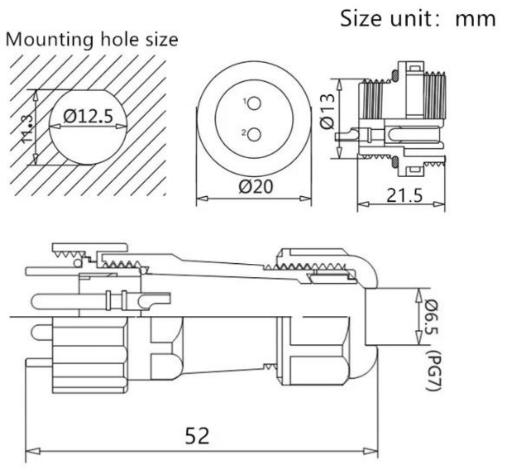


Figure 14. SP13 plug/socket diagram.

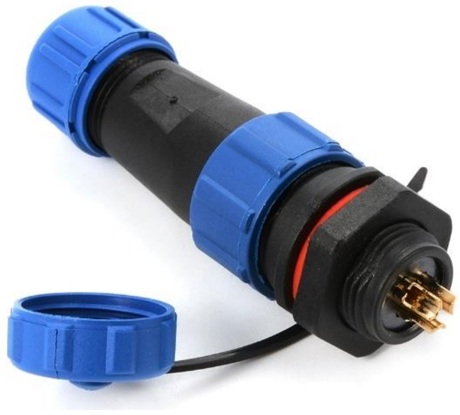


Figure 15. SP13, male and female ends connected.

Figure 16. DC barrel jack plug (male).

Figure 17. DC barrel jack socket (female).

9.1.6 Housing
The sensitive electronic components must be housed in a protective box. The specifics of this will be discussed in 11 Concepts below, though there are a few factors to consider.

9.1.6.1 Material
The housing must be made of some material. This must be durable and weatherproof. There are two primary options for this: sheet steel, and plastic.

9.1.6.1.1 Sheet steel
Pros

- Durable.
- Low cost of entry – sheet metal tools are relatively cheap.
- Low cost of material.

Cons

- Heavy.
- Can rust if not galvanised or painted correctly.
- Scaling up production can be expensive or time-consuming.
- Blocks radio signals, meaning antennae must be mounted on the outside.

9.1.6.1.2 Plastic
Pros

- Light.
- Very easy to produce at scale, after high barriers to entry are overcome.

- 3D printers make rapid prototyping easy. This can also be used for manufacturing, though may be slower.

Cons

- Significantly weaker than sheet steel.
- Very high cost to entry for injection moulding equipment.
- Material used in 3D printing is relatively expensive, so should be avoided in strength-critical applications which tend to use more material.

One key requirement of the node is durability. For this reason, I will construct the housing out of folded and welded sheet steel. The two antennae will be mounted outside of this main housing, to prevent their signals from being blocked by the metal.

It may be possible to design a plastic housing in a way that is sufficiently strong. This, however, would require testing of many different thicknesses, densities, and shapes of housing. I do not have the time for this, so would rather over-engineer the housing with metal.

9.1.6.2 *Waterproofing*
The housing must be resistant to moisture ingress, even in relatively heavy showers. If moisture does enter the node, electrical components may be damaged or destroyed, and metal parts will corrode.

Several techniques exist for reversibly sealing two surfaces.

9.1.6.2.1 O-Ring
An O-ring is required to seal the system against the elements. This will be installed in a small channel around the antenna tower, underneath the steel lid. An O-ring is a small rubber ring which, when pressed on by two objects, prevents water (or other fluids) from flowing between them.

O-rings can be made of many different plastics and rubbers. I only need simple water resistance (not heat or chemical proof), so I will pick the cheapest type.

O-ring dimensions are usually specified by outer diameter (OD) and cross-sectional diameter (CSD), so the formula for inner diameter (ID) is:

$$ID = OD - 2 \times CSD$$

Figure 18. An O-ring of relatively small diameter.

9.1.6.2.2 Silicone sealant O-rings
Silicone sealant is commonly used in bathrooms, kitchens, and around windows, to keep moisture out. However, it can also be used to create a custom O-ring-like seal.

In a real production run, these would likely be replaced with proper rubber seals. However, they are a useful prototyping tool.

9.1.6.2.3 Glands
Glands are intended for providing a watertight entrance for cables into a box.

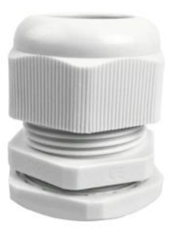


Figure 19. A cable gland

These work by compressing a rubber ring (similar to an O-ring) onto the cable by tightening the cap nut (see Figure 20 below).

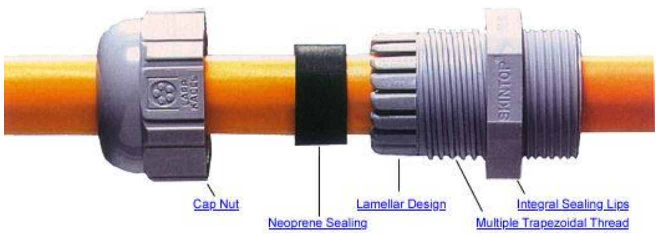


Figure 20. Exploded view of a cable gland.

9.1.7 Detector switch
The detector switch is a component external to the node, which is connected via a waterproof connector.

It must close a circuit when the trap is triggered, and hold it open when the trap is set.

There are a couple of options for this.

9.1.7.1 *Microswitch*
As the name suggests, these are small ('micro') mechanical switches.

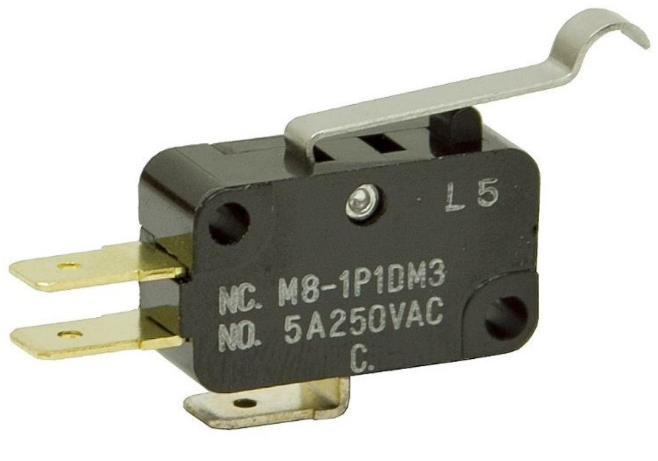


Figure 21. A ubiquitous microswitch, with steel lever arm.

The switch could be mounted in such a way that the lever arm is depressed when the trap is set, and released when the trap is triggered.

This would require a mounting bracket, which screws into a specific location in the trap tunnel or trapping mechanism. A moderate degree of accuracy is required for the arm to be pressed properly.

9.1.7.2 Reed switch

A reed switch is a mechanical switch activated by a magnet.

Figure 22. A small reed switch.

As before, a bracket would secure the switch to the trap mechanism or tunnel. A magnet would be mounted on a moving part of the trap, which would activate the switch when the trap is ready, but not when triggered – or vice versa.

Mounting a magnet on a moving part of the trap is no mean feat. These moving parts are subject to extreme forces in order to kill the pest animal, which may cause the magnet to fall off. Further, securing magnets onto metal parts usually requires accurate drilling into metal. This is not achievable in the field, where many nodes will be fitted – traps cannot all be brought back into a workshop.

This leaves microswitches as the only feasible option. The chosen microswitch must be waterproof, as it is outside of the node's main housing. The "T85" IP67-rated model is cheap and readily available online.

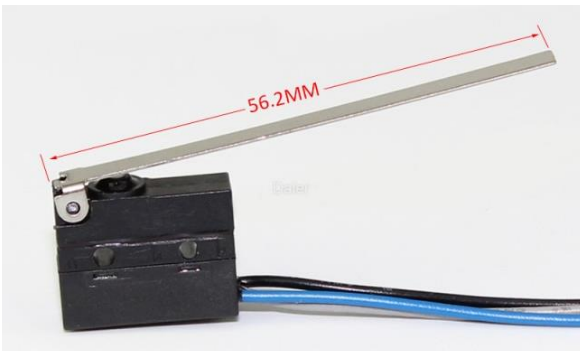


Figure 23. The chosen IP67 (i.e., waterproof) T85 microswitch.

9.2 Some reference photos (mood board)

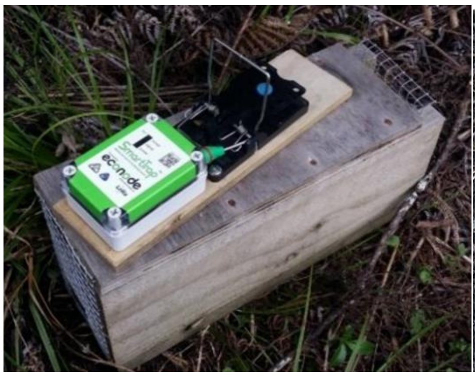


Figure 24. (above) Econode attached to Victor rat trap.

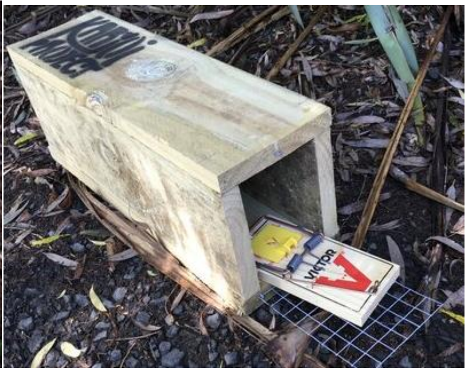


Figure 25. (above) Victor rat trap and tunnel.

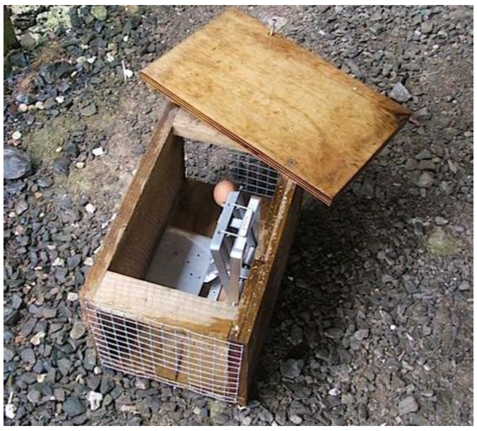


Figure 26. The ubiquitous DOC200 rat and stoat trap.

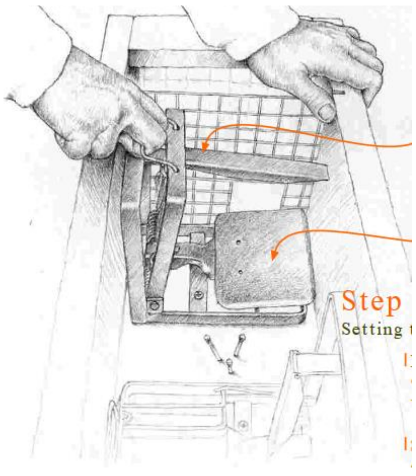


Figure 27. (above) Diagram of a DOC200

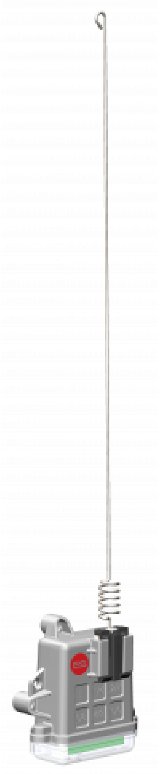


Figure 28. Celium node

9.3 Node full Bill of Materials (BOM)

Part	Purpose / subsystem	Quantity per node	Unit price (exc shipping)	Price per node	MOQ price	Minimum Order Quantity	Supplier
RFM95w 915MHz	Radio to communicate	1	$5.58	$5.58	$5.58	1	CN888 Store via AliExpress
Module Mini 360 DC Buck Converter	Voltage regulator (buck converter)	2	$0.45	$0.90	$2.25	5	FDKJGECF via AliExpress
GY-NEO6MV2 GPS module	GPS module to determine node location	1	$3.46	$3.46	$3.46	1	Wanzai store via AliExpress
STM32F103C8T6 "bluepill"	Microcontroller - brains of the operation	1	$2.83	$2.83	$2.83	1	FDKJGECF via AliExpress
SP13 2 pin plug	Connect detector switch to node	1	$3.00	$3.00	$3.00	1	ZHTCRJ via AliExpress
SP13 4 pin plug	Connect config device to node. Unconnected in normal operation	1	$3.14	$3.14	$3.14	1	ZHTCRJ via AliExpress
DC Barrel jack/socket	Connect battery to microcontroller	1	$0.26	$0.26	$1.31	5	Locheuk Connector Store via AliExpress
SMA edge antenna plug	Connect antenna to RFM95w breakout	1	$0.26	$0.26	$2.61	10	GuoQi Pneumatic store via AliExpress
O-ring 50x47x1.5	Seal the system against water	1	$0.33	$0.33	$3.33	10	U Officer Store via AliExpress
Cable Gland	Seal the antenna against water	1	$0.54	$0.54	$5.44	10	Dashen Electric Store via AliExpress
915MHz antenna	Allow the radio to transmit signals	1	$3.17	$3.17	$6.34	2	Cerxus Store via AliExpress
Waterproof microswitch	Detects when the trap has been triggered	1	$2.91	$2.91	$2.91	1	Daier Store via AliExpress
Total electronics cost per node				$26.39			

This BOM shows the cost per node of all selected electronic components so far. Most of these components are "development boards". In a final production design, discrete components would be used on a highly sophisticated custom PCB, which would cost significantly less.

9.4 Software: node firmware

As discussed in the brief, it is critical that the node consumes as little power as possible. This will allow the node to be maintenance-free for as long as possible. Therefore, the microcontroller in the node must operate in a low-power sleep mode for the vast majority of its life.

The node must be able to be accurately located, so includes a GPS. This is especially important for public traps, which may move slightly and must be easily located by people unfamiliar with the environment. Receiving GPS signals takes a fair amount of power, so the GPS module should be switched off unless it is being used.

LoRaWAN transmissions and GPS receives should be kept to an absolute minimum, because this consumes a large amount of power.

9.4.1 The Arduino framework

Different microcontrollers typically have different ways of interacting with their hardware.

This behaviour is not because of the inherent differences in the microcontroller: it is because they are usually programmed in different frameworks.

As the name suggests, a framework is a common 'dialect' for interfacing with the microcontroller's hardware.

For example, in the STM32Cube framework, which is the 'suggested' framework for use with the STM32, the code to turn an LED on is as follows:

```
HAL_GPIO_WritePin(GPIO_Port, Pin, state);
```

In the Arduino framework, which initially ran on Arduino hardware but now runs on a huge range of microcontrollers (including the STM32):

```
digitalWrite(LED_PIN, STATE);
```

The key point here is not that the Arduino framework is simpler (which it is), but that the one framework allows for the programming of hundreds of types of microcontrollers in the same way. On the other hand, the STM32Cube framework only works on STM32 devices.

This means that I can use code written for Arduino hardware on the STM32. This is incredibly powerful, because there is an enormous catalogue of software 'libraries' available in the Arduino framework to serve various purposes - for example, handling the difficult parts of LoRaWAN, GPS, etc.

I am quite familiar with the Arduino framework. It uses the C/C++ programming languages, which I am also familiar with. Therefore, I will use the Arduino framework.

I will write this code in the PlatformIO Integrated Development Environment. This allows developers to easily build projects across a myriad of boards and frameworks. If I end up needing to switch to Arduino, PlatformIO will automate the few tasks which I would have to do to migrate my code. It also provides a high-quality debugging tool, which should make development easier.

9.4.2 Arduino framework control flow

In the Arduino framework, there are two key functions (blocks of code): setup, and loop.

When the microcontroller is initially powered on, the setup function runs once. This is used to initialise the microcontroller – for example, the code might set up various inputs and outputs.

Following this, the loop function runs repeatedly until the device is turned off. This loop contains the bulk of the software's functionality.

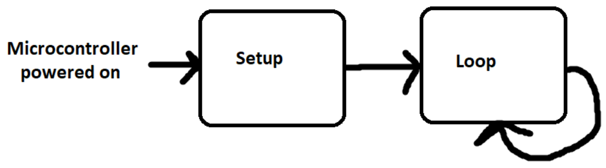


Figure 29. Program execution flow in the Arduino framework.

9.4.3 Interrupts

We need a way for the code to know when the detector switch has been triggered.

One option would be to check the state of the switch every time the loop function runs. This method is called polling. However, when the microcontroller sleeps, this function stops running until the end of the sleep period. This period may last hours or even days, to maximise power savings. This means that a catch may be detected and acted upon far too late.

Instead, the microcontroller can be commanded to immediately stop what it is doing (and wake up from sleep) when the state of the switch changes and do a short task. It then returns to where it was in the main program. This is called an interrupt, since the program execution is being interrupted.

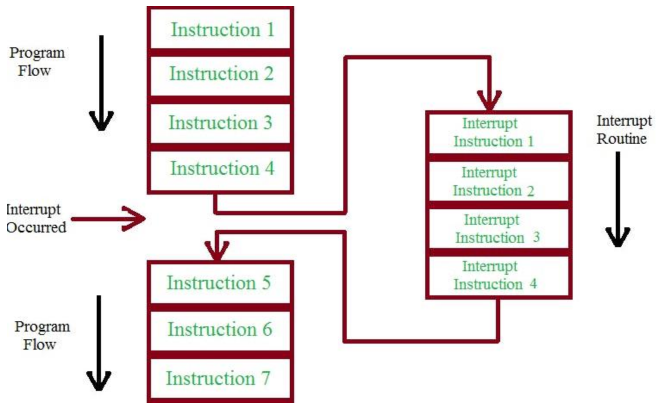


Figure 30. Execution flow of an interrupt.

9.4.4 LoRaWAN flow

LoRaWAN is quite a complex protocol. I will describe what needs to happen from the trap's end.

1. The node must first 'join' the LoRaWAN network via the over-the-air-authentication method (OTAA). It transmits a join-request, sending several parameters describing it, including the DevEUI, AppEUI, and AppKey.
2. The gateway receives these, and, if the node is supposed to be a member of its network, it transmits back a join-accept message.
3. The node stores all parameters derived from the joining process. These are used for transmitting 'normal' data uplink messages.
4. The node can now transmit data uplink messages using the information determined earlier.

This is summarised in Figure 31 below.

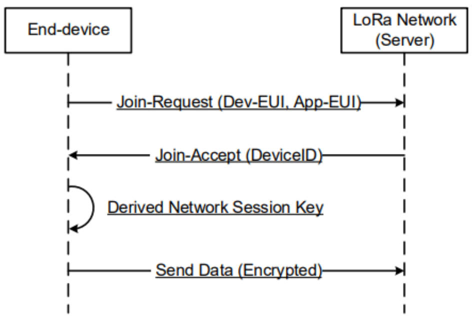


Figure 31. The 'flow' of LoRaWAN packets between the node (End-device) and gateway (LoRa Network).

9.4.5 State machines

Simple embedded devices such as this often use a concept called a 'state machine' to formalise their behaviour. In a state machine model, the device will always be in one of several states. Each state has a clear set of tasks that occur in the state, and defined ways of entering and exiting the state.

For example, a state would exist for joining the network as described above. Another state would describe sleeping.

9.4.6 Required states

A full list of states follows:

9.4.6.1 Joining

Connect to the LoRaWAN network, much like you might connect a cell phone to a WiFi network.

9.4.6.2 Preparing a packet

The device collects the information needed to be sent to the base station (e.g., trap state, GPS location, battery voltage), and builds this into the completed 'packet' to be sent.

9.4.6.3 Sending the packet

The device sends the packet to the base station. It may also be able to wait for an acknowledgement to ensure that the transmission was successful.

9.4.6.4 Sleeping

The device sleeps, saving power.

9.4.6.5 Verify Trap State Change

In an ideal world, accurate information would be read by the node. However, the real world contains electrical interference, switch bounce, and numerous other factors which may lead to an inaccurate reading or false alarm. To mitigate this, after the trap is first detected to have changed, the node waits a couple of seconds, and reads the trap state again. If the two readings match, then we can be sure the trap state has actually changed.

9.4.6.6 Periodic Wakeup Check

The device periodically wakes up from sleep, even if the trap state hasn't changed. When this happens, we need to determine whether it is time to send a periodic "phone home" packet, or just go back to sleep. If the time since the last packet is high, or the battery voltage is low, a packet should be sent. Otherwise, the node can return to sleep.

9.4.7 State machine flowchart

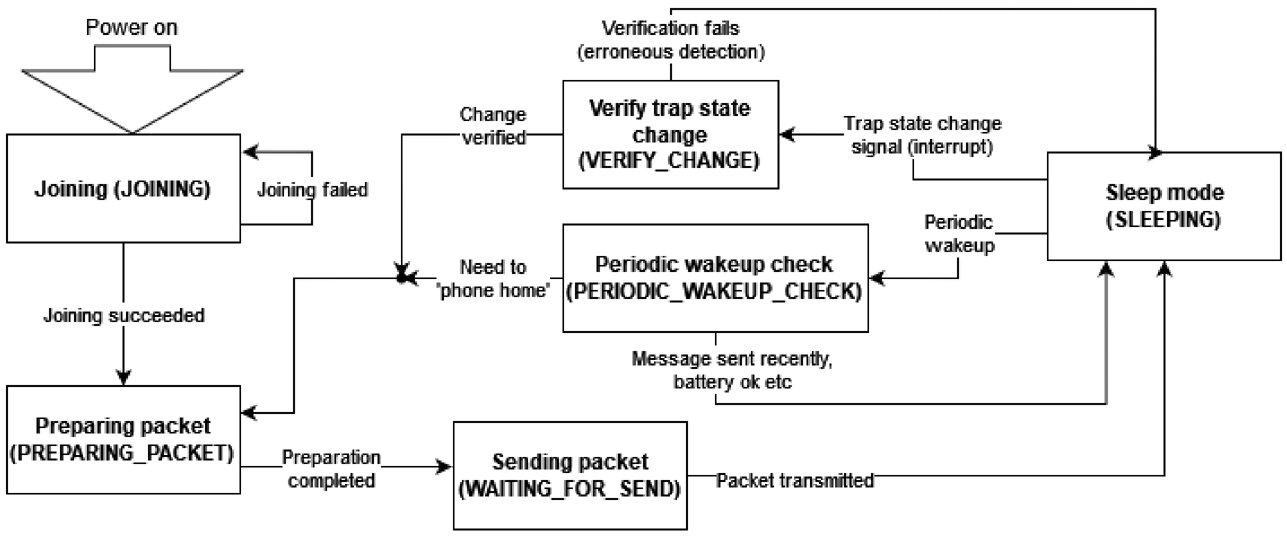

Power on
Joining (JOINING)
Joining failed
Joining succeeded
Preparing packet (PREPARING_PACKET)
Preparation completed
Sending packet (WAITING_FOR_SEND)
Packet transmitted
Verification fails (erroneous detection)
Change verified
Verify trap state change (VERIFY_CHANGE)
Trap state change signal (interrupt)
Sleep mode (SLEEPING)
Periodic wakeup
Need to 'phone home'
Periodic wakeup check (PERIODIC_WAKEUP_CHECK)
Message sent recently, battery ok etc

Figure 32. The node firmware's internal state machine.

9.5 Software: app

I anticipate this aspect of the project to be the most challenging, as I have very limited experience with app development.

In my brief, I establish the need for the app to run on several different operating systems – primarily Android and IOS, though some large-scale trapping operations may use Windows, Mac, or Linux computers.

One way to achieve this would be to write a separate version of the app for each platform, using its native development tools. This is how large-scale developers of apps, such as Facebook, work. However, this is not practical for me as I do not have time to write many different versions of my code.

Instead, I will use an app development framework that allows code to be written once and compiled to many different platforms. Several of these exist, including Flutter (by Google), Appcelerator Titanium, and React Native (Facebook).

All of these options compile to roughly the same targets, though Flutter is the only one to target Linux. Embedded Linux devices could conceivably be used in a large-scale trap management operation, so it is worth having the functionality. Flutter also has an excellent library of pre-written software libraries. Therefore, I will use Flutter for my app development needs.

Traps should be listed on a map, with icons clickable for more information.

Another tab should display the user's profile. This will mostly be used for the public gamified component, so users can view, friend, and interact with each other.

Gamification is defined as the " the use of game design elements in non-game contexts " (Deterding, Dixon, Khaled, & Nacke, 2011), which exploits "people's natural desires for socializing, learning, mastery, competition, achievement, status, self-expression, altruism, or closure" (Lieberoth, 2014).

Gamification is widely applied. Successful examples include Pokemon Go and iNaturalist, both apps that require users to move to a real-world location and complete a task. Uniquely, iNaturalist uses this workforce for the documentation of plant species – this is genuine work that experts get paid to do, much like trap resetting.

Gamification draws off people's natural sense of competition. Users can gain points for each trap they reset. If a trap is unpopular, it will be worth more points – this is supply and demand in action. A user score, similar to Snapchat's snap score, will be assigned to each user and will be visible to their friends. This score will be a combination of the points derived from resetting each trap, plus extra points from weekly 'streaks', and other factors.

For the public component, users need a way of telling the system that they have reset the trap. A QR code or barcode attached to the trap could be scanned, sending a 'secret number' to the backend. This could be correlated with the node's own reporting of when it has been reset, and the user's GPS location.

9.6 Software: server

The backend server is essential for storing the state of traps, and allows the app the work.

Care must be taken to develop the server in a robust and extendable manner - many hundreds or thousands of traps and users may access it, whereas there will only ever be one trap interacting with the trap firmware, and one user interacting with their copy of the app.

9.6.1 Libraries

The most efficient way to build software in the 21st century is to stand upon the shoulders of giants - using free, open-source software libraries and frameworks. Each library is built to serve a particular purpose. For example, Sequelize is a library which serves as adapter between a MySQL database and the rest of the application. This hugely simplifies the job of interacting with the database, like every library simplifies the job it is designed for. Libraries are developed and maintained by teams of volunteers, and are thus thoroughly checked for bugs and sloppy code. This results in a huge collection of free, high-quality software, allowing me to focus on what makes my system unique, rather than re-inventing the wheel.

In the node firmware I am using the Arduino framework, as my 'lowest level'. The analogue of this in the server is Node.js. This is an open-source JavaScript server framework. This essentially serves as the programming language and framework from which everything else is built.

I will be using the following software frameworks and libraries:

NodeJS	Base language runtime
Express	Web server library for NodeJS
MySQL	SQL database allowing for quick storage and retrieval of data
Sequelize	An ORM (object relational mapping) which makes interacting with the database easy
JWT	Helps with authenticating users
BCrypt	Hashing library allowing secure storage of user passwords in the database
ValidatorJS	Ensures data entered is valid

9.6.2 About APIs

An API (application programming interface) is a structured way for two different pieces of software to communicate with each other. I will be using an API for the communication between the app and server. An API does not display text or images to a user - it simply mediates the flow of data.

I need a way of managing different groups of traps differently. Some traps are public, whereas others belong to different private owners. Therefore, I will introduce the notion of a group. A trap belongs to one or more groups. Users are members of one or more group(s). Public traps are all put in the "Public Traps" group, which every user is automatically a member of.

9.6.3 "Groups" concept

Some groups are private and require invitations from group owners/moderators. Members of a group can see the traps belonging to that group.

