MONASH UNIVERSITY
DEPARTMENT OF ELECTRICAL AND COMPUTER SYSTEM ENGINEERING

ECE4094/ECE4095 - FINAL YEAR PROJECT

OpenFence
GPS Based Livestock Management

Alex Muir

Supervised by
Dr. Mehmet Yuce

November 10, 2016

Significant Contributions

- Successfully created a complete system for virtually fencing and tracking livestock.
- Designed a custom PCB containing all required components for the collars operation, with low current consumption down to 4mA.
- Designed the geofence algorithm with boundary checking and a directional audio output.
- Implemented in software a tilt corrected compass by combining available data from the 9 degrees of freedom inertial measurement unit.
- Designed power saving software feature which puts the collar device to sleep during periods without movement, and wakes when motion is detected by the accelerometer.
- Designed a method of intelligent LoRa output power selection, such that the minimum power required is used.
- Implemented a LoRa network with a star topology, achieving a maximum measured range of 670m.
- Designed a flexible and water resistant enclosure for mounting around the neck of livestock.
- Designed and constructed a base station that communicates over LoRa with the collar devices, allowing transfer of logged position data to the online database, and changes to the fence position to be transferred to the collars.
- Designed and developed a web based interface allowing the placement of fence locations on satellite map.
- Designed and developed web based interface for viewing logged animal movements, allowing monitoring of their location in near real-time.
- Implemented the ability to remotely update fence locations and settings on the collar devices, and store these securely on the collar in flash memory.

Poster

MONASH University Engineering

Department of Electrical and Computer Systems Engineering

ECE4095 Final Year Project 2016 — **Alex Muir**

OpenFence
GPS Based Livestock Management

Supervisor: Dr Mehmet Yuce

The Project

OpenFence is a digital livestock fencing system, which uses **GPS** and **stereo audio alerts** to contain animals within a specified boundary, when they have been trained with the audio alert.

The location of the fences can be updated using the **web interface**, and transferred to collar devices wirelessly using LoRa via the in field **base station**.

The movements of the animals are **logged** and transferred to the web database, to allow **analysis** of movements and land utilisation.

The Purpose

OpenFence is designed to make it easier to implement **productive** and **environmentally** friendly grazing practises, such as cell grazing.

Cell grazing has been shown to increase biodiversity, ground cover, **soil carbon** and water retention, however it is not commonly implemented due to the additional expenses of fencing and labour.

OpenFence solves these issues by making it **simple** to set up **temporary** fences and easy move animals across the property.

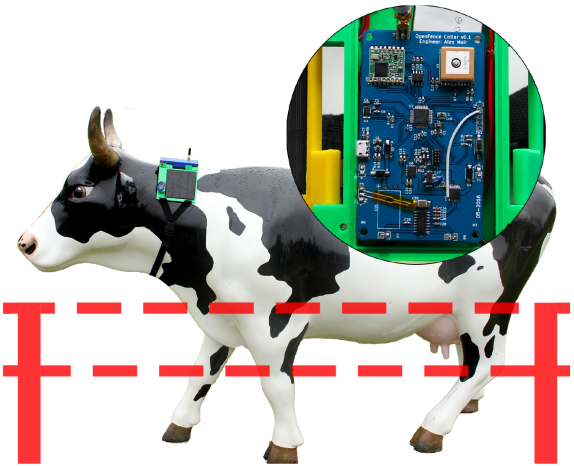


Figure 1: The OpenFence collar mounted on a fiberglass cow. Inset: View of the custom designed PCB.

Overview of Software Components

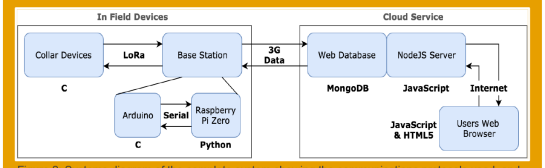


Figure 2: Systems diagram of the complete system showing the communications protocols used, and the devices and software created as part of this project.

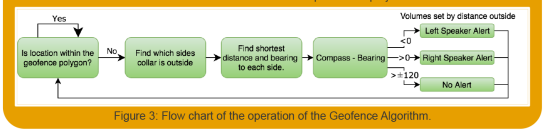


Figure 3: Flow chart of the operation of the Geofence Algorithm.

Collar Circuit Diagram

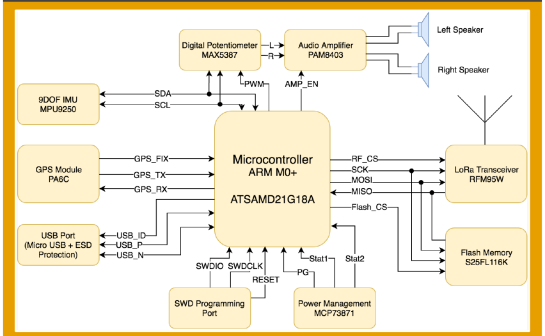


Figure 5: Overview of the components of the collar circuit, with the digital and analog interconnections.

Key Features

Virtual Fence Algorithm: Polygons consisting of up to **255** points can be used to define the boundary.

Intelligent Sleep Mode: Current consumption down to **4mA** in sleep mode, with wake on motion.

Wireless Communications: Range of up to **650m** (in urban testing) using LoRa. Optimal output power selection implemented. Allows the wireless updating of fence locations, collar setting and real time tracking of animal movements.

Tilt compensated compass: Compass accounts for tilt of the collar using 3DOF accelerometer and 3DOF magnetometer.

Web Interface

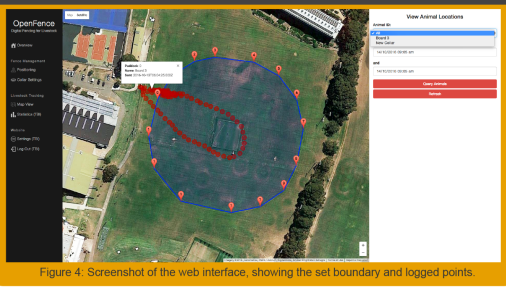


Figure 4: Screenshot of the web interface, showing the set boundary and logged points.

Open Source

All source code, CAD files and PCB design files are available under the GNU GPL v3 license at: https://github.com/plyalex/OpenFence

Executive Summary

This project was created with the intention of filling the need for a method of flexibly fencing livestock, such that sustainable grazing practices such as cell grazing can be implemented more easily. Sustainability of agriculture will be a key area for research and development in the future, and technology may be able to provide some solutions to the issues faced.

In order to effectively tackle this topic, a review of literature was performed first. This report presents cell grazing's benefits on the environment, the existing work in the field of virtual fencing is examined and an in depth explanation of LoRa networking is provided.

OpenFence's aim was to create a complete virtual fencing system, with the collars, base station and web interface all being designed as part of the project. The geofence algorithm created not only determines if the animal is outside the boundary, but also provides a directional alert based on the compass bearing. This is output through the speakers on either side of the collar. The intention of this directional signal is to ensure the animal returns to the boundary in the most direct manner, reducing the possibility of confusion.

A number of software features on the collars were implemented including wake on motion, tilt compensation for the compass and intelligent wireless power selection. By using the accelerometer to produce an interrupt when motion is detected the collars microcontroller can remain in a low power state until there is movement, at which point it wakes to check the GPS position. The tilt corrected compass is necessary, as the collar will rarely be kept level whilst the magnetometer is being measured. The intelligent wireless output power selection ensures that the lowest level of transmission power required for reliable communication is used.

The in-field base station provides the link between the LoRa wireless network and the internet based database, updating the fence locations on the collars and receiving the logged waypoints.

The website that was created allows fences to be defined by clicking on a satellite image of the location. The settings used by each individual collar can be updated from the website. Finally the website also provides a page in which the logged movements of the livestock can be tracked.

As part of the project a small amount animal testing was performed, with the majority of the testing involving humans walking around holding the collar. The solar panels ability to maintain the batteries charge is confirmed and the range of the LoRa network tested, revealing a maximum range of 670m in an urban environment.

Through the project a number of issues were discovered, which could be improved upon in future work, the design of the collar enclosure is the most obvious of these. Along with this, a variety of features could be added to the geofence software such as virtual containment fences if the animal continues too far beyond the boundary.

Contents

List of Figures

List of Tables

1 Introduction

Sustainability in agriculture is an area that is currently receiving a lot of publicity due to issues such as feeding a growing population, soil degradation and climate variations. The current reliance on fossil fuels and the associated exposure to price shifts in production costs is further incentive for the implementation of sustainable improvements. Cell grazing is a method of grazing cattle which has been shown to improve land and increase productivity. Cell grazing involves the use of temporary electric fences or high tension fencing to contain animals in a smaller area and move the livestock every few days. This requires more fencing than typical continuous grazing. This is the problem that OpenFence aims to solve, by creating a digital boundary which can be easily moved by the web interface. Animals are then contained within the boundaries by a collar, using the location points specified by the manager.

Additional benefits of OpenFence include being able to easily exclude fragile areas from grazing, such as rivers and regrowth forests, as well as seasonally inappropriate regions such as where a native bird comes to breed. Being able to track herd movement, and monitor behaviour using the large amount of data that is collected could be useful for researchers and managers.

1.1 Aims and Objectives

OpenFences aim is to allow graziers to focus on the management of their land and reduce the costs and labour associated with cell grazing techniques. Fences are expensive to install and maintain, and arent able to change with the variable nature of feed quality and quantity. Moving livestock frequently to fresh pasture, through gates, is time consuming and difficult. OpenFence aims to remove these inhibiting factors of cell grazing through the flexible nature of digitally defined boundaries.

OpenFence will build upon research performed over the past few decades to create a complete engineered solution, taking it from a research device to a device that is manufacturable and usable for extended periods in the field. As this is primarily an electrical engineering project, the objectives focus on the design of the electronic device and associated software, rather than on animal behaviour outcomes.

The solution should be a small and lightweight collar, with self-contained charging and energy storage, variable alerts of sound and wireless low powered communications. Each collar has a GPS receiver allowing the software to determine if the animal is within the boundary, approaching the boundary or outside the boundary. The devices will be weather-proof and able to withstand harsh outdoor conditions. Keeping the device cost to a minimum will be a key goal.

The collars will be paired with a base station in the field, which will allow the collars to use short-range telemetry, reducing their power consumption compared to a direct internet connection such as 3G data. The base station will also contain its own battery supply and solar panel. The base station will allow uploading of the data it receives from the collar devices to the internet, and download any changes to the fence locations to the collars. This could be placed near watering points as the animals will always come back to this area throughout the day, allowing the transfer of data if they have been out of range.

The management software will run through a web application, which allows the simple placement of fences and viewing of logged GPS waypoints for individual or multiple animals. The benefit of using a web-based application is that it will allow the farmer to check and set the fences from a mobile device in the field, on a computer at home, or for other users to view the data such as an agronomist. This however does not mean that the system can be used to manage livestock remotely, as the health of the animals, access to water and amount of feed must be monitored by an experienced grazier on the ground.

The name OpenFence was chosen because all software and hardware that is being developed

will be made available as Open Source, under GNU General Purpose License version 3.

1.2 Document Outline

This report begins with the literature review, covering in detail the benefits of cell grazing, exploring previous research into virtual fencing and an overview of LoRa networking. Next the process of creating the hardware and software is outlined. An analysis of the operation of the system and results are presented. Finally, a discussion of the projects successes and difficulties are detailed, including recommendations for future work.

The terms virtual fence, digital fence and geofence have been used interchangeably, as they all refer to the same thing.

2 Literature Review

2.1 Cell Grazing

Cell grazing changes the focus of a grazing farm operation from raising animals to managing grasslands. The health of the land must be the priority, as in poor conditions productivity of the farm will be reduced. The most common style of grazing is continuous, where animals are left in an area for long periods of time, often the entire year and eat the grass as it begins to grow back. This results in more stress on the plant and hampered regrowth, eventually resulting in a loss of biodiversity of grasses.

Cell grazing is a time-control grazing method; however, it differs from other time-control techniques such as block or strip grazing in that it has a holistic focus. The focus of cell grazing is on sustainability and optimising profit, rather than on maximising plant and animal growth. Holistic management is a whole farm planning method which aims to deliver successfully on the triple bottom line, with positive environmental, economic and social outcomes. Cell grazing aims to maximise the harvest of sunlight, by implementing the following principals [8]:

- Control rest to suit the growth rate of the plant
- Adjust stocking rate to match carrying capacity
- Plan, monitor and manage the grazing
- Use short graze periods to increase animal performance
- Use maximum stock density for the minimum time
- Use diversity of plants and animals to improve ecological health
- Use large mob sizes to encourage herding

With the goal to [8]:

- Improve the biodiversity of the soil
- Increase the carbon in the soil, through the creation of topsoil
- Increase the drought tolerance of the grassland, through deeper roots and better water holding ability
- Increase animal productivity and health

Pioneers such as Allan Savory have been advocating the benefits of holistic management, and educating others on its implementation for over 50 years. He believes that desertification of land can be reversed by running more livestock, rather than the conventional response to run less or completely remove the animals and rest the land, which does not prepare the land for recovery. When it rains the water runs off the dry, bare land. If livestock are used correctly they can have a beneficial effect as they create many small divots in the ground, where water can soak in, as well as fertilising the soil.

There are many examples of people successfully using this type of holistic management on their land and seeing real improvements. Almost all of the graziers who have won Australian Farmers of the Year awards since 2007 have described cell grazing as the core of their farm plan [9]. As an example, in a low rainfall area of South Australia, one grazier has gone from a ground cover of less than 50% to 70-80% in just 7 years of using cell grazing [10]. This increased ground

cover means that with only 10mm of rain the paddocks produce feed, when previously the water just ran away. For scientific analysis of cell grazing see T. McCoskers report [8].

Soil is the third largest sink of carbon in existence, after oceans and geologic sinks [11], so increasing the amount of carbon in the soil presents a safe and simple way to sequester carbon. Many of our modern farming practices result in a loss of soil carbon, which reduces the productivity of the land, requiring more inputs to maintain fertility. The most effective way to increase the amount of soil carbon is to ensure that there is plant growth in the soil at all times [11]. Perennial grasses with good coverage and animals that are involved in the grasses life cycle have the potential to take carbon out of the air and store it in the soil.

The factors that make cell grazing a less attractive option than continuous grazing are, access to water, more fencing and increased time spent herding animals. Through smart layout of the cells it is possible to reduce the number of watering points required. The cost of fencing these additional smaller paddocks is significant, and therefore a major drawback of cell grazing. There is also additional labour time associated with moving animals frequently, requiring the animals to be rounded up and herded through gates across the property.

2.2 Existing Research in Virtual Fencing

Research into virtual fencing has been ongoing for almost three decades, with groups such as the United States Department of Agriculture, CSIRO and a number of universities all working on testing the viability of containing livestock within invisible boundaries.

Livestock such as cattle have been successfully trained to associate audio alerts with a boundary, when an electric shock was used as part of the training process. There are many papers on the subject by the US Department of Agricultures, Dean Anderson [1]. When no electric stimulation was used in training, as studied by Butler et al., there was a limited reaction by dairy cows to a variety of sounds [12]. The electric stimulation should have a startling effect only, rather than being a painful experience, as it is only designed to get the animals attention. The level of stimulation required is an area requiring further investigation as it is affected by a number of parameters such as animal size, placement and weather [1].

The virtual fencing systems have had a variety of designs, with collars and ear tags being the most common categories. Ear tag systems need to be a lot smaller and have therefore been limited in features and battery life [13]. Collars allow for a larger size and weight device to be carried, however it has its own drawbacks. Rotation of the collar on the neck will result in applying audio alerts and shocks incorrectly, and the device may have issues communicating or receiving GPS signal [1]. Of either type, key design criteria for the device include being small, lightweight and robust, as the animal will be using it for a continuous period and will likely bump into things, lie on it, etc [14]. Being in an outdoor environment the majority of the time the device should be dust and waterproof as well.

Many wireless technologies have been used to communicate with the on animal devices from the long range WiFi of Butler et al.[12], to sub 1GHz systems such as the 915MHz GFSK (Gaussian Frequency Shift Keying) modules of Wark et al. [14]. The topology of the wireless network generally will be either a mesh or a star, with each having its own advantages and disadvantages. Mesh networks provide increased range, as nodes can communicate with each other in order to pass a message back to its intended destination. However, this additional time spent monitoring the air- waves and transmitting increases power consumption. In star networks the nodes communicate directly to the intended destination if it is within range, meaning that communications are short and simple.