9.6.4 Structuring the application

Modern web applications can be very complicated. This necessitates a strict project structure to organise all code, lest 'spaghetti code' be generated. The server can be split into several different modules:

- Router: the router receives the query from the app, and decides what piece of code needs to handle it, based on the URL.
- Controllers: A controller handles a related chunk of the program's functionality. For example, the authentication controller handles logins, registrations, password resets, etc.
- Model: A model describes a table in the database, which represents some object. Models don't describe specific instances of these objects – they are a description of them in general. For example, the user model contains fields email, password, username, signup date, etc.

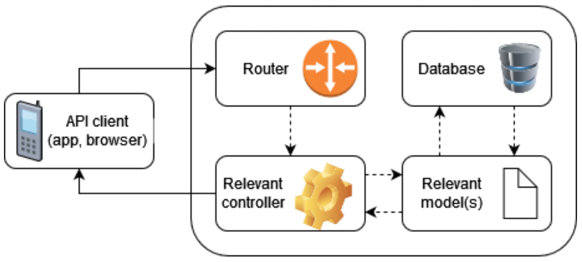


Figure 33. A description of what code subsystems handle a request.

9.6.5 Hashing passwords

Passwords are an everyday part of the internet. However, their storage requirements are unique. They must not be stored in a database in 'plaintext' form, in case the database is hacked. Instead, a one-way transformation is applied to them. This is called hashing them. When a user wants to log in, their password input is hashed and compared to the saved hash. If they match, the password is correct.

9.6.6 Persistent authentication

It would be very inconvenient for a user to have to log in every time they wanted to switch pages on a website. For this reason, the login process must give the user some unique secret 'token' which they can provide with all subsequent requests. When the server receives the token, it knows that the user is correctly logged in, and who the user is. This is the purpose of the JWT library – a token is sent to the user, which is securely encrypted, yet contains information about their username, permissions level, etc. The user sends this back whenever they want to make a request.

10 Stakeholders

At this point, it was time to contact a stakeholder for feedback on my chosen direction, advice on how to make the system fit into its environment, and some traps to design my detector switches around.

I decided that Predator Free 2050 (PF2050) would be a good place to start. They are the Crown-owned charitable company established to oversee our predator free 2050 goal. They provide funding for predator elimination projects, as well as science and technology. They have an annual "Products to Projects" funding system, where projects like mine can apply for grants to make a complete product. Of course, mine is nowhere near the level required. However, it's not impossible that I could enter it next year.

Since PF2050 has experience dealing with designers like me, I decided to contact them. I asked for:

1. Advice on how to make the system fit for the environment it will be deployed in, and more information about how existing sensors are used. Ideally, I'd have a contact who I would work with throughout the year as I create my product.
2. One or more traps to design around. I can make these myself, if need be, but 'real' ones I can have or borrow would be better.

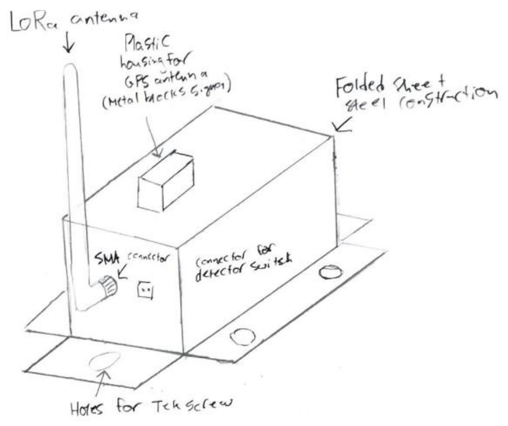
I received a reply from [redacted], the Research and Development Project Support Manager at PF2050. She suggested I contact Simon Croft of Celium, one of the competing trap monitoring systems I discussed in 8 Analysis of Existing Devices above. She also said she would look into getting me some traps to design around.

I emailed [redacted] and unfortunately received no reply – this may be because he doesn't want to share his trade secrets, which is fair enough.

A few days later, I received another email from [redacted] confirming that PF2050 would be willing to buy me a couple of traps. We settled on a Victor Professional + tunnel, and a DOC200. These were mailed to my school.

11 Concepts

11.1 Trap concept 1


LoRa antenna
Plastic housing for GPS antenna (Metal blocks signal)
Folded sheet steel construction
SMA connector
connector for detector switch
Holes for Tek screw

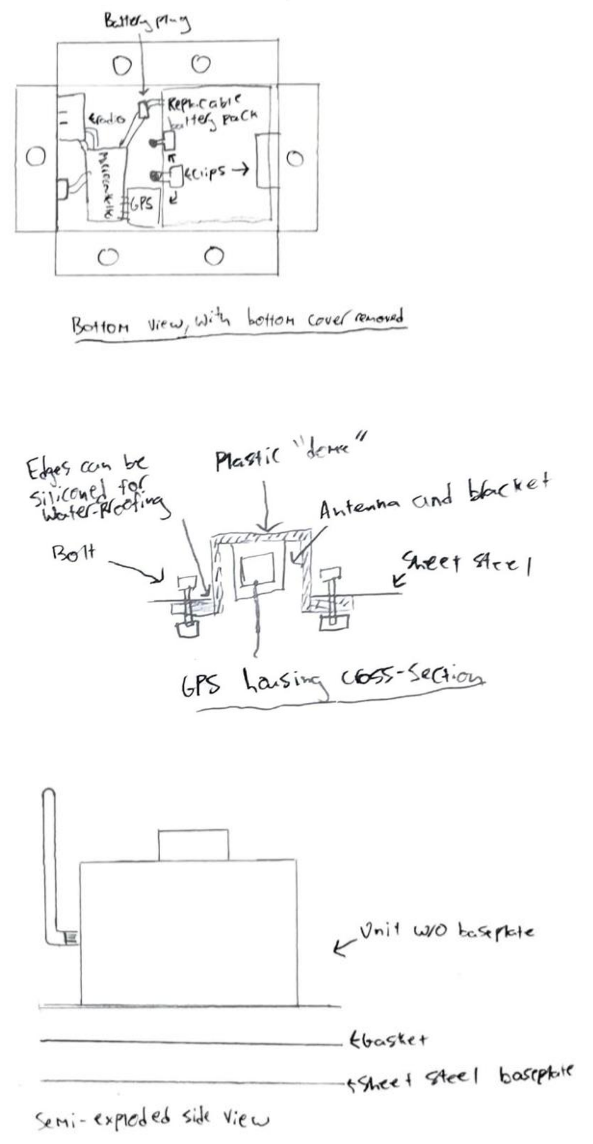


Figure 34. Concept 1 drawings.

This concept consists of a rectangular box made of folded and welded sheet steel. The GPS antenna is mounted in a 3D printed plastic 'dome' to ensure the signal is not blocked by the steel. The LoRaWAN antenna may be vulnerable to damage.

Pros

- Easy to mount.
- Fairly simple construction.
- Strong waterproofing.

Cons

- LoRaWAN antenna may be vulnerable.
- GPS dome is a bit complex and may cause waterproofing issues.
- Battery cannot be replaced without unscrewing unit from trap, as it is accessed from the bottom.

11.2 Trap concept 2

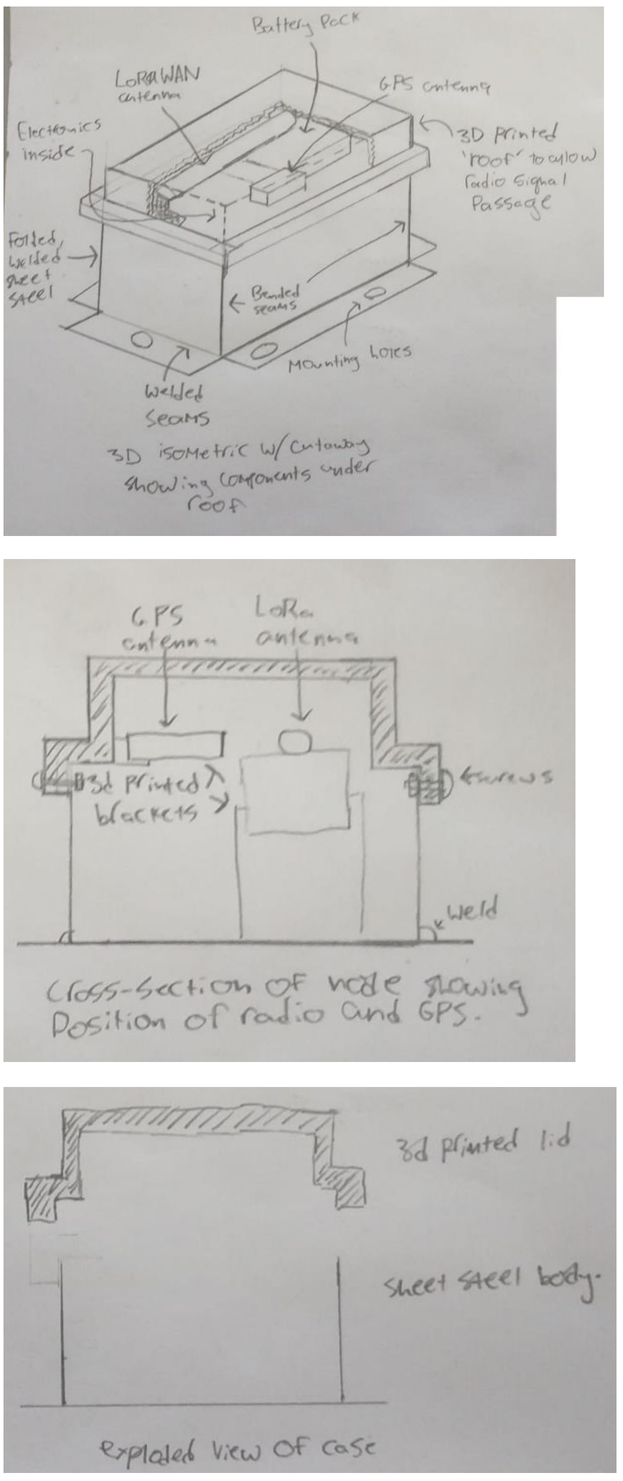


Figure 35. Concept 2 drawings.

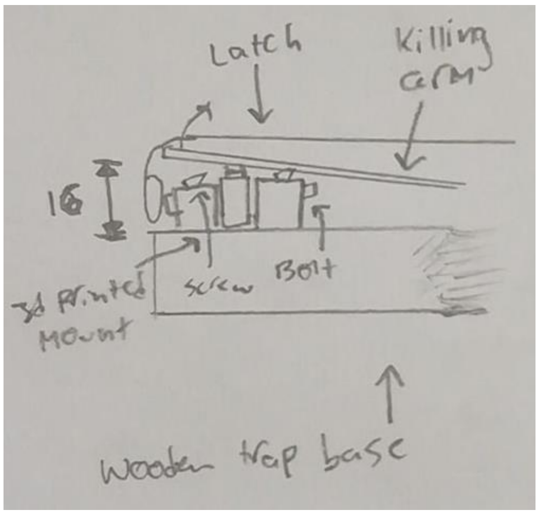
Pros

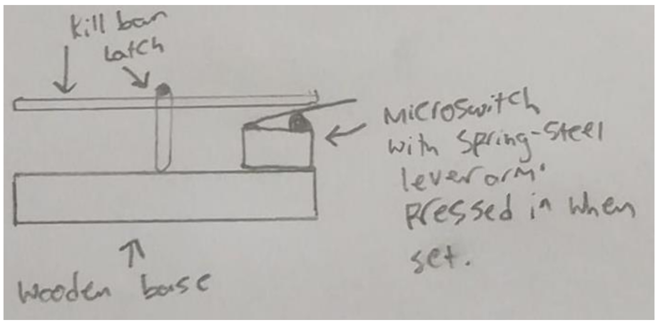
- Both antennae are protected from knocks and are weatherproof.
- Simple base construction.
- Easy to mount on trap.

Cons

- Lid is complex and may cause waterproofing problems.
- Lid is fragile plastic which somewhat defeats the point of making the rest of the body out of sheet steel.

11.3 Detector switch: Victor rat trap




Kill bar
latch
Microswitch with spring-steel leverarm. pressed in when set.
wooden base

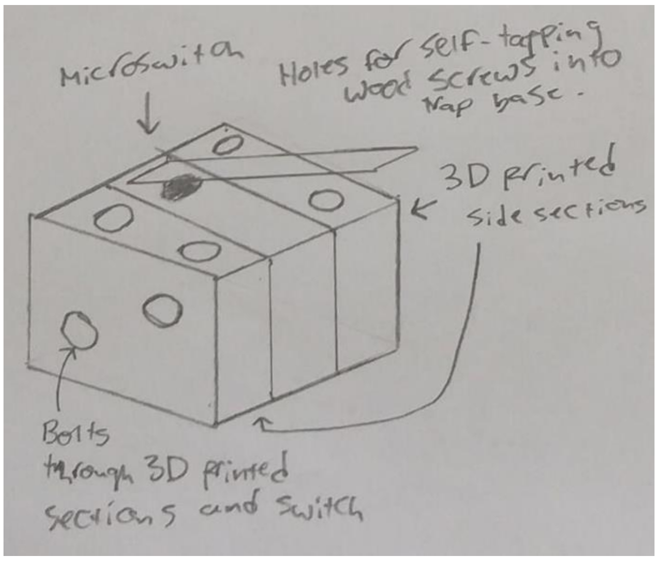

Microswitch
Holes for self-tapping wood screws into trap base.
3D printed side sections
Bolts through 3D printed sections and switch

Figure 36. Victor detector switch concept.

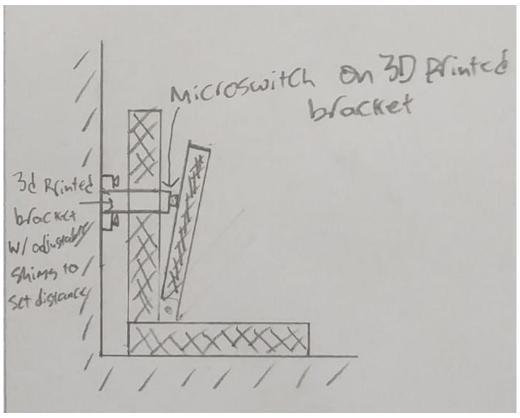
Pros

- Simple design and construction.
- Easy to attach to trap.

Cons

- Requires a waterproof switch.
- Cables leaving switch may be fragile.

11.4 Detector switch: DOC200 microswitch design


Microswitch on 3D printed bracket
3d printed bracket w/ adjustable shims to set distance

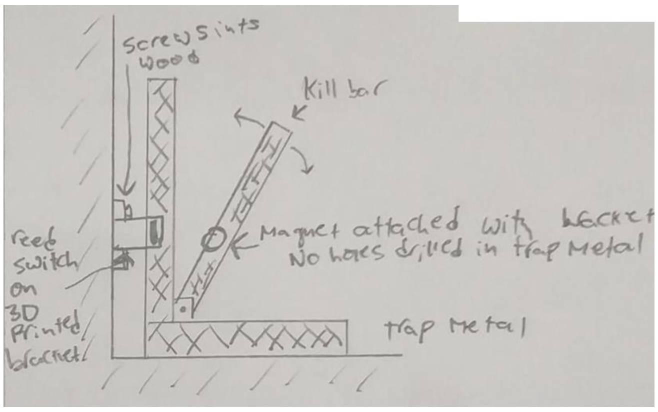
Pros

- Microswitches are simple, cheap, and reliable, as per 9.1.7 Detector switch above.
- Doesn't require any attachments to the kill bar, which would likely fall off from the force.

Cons

- Requires accurate calibration relative to the position of the kill bar. The calibration would likely vary by individual trap as it depends on the position of the trap relative to the box. Shims could be used.

11.5 Detector switch: DOC200, reed switch design


Screws into wood
Kill bar
Magnet attached with bracket No holes drilled in trap metal
reed switch on 3D printed bracket
trap metal

Pros

- Reed switches are cheap and reliable.
- Doesn't require physical contact between magnet (on kill bar) and switch, so less accurate calibration required.

Cons

- There is no good way to attach the magnet to the kill bar without drilling holes in metal, which is not feasible in the field. The kill bar experiences enormous force which would destroy the magnet or knock it off.

As per 9.1.7 Detector switch above, I will use the microswitch design for this detector switch.

11.6 System architecture concept

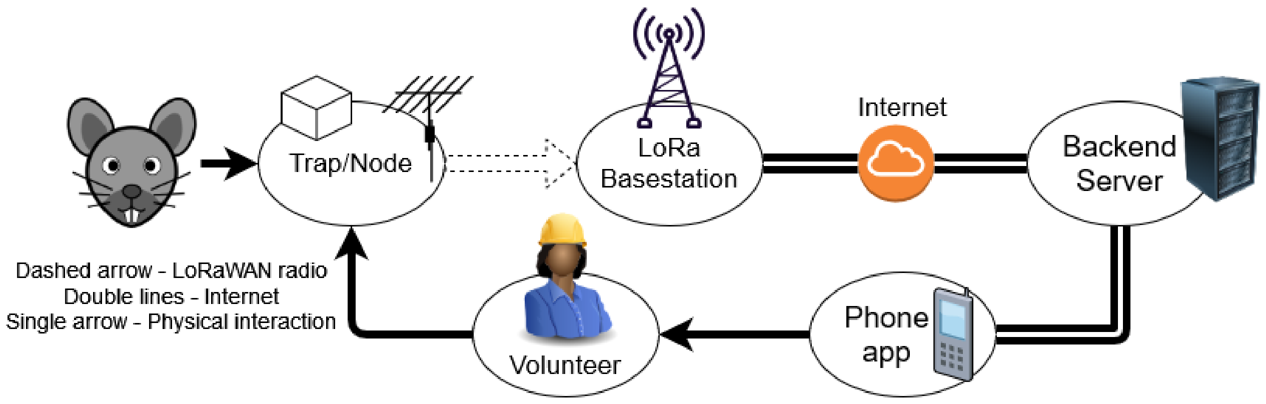


Figure 37. The proposed system in its entirety.

12 Development Drawings

Initially, Trap concept 2 appeared to be the most promising to build upon, as it had much simpler sheet metal components – Trap concept 1 requires a watertight seal between plastic and metal components for the GPS antenna tower. However, after starting to make a full design based on Concept 2, I realised that due to antenna polarisation, the LoRa antenna must be mounted vertically, or else major signal loss will occur. This concept is touched on in 9.1.1.9 Antenna above. In Concept 2, the antenna is mounted horizontally under the plastic shield which would not work with a vertically-polarised gateway. I also was growing to dislike the use of a plastic lid, since this largely defeated the point of having a steel body. It also complicated waterproofing. Therefore, I produced development drawings based on Concept 1.

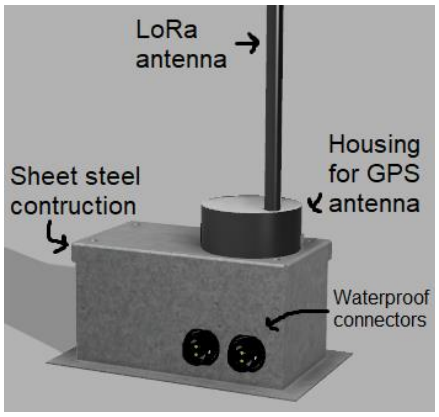


Figure 38. The assembled node.

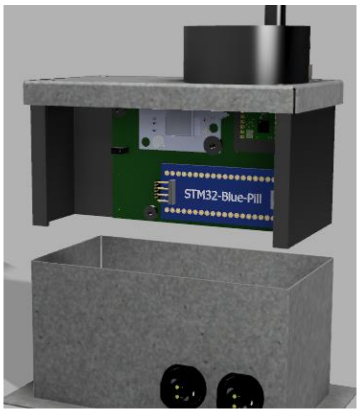


Figure 39. The node being disassembled. The steel lid is fixed to the plastic internal caddy.

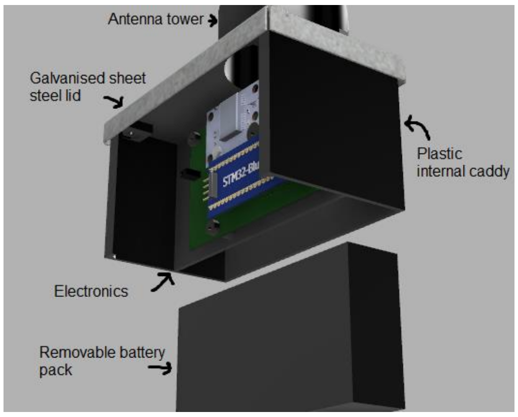
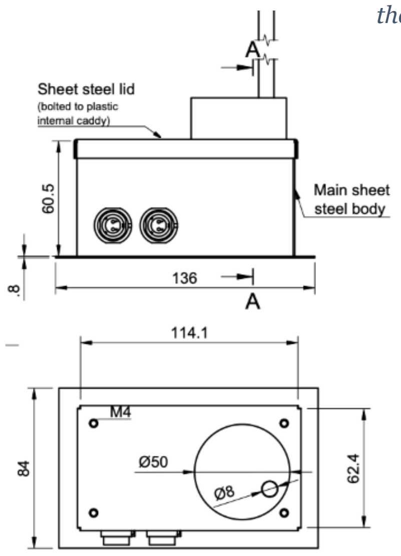
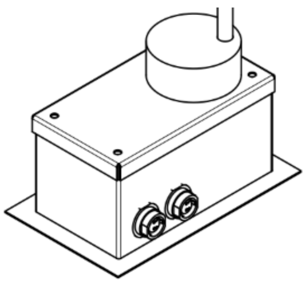
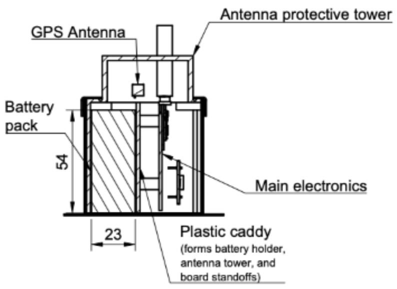
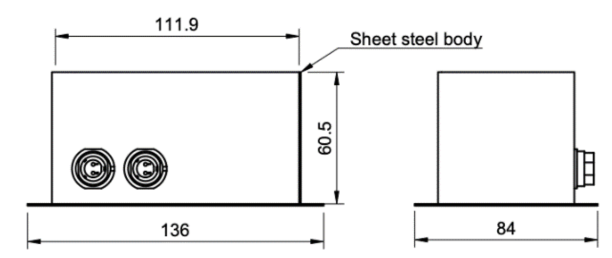


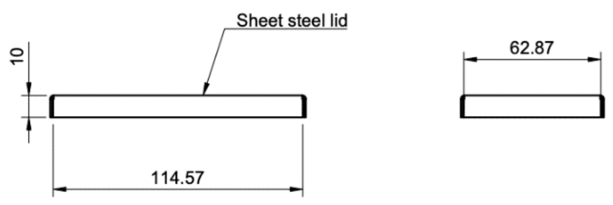
Figure 40. Exploded view, showing the battery pack being removed.

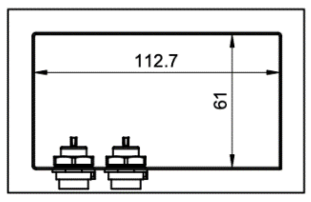


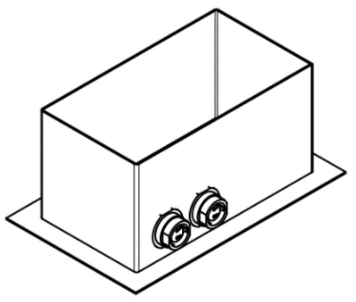


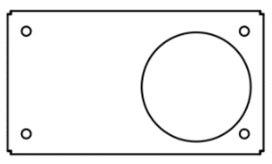


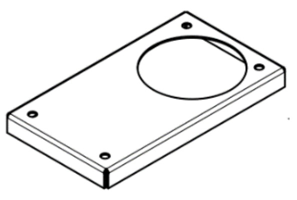

111.9
Sheet steel body
60.5
136
84

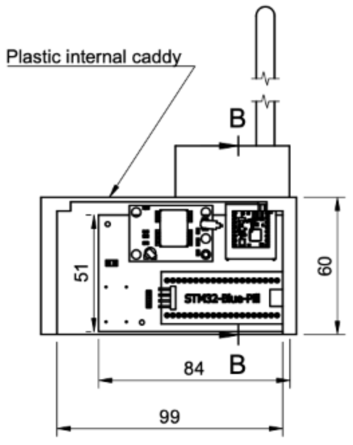

Sheet steel lid
10
114.57
62.87

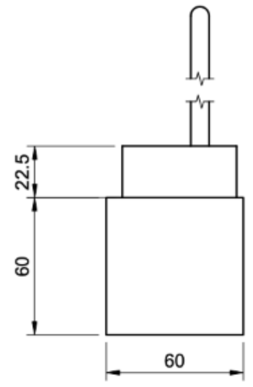

112.7
61

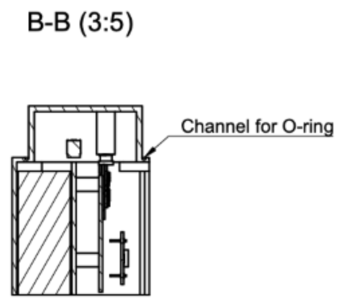


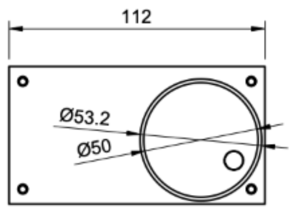


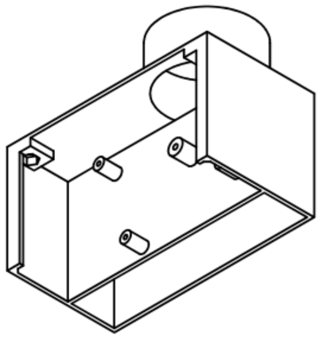


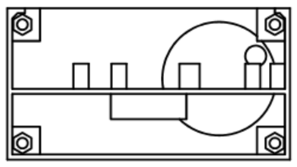

Plastic internal caddy
B
51
60
STM32-Blue-Pill
84
B
99

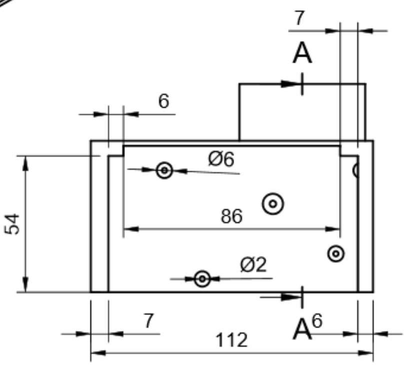

22.5
60
60

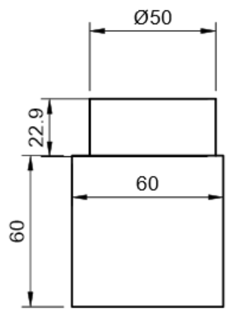

B-B (3:5)
Channel for O-ring

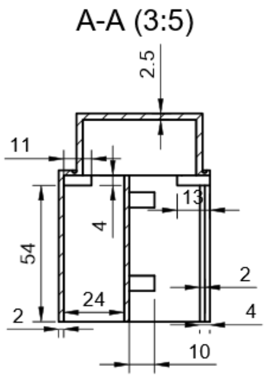

112
Ø53.2
Ø50

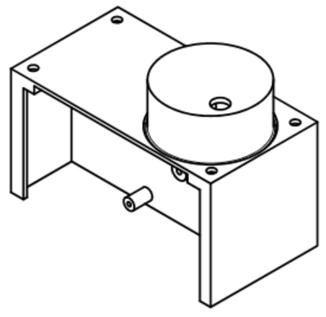


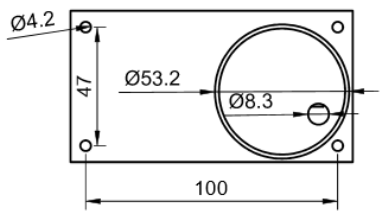


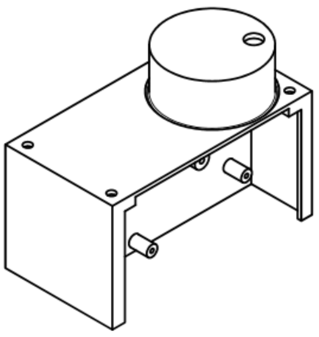

7
A
6
Ø6
54
86
Ø2
7
112
A
6


Ø50
22.9
60
60


A-A (3:5)
2.5
11
13
4
54
2
24
2
4
10




Ø4.2
47
Ø53.2
Ø8.3
100



13 Modelling and Development

13.1 Cardboard modelling

To understand the true scale of my design, I created a cardboard model of the node housing. The antenna cylinder was not created as it is hard to make a cylinder out of cardboard and was not necessary for determining the relative scale of the system.

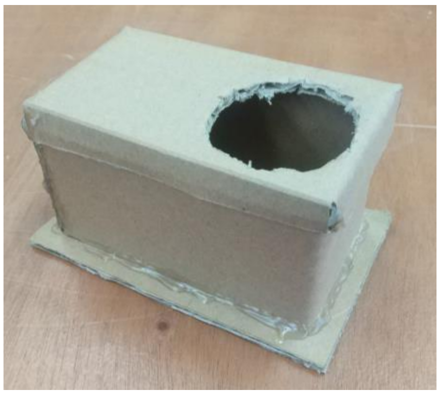


Figure 42. Cardboard model

Figure 41. Lid

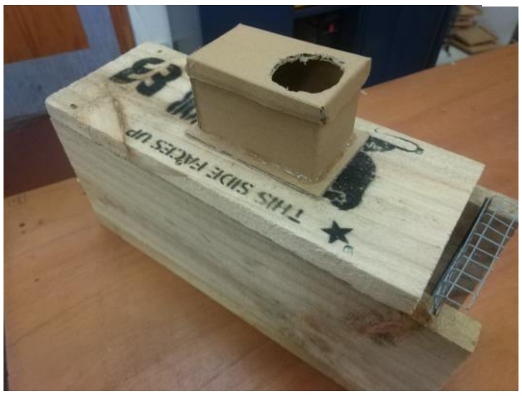


Figure 43. Cardboard model relative to a trap.

13.2 Caddy

13.2.1 About 3D printing

The next stage was to create a functional model of a node out of the real materials, starting with the internal caddy. The caddy is made out of plastic, so it will be 3D printed.

In production, plastic parts are usually made by injection moulding: a steel mould is made, and molten plastic is injected into the mould. When the plastic cools, the mould is split, and the part removed. The process repeats. This has a very high cost of entry – making moulds and buying an injection moulding setup is not cheap. As an alternative, for prototyping, 3D printing is widely used. This takes a relatively long time to create each part, and each part is more expensive than if it was injection moulded – though obviously cheaper than making a new mould for every change.

I will be using the fused deposition modelling (FDM) 3D printing technique, in which a moving nozzle deposits molten plastic, which cools down into a solid object. This is because it is easier, cheaper, and less toxic than the alternatives.

FDM 3D printers can create objects out of several types of plastics. The most common of these is PLA (polylactic acid) derived from plant materials. This means it is relatively biologically safe and friendly to the environment. Another popular plastic is ABS (Acrylonitrile butadiene styrene). This is a mixture of several hydrocarbon-derived chemicals which form into a hard plastic, which is marginally stronger and more heat-resistant than PLA.

My product is to be deployed outdoors. Therefore, it could conceivably break, leaving broken pieces of plastic. To reduce the environmental impact if this happens, I will use PLA, as this is a non-toxic plastic and is somewhat biodegradable. It also produces less toxic fumes in the printing process and is generally easier to work with.

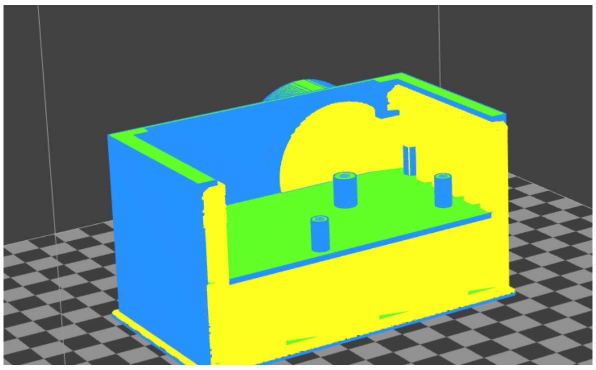
3D printers are not good at printing overhangs. This is because they print layer-by-layer, with each layer supported by the one below it. The largest overhang angle possible is usually around 55 degrees. To allow for overhangs steeper than this, removable supports can be printed along with the main model. This support material is usually disposed of after being removed, so it is a good idea to minimise the amount used.

When operating the 3D printer, I will wear gloves and safety glasses to mitigate the risk of burns. I will use PLA filament where possible, which does not produce toxic fumes. I will not put my hands inside the machine while it is switched on to avoid jammed or burned fingers.

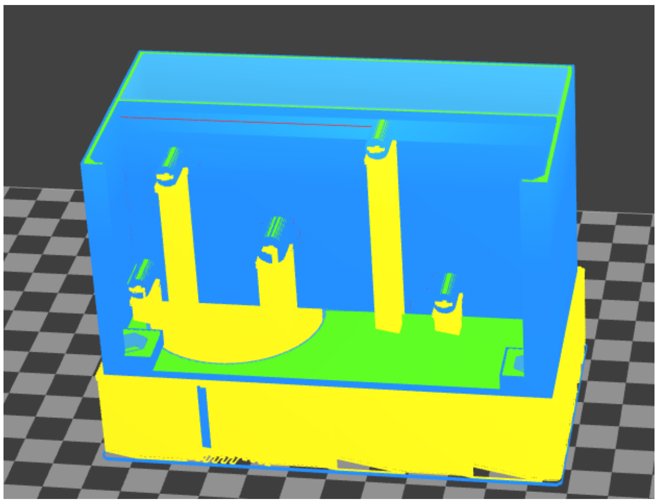
13.2.2 Slicing

Because the part is largely hollow, support material is almost certainly required. It is important that support material is not placed in difficult-to-reach places, where it may not be easily possible to remove it without breaking the part. Two orientations of the part were tested in the printer's software, which reports the amount of filament necessary. Choosing the lowest mass of plastic allows for the most efficient print.

Option A (159g):



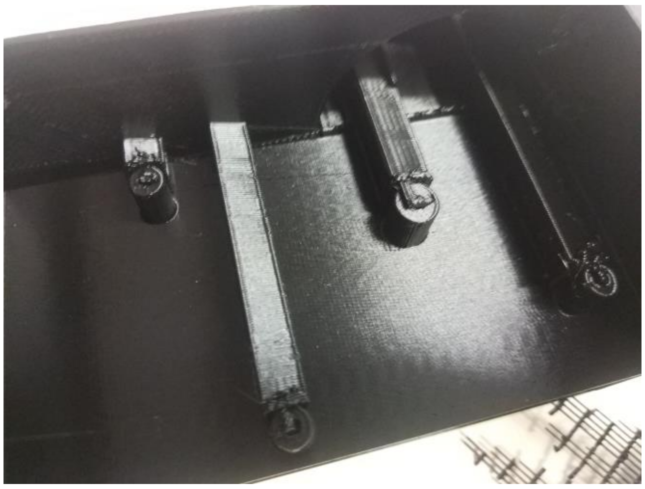
Option B (140g):



In option A, a large amount of support material is used (yellow), especially in the battery compartment which is entirely filled with support. In option B, support exists all around the base (except for the antenna tower). This would be easier to remove than in the battery compartment. Supports also exist for the circuit board standoffs.

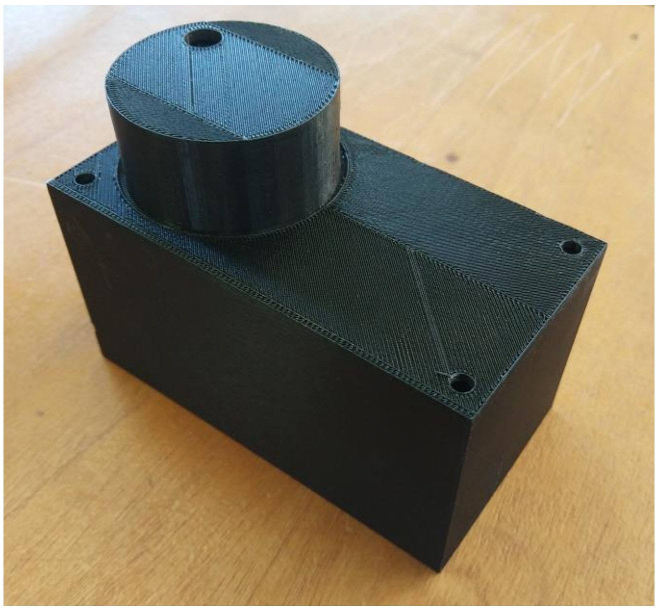
Option B requires less mass of support, and it will be easier to remove. Therefore, I will print in orientation B. Because this is an early model, a high-strength and high-quality print is not required: speed and material-saving are more important. Therefore, the printer is set to 'fast' speed, with a large layer height and low density.

13.2.3 Printing



This photo shows some of the support, prior to removal.

Removing the support material is a fairly messy process. It is clear how much plastic is wasted by using supports, and why minimising their use by changing orientation is important for efficiency.



13.2.4 Evaluation of print

This print worked quite well. The support material was easy to remove and came off cleanly. The plastic is robust, and the walls are thick enough to withstand damage.

One minor issue is the lines on the top face of the box. I believe these are caused by a slight error I made while rotating the part in the slicer software, where the box was rotated one or two degrees from horizontal. This is such a small change that is causes no problems to the box's function, but it causes unsightly lines.

13.3 Sheet steel housing

It was now time to make a model housing out of sheet steel.

Three sheet steel pieces were to be cut, folded, and welded, resulting in two parts: the main 'container', and a lid.

The 3D model of the housing can be turned into a ‘flat pattern’, showing the cuts and folds required.

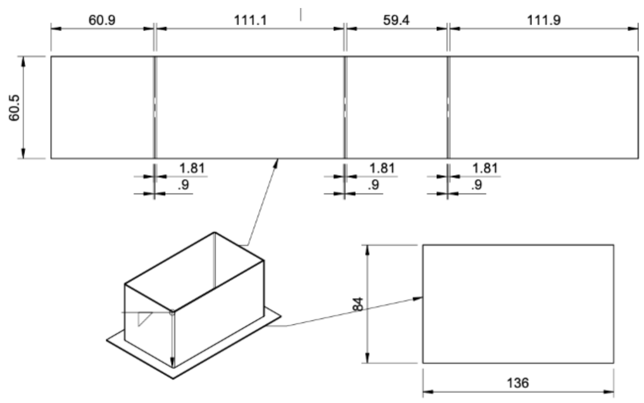


Figure 44. Flat pattern of housing.

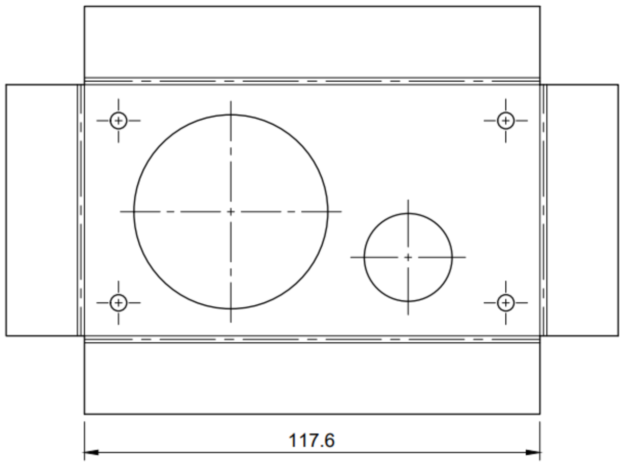
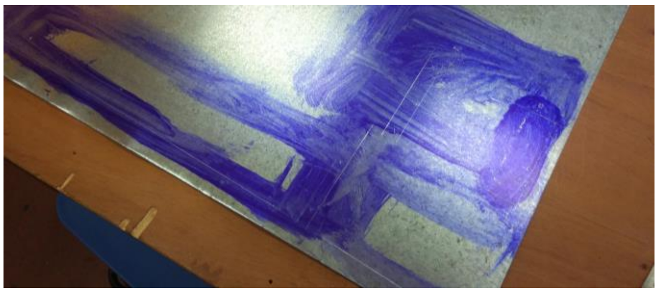


Figure 45. Flat pattern of lid.

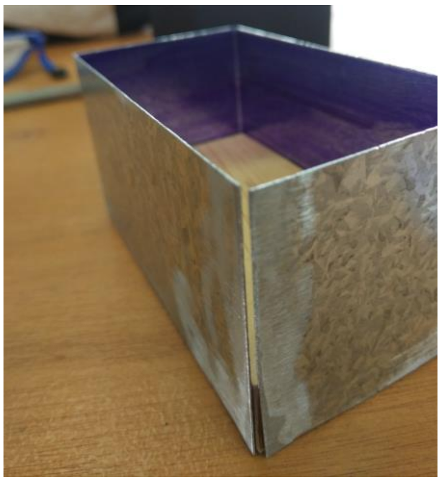


Blue engineer's dye is applied to sheet steel stock to increase the contrast between scribed lines and metal background. Lines are scraped into the stock according to the drawing using a square, ruler, and scriber.

The steel is cut with a bench shear, which operates similar to a paper guillotine. Gloves are worn to protect against the metal's sharp edges.

The cut pieces are again applied with dye, and scribed with lines for where they must be folded. The first piece to do is the container wall, which require 3 bends to go from the long piece in centre shot above to a closed rectangle.

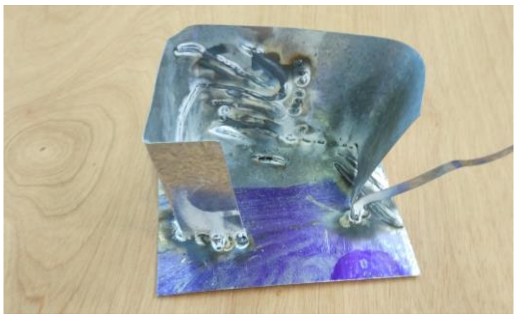
Three bends are made with a cornice brake, forming a closed rectangle.



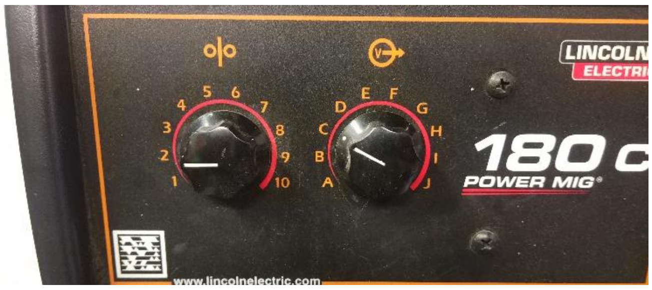
Of course, one of the corners is not connected. This must be solved by welding the corner together in a watertight fashion. A rectangular piece of wood (visible above, inside the walls) is cut to brace the walls in the required shape while welding.

The housing base was made in similar fashion. It requires some holes to be drilled in it for screws mounting to a trap tunnel.

I must now weld the pieces together. Safety equipment is essential when welding. Overalls and an apron are worn to protect my clothing, and a welding helmet is worn to protect my eyes from blinding UV light emitted by the welding arc. Before welding, I need to determine the welding settings matched to the material.

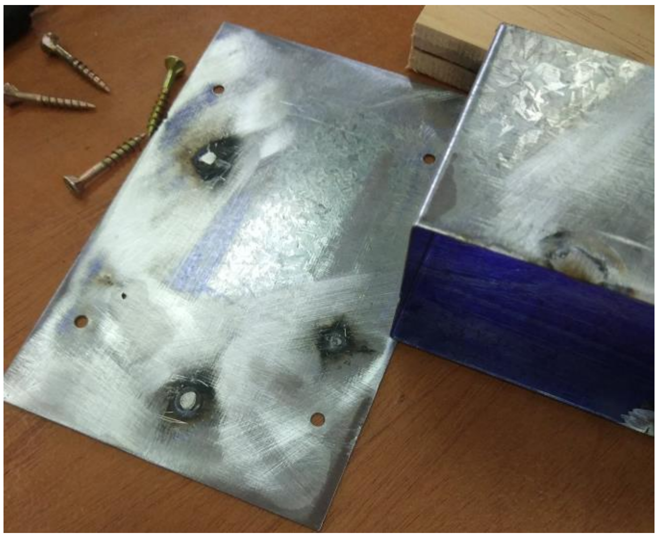


This piece of scrap metal was used to tune the settings, so the welder penetrated the metal sufficiently to make a good bond yet did not put holes in it.



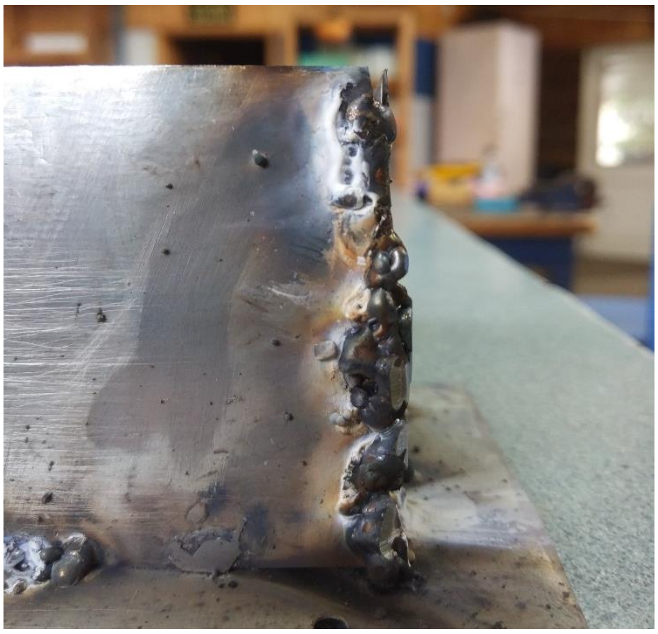
The optimal settings found for this material are shown above.

My photographer held the base and walls together, and I tack welded them. Tack welds are temporary welds that are meant to hold the work pieces together while the main welds are done, eliminating the need for bulky clamps.

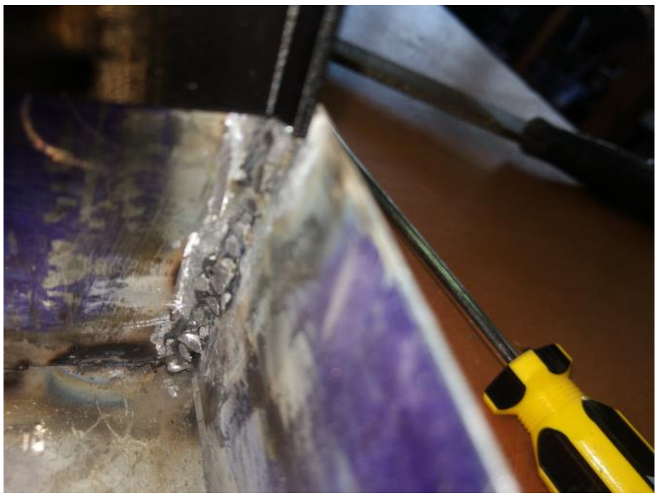


Unfortunately, my strategy of having someone push the parts together did not work very well – there was a large gap between the pieces. I cut the tack welds and was ready to try again.

This time, I clamped the work properly. This will minimise the gap between the walls and base. As before, I welded the parts together.



This is *not* a good weld. I think there are two reasons – I am attempting to weld galvanised (zinc-coated) steel, but the zinc disrupts the welding arc. Also, I have very little experience welding sheet steel. Next time, I will strongly consider not using galvanised steel.



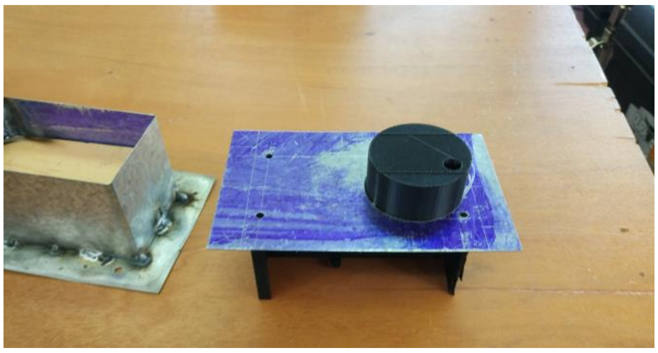
To remove the worst of the welds, leaving only the useful parts, a Dremel and grinder are used. This does not fully work inside the housing, but it is much better. In the future, I should design or weld this in such a way that doesn't result in difficult-to-grind internal welds.

This slightly bulging weld prevents the caddy from fitting. This is easily solvable by cutting some excess plastic from the caddy.

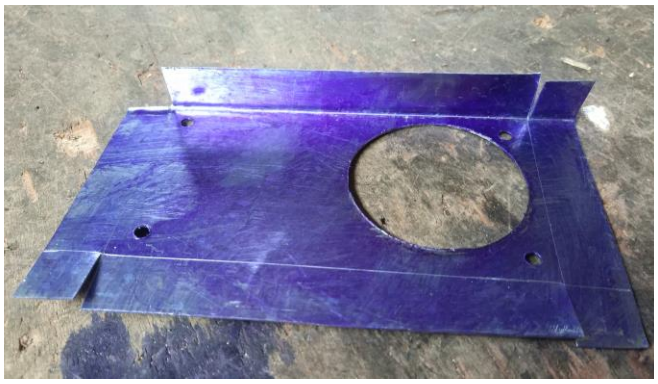
Now that the base box is done, and the caddy fits, I must make the lid. Metal is dyed, marked, and cut as before.

I then drill the holes – 4 small holes for the bolts that hold it onto the caddy, and one large hole for the antenna tower.

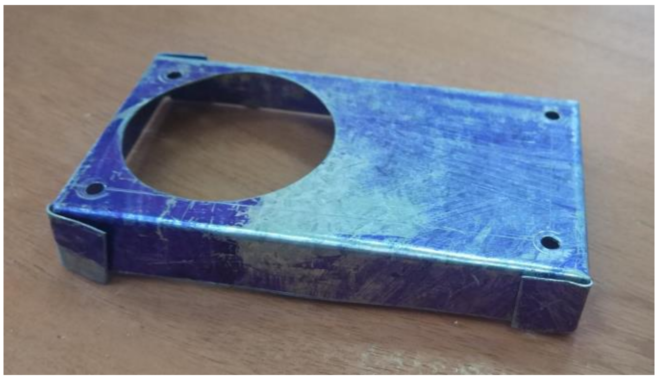
A hole-saw is used to cut the larger hole.



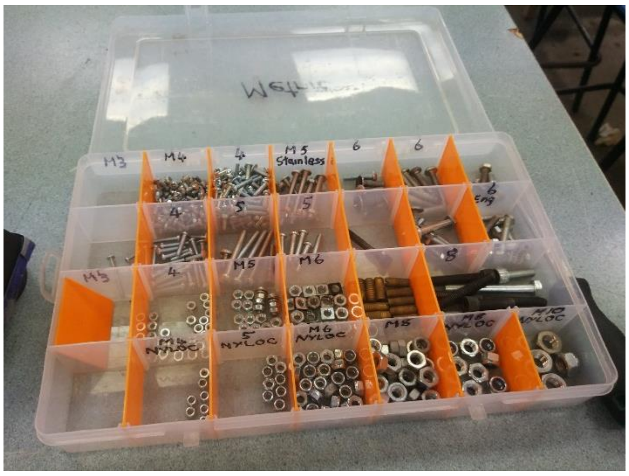
The antenna tower fits through the hole-sawed hole, and the bolt holes line up.



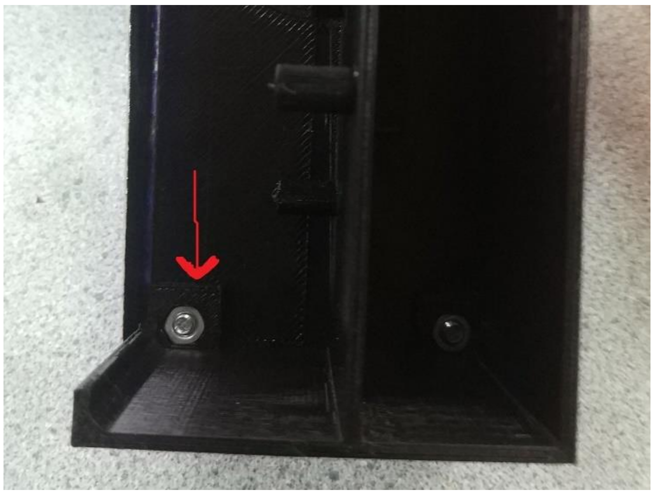
Slots are cut and the sides of the lid are folded down. I initially planned to cut off the extra flaps of metal (like in the top left corner), however I realised that these could help provide more waterproofing and make for easier welding, so I kept the other three. I will keep all 4 in the next version.



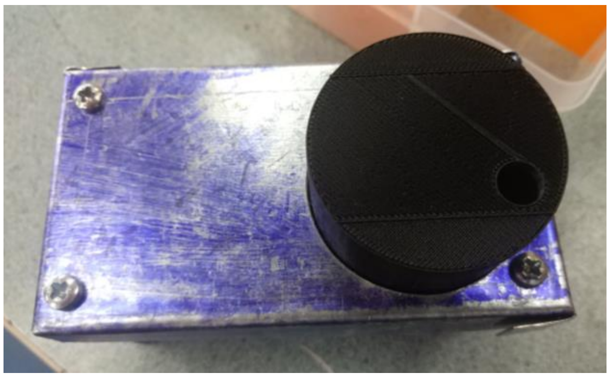
The finished folded lid. Because this is an early model that isn't intended to be waterproof and considering the issues I faced with welding galvanised steel prior, I chose to not weld the corner flaps in place for this iteration.



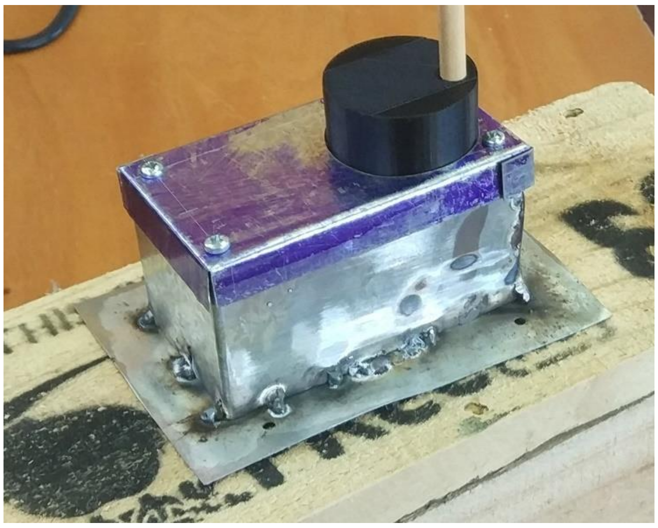
To connect the lid and caddy, matching bolts and nuts must be used. Nuts are pressed into special hexagonal holes in the plastic caddy, while the bolt heads protrude from the lid. My parts are designed for M4 bolts, though this is likely overkill. M3 or even smaller would likely work fine.