In order to operate for extended periods of time without being removed for the animals, recharging of batteries is required. This has most commonly been achieved with solar panels attached to the device, however another possibility is to convert the regular motions of the animal into energy.

Although this has not been done before with livestock there are patents for generating energy from human motion [1]. Due to the varied nature of the weather there may be times where there is limited solar energy received for a number of days, so it is important to have enough battery capacity to operate the system during this period.

Virtual fencing algorithms use a number of methods and have varying levels of complexity. The technique used by Butler et al. [12]was to calculate the distance of the animal to each of the fence sides, if any of these were negative (signifying it is outside that boundary), then an audio alert proportional to the magnitude of the distance was applied. In the method used by Anderson [1], in addition to the distance, the heading of the animal was obtained from the included magnetometer and compared to the bearing of the fence side. These two pieces of information allow for the audio alert to be applied on the side that will send the animal back inside the boundary in the shortest amount of time. This directional virtual fencing technique is shown in Figure 1.

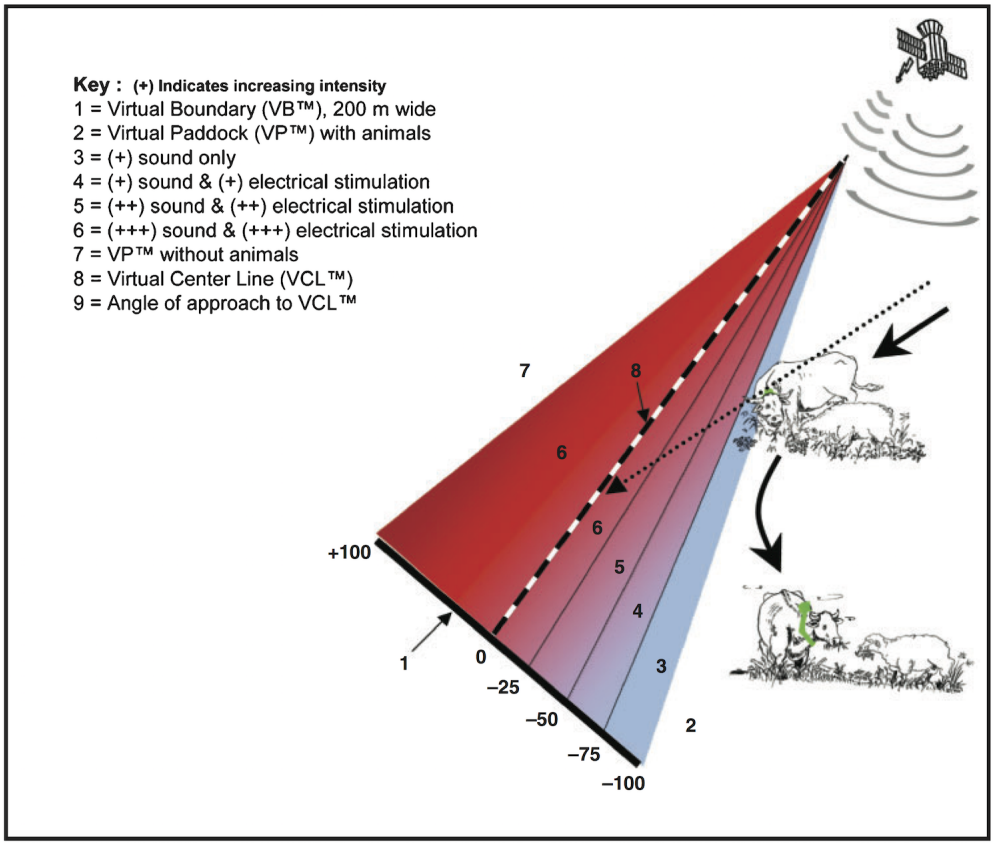


Figure 1: Diagram showing how directional virtual fencing works. Sourced from Anderson [1]

2.3 LoRa Wireless Communications

LoRa is a **Lo**ng **Ra**nge, low power wireless networking protocol created by Cycleo SAS (now owned by Semtech) [15]. It is suited for applications which require low data rates, long battery life and long range. It is a frequency agnostic protocol, however it is primarily operated on in the sub 1 GHz unlicensed Industrial, Scientific and Medical (ISM) bands [16]. In Australia frequency bands such as 433MHz or 915MHz can be used, with 868MHz, another common frequency able to be used in Europe. Sub 1 GHz frequencies have benefits of improved propagation characteristics over more commonly used ISM frequencies of 2.4 and 5.8 GHZ, giving longer range and greater resilience to obstacles [17].

LoRa uses spread spectrum frequency modulation to encode information, this means that the

entire allocated bandwidth is used in communication (wide band), giving it more resilience to noise and interference [18].]. The type of modulation is known as chirp spread spectrum (CSS). Each chirp can either be directed as up or down and is used to encode the information. An up chirp, a linearly increasing frequency pulse over a certain time period is shown in Figure 2, with a down chirp being a linearly decreasing frequency pulse.

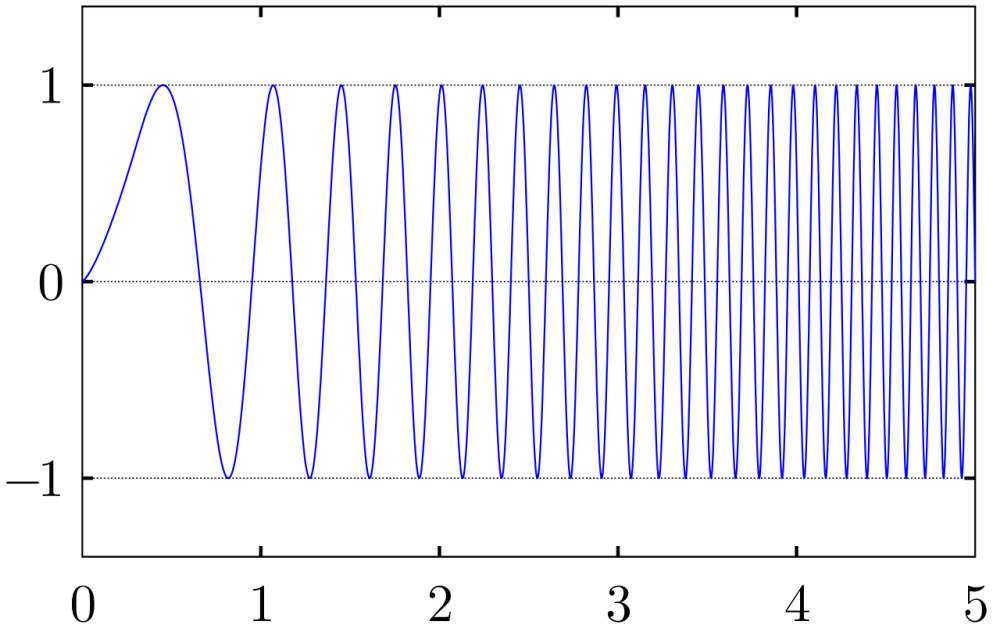


Figure 2: Diagram of an up chirp pulse [2]

The LoRa receivers use a technique called co-channel GMSK rejection to demodulate packets up to 20dB lower than the noise floor, which significantly increases the range, without having to increase the output power, or quality of receiver components [3]. Compared to a frequency shift keying (FSK) system which require a signal 6-10dB above the noise floor this is a huge advantage [19]. Co-channel GMSK rejection works due to the random nature of noise, by averaging it over the period of the chirp it can cancel out the majority of it, making it is possible to successfully communicate below the noise floor [3].

The protocol is flexible with the ability to choose the bandwidth, spreading factor and error correction rates. Some of the many possible combinations are showing in Figure 3. The bandwidth can be chosen to be a range of values between 10.4kHz and 500kHz, with a wider bandwidth providing increased effective data rate. This reduces the time spent transmitting but may be limited by regulation in some countries.

Each symbol (bit of information) is represented by a number of chips (pulse of the spread spectrum code), and the spreading factor determines the ratio of chips per symbol. The larger the spreading factor the longer the transmission will take, however it will be more resilient to noise and interference [3]. Both the sender and receiver need to be set to use the same spreading factor in order to communicate. This allows other devices to communicate at the same time, as long as they are using another pair of spreading factors due to the orthogonal nature of the different factors [20].

As can be seen in Figure 4 the LoRa packet is comprised of three elements, the preamble, the header and the data payload. The payload is used to synchronise the receiver to the data flow. The header can be attached in the case of explicit header modem. However, if payload, coding rate and CRC presence are known in advance, then the implicit header can be used where the header is removed. The header in explicit mode contains the length of the payload, the error correction code

Bandwidth (kHz)	Spreading Factor	Coding rate	Nominal Rb (bps)	Sensitivity indication (dBm)	Frequency Reference
10.4	6	4/5	782	-131	TCXO
	12	4/5	24	-147	
20.8	6	4/5	1562	-128	
	12	4/5	49	-144	
62.5	6	4/5	4688	-121	XTAL
	12	4/5	146	-139	
125	6	4/5	9380	-118	
	12	4/5	293	-136	

Figure 3: Comparison of different bandwidth and spreading factors, showing their effect of the bit rate and sensitivity. [3]

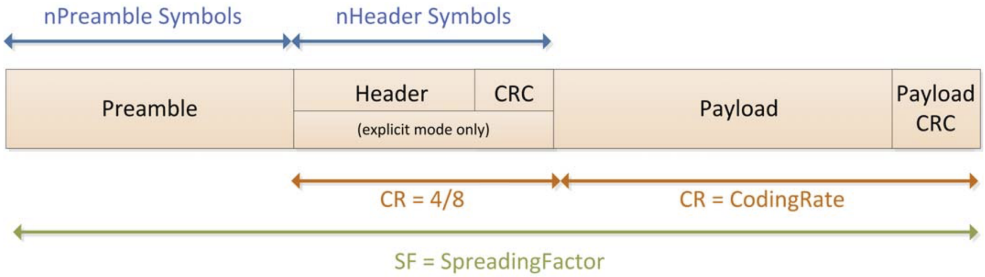


Figure 4: LoRa Packet Structure. [3]

rate, and the declaration of the existence of the payload CRC.

The coding rate is the number of additional bits appended for error checking and correction. LoRa uses cyclic redundancy check (CRC) on both the header and the payload, providing assurance of integrity and error correction. By increasing the coding rate, more bits will be used by the CRC and appended, allowing improved error correction at the cost of additional transmission overhead.

This review has only considered the physical LoRa layer. Above this layer a variety of data link layer protocols can be used such as LoRaWAN, 6LoWPAN or RadioHead. Each different MAC protocol has unique features, supports different topologies and has different use cases. For this project the RadioHead library was used, as it is available for Arduino and provides a simple addressing system for up to 255 nodes. The RadioHead library can also provide star, routed and mesh topologies [21].

3 Project Description

This project can be broken up into three core components, the collar devices, the base station and the cloud software. The hardware of these three components are shown in Figure 5.

Figure 5: The three components of the OpenFence hardware, the collar, base station and web interface.

The software languages and communications protocols are outlined in Figure 6, providing an overview of how the various sections combine together to create the OpenFence system. The in-field devices communicate via 915MHz LoRa and a Raspberry Pi Zero provides a link between the in-field devices and the web software 3G data (WiFi used in testing). The cloud service is made up of the MongoDB database and the NodeJS based server. For this project the server was run locally on machine which was using the website, however this could be installed on an online server and work in the same way.

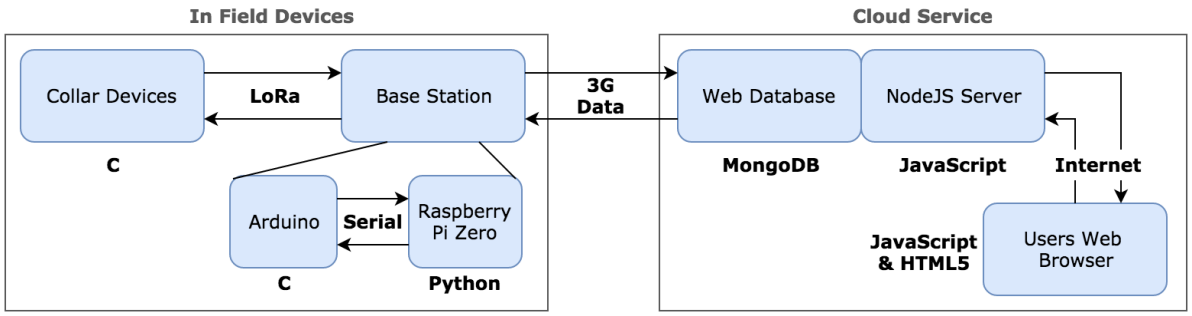


Figure 6: Overview of the software and communications protocols of OpenFence system.

The base station utilises a Rocket Scream LoRa development board (an Arduino based system), connected over serial UART to a Raspberry Pi Zero single board computer. This system was designed to be battery powered, however time constraints did not allow for this. The software running provides a link between the LoRa collar devices and the internet.

The website that was created allows fences to be defined by clicking on a satellite image of the location. The settings used by each individual collar are able to be updated from the website. The website also provides a view in which the logged movements of the livestock can be tracked. The site is built using the MEAN stack, which incorporates MongoDB, Express, AngularJS, and NodeJS to provide a fully functional backend.

3.1 Specifications of Collar

The specifications for the collar device are shown in Table 1.

Specification	Value
Size	150 mm x 370 mm (Collar when flat) 97 x 57 mm (PCB)
Weight	520 grams
Cost	$84 per collar
Current Consumption	3.7mA (min) 36mA (typ) 500mA (max)
Wireless Frequency	915MHz LoRa (Chirp Spread Spectrum) Range of 670m in urban testing
Audio Output	Stereo, up to 78dB
Position Accuracy	$\pm$3m
Features	Directional geofence algorithm Wake on motion - power saving functionality Tilt compensated compass LoRa power optimisation Water resistant enclosure Solar PV charging

Table 1: Specifications of the Collar.

4 Design and Implementation

The process of designing and implementing each of the project stages is detailed in the following sections, breaking the discussion into the three core components. Each of these sections contains discussion of both the hardware and software implementation and the design process.

4.1 Collar

4.1.1 Hardware

The design of the collar electronics began with determining the components required, an extensive review of the many options for each of these components, testing of development modules and finally confirming their selection. Once this was completed the electrical schematic for the collar was created. This was primarily based on the recommended application circuits in the associated datasheets and the development modules. An overview of the circuit can be seen in Figure 7.

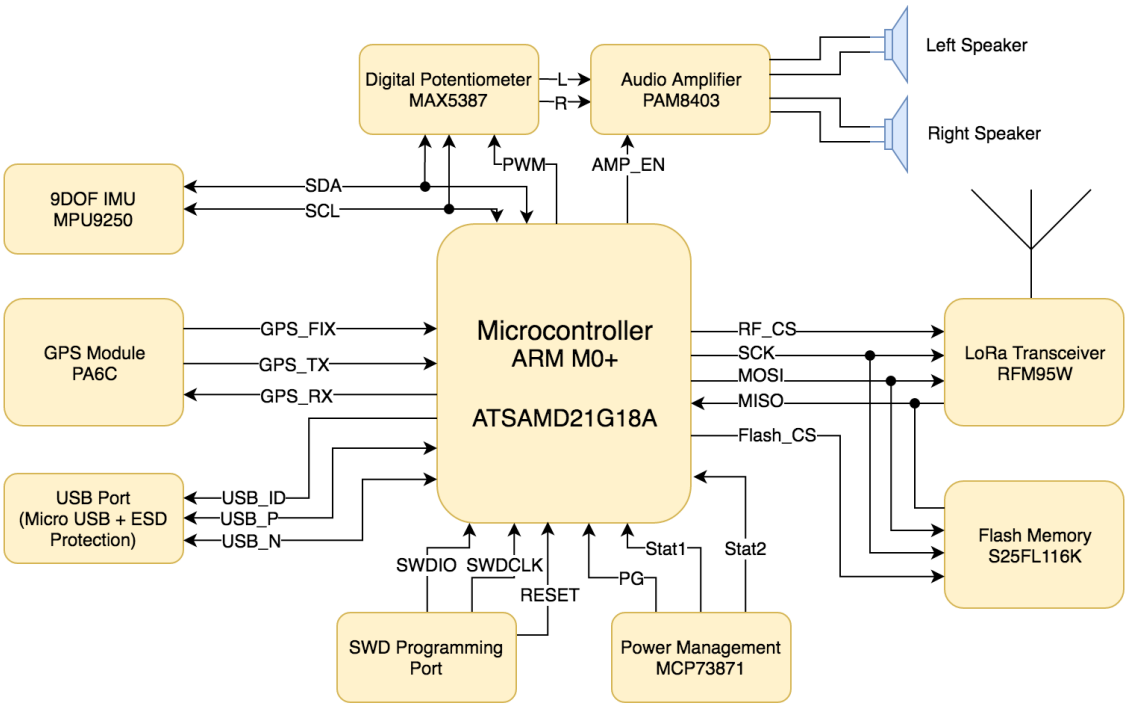
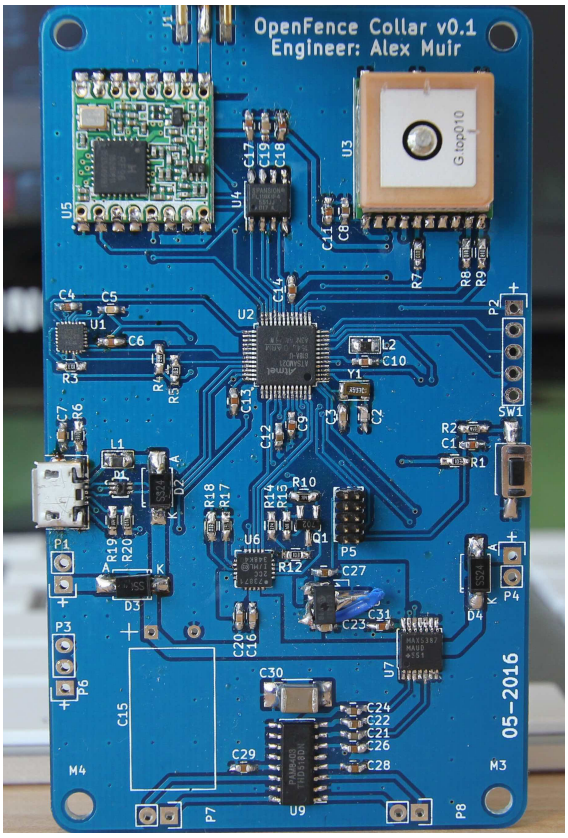


Figure 7: Overview of the digital and analog signals of the collar board.

The software chosen to design the schematic and Printed Circuit Board (PCB) was KiCAD. An open source option was desired to make the project easily accessible to others and meet the design aim of being completely open source. The complete schematic and PCB layout image is available in the appendices and the original KiCAD files can be downloaded from the project GitHub repository.

The design was made to be two layers, however no components were placed on the backside as this would complicate the routing of the design and the final board was small enough for the application, at 97mm by 57mm. The boards were manufactured by PCB Way, a PCB prototyping company based in China, which offers cost effective small run manufacturing. Components were primarily sourced from Element14, AliExpress and the Monash ECSE department supply.

The enclosure was printed by the university's new 3D printing lab.



(a) Assembled OpenFence PCB.

(b) The 3D printed collar.

Figure 8: The hardware created for the OpenFence collar.

4.1.2 Software

The collar software was written using the Arduino language, which is based on C and C++. There was strong support for this language for many of the components used in the collar, with libraries available for the LoRa module, real time clock, GPS, inertial measurement unit, flash memory and serial communication protocols. Using these libraries and the Arduino bootloader adds additional bloat to the software, and could be written as a more streamlined solution by using the Atmel IDE and writing the code for register level interaction.

The total memory used by the collar software when compiled is 48.5KB, which is just 18% of the total 256kB capacity.

4.1.3 Microcontroller

The chosen microcontroller unit (MCU) is the Atmel SAM D21. This microcontroller is based on the low power ARM Cortex M0+, has enough processing power for this project and offers a wide variety of inputs and outputs. I chose the 48pin variant with 256kB of Flash and 32kB of SRAM, this includes 38 I/Os configurable as interrupt inputs, PWM outputs, DAC output, ADC inputs, SERCOMs and many more. The D21 runs at up to 48MHz and has power consumption of 4mA when running, down to 4μA in sleep mode.

It is also an Arduino compatible device so it can use the Arduino Integrated Development Environment (IDE) or the Atmel IDE. The bootloader is downloaded to the chip via the Serial Wire Debug (SWD) interface with a compatible programmer such as the Atmel ICE or Arduino Zero Pro. Once the bootloader has been installed on the device it can be programmed using the native USB interface, with no need for the programmer board.

The external components required on the PCB are minimal with just 8 capacitors, 1 inductor, 1 crystal oscillator, 2 resistors and 1 push button needed. These ensure reliable power supply to the device, provide the clocking source and the reset functionality. In order to program the bootloader the standard 10 pin SWD connector is included. To implement the native USB interface which allows for programming with the Arduino IDE and for serial communications with a host computer, a micro USB port is required. A transient protection module is added between the USB port and the MCU to increase resilience of the board to electrostatic discharge (ESD) events. In order to prevent shielded USB cables from introducing a large amount of noise to the ground plane through their antenna like behaviour, an RC filter is connected between the outer shield and the boards ground as is shown in Figure 9.

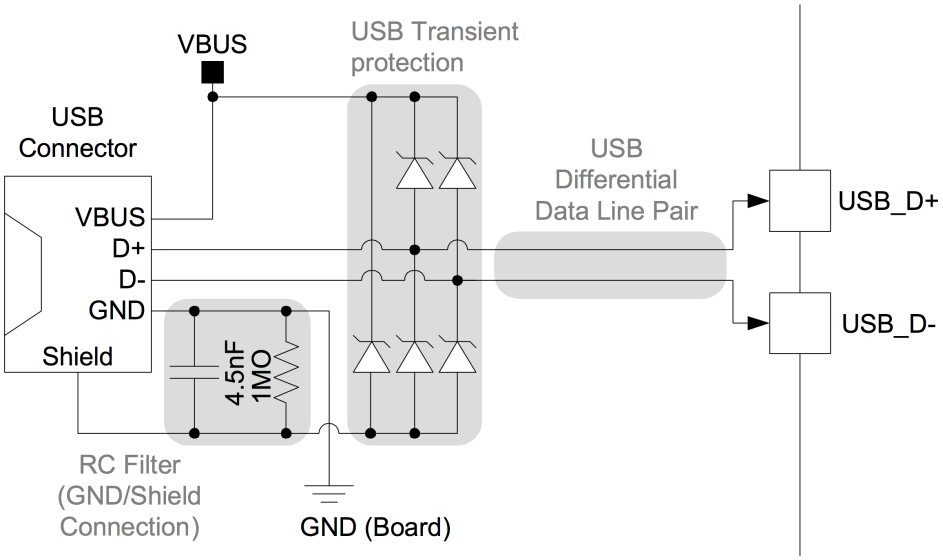


Figure 9: Protected USB Interface Example Schematic [4].

Like many modern microcontrollers, the SAM D21 provides a variety of multiplexing options for each pin, giving the engineer more flexibility when designing their circuits. For example, the auxiliary pins that were brought out to the side of the board for future developments can be used as a digital input, digital output, analog input, PWM output, or as part of an I2S bus or other serial communications. As well as setting the type of pin, pull up or pull down resistances can be enabled. Serial communication interfaces are handled by SERCOMs, of which up to six can be implemented, allowing many protocols to be used including USART, I2C and SPI. The desired multiplexing configuration is set each time the MCU starts up, with any pins that are not explicitly set left uninitialised in a low power state.

The SAM D21 has four different low power states, IDLE 0, 1, 2 and STANDBY, with varying combinations of internal elements disabled in each. Standby mode was implemented for this project to reduce the collars power consumption when processing is not necessary. In standby mode (the lowest power state), all clock sources are disabled and the internal regulator placed into low power mode. The processor can be woken from this state only by an asynchronous interrupt. There were difficulties in getting this to work in a way that an interrupt from the accelerometer would wake the system, eventually it was determined that before sending into standby mode it is necessary to disable the USB interface, otherwise as soon as it goes to sleep, it wakes up, even when not connected to the computer. This issue seems to be common with the SAM D21 as there are a number of forum posts about the situation. The current consumption reduces by 10mA when entering standby, which is discussed in further detail in Section 5.2 on page 32.

4.1.4 GPS

The Global Positioning System (GPS) module selected was the Global Top PA6C. This was chosen due to its excellent price point, power consumption, and the integrated nature of the module. The module is 16mm square and 6.5mm tall, with 13 pins and requires only 5 passive components external to the device.

By using a self-contained module for the GPS, the complexity of the PCB design was significantly reduced. The module is based around the MediaTek MT3339, a chip which has 22 tracking channels, an ARM7 processor and a number of intelligent features which make regaining a location fix faster. The RF section of the module includes the ceramic patch antenna, which feeds into a low noise amplifier (LNA), then a surface acoustic wave (SAW) filter, before connecting to the MT3339. The other components which the MT3339 requires are a 16.368MHz temperature compensated crystal oscillator (TCXO) and 32.768kHz crystal oscillator used for the real time clock of the ARM7 processor.

Communication with the GPS is performed over universal asynchronous receiver/transmitter (UART) with a baud rate of 9600bps. The GPS will communicate to the MCU when it has a new location fix at a user definable frequency. The MCU can alter settings of the module by communicating a command, such as changing the frequency of location outputs (up to 10Hz), the desired output messages or sending the module to sleep.

There are five different National Marine Electronics Association (NMEA) output sentences that can be enabled on the PA6C. The two that are enable for OpenFence are GPRMC and GPGGA, providing the position, time, date, course and fix quality. These sentences are received as comma separated text strings, which must be parsed in order to obtain the individual data quantities. The firmware of the PA6C module can be customized to provide magnetic declination as part of the GPRMC sentence, which would be useful for adjusting the compass (Section 4.1.7) from magnetic north to true north.

Initially parsing the text strings was completed using custom code written as part of this project, but after a more advanced code library, which is compatible with the MT3339, the project moved using the TinyGPS++ library. The key additional feature is checking the checksum attached to each sentence to confirm the message was received intact.

When power is provided to the device it searches for satellites, after 30 to 60 seconds it begins reporting it has a fix (at least 4 satellites being tracked), which can be observed by the 'fix' output pin or read from the transferred sentences. As long as power is maintained to the GPS, it will get a fix more quickly on waking from sleep due to the stored almanac and ephemeris data, the standard times are listed in Table 2. Almanac data contains the orbital patterns of satellites and can aid obtaining a fix, however it is not very precise, this is used in a 'warm' start. Ephemeris data is much more accurate but is only valid for 30 minutes, so if the GPS has had a fix in the last 30 minutes it will be used as part of a 'hot' start [22].

Cold Start	35 seconds
Warm Start	33 seconds
Hot Start	1 second

Table 2: Typical time to first fix of different starting conditions.

The accuracy of the module is stated to be 2.5 to 3 metres, which in the testing of the system seems to be accurate. Positions are reported to six decimal places by the PA6C, providing a minimum precision of 0.11metres [23]. Based on this and the accuracy of 3m, it was decided that 6 decimal places would be used through the system.

4.1.5 Wireless Communications

The wireless communication technology utilised by OpenFence is LoRa as discussed earlier in Section 2.3. This was chosen over other solutions such as ZigBee due to the significant difference in cost, power consumption and range, with LoRa being the superior option in all specifications apart from data throughput. As the amount of information required to be transferred between the base station and nodes is minimal the lower throughput was not such an important factor.

The module chosen to implement LoRa was the HopeRF RFM95W, which contains all of the required radio frequency (RF) components to receive, demodulate and transfer messages to the MCU, and vice versa for transmitting messages. The 16 pin device, requires just two additional bypass capacitors on the 3.3v supply and a connector for attaching an antenna. It communicates with the MCU via the serial peripheral interface bus (SPI) with a chip select (CS) input to alert it when it is being communicated with. The module has a number of outputs, with the DIO0 being used for interrupting the processor and informing it that a new packet has been received.

The selected modulation for this project were chosen be the medium range settings shown in Table 3. The full details on what the modulation settings mean for the LoRa transmission can be found in Section 2.3, on page 5.

Bw = 125 kHz	Cr = 4/5	Sf = 128chips/symbol	CRC on. Medium range.
Bw = 500 kHz	Cr = 4/5	Sf = 128chips/symbol	CRC on. Fast, short range.
Bw = 31.25 kHz	Cr = 4/8	Sf = 512chips/symbol	CRC on. Slow, long range.
Bw = 125 kHz	Cr = 4/8	Sf = 4096chips/symbol	CRC on. Slow, long range.

Table 3: Possible modulation combinations of RadioHead.

The network uses a star topology with the collar nodes only communicating directly with the base station. In order to reduce the power consumption of the nodes they only look to receive packets after they complete a transmission. The star topology was chosen over the option of a mesh network, as it would require the collar nodes to be always monitoring the airwaves and sometimes retransmitting messages, consuming much more energy. Another flaw with using a mesh network for this application is that the position of the animals will be constantly changing, making it difficult for the network to find the most efficient path to a particular end node.

The RadioHead library (GNU GPLv2 open source license) has drivers compatible with the RFM95W, and provides a number of different 'mangers' allowing unreliable, reliable, mesh and routed networks to be created. The reliable manager with ability to transmit and receive acknowledged packets to up to 255 nodes is used for this project. It supports broadcasting to all nodes, and has the ability to classify messages into 16 different categories using 4 bits of flags in the header. The complete message format is shown in Figure 10. It is possible to alter the header to allow more node IDs or message categories, however the standard set up is suitable for this project.

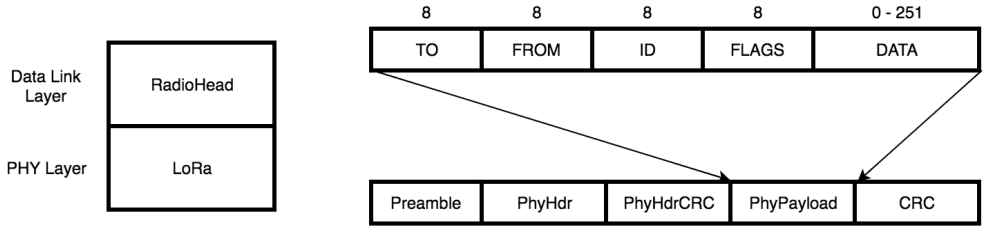


Figure 10: RadioHead and LoRa Network Layers.

There are four types of messages that can be sent between the base station and collar, each with its own unique header flag. The first is a position update from the collar to the base station

and contains the location, time, date and number of alerts and shocks so far today. The second is a fence update from the base station to the collar and contains the fence version, number of corners, and the positions of these corners. Both of these two packets are based around a packet size of 20bytes as can be seen in Figure 11. The third is a settings update designed to be sent from the base station to the node, when the owner changes the settings of that node address. This allows the RFID to be changed remotely, along with setting the thresholds for how often it transmits location data, how much motion wakes it up, turning on testing mode and adjusting the bias values for the magnetometer. The final packet is a magnetometer bias update, which is sent from the collar to the base station after a calibration sequence has been completed, allowing the latest values to be stored in the web database.

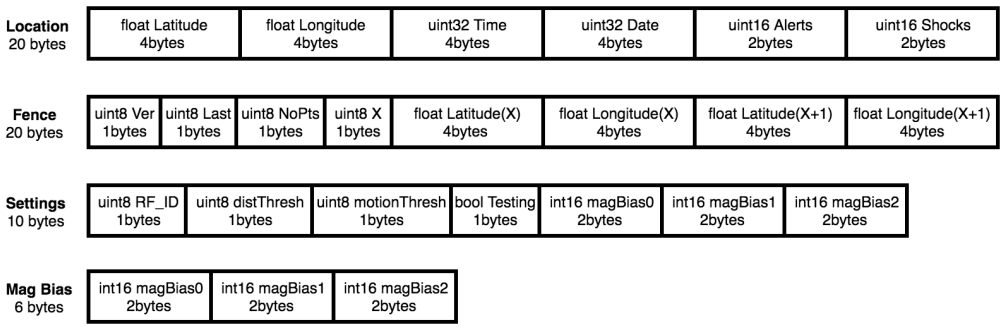


Figure 11: The four different packets used by the OpenFence system.