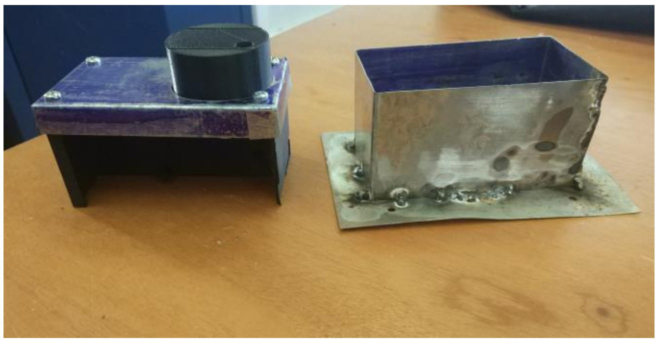
The nut pressed into hexagonal holes in the caddy.



The lid attached to the caddy.



The fully assembled model housing, on top of a trap.



The caddy/lid assembly separated from the base box.

13.3.1 Evaluation of caddy

This first attempt to create a metal housing was largely successful. I did, however, find a few things to change before attempting to make the real thing.

Welding galvanised steel is difficult and a bad idea. Welding necessitates grinding away the galvanisation in many places anyway, leaving exposed metal which must be painted. This defeats the point of using galvanised steel. In the future, I will use normal non-galvanised steel (which is far easier to weld) and paint it.

My tolerances as designed in CAD were a bit too tight. The actual bend radius of the steel was much larger than what I had designed for, and the gap between the lid's lip and caddy walls was too small, making it tricky to get the container's walls in this gap. Therefore, I will adjust my sheet metal design settings and increase the tolerances.

I will not cut off the lid corner's flaps. Instead, these will be folded over and used to aid welding and waterproofing.

The container walls will be welded before attaching them to the container floor, so that the weld can be ground on both the inside and outside.

The caddy will be reshaped to give some room for slightly bulging internal weld beads.

13.4 Electronics

13.4.1 Circuit design

The individual subsystems and corresponding pieces of hardware are discussed in 9.1 Node hardware above. However, we must determine how they are connected together. Different modules communicate over different electrical and software protocols.

13.4.1.1 GPS

The GPS communicates using UART, the same as the communication between the microcontroller and a connected computer. Luckily, the STM32 has three independent UART ports, allowing for both GPS and the configuration interface to run simultaneously. UART requires two data wires - transmit and receive, as well as ground and power for the module. The transmit pin of one module is connected to the receive pin of the other, and vice versa. This allows for bidirectional communication between two devices.

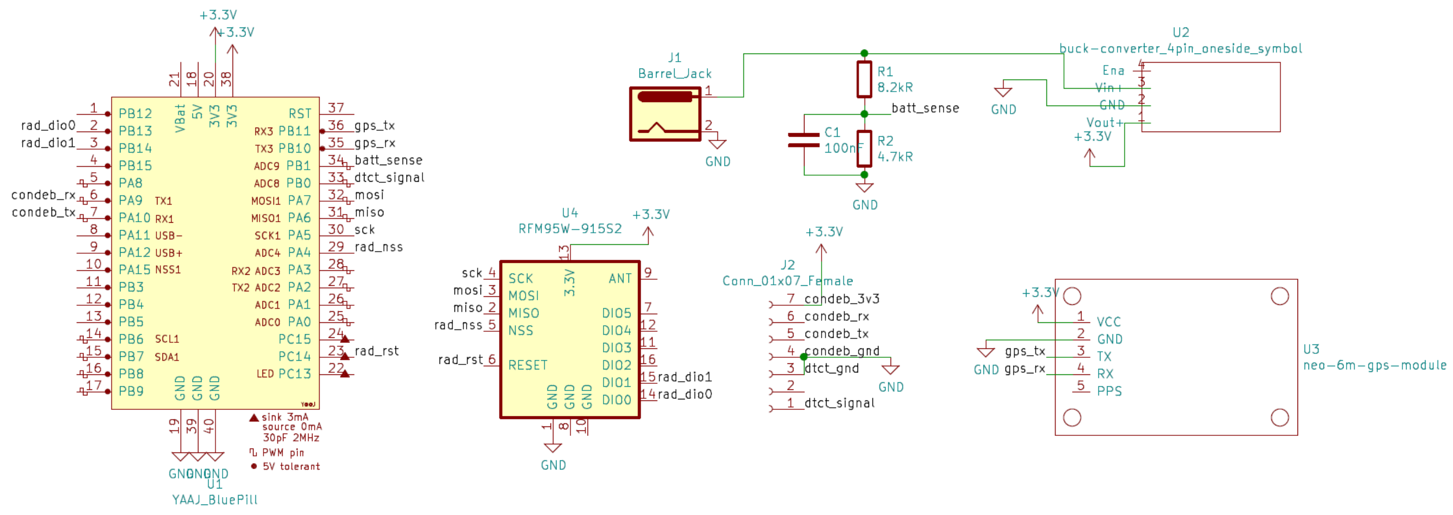
13.4.1.2 RFM95 radio

The RFM95 radio module communicates using the Serial Peripheral Interface (SPI). This is a common data communication protocol which can connect a single 'master' device to multiple 'slave' devices. The microcontroller would be the master device, and the RFM95 the slave. SPI requires three data wires that are shared between all devices on the bus, plus one 'slave select' wire per slave. For my one slave, three data wires are used for SPI. The RFM95 also requires some supplementary connections: reset (restarts or switches off the radio), and DIO0/DIO2 (two pins which carry supplementary information e.g., when the transmit is finished). This is a total of 7 data wires, as well as power and ground.

13.4.1.3 External connections

Connections must be made with the outside world, in the form of the two SP13 plugs. One of these is for the detector switch in the trap, and requires ground and one signal wire. The connection to the configuration and debugging computer (I refer to it as 'condeb') requires a simple UART interface, which consists of two data wires, as well as power and ground. This is 6 pins in total. A socket will be placed on the main circuit board, allowing the SP13 plugs (mounted on the housing) to be removed or connected when necessary. A blank pin will be used to ensure the plug is not inserted upside down, meaning the connector will have 7 pins, only 6 of which are connected to anything.

My circuit design follows.



13.4.2 Printed Circuit Boards

13.4.2.1 *What's a PCB?*

There are a number of ways to build a circuit. The most basic is called freewiring - individual, flexible wires are soldered between all connecting points in the circuit. This takes excessive wire, space, time to solder, and is very error-prone as all connections must be done manually. It also introduces the risk of wires touching each other and shorting out, damaging the circuit. Freewiring is good for basic circuits, but quickly becomes impractical for those as complex as mine.

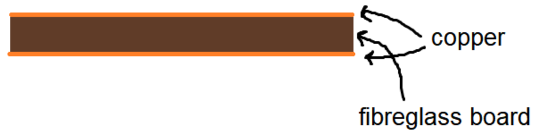
Another way to prototype circuits is by using perfboard. Perfboard is short for perforated circuit board, and is essentially a blank PCB filled with a grid of standardly-spaced copper-plated pads, to which components can be soldered. This holds components in place like a proper circuit board, however all connections must still be made with individual free wires.

The most advanced way to prototype (and produce) circuits is to make a custom printed circuit board. Circuit boards of this type are essentially a flat piece of fibreglass with very thin copper wires attached to the top (and sometimes bottom, if a circuit is especially complex). Proper circuit boards are covered with an electrically insulating 'solder mask' to prevent shorts and to improve the board's appearance. It is, however, possible to produce custom circuit boards yourself. There are several approaches to this, however all start with the same material and have similar basic steps.

I need to make two PCBs: A 'mainboard' which will house all my electronics, and the aforementioned RFM95 breakout board, designed by attexx on Github.

13.4.2.2 *How to make a PCB*

First, copper-clad fibreglass board is purchased. This is the 'blank canvas' of the circuit board world. It consists of a fibreglass board coated on the top (and bottom if double-sided) with a very thin layer of copper metal. The side profile of this material (if double-sided) looks something like this:



Single-sided copper-clad board looks like this:

Note the shiny copper side and brown fibreglass side.

To make a circuit board, copper must be removed everywhere on this board except for where wires are desired. This can be achieved many ways.

Both involve 'masking' the areas of copper that are to be kept (with a chemical-resistance cover), then placing the board in a chemical that removes (etches) the exposed copper. After the unwanted copper is removed, the mask can be washed off by a different chemical, such as acetone.

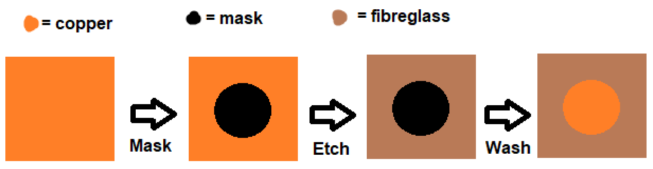


Figure 46. The basic steps of PCB manufacture.

13.4.2.3 *Toner transfer*

One very popular way to produce circuit boards is to use the toner transfer method.

The mask is printed, by a laser printer, onto glossy paper, such as that out of a magazine. The mask is temporarily attached, face down, onto the surface of the copper (which has been pre-cleaned with acetone and steel wool to remove the oxide coating). The paper is then ironed, or otherwise pressed and heated. This, in theory, transfers the toner onto the copper, acting as a mask.

Pros

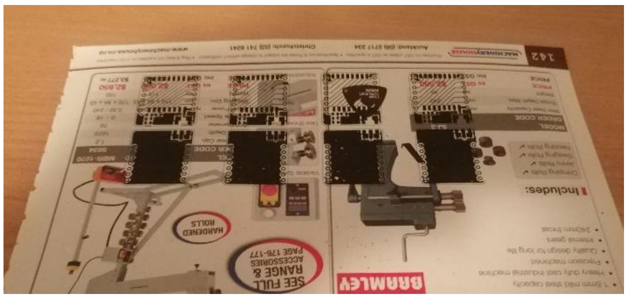
- Quick and simple, if settings are already known.
- Can be done with everyday tools and materials.

Cons

- Lack of repeatability - pressure must be very even and in just the right amount. Incorrect temperature will result in no transfer, or a squashed, 'bleeding' transfer.
- Printer-dependence. Different printers and toners work differently, and some work much better than others.
- Difficulty to tune the settings. A lot of work has to be invested into finding the correct temperature, pressure, time, and other parameters specific to my printer. I don't even know if my printer and toner is very good for toner transfer in the first place.

As this was the simplest option, I had a tiny bit of experience, and was doable with on-hand materials and tools, I decided to try it, despite its shortcomings.

First, the circuit is exported from KiCAD (my PCB design software) as an SVG image. Inkscape is used to duplicate the image, so that multiple images can be printed on a single page. It is printed using a standard laser printer (not inkjet, as the properties of the toner are essential). Glossy paper is used which prevents the toner from firmly adhering to the page, allowing for a better transfer onto the board.



It is essential that the copper surface is clean for the toner to properly transfer. Therefore, surfaces are cleaned with steel wool followed by an acetone wipe. This removes paint, oxides, oils, and miscellaneous grime, leaving only metallic copper.

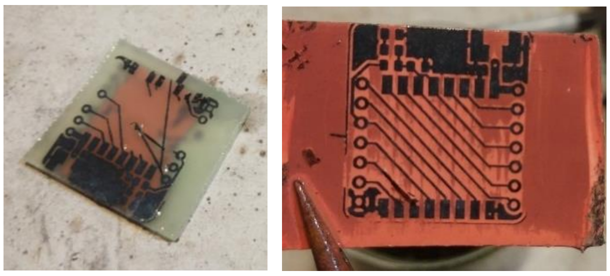
The next step is to attach the paper with the transfer to the clean board, face down. The paper can then be fastened with to the board with tape.

The paper must now be heated and compressed, which should transfer the toner from paper to board. A large amount of pressure must be applied very evenly, at a specific temperature.

One popular way to achieve this is with a standard clothes iron. A piece of A4 paper is folded and placed between the iron and glossy paper for protection from rips and to help distribute heat more evenly.

Once ironing is finished (about 5 minutes, though determining the optimum time is very difficult), the paper can be removed. This is made easier by running the board under water, dissolving the paper, and cooling the board.

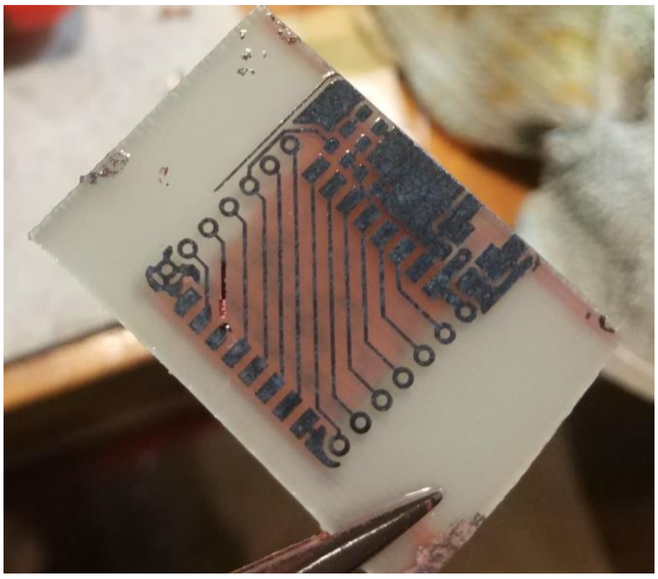
I then placed the board into etching chemicals.



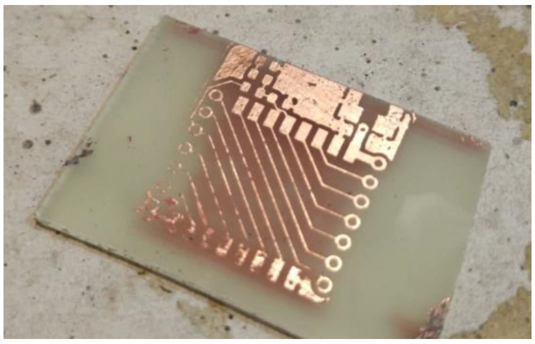
The toner transfer process proved a source of constant trouble and lack of repeatability for me. One common problem was tracks incompletely transferring, or coming off the board during etching. Also note the massive amounts of smudging occurring in the bottom-area of the board, in the left-hand photo above. The best transfer I achieved was the right-hand photo above. The photo was taken mid-etch, observe the pink copper metal with no oxide layer.

Also note the touch-up with a pen in the bottom left corner of the photo. This is a fairly decent transfer, though etching still did not go according to plan.

The same (best) transfer post-etching, before washing:



This appears acceptable, though there is a fair amount of bleeding in the top-right area. However, after removing the mask, it is evident that the mask did not effectively protect all the copper underneath.



The primary issue with this board is the uneven transfer of toner (less in the bottom left corner of the board), resulting in missing copper, leading to a useless board. This is a symptom of the wider issue with toner transfer – it is very difficult to get a repeatable result.

With much more time and effort, I believe I would be able to create a fairly good board, and possibly make the process reliable. Online research suggests that certain brands of printer and toner are better or worse. I was printing on my family's Brother printer, using an aftermarket toner, which is considered to be a bad combination by those in-the-know. It is possible that using a different printer would solve some of my problems.

However, a new method for applying a mask looked more promising.

13.4.2.4 Laser cutter masking

The board is covered with a thin layer of spray paint, which acts as a mask. The unwanted parts of the mask are removed by a laser cutter, essentially burning the paint off. The underlying copper is not affected, as it is a metal which is far harder to burn through than paint. This leaves the mask where it is wanted, and bare copper everywhere else.

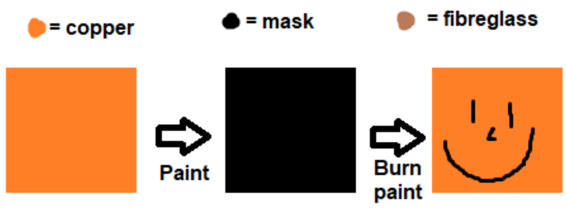


Figure 47. The process of laser-applying a mask. The masked board can then be etched as before.

The goal of this laser cutting is to remove black paint off the surface of a copper-clad fibreglass board without damaging the copper, and without leaving any significant paint behind.

There are three main parameters that determine how the laser acts on the workpiece:

Laser power: This is a percentage scale from 0% to 100% power. A higher power will cut more and faster, however may have a wider area of cutting which can be problematic if very thin, accurate lines are required.

Speed: How fast the laser moves, in millimetres per second. A slower laser will remain on a specific area of material for longer, so will cut more. A faster laser will simply brush over the material, so will make less of an impact. A slower speed will also make the cut take longer, which is best avoided.

Passes: How many times the laser is to go over the same material. Essentially, adding extra passes makes the same cut happen again on the same path and same material. Adding more passes has a similar effect to using a slower speed.

I want to minimise the time it takes for the mask to be lasered, so I will set the power as high as possible, as this has few consequences other than possibly a slightly less accurate cut. If this becomes a problem, I will turn down the power.

Pros

- Very repeatable. Unlike toner transfer, all steps are machine-controlled, meaning that tuning exactly the right temperature and pressure (which don't have to be manually applied) are not problems.
- Less work. Once set up, the laser must just be set up and told to go, unlike toner transfer, where a piece of paper must be cut out, taped down, and ironed, which may take several attempts.

Cons

- It is very slow. The laser is not very powerful, so the process of removing all the required material can take a while, on the order of hours.
- It requires specialised equipment, namely a laser cutter. This is not a problem for me as I am at school, but it means I must finish my boards before I leave or go on school holidays.

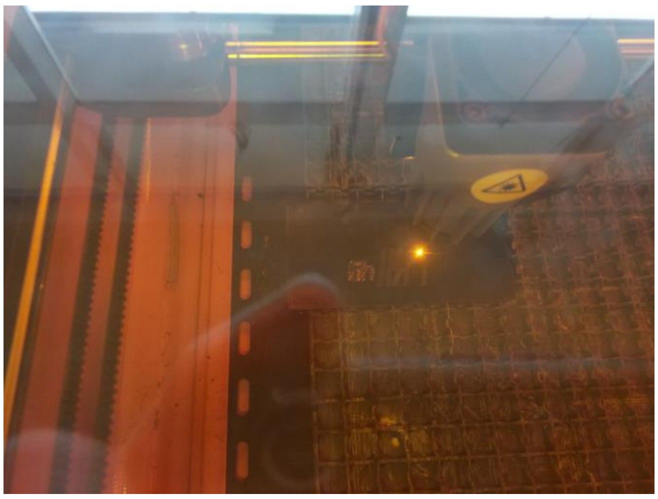
As laser cutters can be dangerous, I will keep my fingers clear of moving parts, keep the laser-proof lid closed when possible, and ensure the fume extractor is active when cutting.

The copper-clad blank is spraypainted black (the most absorbent colour, which should help to absorb the most heat energy from the laser for highest effect):

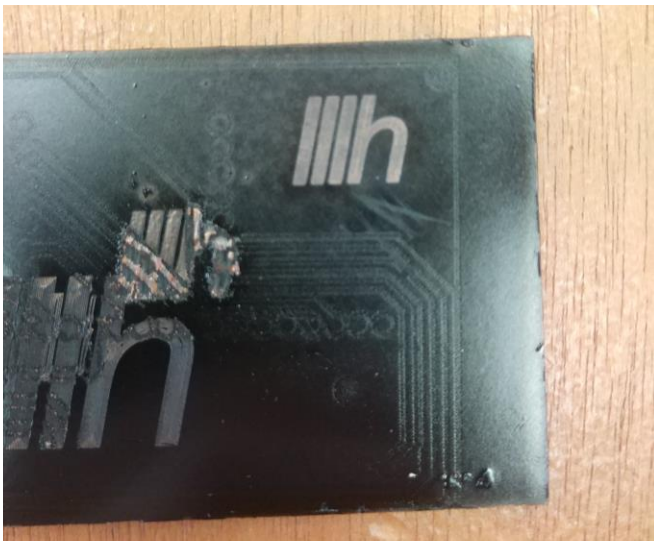
Like 3D printers, laser cutters require software running on a computer to prepare instructions describing how the machine must move. I used the software Lightburn, which is designed for the Emblazer range of laser cutters. I am using an Emblazer 2.

There are a large number of ways that movements can be described to the laser cutter. The simplest is to simply draw shapes inside Lightburn, such as circles, rectangles, and lines. For my test piece, I make the laser cutter draw some text.

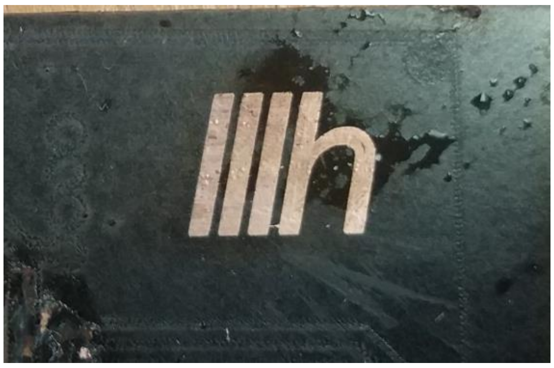
Trial and error with the laser now begins. This takes a few attempts but eventually I get some good settings.



Attempts with the test piece, which is reused from a failed 'real' attempt. Note that in some places, the fibreglass has been burned, meaning that the power is far too high, or speed is too slow. The figure in the top right is quite good, however there is still a thick black coating.



I found that the aforementioned black paint residue could be removed with a wet cloth, revealing copper.

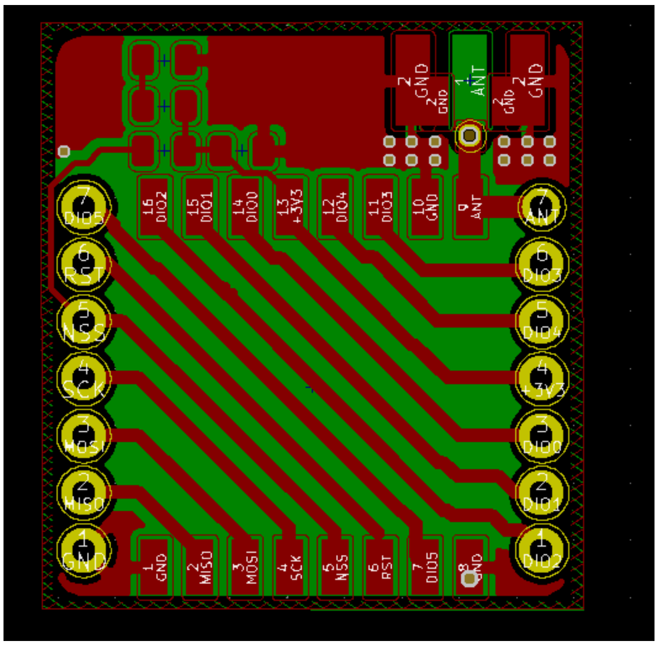


The settings that worked were 5mm/s speed, 100% power, 3 passes, air assist off.

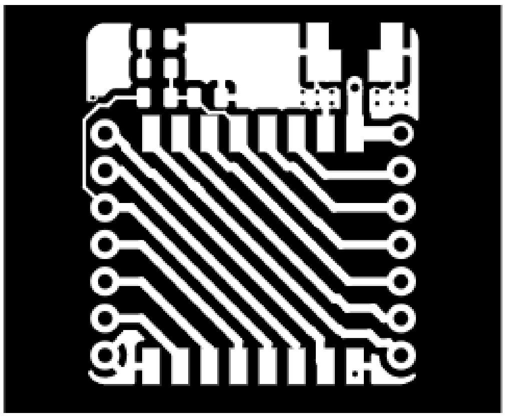
I now move on to making a real board, and etching it.

While it would theoretically be possible to create a circuit board by drawing shapes in Lightburn, it would be inordinately tedious, time-consuming, and error-prone. Instead, circuit boards are designed in specialised design software then exported to Lightburn. I use a free and open-source program called KiCAD to design circuit boards, and export these as SVGs, a form of infinitely-zoomable (without losing quality) image known as vector graphics. These SVGs are exported as a negative image, since the removal of paint translates into the removal of copper, which is the opposite of what may be considered 'normal' for KiCAD.

The following is the circuit board design for the RFM95 breakout board (9.1.1.8 Choosing a hardware module above) in KiCAD. Note that this is a double-sided board, so the green and red images are to be etched into opposite sides of the board. First, I will attempt to make this one-sided as the back side is not really necessary.



The design exported from KiCAD, in SVG format:



The laser will cut (i.e., remove paint) on the black areas, and leave the white alone.

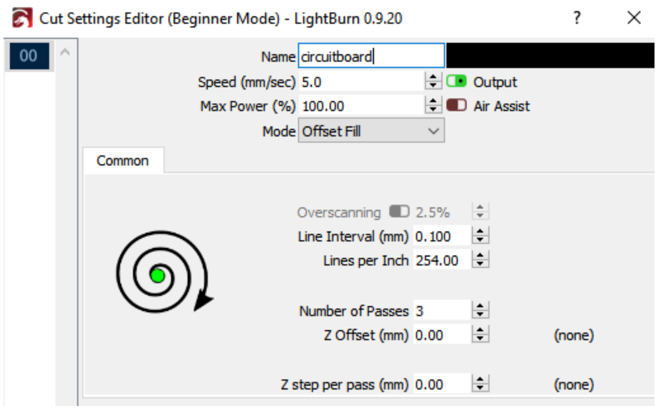
The toolpath can now be generated in Lightburn. There are two options for this: raster, and vector.

Raster is the simplest and most common way of laser-cutting an image. The laser simply scans back and forth horizontally along the image, changing the output power as required. Since this is a black-and-white image, the output power would oscillate between 100% where black, and 0% where white. This is quite inefficient, because a large portion of the image is white, so the laser is only wasting time by being there.

Instead, the vector process can be used. This is where the laser travels only along the path to be cut. This saves a lot of time compared to raster. This also results in smoother, less 'furry' lines which can result from vibrations and variations in raster cutting.

To vector cut, Lightburn must 'trace' the image. This is a process where it finds the edges, and determines what must be filled (black).

The settings found previously were applied. The mode 'offset fill' is Lightburn's name for a filled vector cut.



Cut Settings Editor (Beginner Mode) - LightBurn 0.9.20

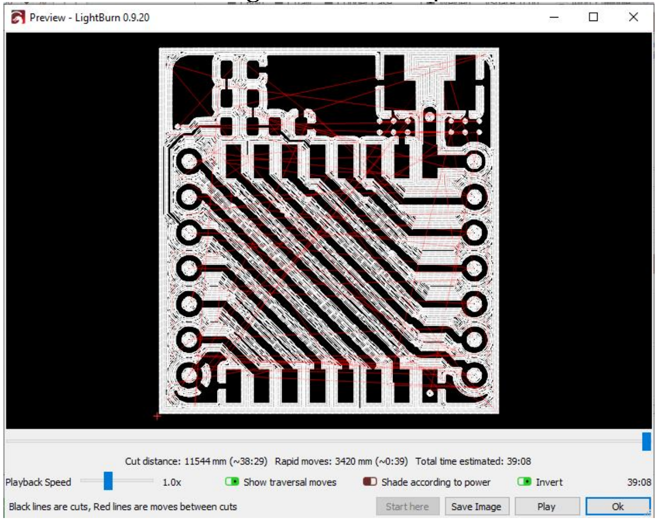
00

Name circuitboard
Speed (mm/sec) 5.0
Output
Max Power (%) 100.00
Air Assist
Mode Offset Fill

Common

Overscanning 2.5%
Line Interval (mm) 0.100
Lines per Inch 254.00
Number of Passes 3
Z Offset (mm) 0.00 (none)
Z step per pass (mm) 0.00 (none)

A preview of the path the laser will take is generated. The estimated cutting time is about 40 minutes.