Each of the packets in Figure 11 are defined as C structures, where each of the values can be set from the latest variables. These structures are then copied into the transmit buffer (array of bytes) using *memcpy* and the size of the structure sets the length of the transmission. When a new packet arrives it is placed in the receive buffer and the header information is extracted. Based off the header flags, the packet type is determined and the appropriate fields are extracted, again based off the set structure of the packets and using the *memcpy* functionality.

Network capacity is limited by the time on air of communications, as if all devices are using the same spreading factor (as they are in this project) only one pair can communicate at a time. Thus by measuring the time to send a packet and receive the acknowledgement, it is possible to determine the maximum number of communications in a second. As is shown in Figure 12, the period of 53ms was measured. Based on this just under 19 communications per second could theoretically be completed. By changing the modulation to use a larger bandwidth, reducing transmission time or by implementing a system where spreading factors are dynamically chosen and using a base station that can separate these messages, parallel communications can be achieved [19].

4.1.6 Audio

In order to produce the varying intensity stereo audio alerts from the collar, an audio source, volume control and amplification are all required. The source is a single pulse width modulated output from the microcontroller, this is sent to the Maxim Integrated MAX5387 digital potentiometer to provide volume control of left and right channels, and then to the Diodes Inc. PAM8403 stereo audio amplifier providing output for the speakers. This can be seen at the top of the diagram in Figure 7 on page 10.

In order to have a waterproof collar, the speakers need to be able to cope with being exposed to water. With this in mind, a small 40mm 0.25Watt speaker with a Mylar diaphragm and plastic casing was chosen. This will not disintegrate like a paper diaphragm would, and will not rust like the more common steel casings on speakers. It also has the added benefit of being lightweight.

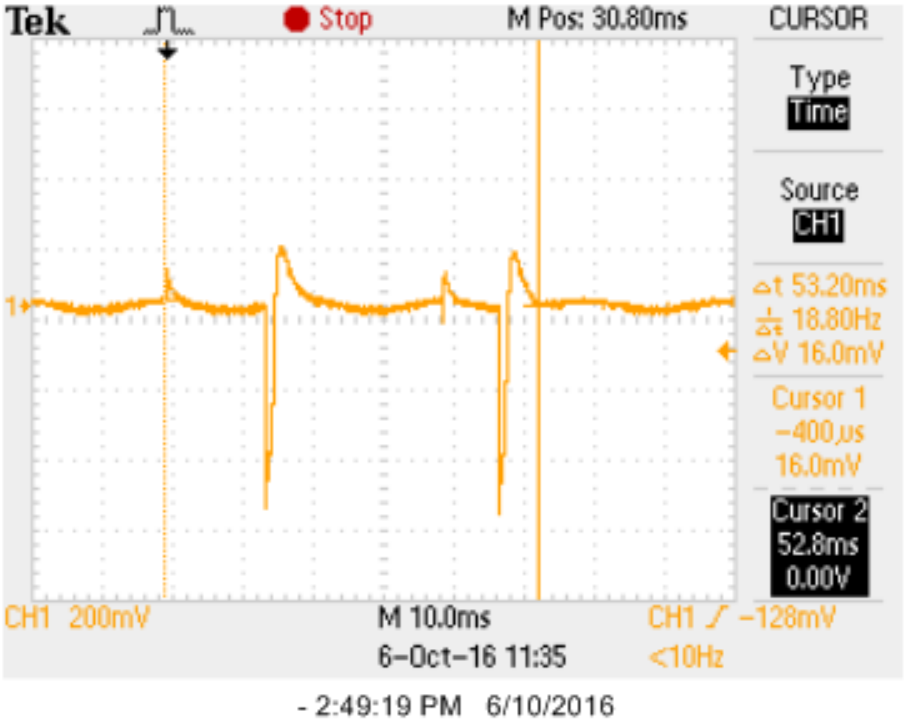


Figure 12: LoRa antenna signal measured on the oscilloscope. The first pair of peaks is associated with the module transmitting and the second pair with receiving.

The original plan for the audio source was to use the 10bit digital to analog converter (DAC), as this would allow more flexibility in the audio that could be produced. Using the DAC, it would be possible to have complex audio waveforms such as a recorded voice, which could be useful when trying to round up animals. However, it found that for producing the simple square wave tone this would require the processor to be interrupted every 1.2 milliseconds to change the output. Compared to using a PWM output this was a significant overhead. Once the frequency and shape of the PWM has been set up it will continue to produce the square wave continuously, and by enabling/disabling the amplifier it is possible to control when this wave is audible. Changing the audio source from the DAC to PWM required changing the pin connected to the digital potentiometer, this was achieved by using one of the auxiliary points and setting the DAC pin to be an input such that the two can never drive the pin at the same time.

Output pulse width modulated tone was set to a frequency of 850Hz as this was found to be successful in the Tiedemann et al. paper [13]. This square wave output can be very loud and is difficult to ignore. Producing this frequency from the PWM output required changing the PWM clock to a slower than standard frequency (1) and then determining the required maximum value of the count register (2).

$$f_{PWMClock} = f_{clock}/Divisor_{clock}/Prescaler_{PWM} = 48MHz/8/8 = 750kHz \quad (1)$$

$$f_{desired} = 850Hz \qquad \therefore Count_{PWM} = 750kHz/850Hz = 882.35 \quad (2)$$

Communicating with the MAX5387 digital potentiometer was done over I^2C. As this was on the same bus as the inertial measurement unit it was necessary to ensure that they had different addresses. By default the address is 0x28 by tying the address set pins to 3.3v for the three least significant bits the address as set to be 0x2F. There are only three commands which can be performed, set wiper A, set wiper B or set both wipers. As it has an 8 bit resolution the value of the wipers can be set to between 0 and 255. In my circuit design, a value of 0 will result the lowest volume, with the loudest at 255.

Audio is measured in decibels, which is a logarithmic unit. Each increase of 10dB represents a perceived doubling of the volume [24]. It is common for audio volumes to be controlled with a 'logarithmic' potentiometer (actual resistance increases exponentially). It was desired to be able to set the volume to a value between 0 and 100% and to have each 10% increase in volume result in a doubling of the volume. This could be achieved using a potentiometer that has a resistive element which scales exponentially. As the digital potentiometer chosen has a linear taper a look up table was created instead. The look up table (LUT) contains 101 values, representing the 0 to 100% scale, with values ranging between 0 and 255, as determined from the rounded values produced by the formula in (4).

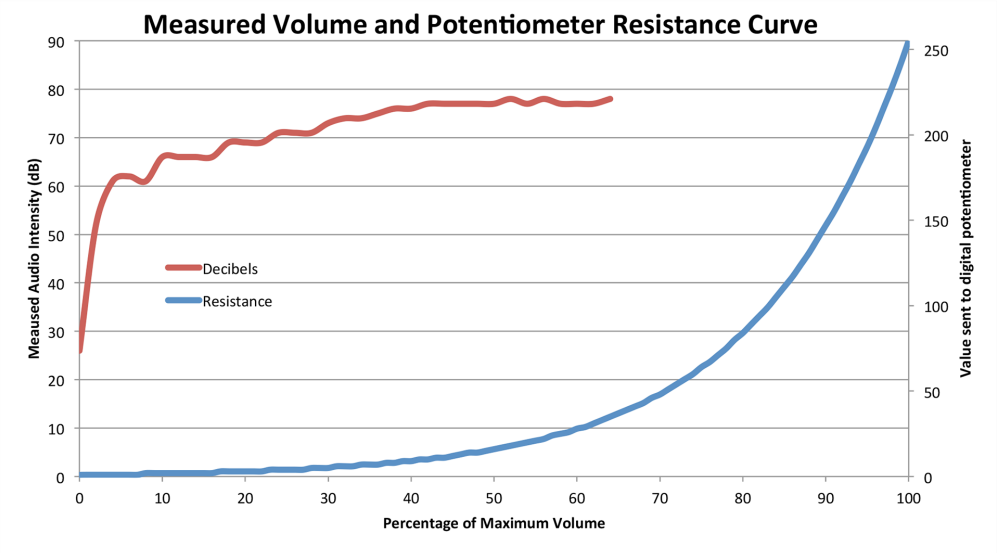


Figure 13: Showing the measured volume and the curve of the wiper position (increasing wiper position results in increased amplitude signal sent to the amplifier).

The resulting testing of the amplifier output is shown in Figure 13. After approximately the 50% point the increase in decibels seems to plateau, this is likely due to the small 0.25 Watt rating of the speakers. In the research papers that listed the maximum audio output used, fall in the range of 87dB [25] to 100dB [12], so the maximum value of the OpenFence system may not be high enough at 78dB. The results in Figure 13 were obtained using a smart phone and would need to be confirmed with a calibrated audio level meter.

$$Finding\ the\ scaling\ factor: \ 255 = 10^{x \times 100} \tag{3}$$

$$Resistance_{uint8\_t} = 10^{0.024065 \times Volume_{0:100}} \tag{4}$$

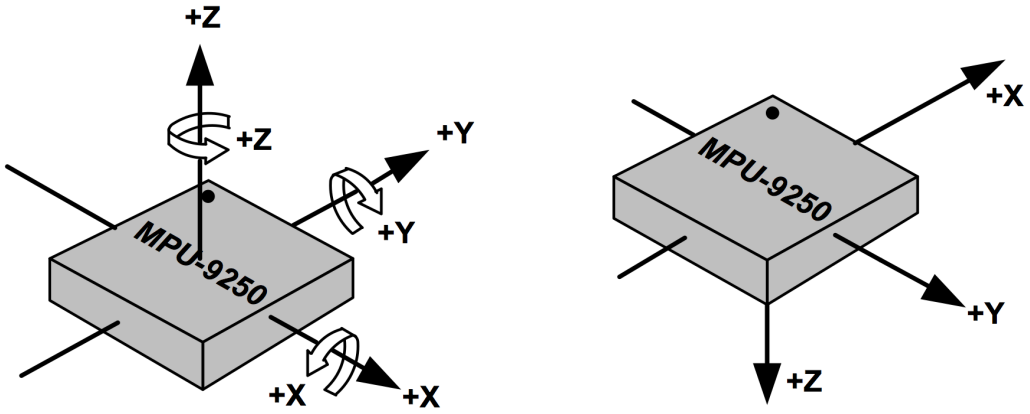
4.1.7 Inertial Measurement Unit

The geofence algorithm requires the compass heading to be available, thus a magnetometer must be included. In order to allow for varying orientation of the collar, a 3 axis (or Degrees of Freedom) magnetometer is required, along with a 3 axis (DOF) accelerometer to determine the orientation. There are many integrated devices available for this task with the Invensense MPU9250 being chosen over other similar sensors such as the Bosch BMX055. The MPU9250 is very popular within the online maker community, which has created a large number of example programs for it. It is available in single units for as little as $4.50 from Chinese manufacturers, as compared to $9.25 for the Bosch device.

The MPU9250 is only available in a QFN package, meaning that there are no leads on the device, and pads for soldering are on the bottom. This requires the use of a reflow oven or the hot

air tool for soldering. It needs just 6 passive components and communicates over the I^2C protocol. The magnetometer is sensitive to ferrous materials and electrical currents, hence it needs to be placed away from other devices and traces, and should not have copper fill underneath it.

Operating the MPU9250 is similar to the MAX5387 in the previous section, it also uses I^2C (address 0x68), and is controlled by addressing registers in the device and reading/writing to these registers. However, in this case due to the more extensive functionality, there are a lot more registers. Simplifying working with this device was Kris Winers MPU9250 library [26], which provided the foundations for controlling the device. The two key functions that were added into this library for OpenFence were the tilt compensated compass and wake on motion functions.



(a) Direction of the accelerometer and Gyro axes.

(b) Direction of the magnetometer axes.

Figure 14: Inside the MPU9250 the direction of the accelerometer and magnetometer axes are not aligned. Sourced from the MPU9250 datasheet [5].

The tilt compensated compass function accounts for any angle that the collar is on to provide the compass bearing relative to the direction of gravity, which is obtained from the accelerometer. The mathematics behind it is just simple trigonometry, however it gets complicated due to the fact that inside the MPU9250 the magnetometer and accelerometer axes are not in the same directions. The direction of the axes are shown in Figure 14.

The first step is to calculate the pitch and roll of the collar, as shown in (5) and (6). The second step is to normalise the magnetometer axes, as shown in (7) and (8). The compass heading is now able to be calculated from the normalised X and Y magnetometer value and adding the magnetic declination for the specific location to correct the magnetic north result to true north as in (9).

$$\text{Roll: } \phi = arctan\left(\frac{-X_a}{Z_a}\right) \tag{5}$$

$$\text{Pitch: } \theta = arctan\left(\frac{Y_a}{-X_a sin(\phi) + Z_a cos(\phi)}\right) \tag{6}$$

$$\text{Normalised X mag: } X_{m,norm} = X_m cos(\theta) + (Y_m sin(\phi) + Z_m cos(\phi)) sin(\theta) \tag{7}$$

$$\text{Normalised Y mag: } Y_{m,norm} = Y_m cos(\phi) - Z_m sin(\phi) \tag{8}$$

$$Heading = arctan\left(\frac{Y_{m,norm}}{X_{m,norm}}\right) + Declination \tag{9}$$

Wake on Motion is a feature built into the MPU9250, which when configured produces an interrupt if the accelerometer detects an acceleration value in the X axis greater than some threshold value (measured in thousandths of a G). This feature was not implemented in the existing library so the function was created based on the datasheet's guideline to the correct settings for the various applicable registers. Placing into this mode also places it into a lower power state, disabling the gyro and reducing the frequency of accelerometer and magnetometer updates. The interrupt is set to wake the MCU from its standby mode, allowing the collar to stay in the low power state until there is motion detected, after which the GPS position can be checked.

4.1.8 Flash Memory

To allow collar settings and fence locations to be recalled in cases where the MCU loses power, a flash memory chip was included, the Spansion S25FL116K is a 2MB chip, with SPI communications and 20 year data retention. It is an 8 pin Small Outline Integrated Package (SOIC) and requires just one bypass capacitor. It is connected to the same SPI bus as the LoRa module, with its own chip select (CS) line allowing the MCU to choose which device it is communicating with.

A software library (SerialFlash) was available for the device with the functionality to create and save files, allowing for simple implementation of the flash memory. Two files were created, one for the fence locations, and one for the collars settings. On start-up of the MCU it will check if the two files exist, if they don't it will create them and store some initial values, otherwise it reads the files and extracts the data to the variables stored in RAM whilst the MCU is running. The files are updated to have the latest values, whenever there is a fence or settings packet from the base station it will overwrite the existing data with this new data.

4.1.9 Power

Powering the collar is an important consideration, as it is desirable to be able to keep them on indefinitely (a number of years is possible). Solar panels are the ideal choice for infield charging, with a rechargeable lithium battery for energy storage and a power management device. The Microchip MCP73871 was the selected component due to its ability to manage a solar input and competitive price when compared to more advanced maximum power point tracking. It comes in a QFN package with 20 pads on the bottom.

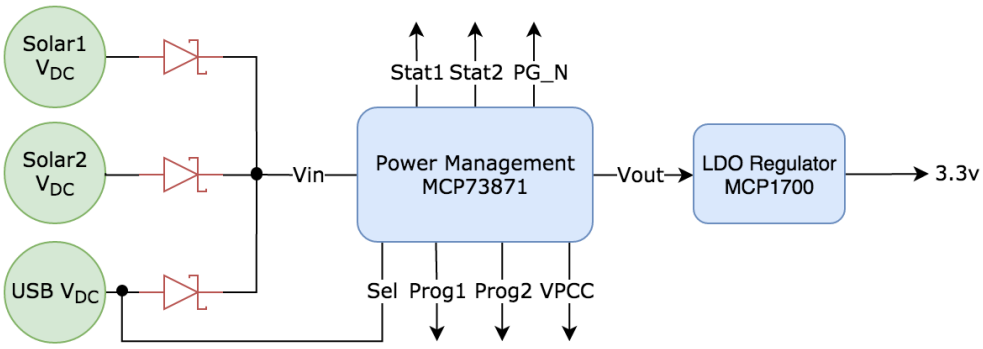


Figure 15: Overview of the power electronics and distribution of the OpenFence collar.

The MCP73871 is configured using a number of resistors, connected to the different pins, PROG 1, 2, 3, Sel, and VPCC. It provides three outputs which are connected to inputs on the MCU, this allows the MCU to observe the state of the power management chip. A simple function was written based on the states in Figure 16, which outputs the state of the device.

The charging method used by the MCP73871 is a constant current/constant voltage (CC/CV) with a selectable final current termination point. The final battery voltage is defined by the model selected, with the 4.2volt version chosen to allow for charging of a lithium ion/polymer battery (which have a full charge voltage of 4.2volts). The current termination point is set by attaching a resistor between the PROG3 pin and ground, the value of which was determined using equation (10).

CHARGE CYCLE STATE	STAT1	STAT2	$\overline{PG}$
Shutdown (V_{DD} = V_{BAT})	Hi-Z	Hi-Z	Hi-Z
Shutdown (V_{DD} = IN)	Hi-Z	Hi-Z	L
Shutdown (CE = L)	Hi-Z	Hi-Z	L
Preconditioning	L	Hi-Z	L
Constant Current	L	Hi-Z	L
Constant Voltage	L	Hi-Z	L
Charge Complete - Standby	Hi-Z	L	L
Temperature Fault	L	L	L
Timer Fault	L	L	L
Low Battery Output	L	Hi-Z	Hi-Z
No Battery Present	Hi-Z	Hi-Z	L
No Input Power Present	Hi-Z	Hi-Z	Hi-Z

Figure 16: Status outputs of MCP73871, sourced from [6].

$$I_{term} = \frac{1000}{R_{PROG3}} = \frac{1000}{50000} = 0.02mA \qquad (10)$$

Voltage proportional charge control is the technique used by the power management device to manage a variable input source such as solar power. The method of operation is that if the input voltage drops below the threshold voltage set at the VPCC pin, it reduces it's charging current. It tries to reach steady state where it is able to maintain the supply to the circuit and keep the voltage above the threshold, with any excess current available being used to charge the battery. The VPCC threshold voltage is set using a voltage divider between the source voltage and ground, with the centre point connected to the VPCC pin. The threshold of the chip is 1.23 volts so the voltage divider must be specified such that it reduces the desired threshold voltage to 1.23 volts, this is calculated using formula (11). The desired threshold was chosen to be 4.5 volts as this would still be a high enough voltage to charge the batteries. A large value of 100kΩ for R_2 was chosen to minimise quiescent current, and equation (12) was solved to give $R_1 \approxeq 270K\Omega$.

$$V_{VPCC} = \left(\frac{R_2}{R_1 + R_2}\right) \times V_{in} \qquad (11)$$

$$1.23V = \left(\frac{100,000}{R_1 + 100,000}\right) \times 4.5V \qquad (12)$$

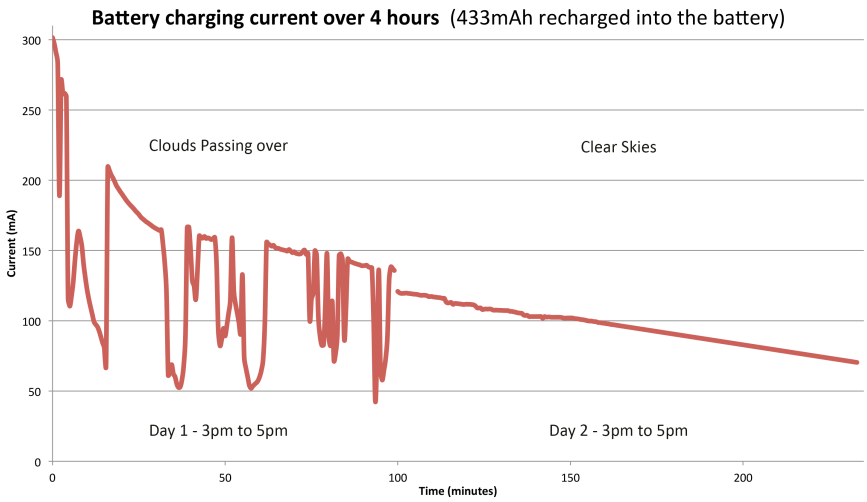
There are three different power sources connecting to the one V_{in} of the MCP73871, the two solar panel inputs and the micro USB port. In order to protect the solar panels from any damaging reverse currents they, along with the USB input, have blocking Schottky diodes. This allows for each of the three sources to be combined with any voltage differences between the sources being prevented from creating reverse currents. Using diodes introduces losses to the power inputs, resulting from the forward voltage drop. Schottky diodes are most often used in these kind of applications as they have a lower forward voltage drop when compared to other diodes, with a value of 0.3 volts measured on the OpenFence board, compared to a common 0.7 volt drop of a standard diode.

The solar panels chosen for the collars are 115mm square and are rated to produce up to 330mA at 6volts (2 Watt). With two of these it can provide enough current to charge the battery, even in sub optimal solar conditions such as overcast days.

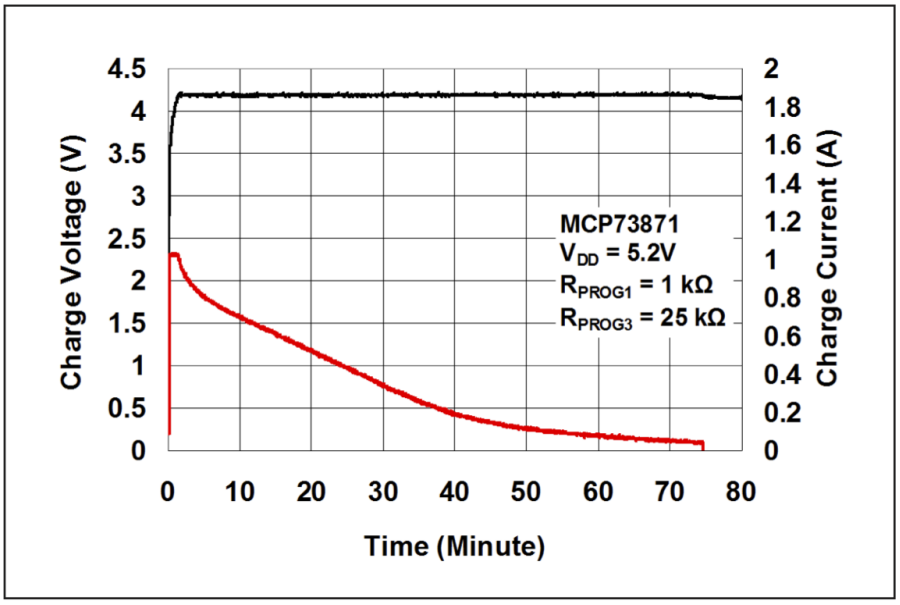
Selecting the 3.3 volt regulator involved determining the maximum current it would need to deliver, and then finding one which exceeded this current whilst aiming for the lowest quiescent current (current consumed by the regulator). The largest drain of current on the board is the audio amplifier, when it is outputting at a high voltage it can draw hundreds of milliamps, however it did not require the drive voltage to be 3.3volts so by feeding this directly from the output of the power management chip, the maximum possible current the regulator would have to supply was

significantly reduced. The Microchip MCP1700 low dropout (LDO) voltage regulator was chosen, with a quiescent current of just 1.6μA and delivering up to 250mA.

Testing the ability of the solar panel to charge the batteries was performed over two days, providing an example of a partly cloudy day and a clear day. The battery was successfully charged from a starting voltage of 3.3 volts to 4 volts. During this period the solar panels were also providing the board with current (sleep mode disabled), and the excess current is used to charge the battery at the rate determined by the MCP73871 based on its CC/CV charge method shown in Figure 17b. By comparing Figure 17a with the expected charge curve, it can be seen that the system follows a similar trajectory.



(a) Plot of the charge current being provided to the Lithium Ion battery. Both days were done in the late afternoon with UV starting at 5 at 3pm and reducing to 1 by 5pm.



(b) Plot of the typical charge curve provided by the MCP73871 datasheet [6].

Figure 17: Charge current plots, comparing testing with the datasheet provided curve.

4.1.10 PCB Design Considerations

Some of the PCB design considerations not already mentioned include the passive components, overall layout considerations and the design of the short antenna waveguide.

The passive component sizes were almost all 0603 (thousandths of an inch) surface mount devices (SMD). This size was chosen due to the cost effectiveness, as it is not so small that the manufacturing techniques become more expensive, and is not costing extra for additional material by being any larger than is necessary. The footprints used in the PCB design were ones designed for hand assembly, meaning they have slightly larger pads on the board and extended clearances when compared to a footprint designed to be used with a reflow oven.

The layout of the board tried to minimize possible interference and noise by placing the RF devices at the top end of the board, the digital components in the middle, and the audio (analog) components at the bottom. The path of the signal wires was also considered to avoid possible capacitive coupling with neighbouring traces. All remaining space on the top and bottom of the board was filled with ground planes, which were stitched together using vias.

The antenna waveguide connects the output of the module to the right angle SMA connector at the top of the board. The distance is 0.5 cm, however it is important to match the impedance of the output and the antenna to ensure the most efficient output and prevent reflections. The coplanar waveguide was determined using Saturn PCB Toolkit. 50 thou with an 8 thou gap to the surrounding ground plane on 1-ounce copper was calculated to provide 50Ω impedance at 915MHz.

4.1.11 Assembly and Debugging

Once the PCBs had been delivered assembly of the boards began. For the first board this was done in stages with testing of each component after soldering it, ensuring that there were good connections and that there were not any issues with the board design. This process took two weeks to complete due to issues which often turned out to be an intermittent connection, especially on the 48 pin MCU and the QFN components.

There were still a couple of issues that were due to the PCB design. The mounting pins on the micro USB connector being about 0.5mm too far apart, to fix this the pins were able to be bent outwards a little such that they could be soldered into the holes and still provide support to the connector.

Another issue found early on was that the regulator pins were reversed on the footprint such that the centre pin which was used as Vout in the design should have been Vin. This was solved by soldering solid core wire to the pads on the board, carefully crossing them and then soldering the regulator on top.

The only other issue with the design was a minor one in which two of the resistors used to set the charge cut-off point and maximum charge rate were reversed on the schematic. All that was required was to switch their placement when assembling the board.

4.1.12 Directional Geofence Algorithm

The core function of the OpenFence collar is to be able to contain livestock within a boundary, which means that it needs to know if the animal is inside the boundary. In order to implement the varying levels of alerts it also needs to determine how far outside the boundary it is.

The fence is a polygon with up to 255 corners (limited only by the choice to use an 8 bit unsigned integer to store the count), with each point being defined by its latitude and longitude, and numbered in a clockwise direction. We also know the lat/lon of the collar device, which is obtained from the GPS module. The geofence algorithm then performs a number of steps, outlined in Figure 18, to determine if the animal is within the boundary, and if outside it, how far outside in order to determine the appropriate alert volume. The directional nature of the algorithm comes when comparing the collars (tilt corrected) compass heading with the calculated bearing perpendicular to the side of the fence that the collar is outside.

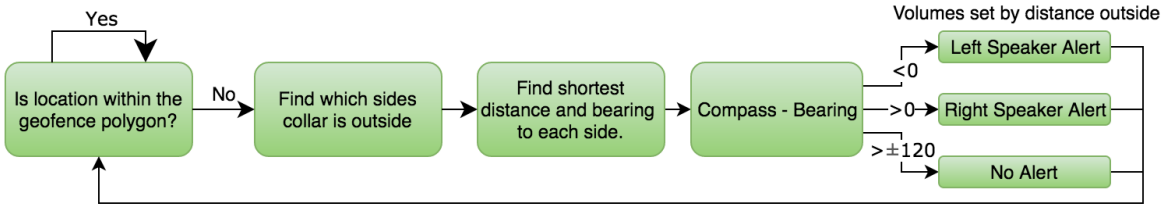
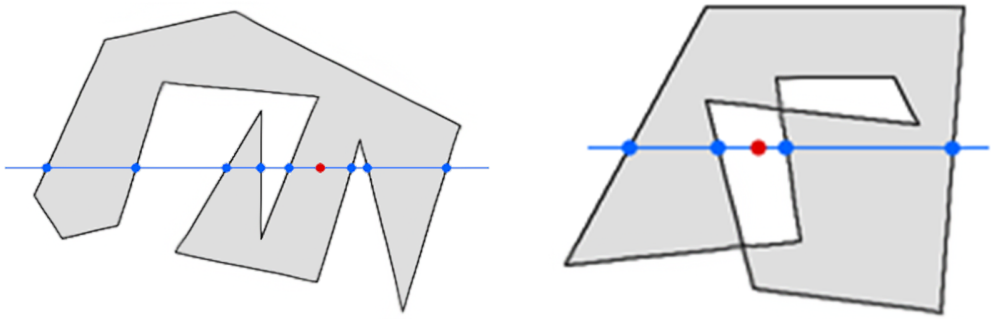


Figure 18: Overview of the directional geofence algorithm used by OpenFence.

The first step of the algorithm is a simple calculation to determine if the collar's latitude and longitude is inside the polygon of points. After researching a number of different ways to find if a point was within a polygon, in a way that used minimal computing power, the method described by D. Finley was used [7]. An example of how this method is works is shown in Figure 19.



(a) Inside the boundary as there are an odd number of nodes on both sides on both side of the test point.

(b) Outside the boundary as there are an even number of nodes on both sides on both side of the test point.

Figure 19: Example of how the Point in Polygon method works. Images sourced from [7].

The next step is only calculated if in the previous step it was determined that the collar was outside the boundary. Calculate the fence normal vectors (Fn) for each side, such that the normal always points towards the inside of the boundary (this is why the fence must be defined in a clock wise fashion). Next find the vector of the animal p to a point on that side of the fence; in this case the end is used for simplicity, although it can be any point along that boundary line. Then use the dot product of this vector and the normal vector to find the value of d. If d is negative then it is outside the boundary on that side, positive it is inside and if d is zero it means it is on the fence line.

$$d = (p - F_p) \bullet F_n \tag{13}$$

Once the sides that the animal is outside has been determined the shortest distance to each side is determined. In order to find the closest point on each side of fence, the animals location is projected onto the segment and the distance between the projected point and the animal calculated. If the animals location does not project onto the segment then the distance to the nearest end of the segment is calculated, as is shown in Figure 21.

The side with the maximum distance is used as the side which the animal is truly outside of, hence it is used for all further calculations. The volume is set proportional to this maximum distance. The bearing from the projected point and the animal is calculated using the formula (14).

$$\theta = arctan\left(\frac{sin(lon_{ani} - lon_{proj}) \times cos(lat_{ani})}{cos(lat_{proj}) \times sin(lat_{ani}) - sin(lat_{proj}) \times cos(lat_{ani}) \times cos(lon_{ani} - lon_{proj})}\right) \tag{14}$$

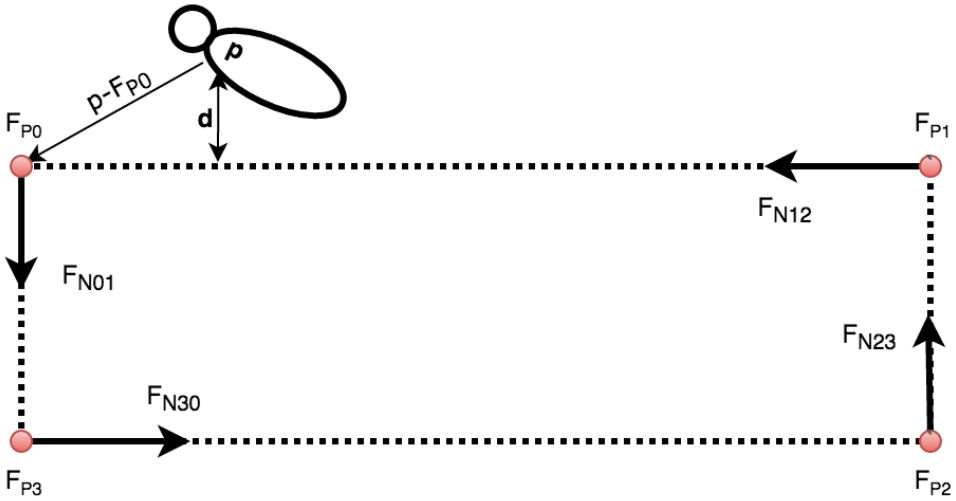


Figure 20: An example of how the geofence calculation (13) is used.