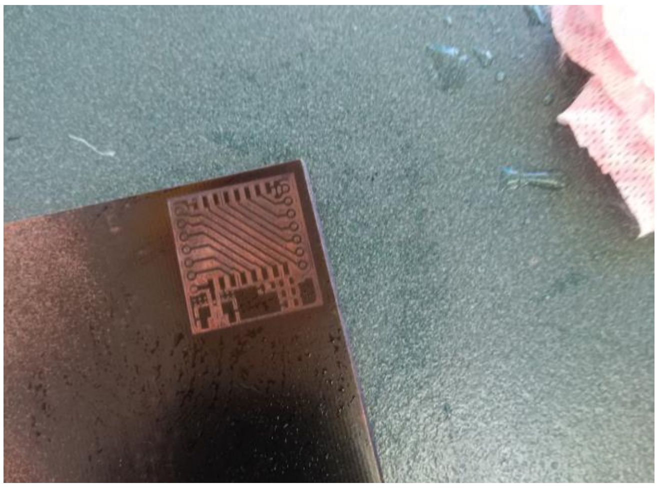
Preview - LightBurn 0.9.20

Cut distance: 11544 mm (~38:29) Rapid moves: 3420 mm (~0:39) Total time estimated: 39:08

Playback Speed 1.0x Show traversal moves Shade according to power Invert 39:08

Black lines are cuts, Red lines are moves between cuts

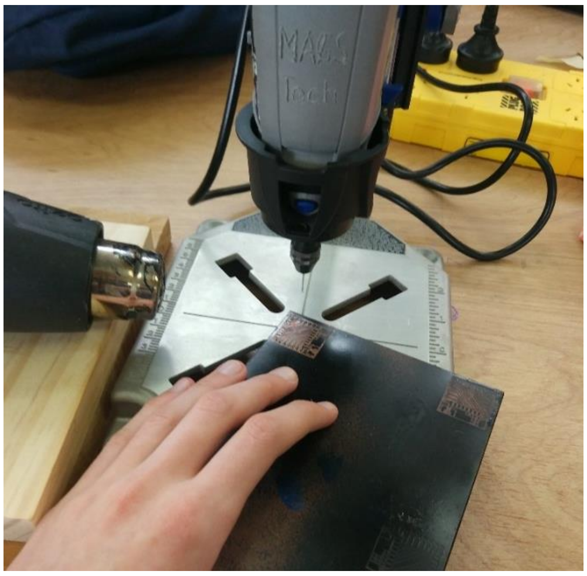
The PCB is prepared by cleaning and spraying it as before. It is then loaded into the cutter, and cut.



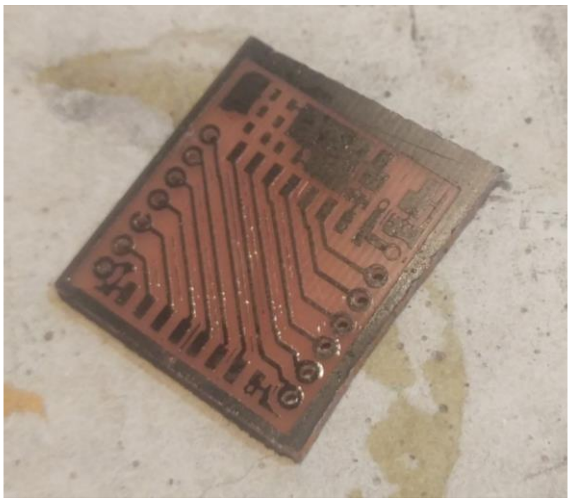
I now drill holes where the pins will go. I decided to do this before etching, as I was afraid the copper may delaminate from the fibreglass if I attempt to do this following etching.

Unfortunately, the chucks of most drills are too small for the 1mm drill I used. Luckily, the school had a tiny chuck for a Dremel, and a drill press type stand for it.

Safety glasses were used while drilling. A heat gun was used (set on very low heat) to blow dust away).

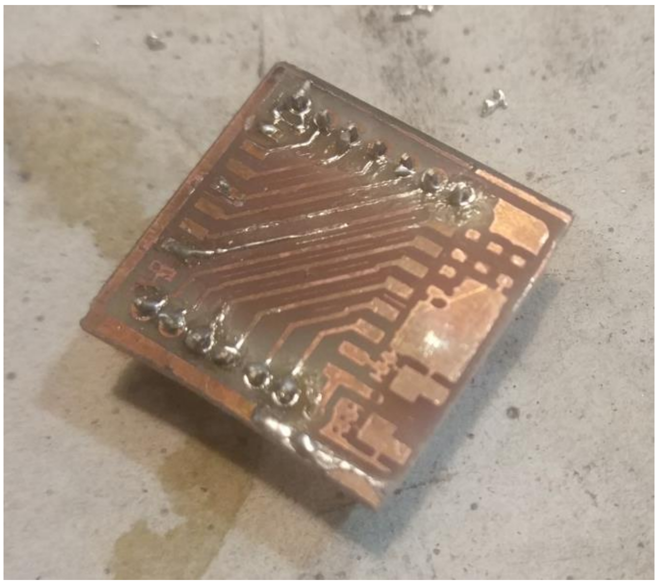


I now had to etch the board. There are numerous etching chemicals. I chose to use Copper (II) Chloride. I had used this previously, and still had some left over. It was somewhat ‘worn out’ from previous etchings, so it was regenerated with hydrogen peroxide. Gloves and glasses were used when dealing with these chemicals. Etching lasted around 1.5 hours until all the exposed copper was gone.



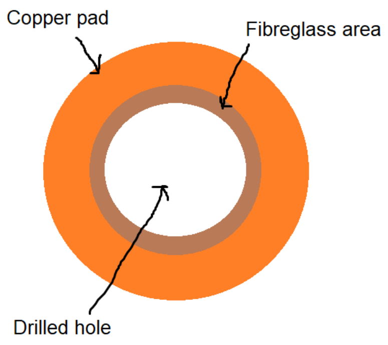
This etched board looked quite good to me. I noted that the tracks and pads were a bit thinner than expected.

I then washed the mask off with acetone, and began soldering pins onto the board. After washing, it became apparent that some of the tracks had worn away. I attempted to fix these by soldering a wire on. However, this wouldn’t work as the RFM95 module must sit where the wire is.



There were a couple of other problems. First, solder was reluctant to stick to the copper. This is unusual, and may indicate that there is a remaining chemical coating on the copper. This is possibly acetone, which tends to leave a thin oily film. Cleaning again with isopropanol didn't seem to make much of a difference, but perhaps I simply wasn’t cleaning hard enough or with the right solvent.

Second, there was a gap between the copper of the pad, and the hole.



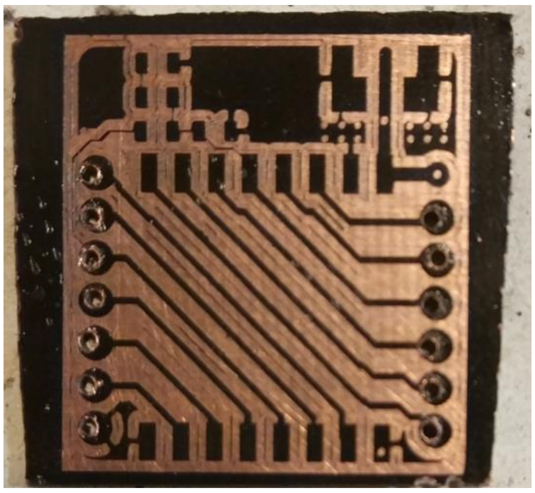
This is a problem because the flow of solder from the pin to pad was interrupted, making soldering difficult and joints very weak.

I can be sure that the copper *was* there before etching, because the mask existed right up to the drilled hole. This means that the etchant must have crept under the paint, around the pre-drilled holes. This is also probably the reason why the tracks are so thin – etchant has moved under the paint in some areas, etching the copper despite it still have paint on it. This is likely the result of my excessive etching times. Guides on the internet tend to suggest 15-30 minutes, whereas mine took more than an hour.

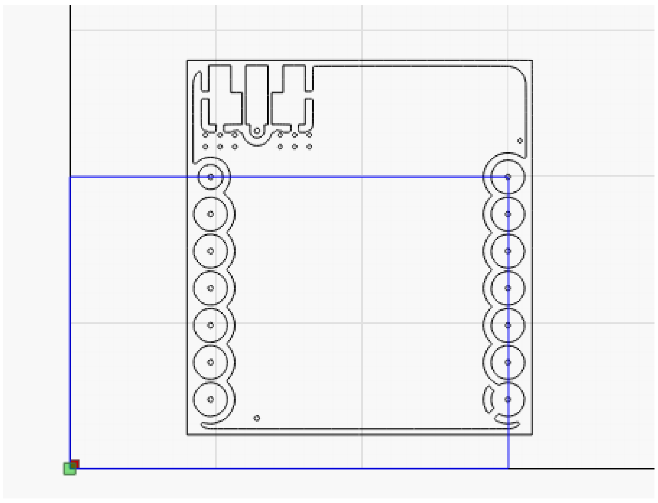
There are two ways this could be remedied. First, holes should be drilled following etching. This will make it harder for the etchant to get under the paint. Secondly, etching time should be reduced if at all possible. However, I need to etch until all the unmasked copper is removed. I believe this etching is taking a long time because there is still some paint residue on the board – like what I wiped off, yet smaller.

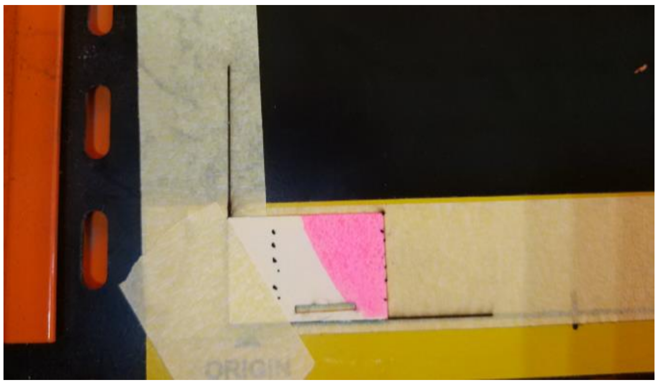
While experimenting, I tried several different laser power settings, to see if a different power would yield less paint residue. To my surprise, 5% power made an extremely sharp, clean cut through the paint, appearing to leave almost no residue. I suspect the higher power was causing the paint to form a different, stickier polymer. The lower power causes some other chemical or physical process to remove it without producing as much of this other polymer.

I also decided to attempt to make a double-sided board. The initial steps for this are the same as before. To produce double-sided boards, I must drill the holes prior to etching (despite deciding this is not ideal) in order to line up the two faces.

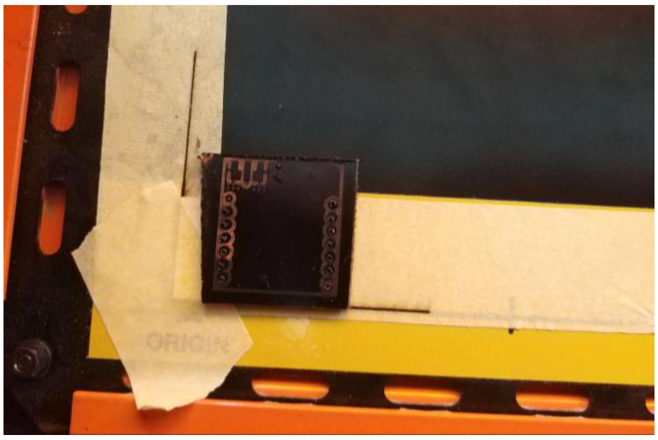


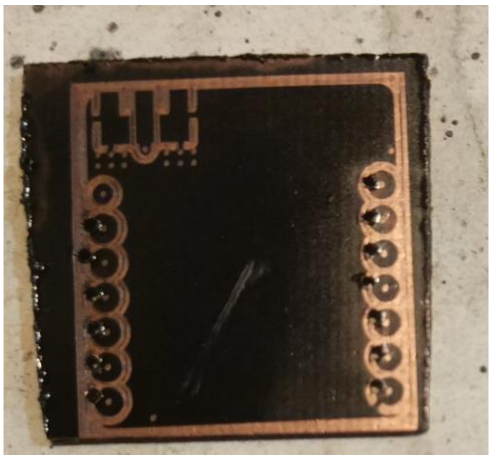
I then laser-cut a cardboard jig which used sewing pins to go through the holes in the board, precisely aligning the backside of the board. The jig is shown in blue below. The pins passed through two holes (one on each opposite corner).



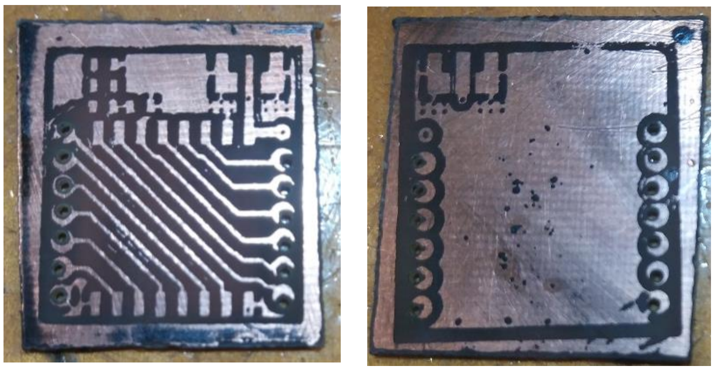


The board could then be aligned in the jig and lasered (with 5% power, as discussed).





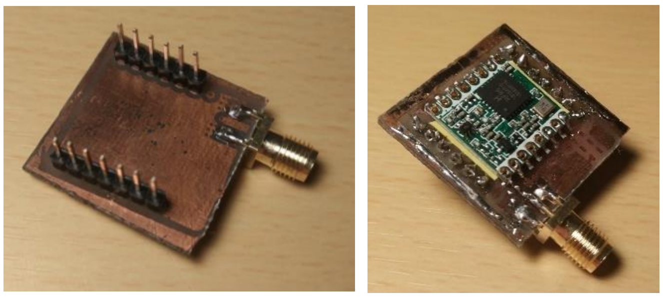
As you can see, much more fine detail is retained, and the board appears quite shiny. The proof will be in how well the etching goes.



This etching worked quite well: the traces have some girth to them, though there is a bit of a gap between the pin-holes and pad.

This is, however, sufficient to make a proof of concept.

Components (the RFM95W, SMA antenna connector, and pins) were soldered on:



13.4.3 Evaluation of PCB making

Creating circuit boards took far longer than I anticipated. Each step in the process of making them is relatively easy, however a successful board is contingent on all the steps working well, meaning there are many points for failure to occur requiring me to start over.

I now know the rough laser-cutter parameters required, so making more PCBs will be much quicker. There are still definitely things to improve – there is still a large gap between pin-holes and pads, and etching takes longer than it should. I will address these issues when they start causing problems – for now, the method is sufficient.

When my circuit board designs are perfected, I may choose to have them professionally made by overseas prototyping companies such as JLCPCB. This is not practical for untested designs, since it costs money and takes time (on the order of a few weeks) to have boards made and delivered.

13.5 Server software

As discussed in 9.6 Software: server above, I will be using the NodeJS framework to create a web API. This API has a number of routes, each corresponding to a different function within the API. For example, the /auth/login route allows the user to submit a login request.

13.5.1 List of required routes

Name	Function	URL	Request contents	Response contents
Login	Logs the user in	/auth/login	Username/email, password	Authentication token
Register	Allows a new user to register an account	/auth/register	Personal details	
Forgot password	Allows a user to reset their password if they forget it	/auth/forgotpwd	Email	
Get trap list	Get a list of traps in the vicinity. Depending on the user's 'level', they may be able to see more traps	/trap/traplist	Auth token, current location, filter options	List of traps
Reset trap	Allows a user to claim that they have reset a trap	/trap/resetclaim	Auth token, location, trap information	
Trap information	Fetch detailed information about a trap	/trap/trapdetails	Auth token, trap id	Trap information
Add new trap	Add a new trap to the system			
Get user profile	Fetch somebody's profile	/social/profile	Auth token, username or user id	User profile, error message
Set user profile	Allows a user to update their own profile	/social/profile	Auth token, new user profile	
Upload picture	Upload a picture, such as a post or profile picture	/social/picture	Auth token, image	Error message, image id
Get picture	Access a picture uploaded by someone by id	/social/picture	Auth token, picture id	Error message, image

Delete picture	Allows a user to delete a picture if they own it	/social/picture	Auth token, picture id	Error message
Make post	Allows user to share a catch with an image	/social/post	Auth token, post	Error message, post id
Get post	Gets a post by id	/social/ post	Auth token, post id	Error message, post
Delete post	Deletes a post if the user owns it	/social/post	Auth token, post id	Error message
Get feed	Get a list of post ids made by friends. Similar to instagram feed	/social/feed	Auth token, feed page number?	Error message, list of post ids
Trap information	Load information from a trap when it makes a LoRaWAN broadcast	/trapb/update	Trap information, secret key	Error message

This is obviously a lot of work. However, the basic functionality can be written first, and the non-critical parts like social media added later. I will start with authentication since this is required for all other features. I will then move on to trap information and trap backend, groups, then social.

13.5.2 Models

As discussed in 9.6 Software: server above, models represent data stored in the database. Models are good on their own, but are especially helpful when linked together. For example, there exists the User and Profile models. Every user has exactly one profile. This allows us to easily link distinct pieces of data together. More complex model associations exist, such as "many to many". For example, a trap may belong to many groups. A group may have many traps. Therefore, there exists a many-to-many relationships between groups and traps. The following diagram describes all relationships using crows-foot notation.

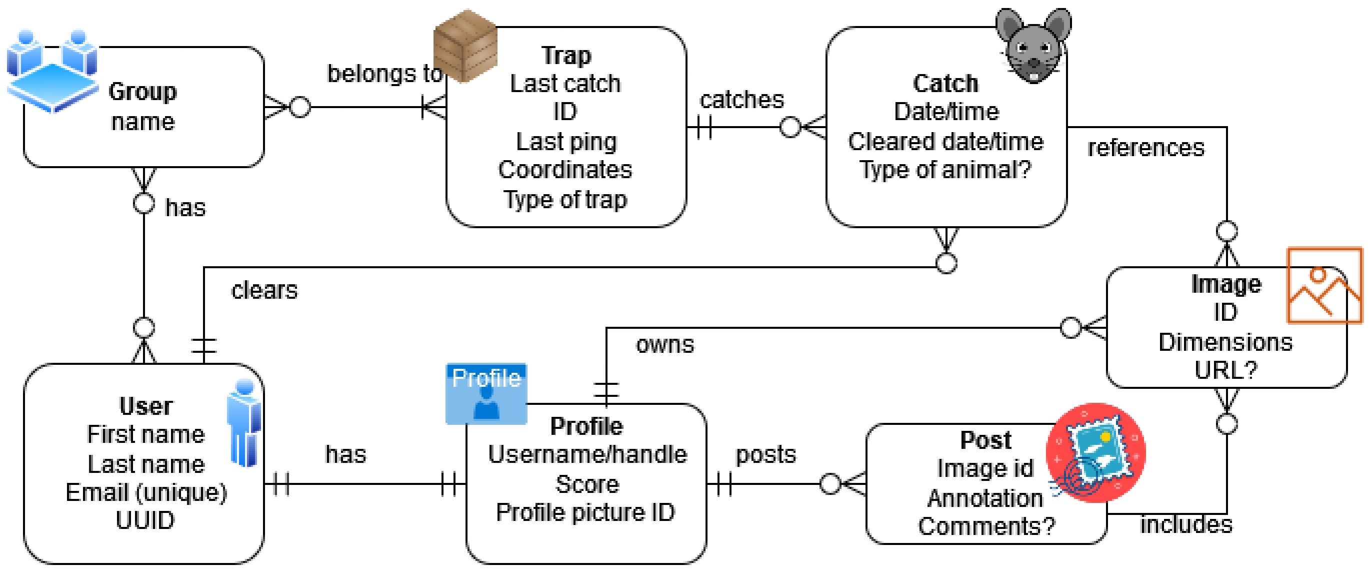


Figure 48. All the models, their contents, and the connections between them.

I will implement my models in roughly this order, to flesh out the basic functionality then add fancy features:

1. User
2. Profile, partially (to figure out how basic one-to-one associations are implemented)
3. Group (an important junction node, and a slightly more advanced association to learn)
4. Trap
5. Catch (at this point, the system will be functional, albeit quite basic)
6. Image
7. Finish profile
8. Post

14 Stakeholder Feedback, Round 1

Now that I had a pretty good idea about the design of my system, I showed my design to my stakeholders.

14.1 Predator Free 2050 ([redacted])

I had already talked to [redacted] in 10 Stakeholders above, where she provided me with traps to design around. I emailed her copies of my development drawings and a render showing what the node would look like attached to a trap.

[redacted] was quite positive about my design, though had two main points about my design.

If the antennae are not removable, as I had until this point intended, nodes would be extremely bulky (and fragile) when in transport. This would seriously encumber their deployment, as they would have to be stacked to be transported, which would not be possible if they had antennae permanently mounted. This problem necessitates a slight design change.

Also, the weight of tools required for installation may cause challenges, as many traps can only be accessed on foot. I don't think this is really a problem, since installation only needs to be carried out once, and bringing tools to traps is still far more practical than bringing all deployed traps to a workshop. Of course, newly deployed traps can have nodes fitted in a workshop prior to deployment. Plenty of trap maintainers already carry electric drills to make opening traps easier. The only tools required would be screwdrivers, and maybe a drill.

I replied to her, discussing my proposed changes for the first problem (15 Development Drawings, Stage 2 below), and explaining my understanding of the tool situation. She confirmed that trap maintainers do often carry drills. She also raised the point of the weight of the nodes. This is valid, since they are partially made from steel and contain relatively heavy batteries. However, I don’t think the weight would be significant compared to heavy wooden and steel traps.

14.2 Prof. Rachel Fewster

Upon learning of my project, my calculus teacher suggested that I meet with his former lecturer, Professor [redacted] at the University of Auckland. She coordinates the CatchIT project, one of the largest and most comprehensive platforms for trap data entry, storage, and analysis. [redacted] mostly commented on the software component, as this is her specialty. She describes trap software platforms as either “inputs” or “outputs”. Inputs take real-world trap data (either from manual data entry or an automatic system) and put it in a database. Outputs display and analyse this data.

[redacted] sees no problem with a variety of inputs, as each may have its separate use, but identifies a problem with fragmented outputs. This fragmentation leads to several (potentially competing) groups trying to collect data, and increases the onus on all input developers to cater to many outputs.

She believes that the urban trapping is already fairly well done. There is no shortage of volunteers to reset and log urban traps because of the large population nearby. Urban areas also make up a small proportion of New Zealand's land area, and an even smaller proportion of land area where wildlife is likely to reside.

Instead, she sees my project having a much more significant effect in the back country. Despite their flashy branding, existing systems like Econode and Celium do not appear to be widely used, and do not appear to be integrated with systems like CatchIT and TrapNZ, resulting in even more fragmented outputs. She agrees that this may be due to their price, an aspect which I can conceivably improve upon.

The public-access aspect of my system could be much more useful in the bush. Many traps can only be accessed via a long bush walk, meaning that they are checked and serviced infrequently. There is also preliminary evidence that suggests certain traps 'run hot' for a period of time after a catch. Unfortunately, if they are not checked for several weeks after a catch, then only the first visiting animal is caught. If, on the other hand, passing recreational trampers are able to reset a trap, it may be back in service in a matter of days.

Trapping groups usually do not want other people (the public) resetting their traps because they lose the valuable data, and these people may not be well-informed on how to reset the traps. My system would likely mitigate both of these issues - members will be informed on how to reset traps, and data would be collected even before someone turns up to reset it.

Both Dr [redacted] and I often look in traps when we stumble upon them in the outdoors. She admits that she would likely plan her weekly bushwalks to pass a trap that needs resetting. A 2017 Mountain Safety Council report found that more than 900,000 Kiwis tramped that year. If even a tiny subset of these people reset the traps they came across, many more pests may be eliminated.

15 Development Drawings, Stage 2

15.1 Findings and changes from client feedback

Findings

- Antennae must be removable, yet still be waterproof when connected.
- Weight should be minimised where possible.

- Tool requirements should be minimised.
- I should avoid conglomerating data in my system except for the essentials. It should be available for easy, automated export.
- Public traps may be most useful in the backcountry rather than urban areas.

Solutions

The main new problem is the question of how to make the antenna removable. To achieve this, I will use a cable gland, like I discussed in 9.1.6.2.3 Glands above.

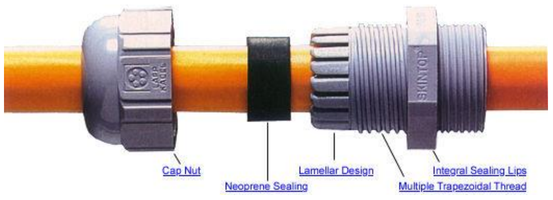


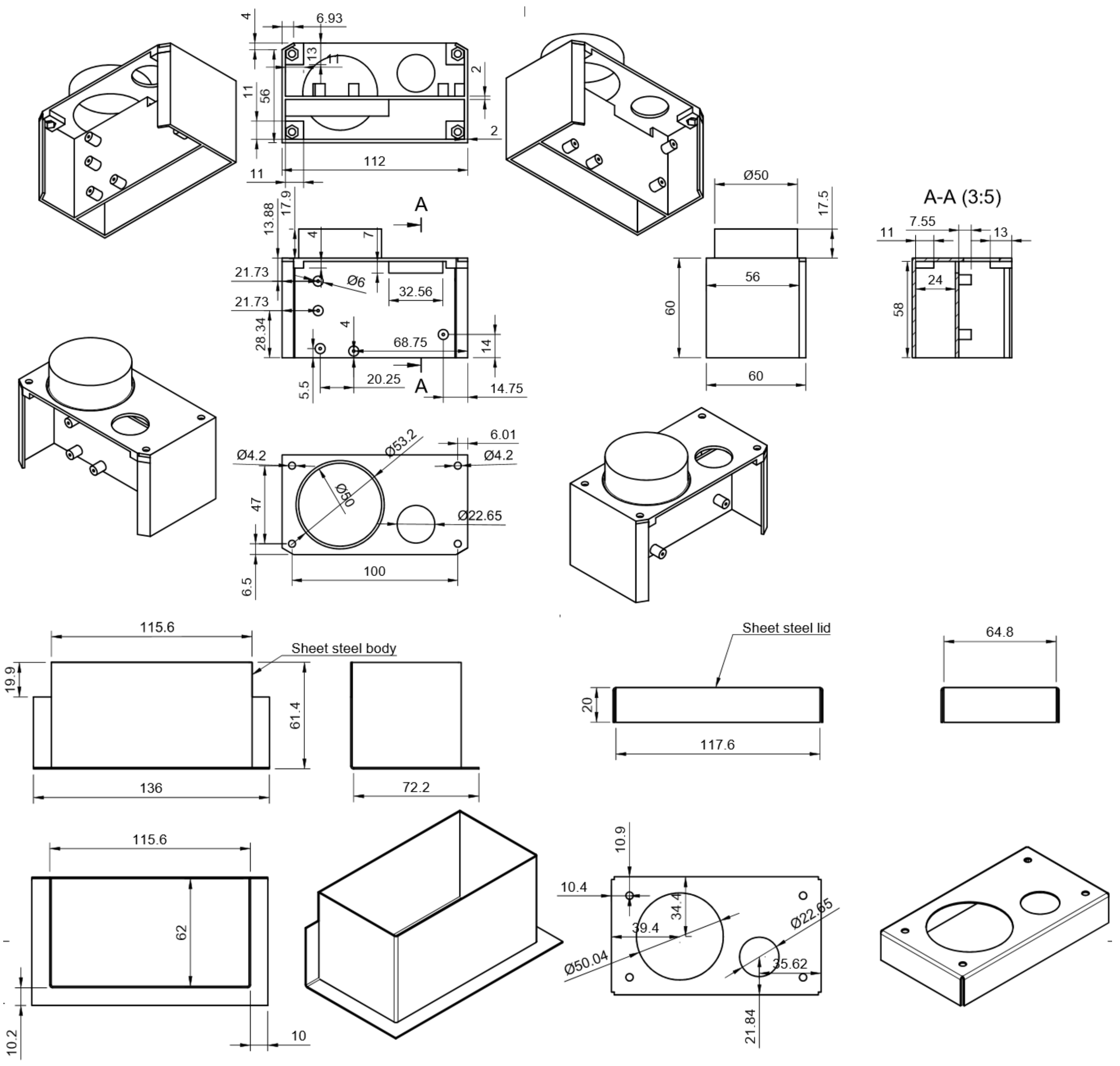
Figure 49. A cable gland exploded view. The antenna is held with a waterproof seal with the black ring.