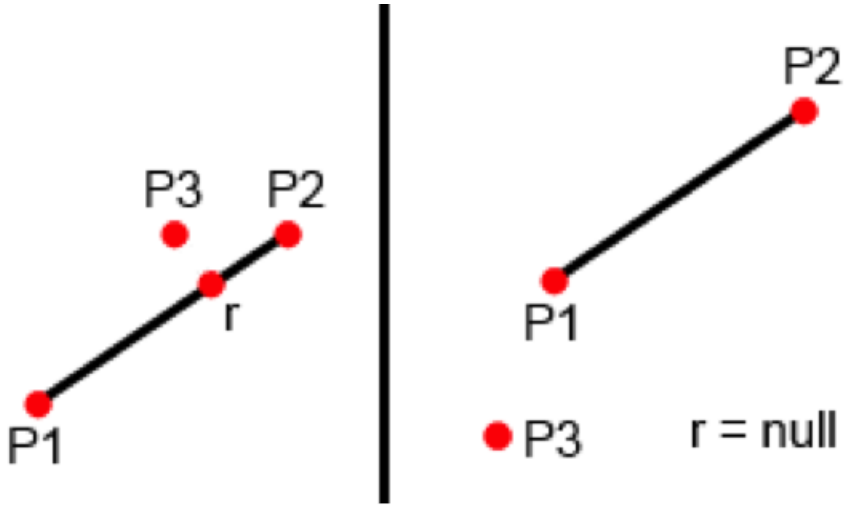


Figure 21: Projecting a point onto a segment. Left shows when the point can be projected, right shows the case when the closest end of the segment is used.

Once the bearing to the fence is known it is added to the compass bearing and based on the result, it determines which speaker to produce the alert from. This directionality ensures that the animal is pushed back inside the fence in the most direct manner, reducing the possibility of confusion. Once the resulting addition is greater than ± 120 deg all alerts are disabled as the animal is now pointing the correct direction to enter back within the boundary as shown in Figure 18 on page 23.

In order to test that these equations worked, the functions were implemented in a desktop C program, which was fed set of fence locations and then trialled a number of different test points. The output values were then compared with the expected result, based on intuition and other sources of this information.

There are various formulas available to find the distance in metres between latitude/longitude points. The differences are summarised in Table 4, with some examples of their accuracy over various distances. Whilst Haversine is not perfectly accurate (it assumes the earth is a perfect sphere), it has been used as the baseline for this comparison, as it is the most accurate without using very complex calculations. As can be seen for distances less than 1 km Equirectangular Approximation was slightly more accurate and requires much less intensive calculations. For this application it was chosen to be the best choice.

Name	Haversine	Spherical Law Of Cosines	Equirectangular Approx.
Square Roots	2	0	**1**
Trig Calculations	7	6	**1**
Accuracy of 100m	-	1E-5 %	**1E-10%**
Accuracy of 1km	-	2E-7 %	**7E-8 %**
Accuracy of 10km	-	**5E-10 %**	1E-5 %
Accuracy of 1000km	-	**∼0%**	0.02%

Table 4: Comparison of three different distance formulas. The accuracy at measuring distances of various scales is compared.

4.1.13 Intelligent Power Saving

In order to reduce the power consumption of the collar electronics two core features were used. There are the wake on motion function and the intelligent LoRa output power scaling.

Many of the components used by the collar have low power states which can be enabled by the MCU, reducing the systems power usage when though components are not required. In order to send the MCU to sleep as well there needs to be a signal which wakes it up when it is time to perform more processing. Often this is done with a timer interrupt such that it is woken at regular intervals. However for this project due to the inclusion of the MPU9250 inertial measurement unit, the MCU is configured to wake when there is motion. The IMU generates an interrupt when there is motion above the configured threshold. The complete power saving logic is shown in Figure 22.

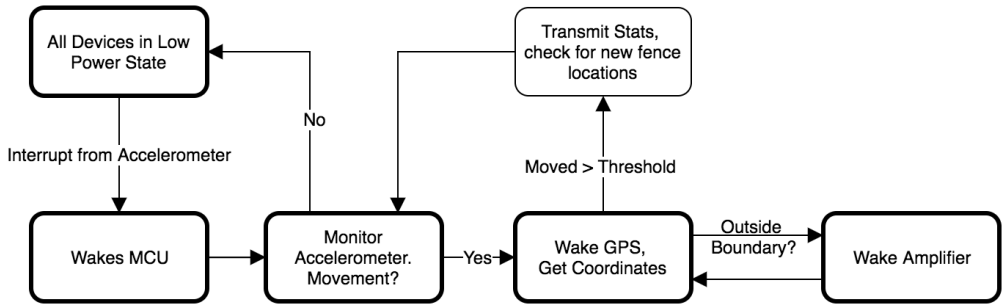


Figure 22: Flowchart of the power saving function OpenFence collar.

The intelligent LoRa output power scaling works by aiming to use the minimum required output power that successfully transfers to the base station. Due to the reliable nature of the RadioHead communications manager being used, every addressed packet sent expects to receive an acknowledgement (ACK) of receipt. If it does not receive an ACK it tries twice more before returning an error.

The power scaling mechanism was designed based on RadioHead returning an error if the packet transfer was unsuccessful. It begins with the lowest output power, 5dBm, and attempts to send. If it is unsuccessful the power is increased by 5dBm until the packet it transferred. The next time it sends a packet though, it does not start with 5dBm but tries the last successful power level minus 5dBm. This means that it will still reduce the power if possible, but it will also not need to progress it's way through attempting to send at all lower power levels first. As can be seen in Figure 23 the current used by the LoRa module increases considerably so using the minimum power level will be beneficial to power consumption, as well as reducing the impact on neighbouring RF users.

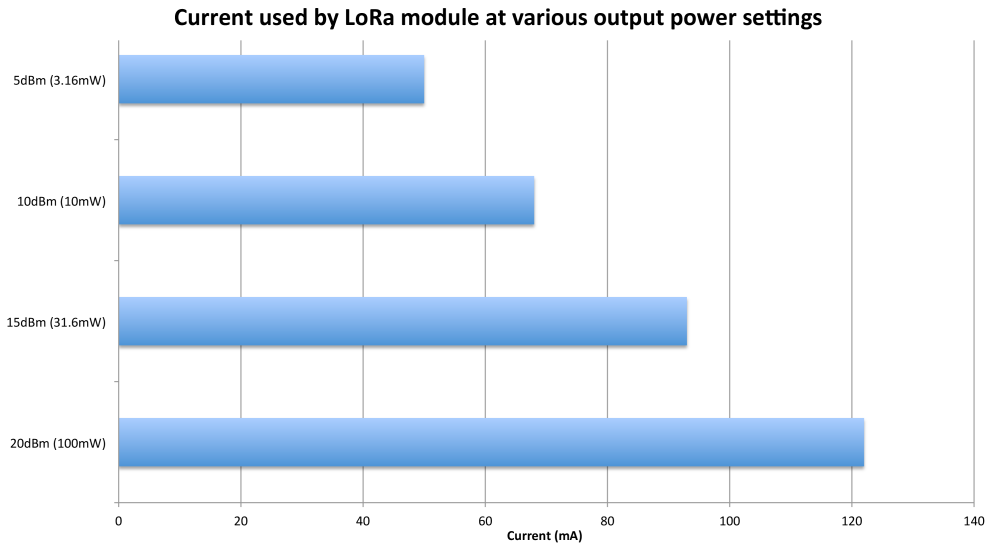


Figure 23: Current draw of the LoRa module (the boards current when the LoRa module is in sleep mode was subtracted) for different output power levels.

4.1.14 Enclosure

The collar enclosure was designed in Solidworks as four components that fit together. The two sides hold the solar panels and speakers, with the wires from each running in channels underneath the solar panel and into the hollowed out hinge. This method of running the wires through the hinge allows for a water resistant entry for the wires into the main compartment. The centre section is made up of two parts, the base which has the other side of the hinges on it, and the lid which has a hole for the antenna. In order to create a water resistant seal between the base and lid, a small recess and lip was placed where the two pieces meet where a rubber seal could be placed.

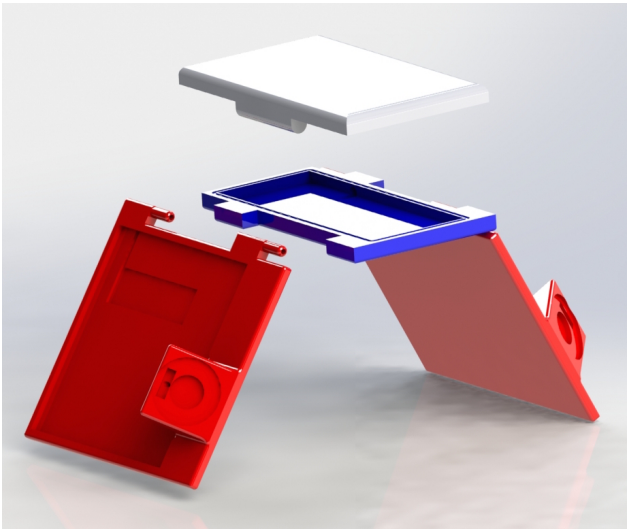


Figure 24: Collar design in Solidworks, showing the wire channels.

The original plan was for there to be a separate battery enclosure 3D printed as well, however for the testing required for this year, it was deemed unnecessary and so the battery was placed in the main compartment along with the PCB. This did have the drawback of the design being more top heavy than intended.

4.1.15 Bill of Materials

The total cost for materials for each collar is outline below.

Part No.	Description	Qty	Cost ($AUD)
ATSAMD21G18A	Arm M0+ Microcontroller	1	$7.46
MPU9250	9DOF Inertial Measurement Unit	1	$3.27
PA6C	GPS Module	1	$12.28
RFM95W	LoRa Transceiver Module	1	$13.40
S25FL116K	2MB Serial Flash Chip	1	$0.67
MCP73871	Battery charge controller	1	$2.96
MCP1700T	3.3 volt LDO regulator	1	$0.72
PAM8403	Stereo Audio Amplifier	1	$0.30
MAX5387	Dual Channel 50KΩ Digital Potentiometer	1	$1.71
FC-135	32.768kHz Crystall Oscillator	1	$2.90
FCI-10118193	Micro USB 2.0 Type B	1	$0.47
USBLC6-2P6	USB ESD Protection Device	1	$0.46
Right Angle SMA	SMA Antenna Connector	1	$1.16
Button	SMD Micro Switch 3*6*4.3H mm	1	$0.03
Passives	0603 Resistors, Capacitors, Inductors	49	$1.20
Diodes	2A Schottky Diode	3	$0.18
PCB	Custom Manufactured PCB	1	$2.36
AS3004	40mm 0.25W Mylar Speakers	2	$3.50
Solar	6V 2W Solar Panels	2	$7.00
Battery	2200mAh Lithium Ion Battery	1	$8.00
Enclosure	3D Printed Enclosure	1	$10.00
Strap	Cord Material & Buckle	1	$4.00
	Total:	72	$84.03

Table 5: Bill of materials for the collar.

4.2 Base Station

4.3 Hardware

The base station consists of a laser cut acrylic panel with brackets to connect to the upright pole, and the enclosure which contains the electronics. Solar panels are attached to the front of the acrylic panel, of which the angle can be changed when it is set up in the field to allow for maximum solar efficiency.

The electronics utilised include a Rocket Scream LoRa development board (an Arduino based system), connected over serial UART to a Raspberry Pi Zero single board computer. The development board has a Lithium battery charge controller which uses the solar input to charge the base stations battery. In order for the Raspberry Pi to operate USB devices such as a 3G data modem, it needs the battery voltage (varying between 3.3v and 4.2v) to be stepped up to 5 volts. To do this a boost converter and voltage regulator are paired together to produce a consistent 5 volt output.

The LoRa antenna is placed on top of the upright pole, providing the maximum height available. To connect from the SMA connector on the development board to the antenna a 50Ω coaxial extension is used.

Due to difficulties in getting the power electronics to work in time for the finalisation of the project, the boost converter and battery were removed. Instead the system is being powered via an external USB power supply.

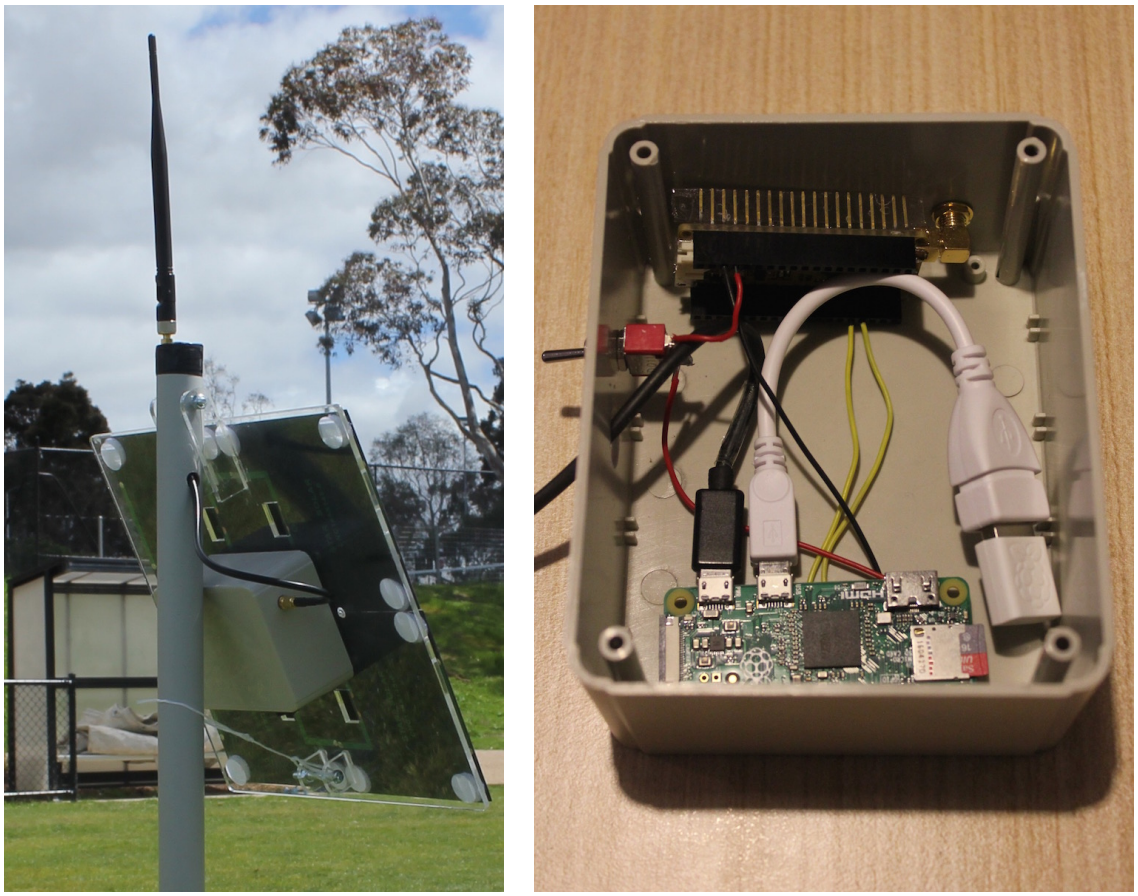


Figure 25: Showing the back of the base station and the electronics inside the enclosure.

4.3.1 LoRa Development Board Code

The code running on the Arduino is primarily just relaying messages received over LoRa to the serial port. The LoRa module is configured to always be listening for incoming messages from the nodes. When it receives a message it checks it's stored list of collar IDs to see if that collar has any changes to the settings that need to be transferred. It also checks if the fence version on the collar is the latest, and if it also needs updating it will send as well. Transferring a received LoRa packet to serial involves attaching the message flag and the sender ID to the received buffer of data.

Serial data sent to the Arduino from the Raspberry Pi is placed into two arrays of C structures. The first one is the fence locations; as the serial data received from the python script is structured in the same way as the packets that are sent to the nodes, the fence packets are stored as they are received. This overwrites the previous version of the fence. For the settings updates, there is an array of 256 settings structures, one for each possible node ID. These have a boolean flag which when true means that the settings were updated and therefore the next time a packet from that node is received it will transfer the updated values.

4.3.2 Python Code

The Raspberry Pi program is written in Python as it has the ability to both communicate with a serial device and connect to a MongoDB database. The program waits for messages to arrive over the serial port. When it receives a message it extracts the information from the struct and transfers the information into a JSON (JavaScript Object Notation) formatted string which is then inserted into the database.

Every 30 seconds (or any desired frequency) the Python script checks the database for changes

to the fence version, or for any collar settings which have the updated flag set to true. If it finds either of these the updated JSON is downloaded and placed into the appropriate structure for transferring to the Arduino over the serial port.

4.4 Web Interface

In order to make the system much more user friendly a website was developed to manage the collars. This allows fence positions to be updated easily using a clickable satellite image of the ground, and allowing complex shapes to be created, with polygons of up to 256 points. The ability to view logged location is overlaid on the map, and the tally of alerts received recently can be viewed. The settings of individual collars can be updated as well.

The web server is based on the MEAN stack, which incorporates MongoDB, Express, AngularJS, and NodeJS to provide a fully functional backend that can store information in a database, serve web pages and provide single language site development. The NodeJS server runs on the users computer, and serves the website when it is requested.

The MongoDB database is hosted online by MLabs which provides free MongoDB databases up to 500MB in size, which is more than enough for the usage of this project. The database contains three collections, the list of animals, the list of logged animal locations and the list of fence locations.

In order for the database to communicate to the base station, a number of fields are used to indicate updates. For an update to the fence locations, the fence version increments, which is noticed by the base station, and thus it updates to the latest version. For a collar settings update, each database entry contains a boolean field which is set when the settings have been updated, then once the base station has downloaded these new settings it sets the field in the database to false.

The website is written using HTML5 and CSS, as well as the JavaScript. The Google Maps JavaScript API was used to embed and control the satellite map.

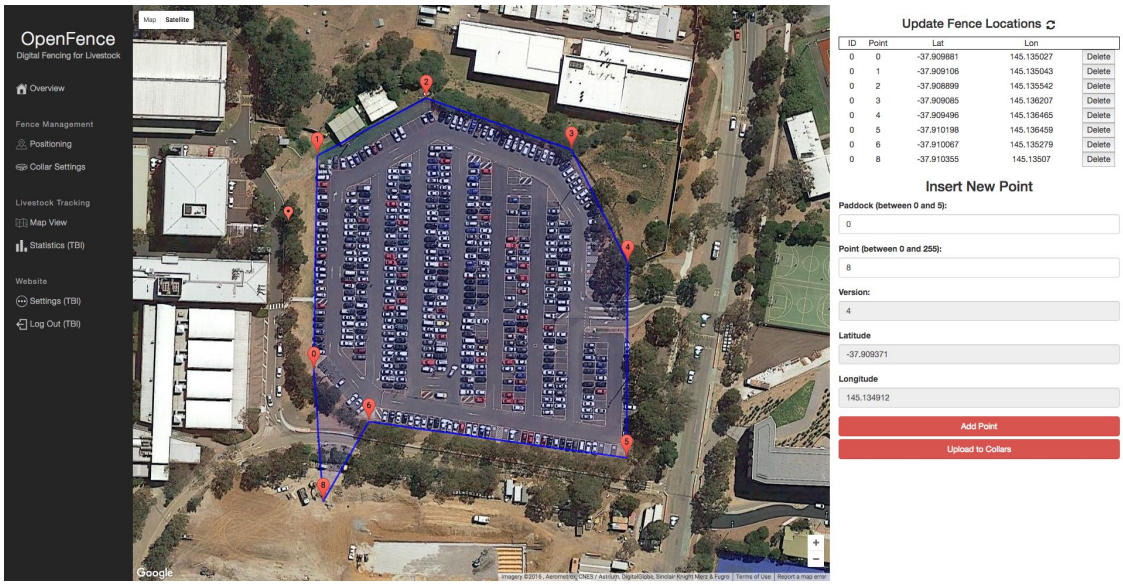


Figure 26: Fence creation page.

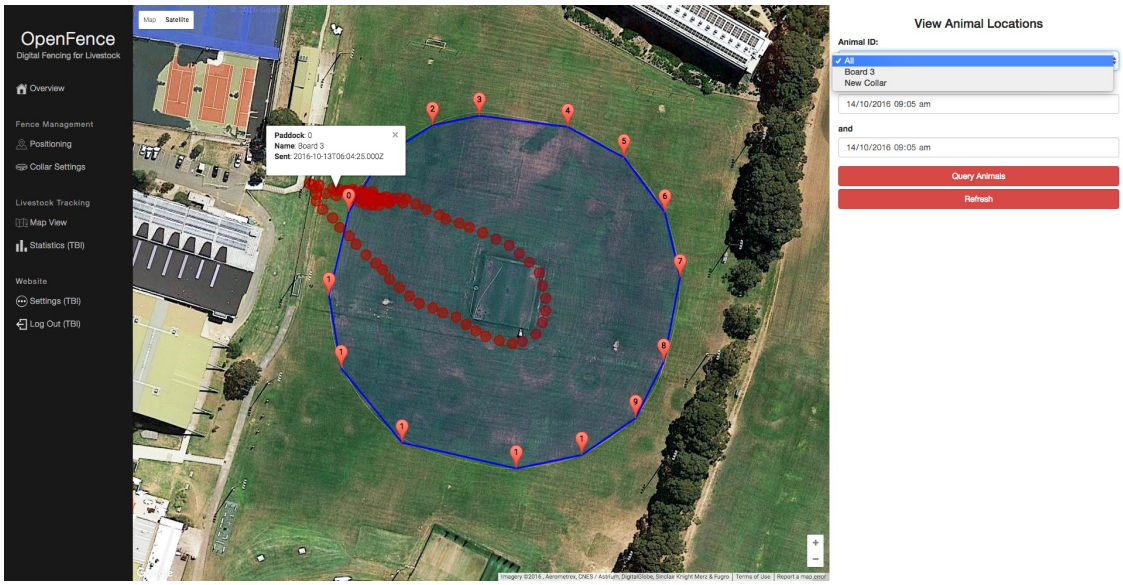


Figure 27: Animal tracking page.

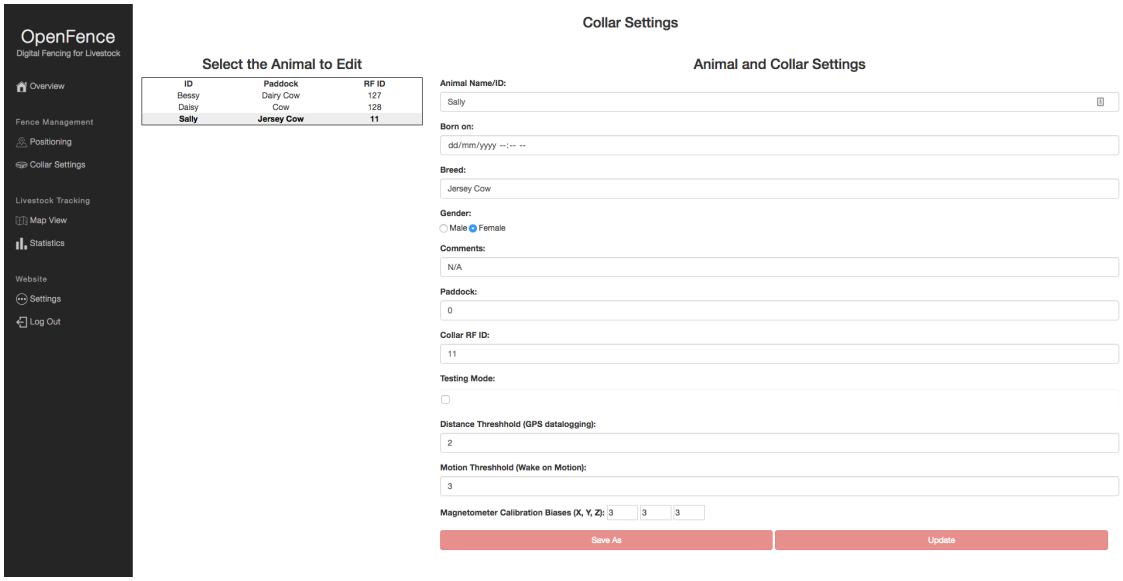


Figure 28: Collar settings page.

5 Analysis and Results

5.1 Animal Testing

Though it was not expected that it would be possible to do any on animal testing, thanks to the interest of Mooroopna dairy farmer Jeff Odgers a simple trial was performed. This allowed for the fit of the collar to be trialled, as well as to observe the reaction of a cow to having the collar around its neck. The response to audio alerts were not tested, to avoid any unnecessary stress to the animal in case of an adverse reaction.

A cow was selected, rounded up and placed in the cattle crush, which holds the head still and provides access to their side and lower neck. This was where the first issue became apparent as the location of the bars which keep the cows head still were in the spot where the collar would ideally sit. The collar was therefore attached lower and the cows head released so that it could be moved up the neck and tightened. This is not an ideal solution as you have less control over the animal at

this point, and doing this for a large number of animals would be unnecessarily time consuming.

The second issue is the shape of the collar, with its flat top, it did not match the shape of the cows neck. The cows neck come to quite a sharp point at the top and is closer to being a triangular shape than rectangular. The circumference of the neck was about 85cm and the adjustable buckle and webbing al- lowed for a good tight fit. Due to the incorrect shape and top heavy nature of the collar, when the cow was released to walk around the yard, the collar rotated until it was upside down. This was a known flaw in the final design, with the original plan including battery at the bottom of the collar in a separate enclosure to balance the weight. In the interest of simplicity and time it was moved to be in the enclosure with the PCB.

Figure 29: Cow in the cattle crush.

The cow initially tried to lift its front leg up to thecollar when it was first let out, however it soon realised that it could not reach it and gave up. After the initial settling time the cow did not seem worried about having the (upside down) collar around its neck for the 20 minutes that it was on. The 3D printed enclosure was reasonably robust and suffered no damage during its time on the cow. One of the solar panels fell out as it had not been permanently glued in and was only friction fit, resulting in some scratching. Protecting the solar panels is something which should be considered in a further design as being on the side of the animal it is likely to brush up against objects, potentially damaging it and reducing its efficiency.

Whilst the collar was on it began raining quite steadily, which provided a good test for the waterproofness of the collar. The collar was around the cows neck, upside down, in the rain for about 5 minutes, however once it was taken off and opened there was no sign of any moisture within the section containing the battery and PCB.

Figure 30: Showing the collar rotated due to the top heavy nature of the design.

5.2 Power Testing

Power consumption of the collar is an important factor as it would be limiting if the collars had to be removed for charging. Thus keeping power use to a minimum will allow for the board to operate on battery power for a number of days even if there is limited solar charging.

The estimated current consumption based on the components datasheets is shown in Table 6. The measured current of the board is showing in Figure 31. Comparing the two it can be seen that they are quite similar, with the estimated average current of 33.6mA being only 3mA lower than the measured value of 36mA with everything running. Putting the GPS to sleep provided the largest reduction in current consumption, with a 17mA reduction which is similar to the expected 20mA. Sending LoRa into its sleep state from idle state saves the expected 2mA. Placing the IMU into a lower power state, which disables the gyro and checks the accelerometer and magnetometer less frequently, reduced the current by 3mA. This 3mA reduction as opposed to the 0.7mA reduction expected if it was in the datasheets 'average' state could explain the difference in the estimated and measured average currents.

	Peak	Average	Standby
MCU	7mA	2mA	0.02uA
GPS	25mA	20mA	7uA
LoRa	120mA	1.6mA	0.2uA
IMU	3.7mA	730uA	8uA
Flash Storage	20mA	15uA	2uA
Digital Potentiometer	-	-	1uA
Audio Amplifier	300mA	9mA	1uA
Charge Controller	3.75mA	260uA	28uA
Voltage Regulator	4uA	1.6uA	-
Total	480mA	33.6mA	47.2uA

Table 6: Estimated Power Consumption

The measured standby current is significantly higher than the estimated standby current at 4mA compared to 68μA. Testing has determined that the following are not responsible, the Schottky diodes, pull up resistors, voltage dividers and the voltage regulator. Further investigation would be necessary to determine the source of the additional losses.

Determining the amount of energy required to operate the collar for the entire day is difficult without an extended trial of the device on livestock, as there are many variables that affect the energy use. In order to calculate a conservative daily consumption of power some estimations were used. The first is that cows spend 14 hours of the day lying down [27], and thus the remaining 10 must be spent moving around, grazing etc. It was assumed that during those 10 hours the collar is never able to enter sleep mode, making it an upper value for the energy used. Another conservative estimate used is that the collar will spend 10% of the 14 hours that the cow is lying down, checking the GPS due to motion causing the system to wake up. A distance travelled of 1.5km in a day allows the determination that 750 packets that will be sent to the base station, based on a threshold distance of 2m. Using these estimates and the measured current consumption values in Figure 31, a possible daily energy consumption breakdown was created and is shown in Figure 32.

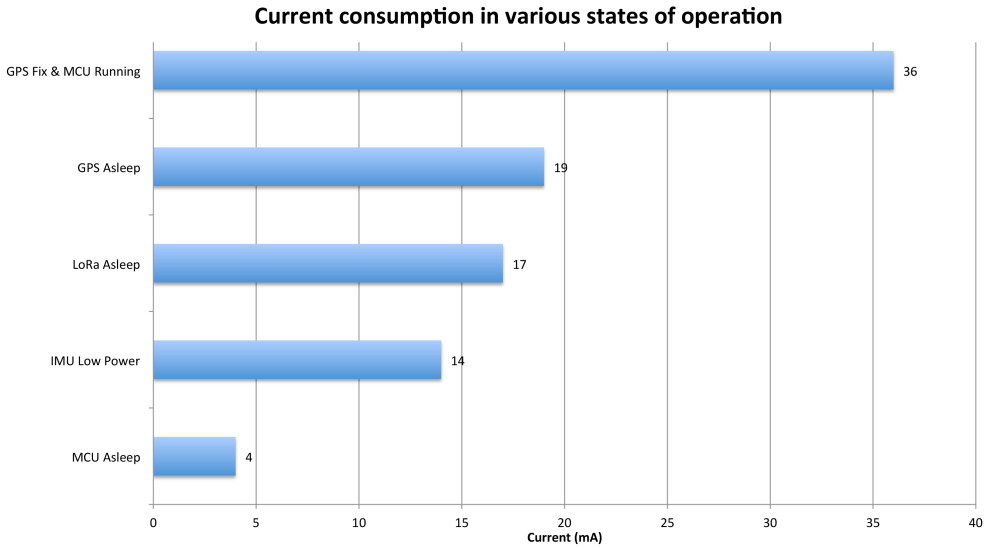


Figure 31: The current consumption of the board, showing the reduction as components are put into their low power modes.