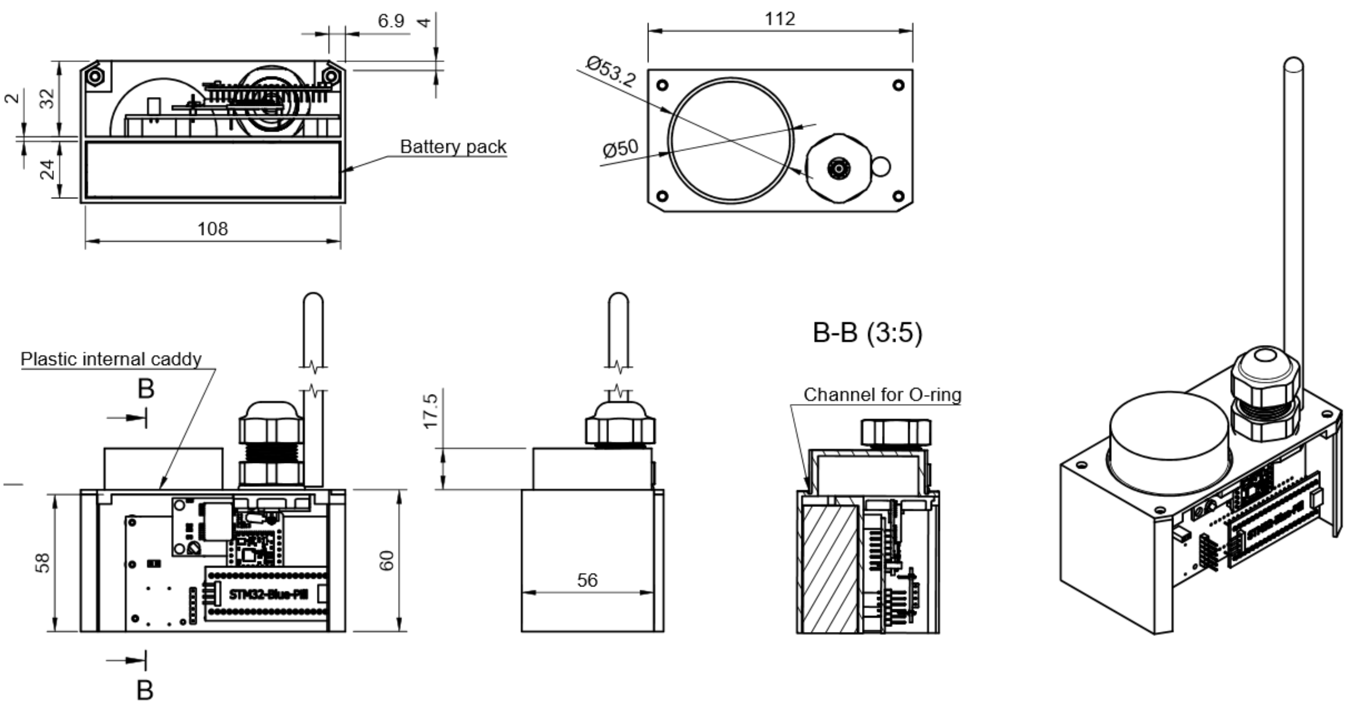
15.2 Other changes

Modelling the sheet steel caddy and other parts showed some flaws with the design. These will be fixed in the following ways.

- Change the shape of the exterior metal to allow for mounting to the side of an object.
- Reduce weight where possible.
- Use non-galvanised steel and paint it.
- Increase tolerances in CAD, especially bend radius of sheet steel parts.
- Change the shape of the plastic caddy to allow for weld beads.
- The "flaps" resulting from cutting the lid corners will be kept and used to aid welding.

15.3 Drawings



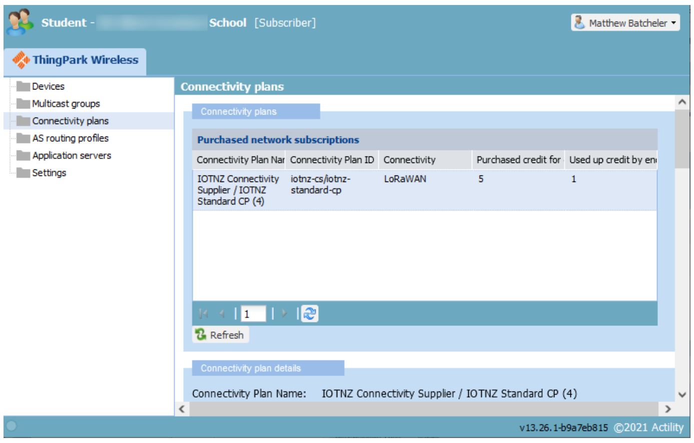


16 Spark's LoRaWAN Network

To develop and test my system, I needed access to a LoRaWAN network. I could purchase my own gateway and establish a private network, however accessing Spark's existing network would offer me more reliable service over a wider range, and experience in integrating my system with commercial LoRaWAN backends like Spark's.

I began by contacting their IOT Support email address and explained my situation as a technology student. I was advised to complete a "Connected IOT Low Power Trial Application", which I did.

A few days later, I received an email welcoming me into their IOT programme. I could access their Spark backend (called "ThingPark"), and provision up to 5 devices on the network.



Purchased network subscriptions

Connectivity Plan Nar	Connectivity Plan ID	Connectivity	Purchased credit for	Used up credit by en
IOTNZ Connectivity Supplier / IOTNZ Standard CP (4)	iotnz-cs/iotnz-standard-cp	LoRaWAN	5	1

Connectivity Plan Name: IOTNZ Connectivity Supplier / IOTNZ Standard CP (4)

Figure 50. The ThingPark wireless backend interface, where I could provision and configure LoRaWAN devices.

17 Working Proof of Concept

17.1 Goal

After creating a working (though not perfect) RFM95 breakout board and getting access to the Spark network, I decided to make a proof of concept. This goal of this would be to transmit a message over the network when a button is pressed.

17.2 Hardware

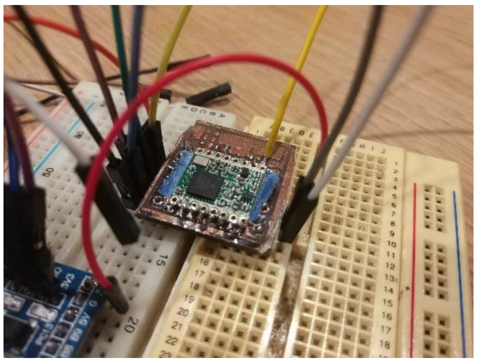


Figure 51. The RFM95 module in the breadboard.

A "breadboard" was used to connect the breakout board to my STM32 microcontroller, as per the circuit designed in 13.4.1 Circuit design above.

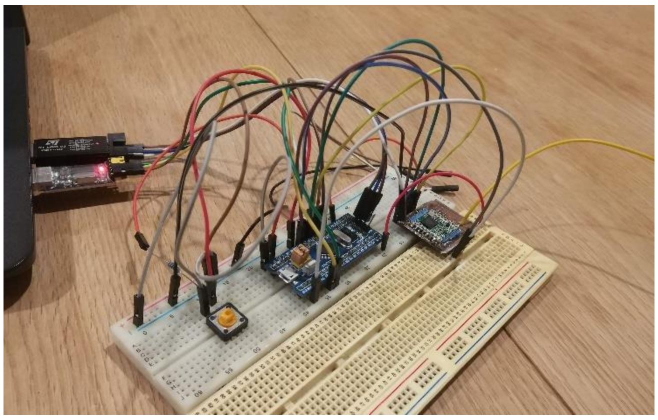


Figure 52. The proof-of-concept hardware. It is connected to a computer for programming and testing. All of these wires will later be condensed into a circuit board.

I will not connect a GPS, voltage regulator, or battery monitoring circuit as these are unnecessary for the proof of concept and would make the breadboard wiring even more messy.

17.3 Node firmware

I now needed software to run for the STM32 to control the radio. These are several libraries for this purpose. I chose the fairly low-level, open-source arduino-lmic library, maintained by the company MCCI.

There are higher-level options available, which would make implementing them easier, however there are three main disadvantages to these:

- Lack of control. This may mean I cannot put the radio into the exact low-power state I want, or any combination of other issues.
- Inability to transmit plain LoRa (not LoRaWAN) messages from device to device. This would be essential if I were to implement mesh networking, as discussed in 9.1.1.2 To Mesh, or not to Mesh above. Therefore, I would rather face some initial difficulty learning the library rather than have to completely replace the library and rewrite my LoRa code at some later date when I implement mesh.
- In a similar vein, extensibility. I would rather have more features at a higher initial cost, even if I don't initially use them, because it will save a lot of headaches later if I have to change library to accommodate them.

Therefore, I will use this library.

For my early tests of the radio, I simply use their demonstration program, which transmits the same packet ("Hello, world!") every 60 seconds. This basic demonstration serves two purposes:

- Verifies that the radio is connected to the STM32 correctly and is indeed transmitting.
- Verifies that the transmitted LoRaWAN packet is received and understood by Spark.

17.4 Testing and debugging

I therefore put my device's identifying numbers into both the Spark system and my device, then uploaded the program to the STM32. Unfortunately, the device consistently reported the following through its debug output:

```
Starting
Packet queued
2700: EV_JOINING
239390: EV_TXSTART
641225: EV_JOIN_TXCOMPLETE: no
JoinAccept
```

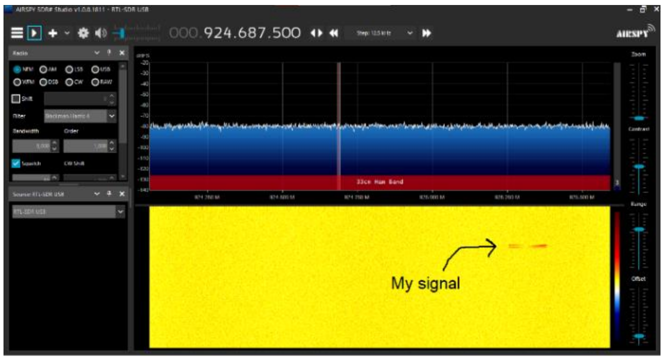
Breaking this down:

1. Starting: the device has turned on.
2. Packet queued: a packet has been queued to be transmitted.
3. The numbers at the start of each following line are a timestamp.
4. EV_JOINING: the radio is attempting to join the LoRaWAN networking with the OTAA authentication method. See 9.4.4 LoRaWAN flow above.
5. EV_TXSTART: The radio is beginning its transmission.
6. EV_JOIN_TXCOMPLETE: no JoinAccept: The transmission has ended, yet no reply has been received from Spark's end. This is obviously not ideal.

There are several possible explanations for this failure:

- The radio is not actually transmitting.
- The numbers I have input to the program are incorrect, or somehow mismatched to Spark's end.
- The radio signal is too weak.

The easiest thing to verify is that the radio is transmitting. I used a device called a Software Defined Radio (SDR), which allows me to 'listen in' to a wide range of frequencies. I can therefore connect the radio to the computer, tune to the correct frequency, make the device send, and see if the SDR receives anything.



Luckily, it appears to transmit. This rules out one of my failure modes. The other two are hard to differentiate, however I hypothesise that my makeshift antenna is not sufficient for the signal to penetrate indoors, which is not really what LoRaWAN was designed for anyway. I therefore go outside, and successfully receive a signal.

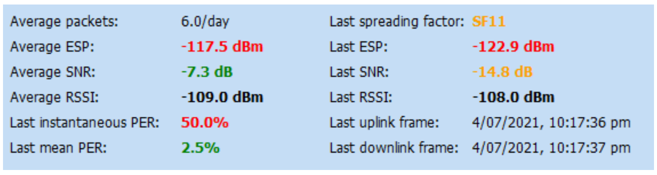
The following now appeared on the STM32 output:

```
1       Starting
2       Packet queued
3       2436: EV_JOINING
4       236574: EV_TXSTART
5       601184: EV_JOINED
6       netid: 6291470
7       devaddr: E01C14XX
8       AppSKey: 14-93-17-22-6E-62-18-0E-
        4C-9B-31-XX-XX-XX-XX-XX
9       NwkSKey: 7B-B6-7B-84-56-C9-06-AC-
        5A-D8-BE-XX-XX-XX-XX-XX
10      608844: EV_TXCOMPLETE (includes
        waiting for RX windows)
```

Breaking this down from line 5 (lines 1-4 are the same as before):

- EV_JOINED: The join request has been accepted, so the node is now part of Spark's network.
- Lines 6-9 show various encryption keys and parameters that have been determined by the OTAA join process. An understanding of what these do can be attained from the link to the LoRaWAN explanation.
- EV_TXCOMPLETE (includes waiting for RX windows): The transmission is complete. Unlike before, it has succeeded.

Spark's online LoRaWAN management software shows the following information:



Average packets:	6.0/day	Last spreading factor:	SF11
Average ESP:	-117.5 dBm	Last ESP:	-122.9 dBm
Average SNR:	-7.3 dB	Last SNR:	-14.8 dB
Average RSSI:	-109.0 dBm	Last RSSI:	-108.0 dBm
Last instantaneous PER:	50.0%	Last uplink frame:	4/07/2021, 10:17:36 pm
Last mean PER:	2.5%	Last downlink frame:	4/07/2021, 10:17:37 pm

The signal received by Spark is rather weak, though this is probably because of the lacklustre aerial. This also explains why it did not work indoors – the building blocked the already weak signal.

I then programmed the device to respond to a button press, triggering it to send a packet.

17.5 Video demonstration

A video demonstration of this system working is available [redacted]

To summarise the video – the time when the last packet was received was noted. I pressed the button (which models the detector switch inside the trap). The messages on screen show the node's output when transmitting, as above. After reloading the Spark page, the time of last receipt has changed.

Despite having a very poor antenna, the signal still travelled 871 metres (to the tower indicated by the Spark software) in non-line-of-sight conditions. This is promising for the future, as much stronger antennae will be used, and transmission conditions will likely be better.

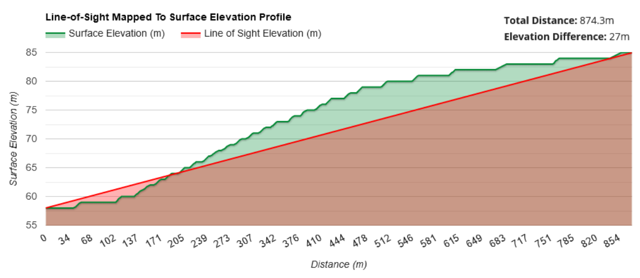


Figure 53. The signal is travelling in non-line-of-sight conditions, and with a poor antenna. Despite this, it works.

18 Full Prototype

It is now time to make a functional prototype. As before, there are three aspects to this: node hardware, server software, and the app.

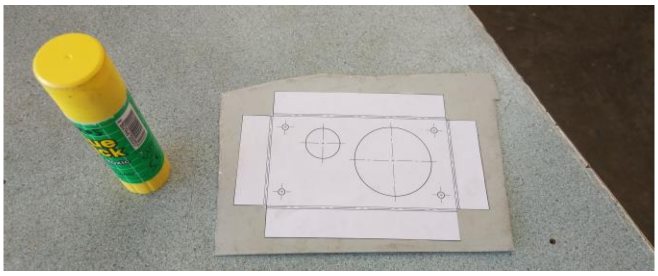
18.1 Node hardware

This consists of three components: the sheet steel outer, the 3D printed plastic caddy, and the electronics that inhabit the caddy.

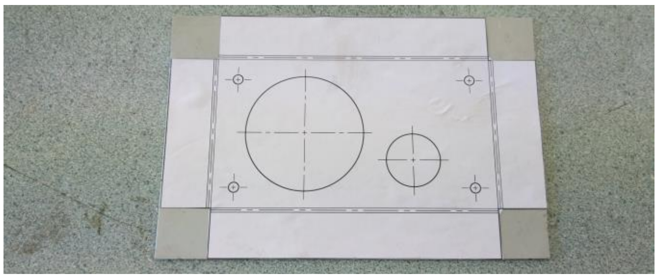
18.1.1 Sheet steel housing

First, I will create an updated housing as per 15 Development Drawings, Stage 2 above.

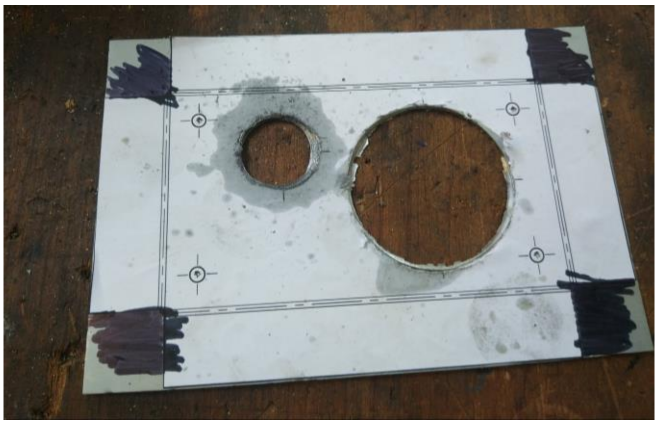
I decided to start with the lid, as this could be tested with the old box. Scale plans were printed onto normal A4 paper, then glued to the sheet steel.



This piece of metal was cut and drilled to the correct size using a bench shear and tin snips.

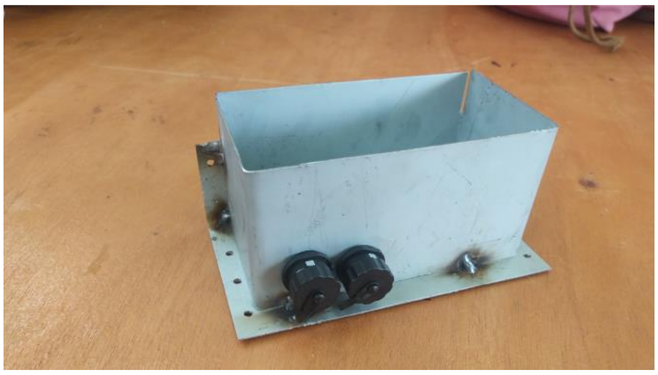


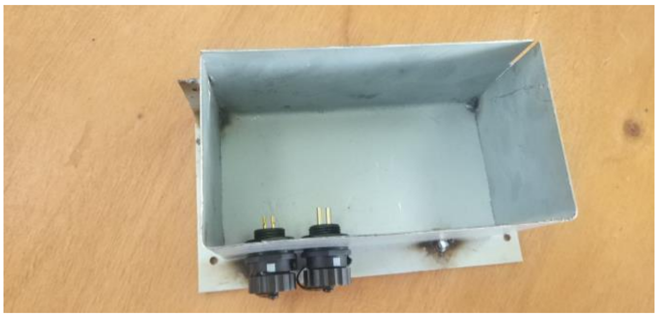
Hole-saws were used to cut the holes for the gland and GPS antenna 'tower'.



The lid was then folded into shape.

The same process was now repeated for the base box. This consists of two folded sheet metal pieces, which are welded together. I will not show this whole process, for brevity, as it is the same as before.

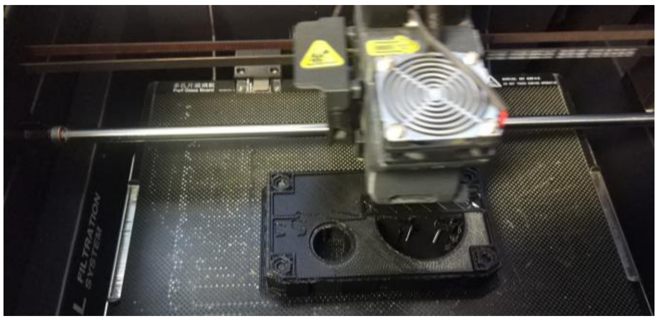


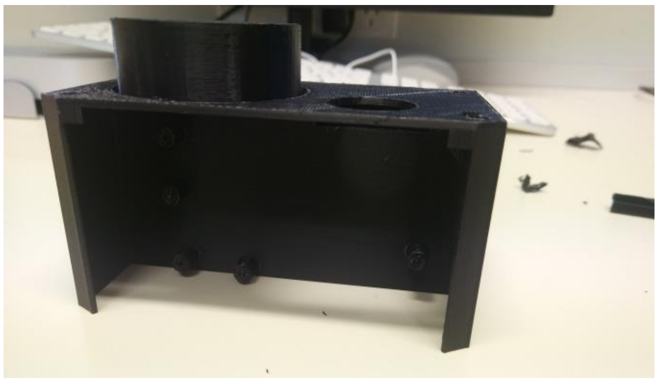


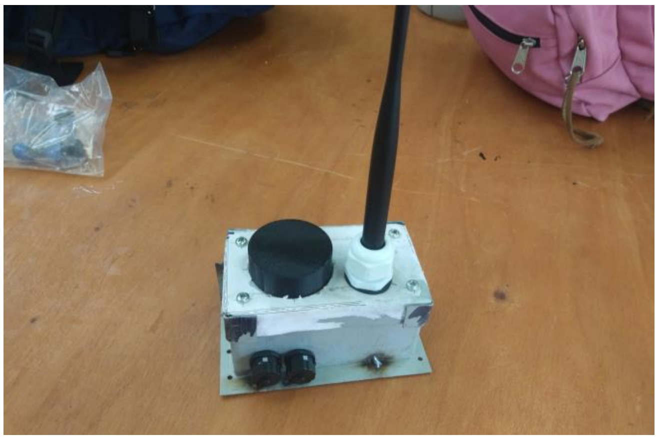
The box was only tack welded together – this is obviously not waterproof, and I will complete welding before deploying it in a wet area.

18.1.2 Caddy

When I redesigned the sheet steel housing, I also updated the caddy to reflect the necessary changes as determined by my client feedback. This caddy was 3D printed.



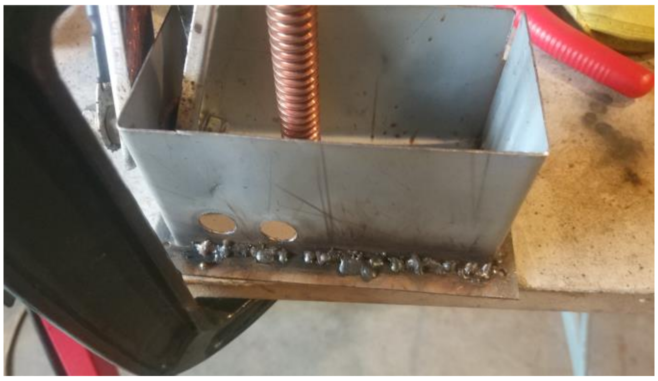




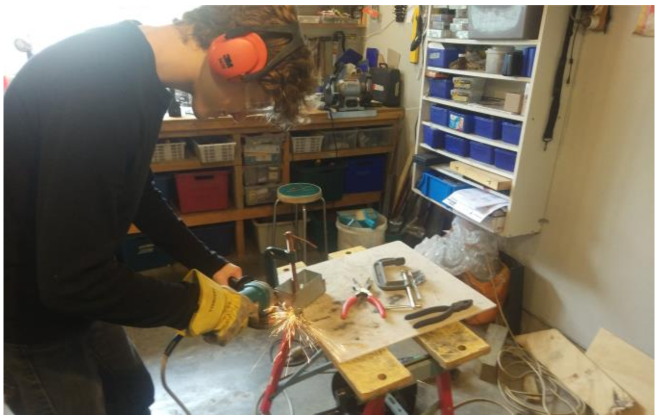
The housing and caddy were now almost completed. As noted, the hardware must be waterproofed.

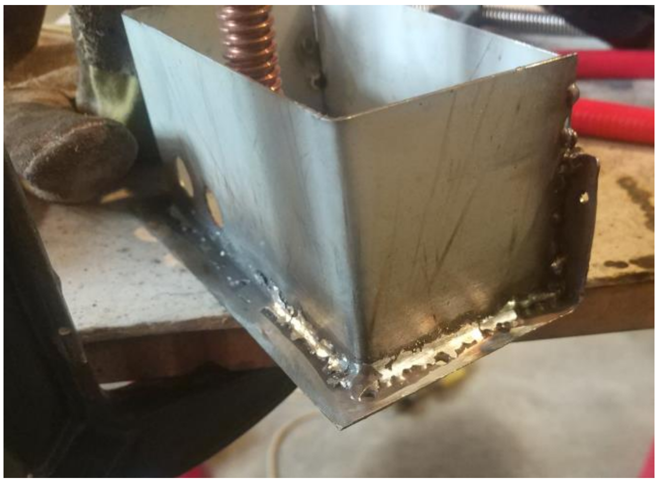
18.1.3 Waterproofing

I need to waterproof the welds and other components. Welding continuous beads onto thin sheet steel without melting holes in it is difficult. Because of this, I will weld in small pulses, limiting heat build-up.



These welds were ground down, removing excess weld. I then went over all the spots I had missed again.

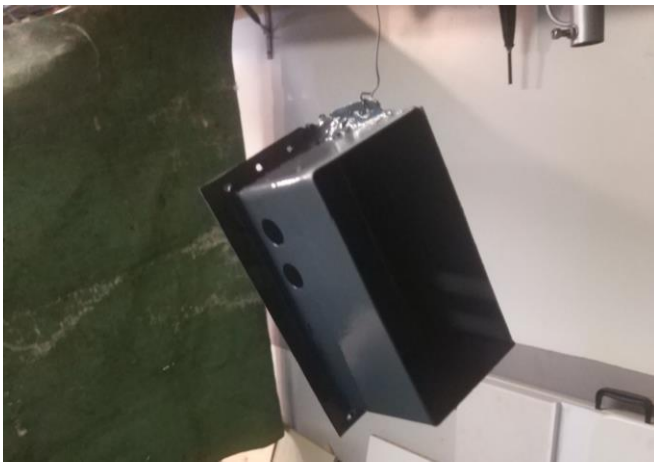




After a few repetitions, I was happy with the welds and tested the waterproofing.

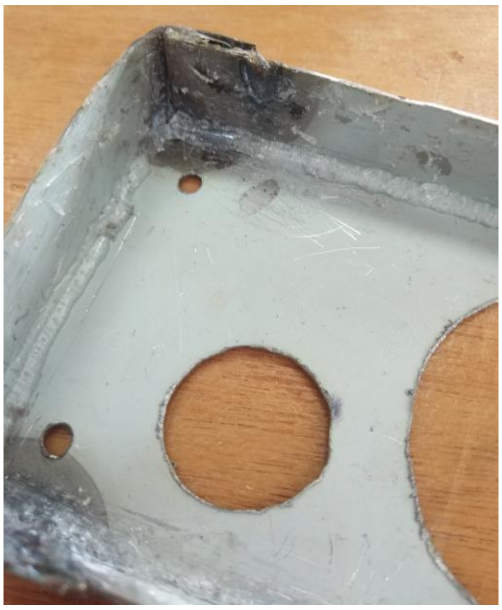
It appeared to be waterproof. In reality, it would never be submerged like this, though it is good to check.

I then painted the box. This protects the exposed metal (especially the areas I ground) from rust, and makes the box slightly better looking.



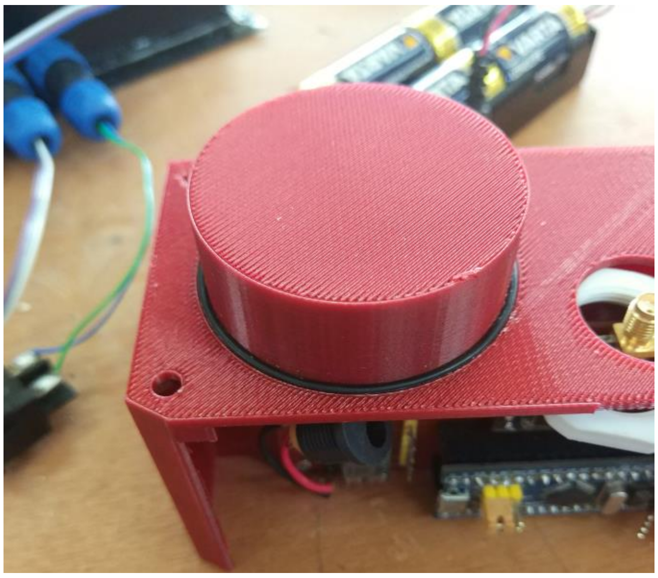
The lid is one of the most critical components of the node, in terms of waterproofing. This is because it will bear the brunt of rain and weather.

To seal the lid to the base box, a thin bead of silicone sealant is run around the perimeter of the lid. This presses down on the rim of the box, forming a watertight seal.



To seal around the GPS antenna tower's hole, an O-ring is used.

This is compressed between the plastic caddy and metal lid, preventing moisture ingress.



The gland through which the antenna passes is waterproof by design. However, up to this point, the antenna had an indent which allowed it to bend. This indent would let water pass through the gland. To remedy this, the indent was filled with araldite epoxy, then covered with heatshrink.

18.1.4 Electronics

I spent a great deal of time at the functional modelling stage developing a method to create circuit boards. A summary of my method up to this point follows:

- Clean and spraypaint the blank copper-clad board.
- Laser-cut the negative of the desired board.
- Wipe down the board with a wet rag.
- Place the board in an etchant solution - either ferric chloride or copper chloride. This dissolves away the exposed copper, but not the painted areas.
- When this is finished (several hours), remove the board, clean it, and remove the paint with acetone.
- Drill holes.

Unfortunately, the extremely long etching duration led to masked tracks being eaten away, since etchant would creep under the paint. This was sufficient for a proof-of-concept board, but would need to be improved for a full prototype.

Further research suggested that etchant solution could be applied with a sponge. The extra friction provided by this would aid the removal of copper, meaning that the etching would take much less time. This prevents removal of paint, increasing the quality of the board.

This worked very well, bringing etching time down to around 20 minutes. The previous issue of having a gap between the pad and pin-hole was solved due to drilling after etching, and the much-reduced etching time.

I now had to produce a new mainboard and RFM95 breakout for my prototype. Both boards were made in parallel. First, I cleaned the copper-clad board to allow the most efficient etching.

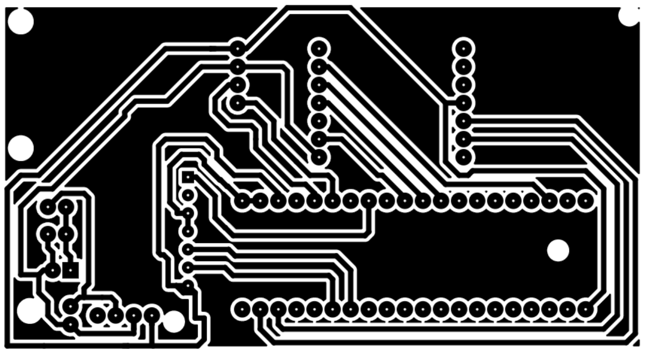
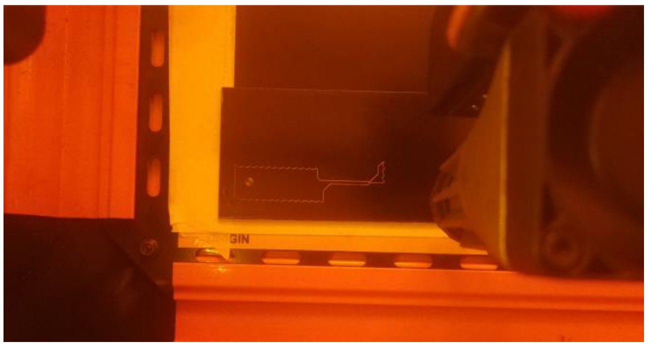
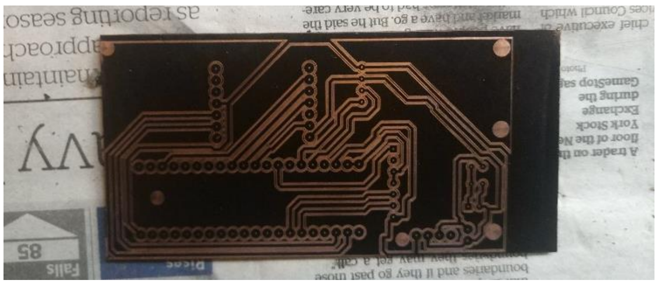


Figure 54. The "mainboard" design to be etched. The RFM95 breakout board design remains the same, designed by attexx.

The board was spraypainted and laser-cut, as per 13.4.2.4 Laser cutter masking above.

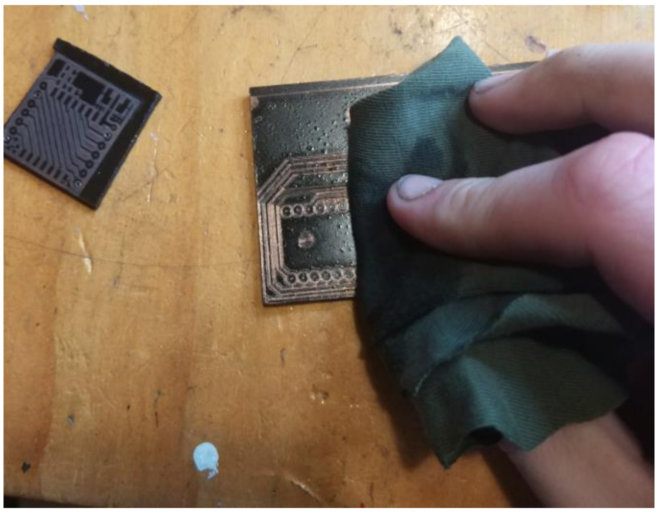


Note the grey residue covering the copper tracks. This is chemically reacted paint, which can be removed by wiping with methylated spirits or water.

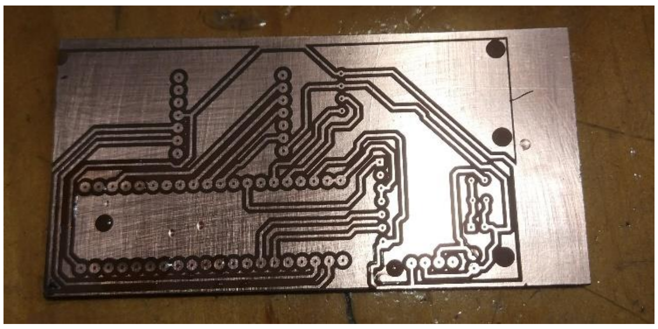


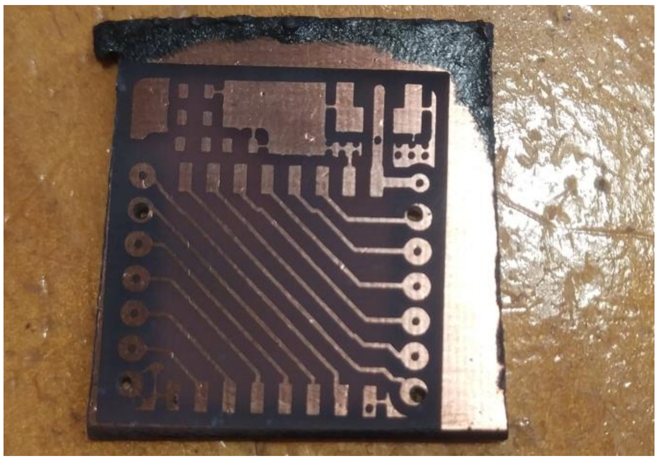
The board must now be etched. The aforementioned sponging technique was applied with great success.

The mask was now removed with acetone.



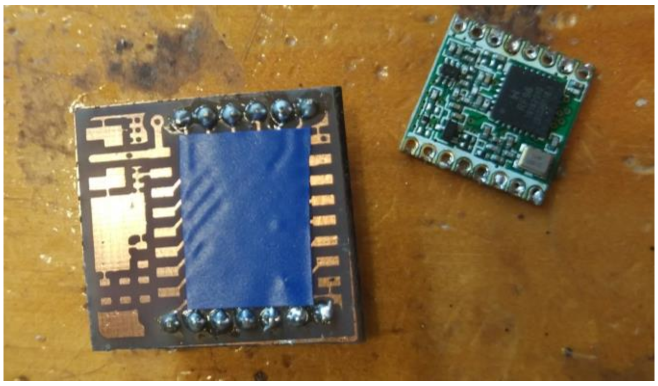
I am very happy with the quality of this board. The straight tracks have sharp edges, and are not eaten away like before. The holes have plenty of copper to drill into, meaning I won't have trouble with gaps between the pin and pad.



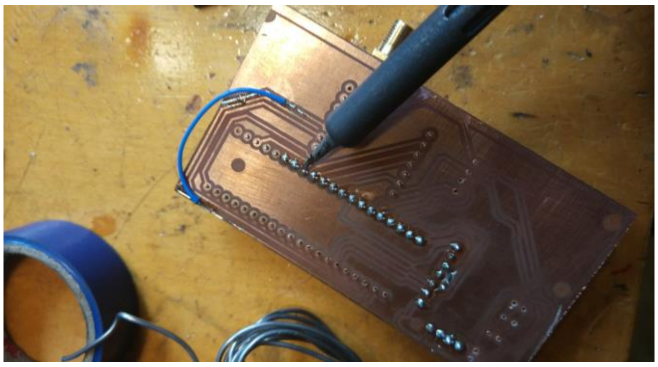


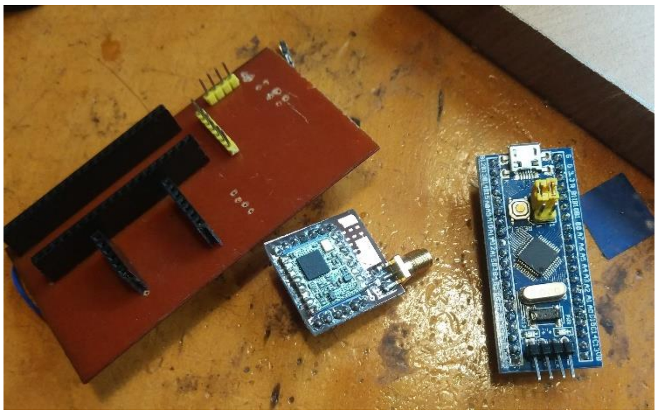
When the PCBs were made, it was time to attach components. At this prototyping stage, I will not attach the GPS as this is not essential for the device to function normally, and getting it to enter and stay in the optimal, low-power mode requires a lot more research.

The radio module was attached with a piece of tape to prevent shorting on the bottom of the board.

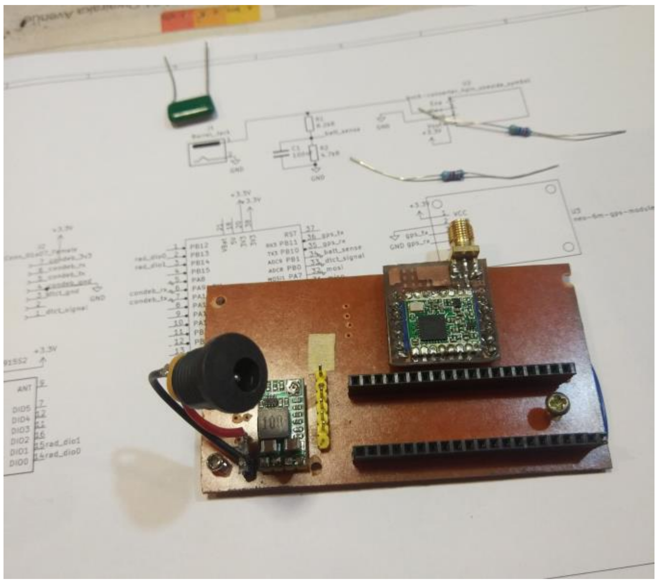


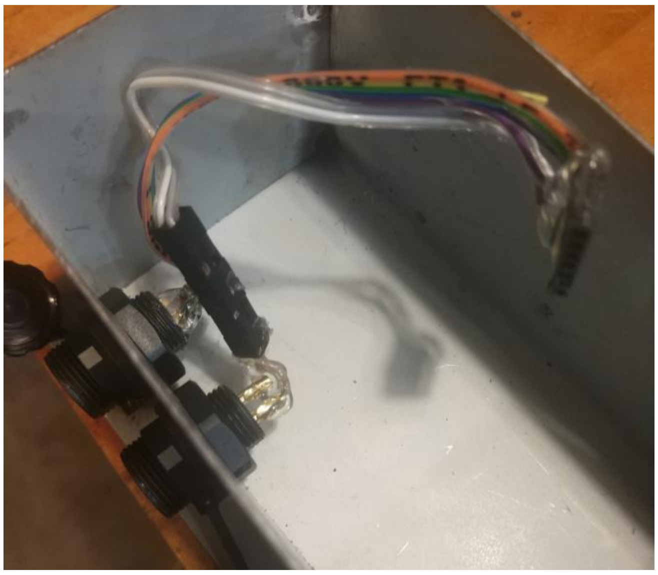
A similar process was repeated on the other components, and other board.





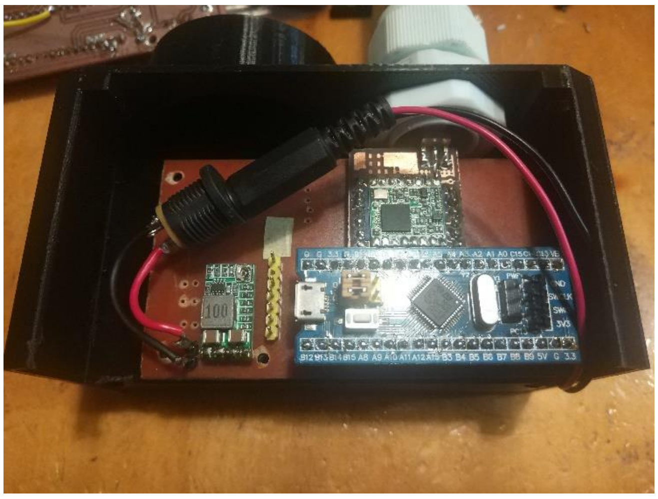
The voltage regulator and battery voltage monitoring circuit were added.





A small removable 'wiring loom' was created to connect the external SP13 plugs to the mainboard.

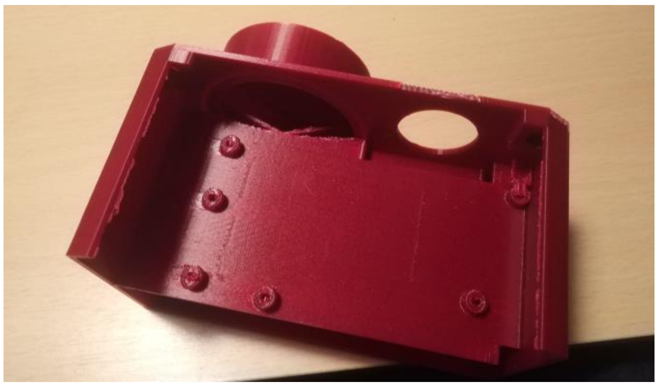
It was also necessary to connect the battery to the caddy, so the barrel connector was added.



18.1.5 Battery fitment issues

Unfortunately, when installing the battery pack, I found that it was too thick - the supplier's dimensions of the pack were for when the pack was empty, not when batteries were installed.

This was an easy enough fix – the CAD model dimensions were adjusted, and a new caddy was 3D printed, this time in a nice burgundy red colour.



18.2 Node firmware

It was now time to implement the state machine designed in 9.4 Software: node firmware above.

The bulk of this programming was relatively easy; although I ran into a strange error while attempting to get the microcontroller to enter sleep mode to save power whilst in the SLEEPING state. For some reason, it would intermittently enter sleep mode, then instantly exit it. Sometimes this error would occur, and other times it would not. Obviously, I needed this behaviour to be reliable.

I initially suspected the problem to be power-related. I tried another power source to little avail – the problem was still as intermittent and confusing as ever.

I next began to suspect the "wakeup" pin on the microcontroller. The STM32 has several sleep modes. The sleep mode I was aiming to get into was "deep_sleep". This sleep mode could be exited via an interrupt or a timer alarm, just like I needed. Another sleep mode, "shutdown", required a rising edge on this pin to wake up. Since the pin was not connected to anything, it was "floating", meaning the voltage would fluctuate randomly. I hypothesised that the random fluctuations were triggering the wakeup pin, causing the microcontroller to instantly wake itself up from deep_sleep mode. I connected this pin to ground, forcing it to remain at 0V. This, too, did not work, which makes sense – the wakeup pin is only supposed to do anything in shutdown mode.

After some more research, I began to suspect that I had a counterfeit STM32, which had some minor hardware differences which caused sleep mode to fail. I tried on a different board with the same results, though I would not be surprised if these are both counterfeit – such counterfeits are very prevalent.

Until this point, I was using the STM32LowPower library to enter deep_sleep mode. This handles all the complicated low-level tasks, including:

- Disable interrupt requests (IRQ).
- Configure wakeup modes such as UART, timer, and interrupt wakeups.
- Enter deep sleep mode using the low power regulator (instead of the main regulator).

While experimenting with running these individual functions manually, instead of through the library, I discovered that using the main regulator instead of the low-power regulator worked. This implies that the counterfeit STM32 has a non-working low-power regulator. Therefore, I re-implemented the library's deep sleep function with the necessary changes. This worked well for sleeping my device.

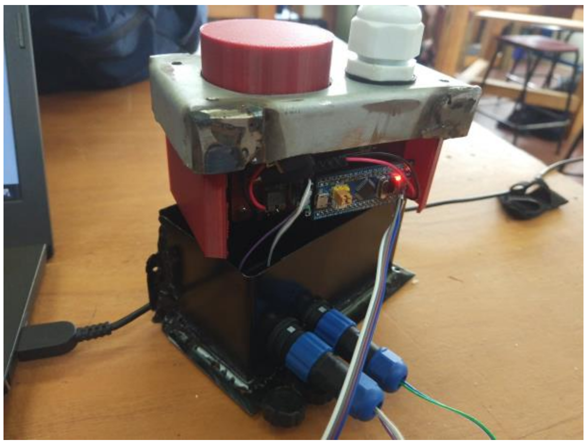


Figure 55. Programming and testing the device.

18.3 Server software

In 13.5 Server software above, I planned a NodeJS-based backend API. This plan remains mostly unchanged. Several database models were linked together with various relationships. For example, a group may contain multiple users and multiple traps. A user may be a part of multiple groups. A trap may also belong to multiple groups.

18.3.1 Models and routes

To create a working prototype, several of these models and interactions needed to be implemented. I implemented:

- User
- Profile (partial)
- Group
- Trap
- Catch

I have not yet implemented these models, as these are not necessary for a basic prototype:

- Image
- Profile (fully)
- Post
- Comment

I have implemented the following routes:

Name	Description
Register	Register a user into the system
Login	Log in to the system
List all traps	List all traps
Query user details	Get a user's details
Create user profile	Create your own profile
Edit user profile	Edit your own profile
Create trap	Add a trap to the system
Process catch from ThingPark	Called by ThingPark when the trap sends a message
Catered traplist	Get a list of nearby triggered traps
List groups	List all groups in system
Create group	Create a new group
Rename group	Rename a group
Delete group	Delete a group
Add traps to group	Add traps to a group
Add users to group	Add users to a group
List traps in group	List traps in a group
List users in group	List users in a group

While beginning to implement the mobile phone app, I realised that I needed to make the outputs of each route more formal. At this point, each route might return different pieces of information depending on various internal factors. For example, a "Get catered trap list" query would not return an error field if there was no error, and would return no trap list if there was an error. This is problematic when developing the app, since it needs to know exactly what to expect, regardless of whether it failed or succeeded. Datatype was another issue - data can be classified into one of several types. An error is of the error datatype, a UUID is of the string datatype, a catch date is of the Date datatype. If the datatypes being sent by the server do not match the datatypes expected by the app, problems will occur.

18.3.2 Statically typed interfaces

To solve this problem, I migrated my server software to TypeScript. This is a derivative of JavaScript that has 'strict' data types, rather than JavaScript's dynamic typing. This makes it much easier to formalise each API route.

To do this, I created an 'interface' or 'model' for each route. This is exactly what it sounds like - a model version of a response, acting as an interface between the app and server. Note this is a similar concept, though different execution, of the database models previously discussed. It describes the name and type of each field returned, allowing for standardisation and formalisation between the app and server.

Here is an example of the model response for the login route:

```
interface LoginResponseModel {
    success: boolean; //did the request happen successfully
    token: string; //JWT token generated
    error: APIErrorModel //error object
}
```

The name of the interface is LoginResponseModel, because it is a model object for a response to a login query.

This response has three fields: success, token, and error. Even if there is no error, a dummy error object will be returned, since the same three fields must be returned every time.

The types of these fields follow on the right of their names, after the colon. Success is a boolean (a true or false value), describing whether the query succeeded or not. Token is a string of characters. Error is a custom type I made, which describes any one of several errors that could be encountered by the server while attempting to process a request.

Once these routes were implemented with statically typed interfaces, I was ready to create the app.

18.4 App

I anticipated this aspect of the software to be the most challenging - I had extremely limited experience with app-making and had already decided to use an app framework completely new to me: Flutter.

The good thing about Flutter is that you write code once, and it is compiled to many different systems, including Android, IOS, web browsers, and Windows/Mac/Linux desktop. This is enormously useful, because otherwise code would have to be written in each of these separately.

Flutter uses the Dart programming language. This is also new to me, though it is quite like C++, which I am familiar with.

18.4.1 UI Design

The basic concept in Flutter is the 'widget'. The app consists of multiple widgets nested inside each other - for example, a Scaffold widget (sets up page, header bar, bottom button bar) may contain a Center widget (centres its children), which contains a text widget (displays some text).

My app design consists of several main tabs, linked together by a "BottomNavigationBar" (referred to as a tab bar in the concept), like Instagram.

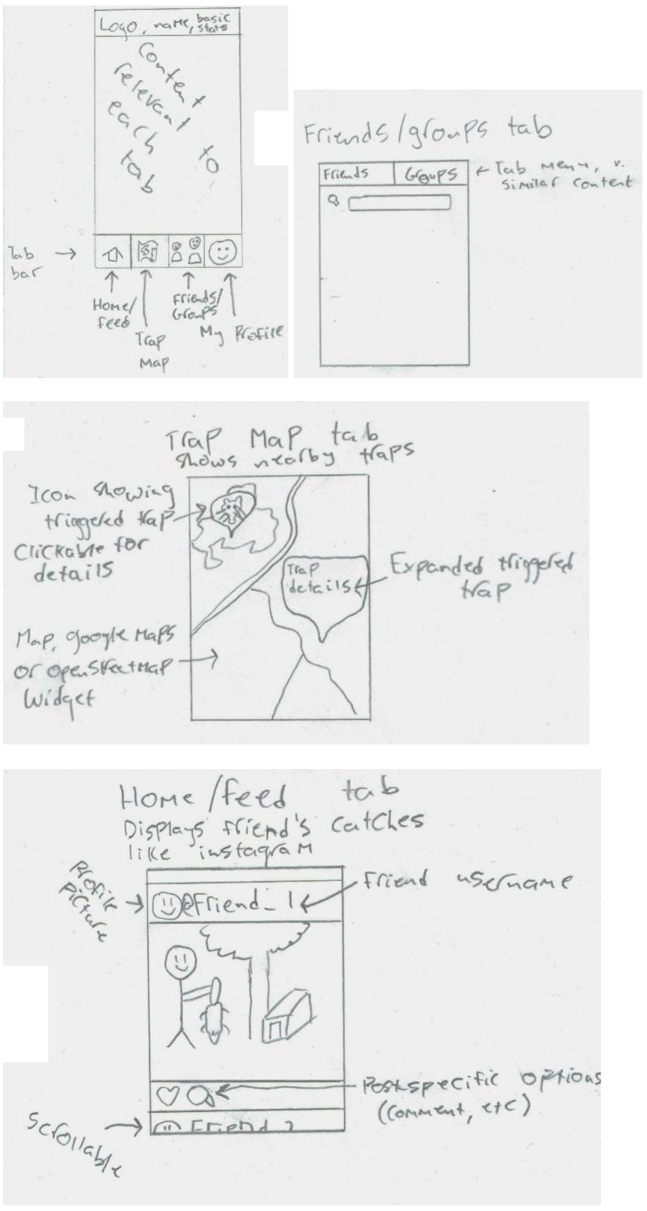


Figure 56. App UI concept.

As with the server, I will first make only the functions necessary for a working prototype. This is primarily the map page, which shows nearby traps and allows them to be interacted with individually. A preliminary login page is also needed, though this will later be integrated into a settings page.

18.4.2 API requests

To log in, fetch trap data, and conduct all the app's other functions, requests must be made to the backend server. These requests must follow a formal predetermined structure, with regards to fields and field types (18.3.2 Statically typed interfaces above). For this reason, interfaces mirroring those used on the server are used for all requests.

Below is the login response model in Flutter. Note the similarities to the NodeJS model in 18.3.2 Statically typed interfaces.

```
class LoginResponseModel extends  ResponseModel{

  bool success; //did the query succeed?
  String token; //authentication token
  APIErrorModel error; //error object, or NoError if success

  … //some extra code hidden for brevity

}
```

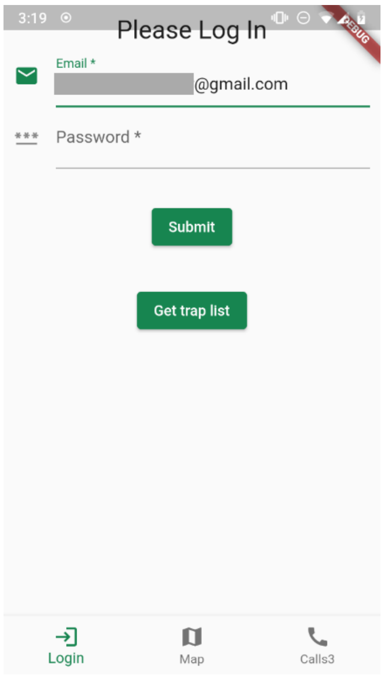
18.4.3 Stored or load-on-demand

Several pieces of information must be stored between app restarts, such as username, password, and authentication token - the user does not want to have to log in every time they start the app.

On the other hand, it is not desirable to store other pieces of information like trap locations (these may change and become outdated) and posts (these rapidly become outdated and waste storage space).

To store information on the phone's local storage, I will use the shared_preferences Flutter library, which is designed for this exact purpose. Data to be stored is labelled with a 'key', which may be searched on at any time.

18.5 App


Please Log In

Email *
[redacted]@gmail.com

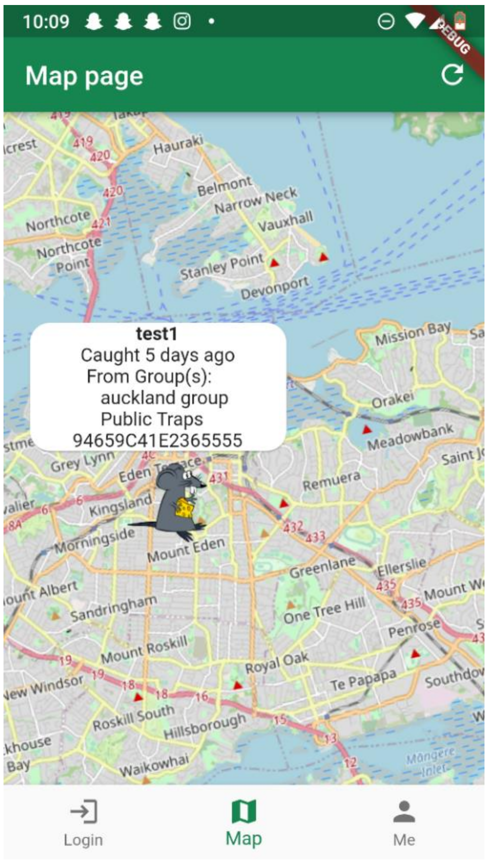
Password *

Figure 57. The prototype login page.

The email address has been retrieved from shared_preferences, as I logged in earlier.

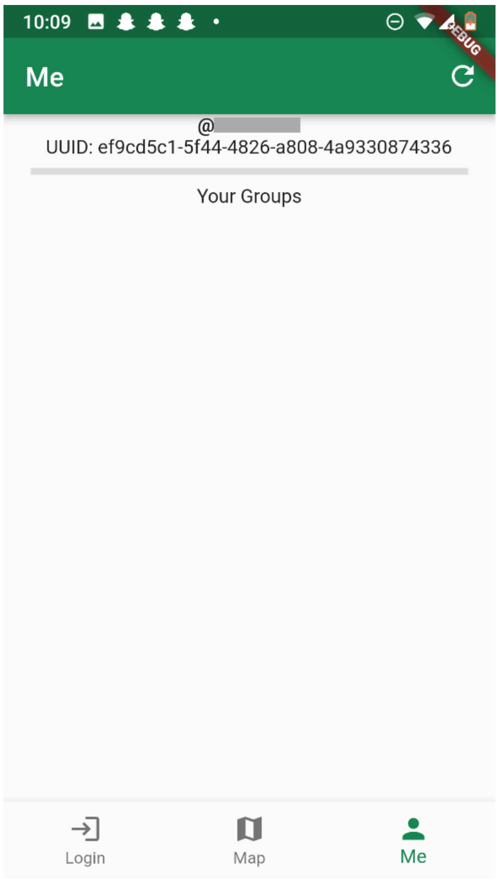
The password is not shown for security reasons.

As discussed above, this login page will not have its own button on the BottomNavigationBar


Map page

test1
Caught 5 days ago
From Group(s):
auckland group
Public Traps
94659C41E2365555

Figure 58. I have zoomed in to the traps and tapped on one. This shows a popup of the trap's information.


Me

@[redacted]
UUID: ef9cd5c1-5f44-4826-a808-4a9330874336

Your Groups

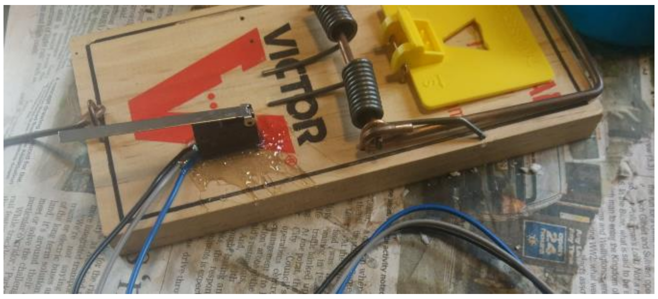
Figure 59. The "Me" page shows some basic information about the user. Eventually, it will display groups, friends, and posts.

18.6 Detector Switch

It was now time to attach the node to the trap. This requires a detector switch specific to the chosen trap, as discussed in 9.1.7 Detector switch above. I opted to use the Victor Professional rat trap over the DOC200, as I did not want to place neighbourhood pets under any unnecessary risk – even though the DOC200 is designed to only catch rats and stoats, there is still a small chance.

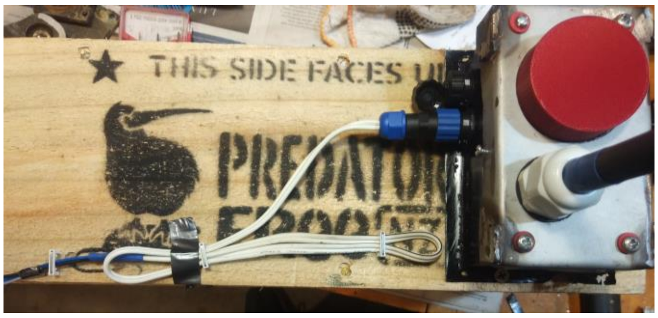
In 11 Concepts above, I created some detailed designs for attaching a detector switch to the trap mechanism. Due to time constraints, I opted to use a simpler solution for the prototype's detector switch.

The ideal location for the microswitch was marked, then the switch was glued with Araldite epoxy.



The wiring was then completed – the switch was connected to the male end of the SP13 plug.

The whole assembly was mounted to the trap.



19 Testing in Environment

The system was now functional, so was ready for a real-world test. First, I demonstrated its functionality in this video:

[redacted]

In this testing, I wanted to find out:

- Does the node survive the elements for an extended period of time?
- Does the node communicate with the Spark LoRaWAN gateway?
- If yes, how reliable is the connection, and what is the signal strength?
- If a catch is made, does the rest of the system work? Do I check the app and notice a change?
- Does the battery voltage drop significantly?

The node was programmed to "phone home" every hour. This allowed me to know exactly how the trap was working, and when signals weren't getting through.

The trap was placed in a quiet location in which rats and other predators were most likely to inhabit and mostly left alone for 6 days.

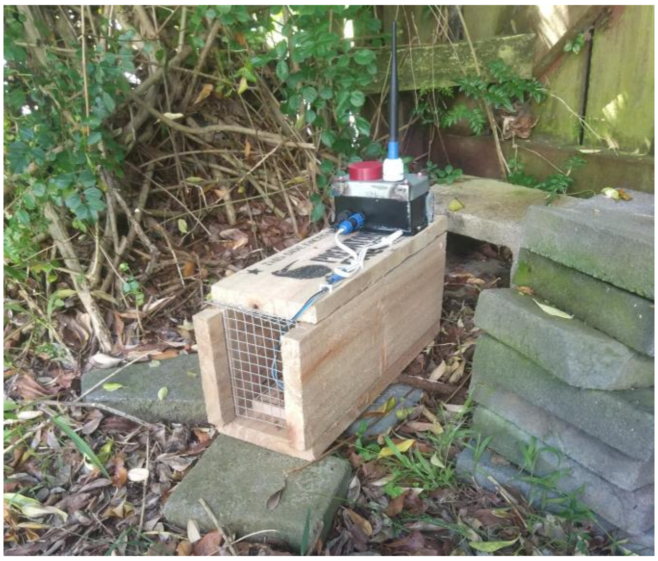


Figure 60. The trap in its new home.

Unfortunately, the trap did not catch anything. Despite this, I still yielded valuable data from this test.

Since the trap was supposed to phone home every hour, I could plot each signal received. Any gaps in the signals mean that a transmission did not make it through.

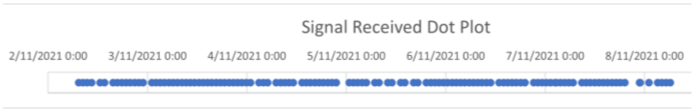


Figure 61. Each dot shows a transmission received from the trap. A gap indicates a missed transmission.

There are clearly a few missed transmissions, though the vast majority of packets are getting through. It must be noted, as in 17 Working Proof of Concept above, that the radio is operating in non-ideal conditions – it has no line of sight to the receiver. Of course, we cannot guarantee that the real trap will have ideal line-of-sight conditions, so it is important to know that the node performs acceptably everywhere.

We can also plot the RSSI of each received transmission. RSSI (Received Signal Strength Indicator) tells us just that – the power of the received signal. A higher number is better, since the signal is more powerful.

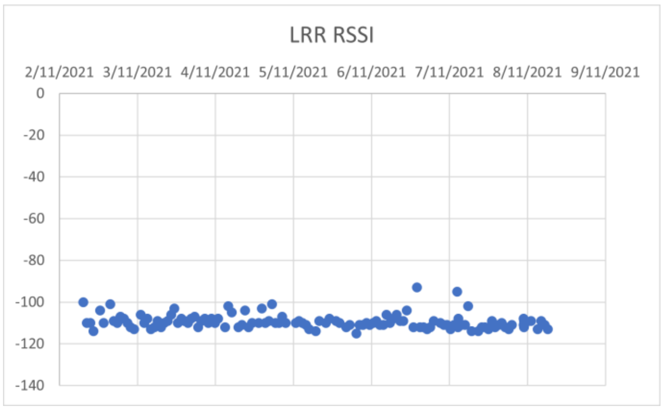


Figure 62. The RSSI over the course of the evaluation period.

The RSSI appears to be quite stable, with a small amount of spread in the y-direction. Most spread appears to be *above* the trend-line, which may indicate that performance gets better, but not worse. In actual fact, this is more likely to be an example of survivorship bias – only signals which are above the trend-line are strong enough to be received, so we do not see much scatter below the trend-line since those examples do not make it through.

The battery voltage was initially around 6.2 volts. Over the course of the trial, this dropped to around 6.0 volts. I am unsure whether to be worried about this change. On one hand, this is a fairly significant drop over a mere 6 days. On the other hand, battery voltages naturally drop much quicker towards the starts and ends of their discharge cycle. Also, the node is transmitting a packet every hour, which could be consuming a lot of power. More research is warranted.

20 Stakeholder Feedback, Round 2

I again contacted my stakeholders, Prof [redacted] of UoA and Olivia Rothwell of PF2050.

[redacted]'s full letter is included at the end of this report. She was very positive about my project, describing it as a "virtuoso piece of technology". As a statistician, she naturally identified it as a useful tool in data collection, especially in the urban backyard environment. Her main question was also data related – she asked, "whether the app allows users to edit the data-log entry – for example if a trap is deliberately set off (as on the video), is there a way to record that this was a false trigger and not a real rat?". At this point, there is not – this is due to the software's lack of maturity. In a more complete version, however, this would certainly be possible, for both private and public traps.

[redacted] had many questions. She made the important point that a low BOM cost is good, but a company must pay its costs and employees, and make a profit somehow. Existing companies invest a lot of time and money into helping customers set up their technology. This is related to my goal of accessibility – I want to make a system which can be set up by almost anybody without requiring expert help, which would reduce these costs. Also, a shift from the use of expensive development boards to discrete components would come with a drop in BOM costs. However, my other production costs remain high due to laborious sheet steel work.

[redacted] liked the idea of gamification. She made the point that shifting between communications protocols will be expensive, and it may be better to stay in the city where Spark's LoRaWAN coverage is available. While installing connectivity off-grid would be costly, the same node hardware and software could be used, due to LoRaWAN's adaptability. Mesh networking would further reduce this cost and hopefully make installation easier, since range concerns would be minimised.

She emphasised the need for extensive testing before putting my product to real-world use. She also provided a link to the *Predator Control Data Standard*, for data export to other systems. This is quite similar to how I store data internally already, so would be relatively easy to implement.

She highlighted the importance of "innovators like you [me]", and the availability of R&D grants to projects like this.

21 Sustainability

As a device intended to benefit the natural world, it is essential that the materials used do not cause it harm. As such, I shall discuss each material and its sustainability.

21.1 Sheet Steel

Sheet steel was used to construct the outer housing. Steel can be recycled an infinite number of times and is thus considered by many the most sustainable metal' (Cain, 2020).

21.2 PLA (Polylactic Acid)

PLA is used in the 3D printed internal caddy.

PLA is made from plant starch and is both recyclable and compostable (in the correct conditions). My product is designed in such a way that would make reuse of this part easy, meaning that even if a node is broken (e.g., from water damage), it is likely that the caddy would survive and could be reused.

21.3 Battery (NiMH, D-cell)

NiMH batteries are used to power the device.

My device accepts any D-size battery, meaning that rechargeable NiMH batteries can and should be used, as opposed to disposable batteries. Their rechargeability means that they will last much longer than disposable batteries. When they are worn out, they can be recycled. Nickel is not a ROHS metal, however it does have some environmental impact, which can be minimised through correct recycling procedures (McMahon, 2013).

21.4 Fibreglass circuit boards

PCBs are used to host the electronic components used in my device.

Circuit boards make up the bulk of e-waste, which is currently a large problem for the world. One main reason for this is the nature of consumer electronics - every year, a new, shinier device is released, meaning that all the old ones are thrown out. This will not be the case with my device, which (hopefully) will receive few hardware iterations, so will remain in use for a long time. Circuit board recycling does exist and is aided by the use of non-hazardous electronic components, such as lead-free solder. Lead-free solder has a higher melting temperature than normal leaded solder, making it harder to work with. As such, it is not feasible for my prototype to use lead-free solder. My prototype uses several individual electronic 'modules' soldered onto a main board. When my prototype is no longer useful, these modules can simply be removed and reused individually for some other device, minimising waste (EPA, 2012).

22 Evaluation and Moving Forward

This project was a success. I produced a functional prototype which completed almost all the functions of the existing commercial options for a small BOM cost.

Despite this, I did not complete all the points on my brief – this make sense, since my brief described a system complete with gamification and all other features, which would reasonably take more than a school year to produce.

22.1 Fitness for purpose

As shown by my evaluation period, the node hardware works quite well in its environment. The radio link is acceptably reliable. The system replicates most of the features of commercial options.

For me to be comfortable deploying the node in a public place, a GPS system would have to be active. This is not currently the case, though would not take much more effort. As it stands, this aspect of the system is completely usable for my secondary and tertiary stakeholders – those on private land.

The system is not currently fit to publicise traps and promote volunteering via the app. This would primarily be a matter of improving the software.

22.2 Improvements

There are several things which I would like to add to the system, or change, to meet my brief or meet it better.

22.2.1 Things to add

- Complete the app and server software, including adding gamification and social media features.
- Make the node configurable by the end user, via the same app.
- Negotiate a way for end users to pay for Spark network access via a TrapApp company.
- Develop a TrapApp LoRaWAN gateway, or add the ability to use an arbitrary LoRaWAN network so any gateway could be used.
- Add the GPS module to the node, completing its intended feature set.
- Add mesh networking to the node. This is almost exclusively a software change to the node's firmware.

22.2.2 Things to change

- Make the housing easier and cheaper to manufacture – currently, all the sheet metal work takes a very long time which is not feasible for mass-production. A material change (e.g., from metal to plastic) could be considered, though extensive strength testing must be completed.
- Investigate voltage regulator performance. Initial tests suggested that the advertised specifications were not accurate, to nobody's surprise. A linear regulator may be more efficient with this low current draw.
- Further investigate why sleep mode would not work with the low-power regulator, as using the main regulator consumes extra power.
- Improve radio performance of the node. The current radio operates at 20dBm, which is much less than the 30dBm legal limit. Significant radio performance benefits may be yielded by changing this, though finding a 30dBm radio module appears to be challenging.

22.3 Moving forward

Next year, I will be studying engineering at university. This means that this project will have to be shelved. Nevertheless, I have learned an enormous amount about conservation and this technology.

It is possible that I will revisit this project later in my life, perhaps as a university project, or even a business – if someone hasn't made a fully-working version of my idea by then.

23 Reflection

I can say with certainty that this is the most complicated engineering project I have completed. It is also my first 'full stack' system – it has hardware, firmware, a server, and an app. Getting all of these components to integrate was certainly a challenge, and I learned a great deal.

I have learned a significant amount about the technologies involved – I started with next to no background in app development, LoRaWAN, STM32 use, developing a working API/database server, TypeScript, etc. I used the knowledge I already had to work up to these heights, making the learning process much more manageable – for example, I began by using the relatively-familiar NodeJS and eventually identified a need which required me to upskill into TypeScript. This change was made easier by my inherent understanding of the context where TypeScript would be useful (since I had just run into the problem), and some motivation to learn it.

I could definitely have improved on my time management – I spent nearly an entire school term developing and refining my circuit board production process (another thing which I initially had very little experience in). If I had spent a bit more time researching and less time making failed test pieces, I might have discovered the sponging technique earlier and saved time. Alternatively, I could have simply ordered boards from a manufacturing company, since I didn't end up changing them that much. Of course, I was working simultaneously on other aspects on the project, so not all was lost.

I am fairly happy with the compromises I made because of my lack of time. I did not become bogged down insisting that everything must be perfect. For example, I deferred implementing GPS despite having already invested some time into researching it. I would have liked to have made a more complete software package, however it was the same time constraints that prevented this. In some ways, software is the best part of the project to be underdeveloped – there is a physical product to show, and it completes a function linked to an app.

As well as learning about the technologies involved, I learned a huge amount about the nuances and details of conservation. I have become more conscious of the immense amount of volunteer, scientific, and regulatory work surrounding the protection of our native species. It has also given me a new perspective of the role of engineers and scientists in sorting out this mess – conservation requires input from everybody, not just the bush-whackers working on the ground. I look forward to my continued involvement in conservation.

Thank you again to all those who helped with my project.

24 References

Cain, M. (2020, December 11). *Is Steel A Sustainable Design Material?* Retrieved from RealSteel Center: https://realsteelcenter.com/blogs/interior-design-tips/is-steel-a-sustainable-design-material-the-answer-might-surprise-you

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining "gamification". *MindTrek '11*, 9-15.

DOC. (2021). *Predator Free 2050: Practical Guide to Trapping 2nd Edition.* Wellington: Department of Conservation.

Environmental Protection Agency NZ. (2006). *1080 Reassessment Application.* Environmental Protection Agency NZ.

EPA. (2012, October). *Printed Circuit Board Recycling Methods.* Retrieved from EPA: https://www.epa.gov/sites/default/files/2014-05/documents/handout-10-circuitboards.pdf

Forest and Bird. (2018, April 16). *Frequently Asked Questions about 1080.* Retrieved from Forest and Bird: https://www.forestandbird.org.nz/resources/frequently-asked-questions-about-1080

Hamari, J. (2019). Gamification. *The Blackwell Encyclopedia of Sociology*, 1-3.

Jones, C., Warburton, B., Carver, J., & Carver, D. (2015). Potential Applications of Wireless Sensor Networks for Wildlife Trapping and Monitoring Programs. *Wildlife Society Bulletin*, 39(2):341–348.

Lieberoth, A. (2014). Shallow Gamification: Testing Psychological Effects of Framing an Activity as a Game. *Games and Culture*, 10 (3): 229–248.

McMahon, R. (2013, March 12). *How to choose the most sustainable battery for small portable devices?* Retrieved from Stack Exchange: https://sustainability.stackexchange.com/questions/616/how-to-choose-the-most-sustainable-battery-for-small-portable-devices

Meek, P. D., Ballard, G., Milne, H., Croft, S., Lawson, G., & Fleming, P. J. (2020). Satellite and telecommunication alert system for foot-hold trapping. *Wildlife Research* , 48(2) 97-104.

Ministry for the Environment & Stats NZ. (2019). *New Zealand's Environmental Reporting Series: Environment Aotearoa 2019.* Ministry for the Environment and Stats NZ.

Norton, D. A., Young, L. M., Byrom, A. E., Clarkson, B. D., Lyver, P. O., McGlone, M. S., & Waipara, N. W. (2016). How do we restore New Zealand's biological heritage by 2050?. *Ecol Manag Restor*, 17: 170-179.

Royal, T. A. (2007, September 24). *Kaitiakitanga – guardianship and conservation - Rāhui – prohibitions.* Retrieved from Te Ara - the Encyclopedia of New Zealand: https://teara.govt.nz/en/kaitiakitanga-guardianship-and-conservation/page-6

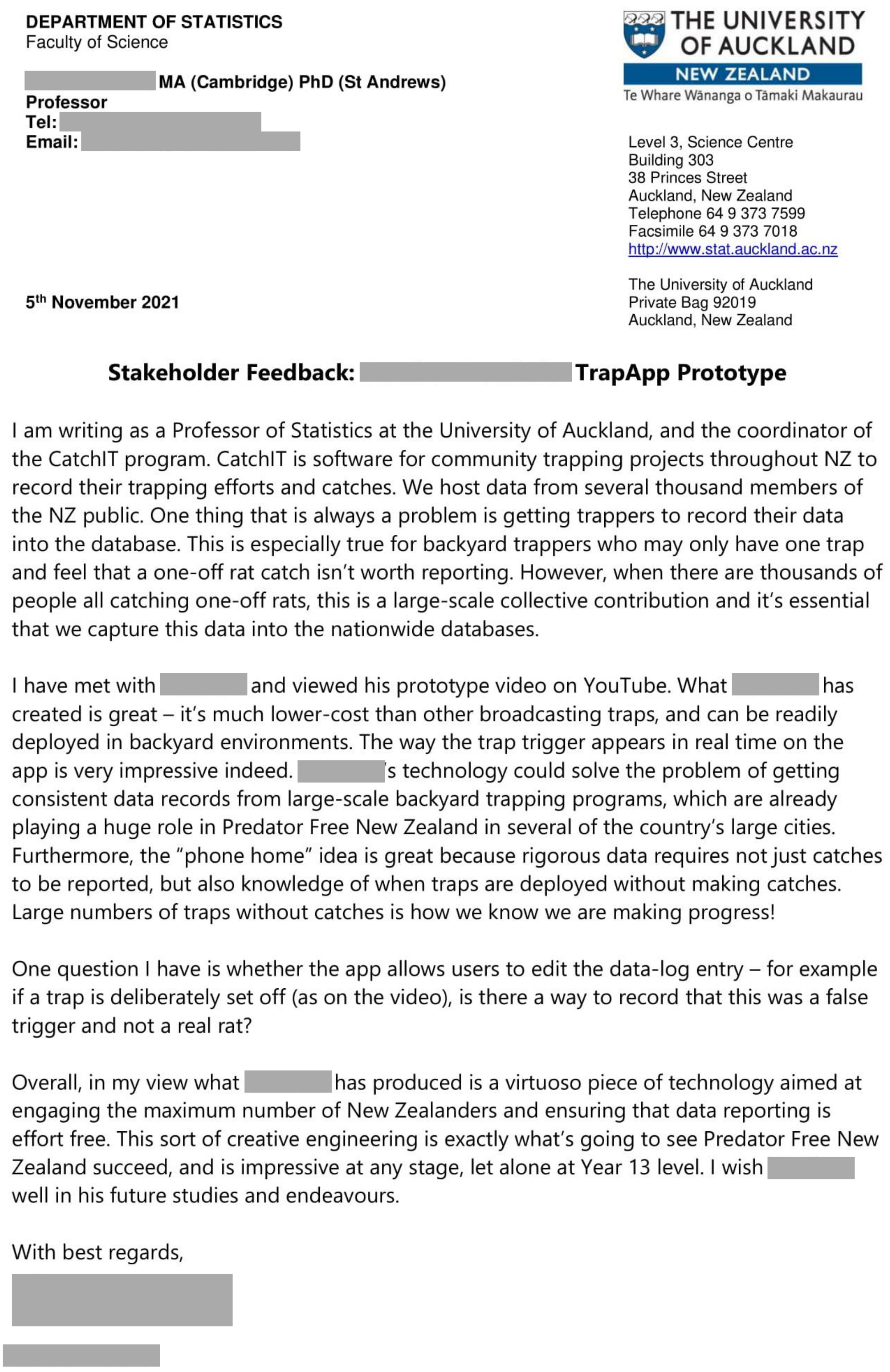
Spiess, A. (2017, August 27). *#155 The 5 Best Solar ChargerBoards for Arduino and ESP8266.* Retrieved from Youtube: https://www.youtube.com/watch?v=ttyKZnVzic4

Stock, R. (2018). *Mycoplasma bovis slaughter pushes annual cow cull total higher.* Stuff.

Timoti, P., Lyver, P. O., Matamua, R., Jones, C. J., & Tahi, B. L. (2017). A representation of a Tuawhenua worldview guides environmental conservation. *Ecology and Society*, 22(4):20.

Warburton, B., Jones, C., & Ekanayake, J. (2015). *Remote monitoring of traps using wireless-based systems.* Napier: Landcare Research.

25 Professor [redacted]'s Final Feedback



DEPARTMENT OF STATISTICS
Faculty of Science

[redacted] **MA (Cambridge) PhD (St Andrews)**
Professor
Tel: [redacted]
Email: [redacted]

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Level 3, Science Centre
Building 303
38 Princes Street
Auckland, New Zealand
Telephone 64 9 373 7599
Facsimile 64 9 373 7018
http://www.stat.auckland.ac.nz

The University of Auckland
Private Bag 92019
Auckland, New Zealand

5th November 2021

Stakeholder Feedback: [redacted] TrapApp Prototype

I am writing as a Professor of Statistics at the University of Auckland, and the coordinator of the CatchIT program. CatchIT is software for community trapping projects throughout NZ to record their trapping efforts and catches. We host data from several thousand members of the NZ public. One thing that is always a problem is getting trappers to record their data into the database. This is especially true for backyard trappers who may only have one trap and feel that a one-off rat catch isn't worth reporting. However, when there are thousands of people all catching one-off rats, this is a large-scale collective contribution and it's essential that we capture this data into the nationwide databases.

I have met with [redacted] and viewed his prototype video on YouTube. What [redacted] has created is great – it's much lower-cost than other broadcasting traps, and can be readily deployed in backyard environments. The way the trap trigger appears in real time on the app is very impressive indeed. [redacted]'s technology could solve the problem of getting consistent data records from large-scale backyard trapping programs, which are already playing a huge role in Predator Free New Zealand in several of the country's large cities. Furthermore, the "phone home" idea is great because rigorous data requires not just catches to be reported, but also knowledge of when traps are deployed without making catches. Large numbers of traps without catches is how we know we are making progress!

One question I have is whether the app allows users to edit the data-log entry – for example if a trap is deliberately set off (as on the video), is there a way to record that this was a false trigger and not a real rat?

Overall, in my view what [redacted] has produced is a virtuoso piece of technology aimed at engaging the maximum number of New Zealanders and ensuring that data reporting is effort free. This sort of creative engineering is exactly what's going to see Predator Free New Zealand succeed, and is impressive at any stage, let alone at Year 13 level. I wish [redacted] well in his future studies and endeavours.

With best regards,

[redacted]

[redacted]