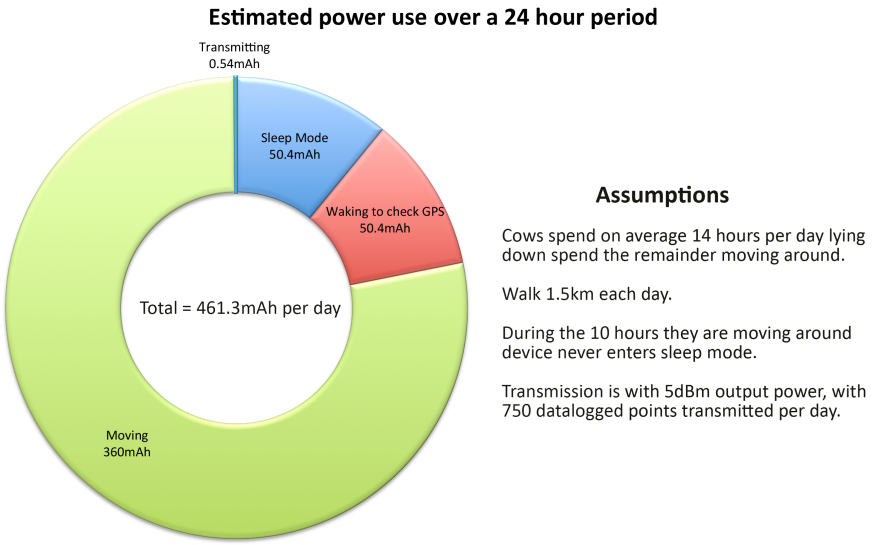


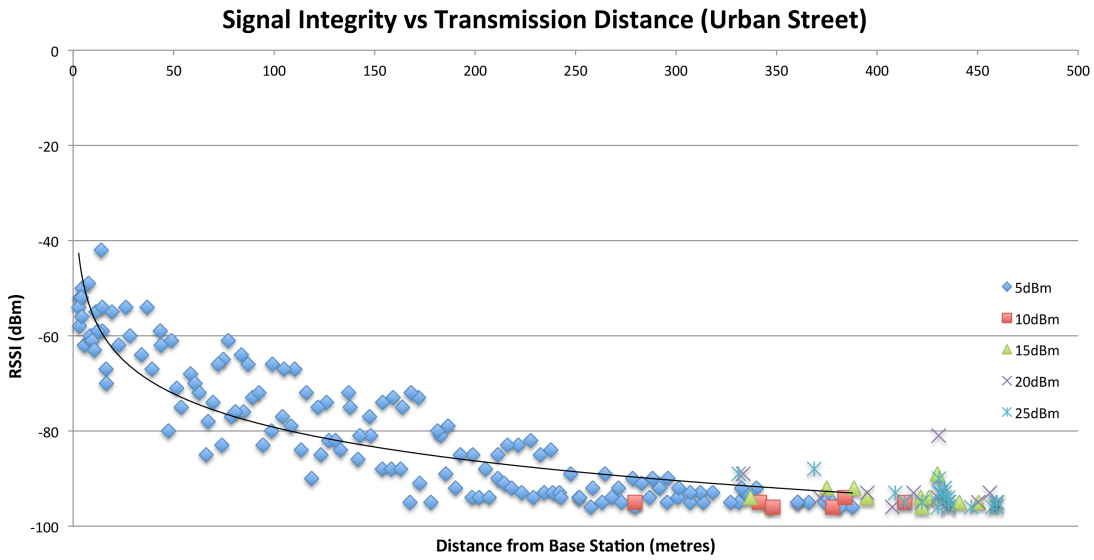
Figure 32: Estimate of the power consumption over an average day.

5.3 Range Testing

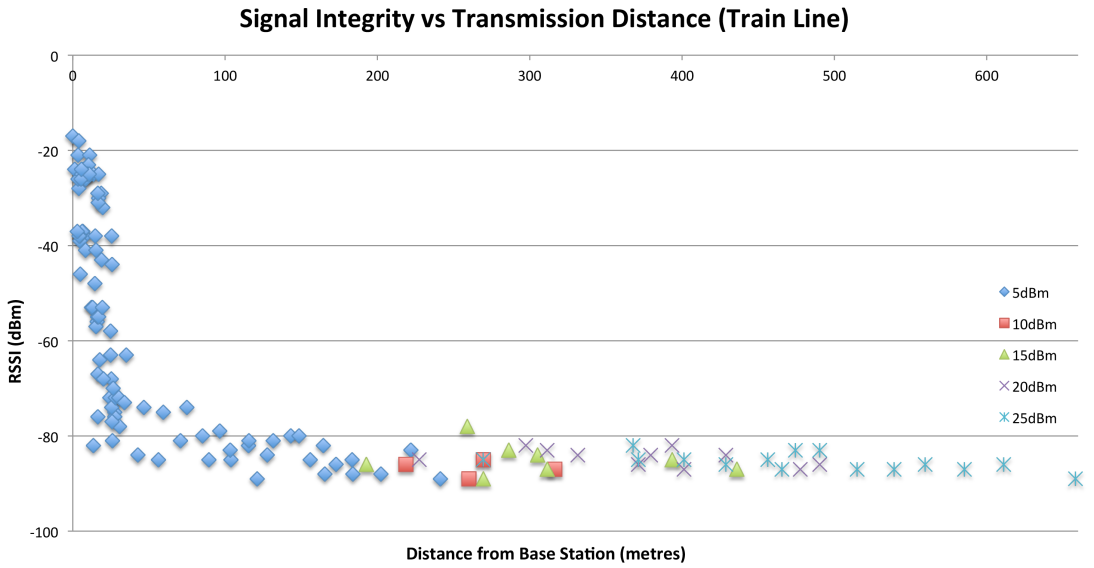
Range testing of the LoRa communications was performed in an urban environment as there was limited access to a nearby space that was large enough to effectively test the line of sight range of the modules. Two different locations were trialled, one on a suburban street and the other all the bike path beside a train line. Antennas used were a 5dBi gain antenna on base station and a 3dBi on the node.

The range achieved was quite good, although it is hoped that it would be further in a clearer environment. With a maximum range of 460m on the street and 670m beside the train line. Changing the modulation could improve distance at the cost of transmission duration (resulting in congestion).

The two figures below show the minimum output power algorithm working as the distance increases.



(a) Maximum range of 460m on an urban street, with buildings and trees lining the edges.



(b) Maximum range of 670m along a train line, better line of sight, however there was an overpass at about 350m.

Figure 33: Received signal integrity (RSSI) compared to the distance away from the base station. The legend shows the output power which was used to transmit each packet.

6 Discussion

6.1 Leaky Fences

Due to the accuracy of the GPS location data, the fence boundaries become somewhat variable. This is a factor that would have to be accepted by the grazier as a quirk of the system. This could result in areas along the boundary being inconsistently grazed. When the scale of a paddock is 100m square this $\pm$3m variability at the boundaries is not as obvious an issue. Using satellite images to place the fences is another possible source for inaccurate boundaries, as they have varying levels of accuracy depending on the location.

It is possible to improve the accuracy of the GPS position with higher quality receivers, with the addition of differential GPS (DGPS) in areas which have coverage, or by using a module with support for concurrent GPS/GLONASS tracking.

6.2 Electric Shocks

Based on the research articles on the topic of virtual fencing it appears that the best way to effectively contain livestock within a boundary is to use an electric shock as negative reinforcement in combination with the audio alert. As was discussed in Section 2.2 after only a day or two of training with the system livestock learn what the audio alert means and will remain within the boundary with only the audio feedback.

As I have no training in animal behaviour, I felt that it would be inappropriate to implement an electric shock in any testing performed on animals. With the advice of trained professionals it could be possible to design an appropriate system, however there are many regulatory requirements to meet with any animal testing as well.

6.3 Security

Security is an area which would need to be improved upon in a commercial system. Wireless communications between collar nodes and the base station are not encrypted, and could therefore be intercepted by anyone with a LoRa receiver. Additionally, as there is no authentication that the received message is from the owner, a nefarious person could change settings, move fences or even disable the system.

The website would need to have a login, as well as segregation of different user data. For this project only one user was required so this was not investigated.

6.4 Floating vs Fixed Point

The geofence algorithm uses calculations that have many decimal places, which are stored as floating point numbers. However as the Atmel SAM D21 does not have a floating point unit (FPU) it may be more efficient in terms of processing time to implement fixed point arithmetic. Due to the time constraints of the project it was not investigated further, and given that there did not appear to be a noticeable delay caused by geofence calculations, it may not be necessary.

Complicated floating point operations such as multiply, divide or trigonometric calculations are only required when the collar is outside the boundary, as the 'point in polygon' test requires only comparisons of the floating point values.

6.5 Hardware Design

The hardware design of the collar was primarily designed as an aid to show what the system could possibly look like. When used on a real cow, the many flaws and areas for improvement in the design became apparent. The flat shape of the middle section did not fit well to the angle of their neck and the collar would need to be very robust to handle the daily activities of the animals, especially if they decided that they wanted to get rid of the collar. The design of the hinge system, whilst allowing for flexibility in the size and shape of the neck it fits around, would be an especially weak point to shearing forces.

Protecting the solar panels and antenna would also be important. The antenna sticking up from the top is not ideal, even at 5cm high as it could easily be broken off. This is a more difficult issue to solve as having a vertical dipole antenna provides the simplest and most consistent reception. Placing a dipole antenna flat would result in a blind spot for poor reception when the antenna is parallel to the direction of the base station. One possible solution would be to use two patch antennas, one on either side of the animal.

6.6 Other Issues

During testing the system there have been many issues, which finding the source of required a lot of time to solve, despite often being a small problem. Unknown software lock ups were a relatively common occurrence until it was determined that they were being caused by brown outs, when not enough current can be supplied to the board. This often occurred when the board was being powered off a laptop and the collar speakers were being used or the LoRa module was outputting at a high power level. When this occurs the MCU goes into a state where a reset is required to recover it.

In order to prevent a random lock up of the system such as the above example, from causing a collar to stop working out in the field, the SAM D21's watch dog timer functionality could be enabled. This is where the MCU regularly resets a timer, preventing it from timing out. If the system has an issue that causes it to not reset the timer before it elapses, it will trigger a reset of the MCU, hopefully allowing for recovery.

7 Project Management

7.1 Project Timeline

During the first few weeks of the project a Gantt chart was created with the complete list of tasks, estimated durations of each task and the aimed completion date. This proved extremely useful in keeping the project on track, with such a large amount of work to be completed having many intermediate deadlines was helpful and motivating. An overview of the key tasks completed during the year is provided in Table 7.

Task	Duration (Weeks)	Completed
Geofence algorithm	2	S1, W5
Component selection and testing	6	S1, W7
Collar enclosure design and manufacture	2	S1, W8
Schematic and PCB design	2	S1, W10
PCB manufacture	1	S1, W11
Transmission of Geolocations over LoRa	2	S1, W11
Design and printing of collar enclosure	1	S1, W12
Web interface - Click to place fences	4	S2, W2
Assembly and testing of custom PCBs	4	S2, W3
Compass - Tilt correction	2	S2, W5
Accelerometer Wake on Motion	2	S2, W6
Web interface - Wireless settings update	1	S2, W7
Various testing and debugging	4	S2, W11
Project poster	1	S2, W11
Report and video	3	S2, W12

Table 7: Task timeline, showing the number of weeks spent on that task and when it was completed.

7.2 Version control

GitHub was utilised for version control of the software and PCB design, allowing for changes to be rolled back if there were issues with them. It also provided security in that there was not only one copy of the project work. In total there were 57 versions committed to the repository over the course of the year.

8 Conclusion

Sustainability of agriculture will be a key area for research and development in the coming years, as there are many challenges and areas for improvement. Finding where technology can play a role in supporting improvements in sustainability will be one aspect of this evolution.

OpenFence was a project with big ambitions, to create not only a collar which could be used for containing livestock within a digital boundary, but to create a complete solution. This included a usable web interface for placing fences and tracking animal movements, and an in field base station for relaying messages from the collars to the internet.

The project was successful in the in achieving its key objectives. The directional geofence algorithm produces stereo audio alerts, with the volume increasing as the distance outside the boundary increases. The web interface created allows for the easy placement of fence boundaries and remote updating of settings. These are transferred wirelessly to the collar devices via the base station.

Two minor objectives that still require more work to be improved or completed are the collar enclosure design and the power electronics for the base station. The shape of the collar proved to not be well suited to the shape of a cows neck. With been more consultation and measurements performed beforehand the original design could have been improved. The base station requirement for 5 volts complicated the design of the single cell battery charging circuit, and thus could not be completed.

The results obtained for power consumption were quite good, with power in sleep mode down to 3.7mA. This combined with the power saving functionalities of wake on motion and intelligent LoRa output power selection allow for a device that can last for a long time on battery. However it should be possible to reduce the power consumption further with improvements in the components selected.

The range of the LoRa communications was less than was hoped could be possible, however it still was able to reach a 670m distance in an urban environment. It is possible that in a less dense environment it would be possible to achieve longer range.

Extra features that were not part of the original objectives, but significantly improved the system were the wake on motion sleep mode and the tilt corrected compass. By using the accelerometer to produce an interrupt when motion is detected the collars microcontroller can remain in a low power state until there is movement, at which point it can check the GPS position. The tilt corrected compass significantly increases the reliability of the system, as the collar will rarely be kept level whilst the magnetometer is being measured.

9 Future Work

Some recommendations for future work and improvements include:

- If the animal travels too far beyond the boundary such that it is unlikely to find it's way back, create a new temporary boundary around the animals location. This temporary boundary could then progress back towards the boundary again until it can again be contained in the original fence.
- Allow for the creation of paths which the paddock boundaries travel along. Boundaries moving at a rate of 1m an hour for example could be created.
- Designing of the system to use wireless speakers could permit the processor, GPS and radio to be enclosed separate unit, allowing the speaker to be placed closer to the ear on an ear tag.
- Design an appropriate antenna which can be laid flat on the animal, such that it is less likely to be damaged.
- Design a way to protect the solar panels such that they are able to achieve maximum power conversion, but not likely to be damaged.
- Statistics could be calculated from the data, with mean distance travelled, time spent in one area or finding close associations between particular animals being possible interesting results.
- Detection of sick, stolen or missing animals, if a collar has not transmitted for a certain amount of time this could raise an alarm.

References

[1] D. M. Anderson, "Virtual fencing past, present and future," *The Rangeland Journal*, vol. 29, pp. 65–78, 2007. [Online]. Available: http://jornada.nmsu.edu/bibliography/07-018.pdf

[2] Georg-Johann, "Plot of the linear chirp," 2010. [Online]. Available: https://commons.wikimedia.org/wiki/File:Linear-chirp.svg

[3] Semtech, *SX1276/77/78/79 Datasheet*, 2016. [Online]. Available: http://www.semtech.com/images/datasheet/sx1276.pdf

[4] Atmel, *SAM D21E / SAM D21G / SAM D21J Datasheet*, 2016.

[5] Invensense Inc, *MPU-9250 Product Specification, revision 1.0*, 2014.

[6] Microchip Technology Inc, *MPC73871: Stand-Alone System Load Sharing and Li-Ion/Li-Polymer Battery Charge Management Controller Datasheet*, 2013.

[7] D. R. Finley. (2007) Point-in-polygon algorithm determining whether a point is inside a complex polygon. [Online]. Available: http://alienryderflex.com/polygon/

[8] T. McCosker, "Cell Grazing - the first 10 years in Australia," *Tropical Grasslands*, vol. 34, pp. 207–218, 2000.

[9] M. Kiely, "Grazing systems dont work: Tell the farmer of the year," 2011. [Online]. Available: http://www.carbonfarmersofaustralia.com.au/carbon-farmers-of-australia-blog/grazing_systems_don-t_work_tell_the_farmer_of_the_year_

[10] N. Sleep, "Cell grazing system in low rainfall areas," 2016. [Online]. Available: http://agex.org.au/media/cell-grazing-system-in-low-rainfall-areas/

[11] M. Bell, D. Lawrence, "Soil carbon sequestration myths and mysteries," *The State of Queensland, Department of Primary Industries and Fisheries*, 2009. [Online]. Available: https://www.daf.qld.gov.au/plants/field-crops-and-pastures/broadacre-field-crops/soil-carbon-sequestration

[12] Z. Butler, P. Corke, R. Peterson, and D. Rus, "Virtual fences for controlling cows," in *Proc. IEEE International Conf. on Robotics and Automation*, vol. 5, 2004, pp. 4429–4436.

[13] A.R. Tiedemann, T.M. Quigley, L.D. White, W.S. Lauritzen, J.W. Thomas, and M.L. McInnis, "Electronic (fenceless) control of livestock," 1999.

[14] Tim Wark, Chris Crossman, Wen Hu, Ying Guo, Philip Valencia, Pavan Sikka, Peter Corke, Caroline Lee, John Henshall, Kishore Prayaga, Julian OGrady, Matt Reed, Andrew Fisher, "The design and evaluation of a mobile sensor/actuator network for autonomous animal control," *Information Processing in Sensor Networks*, 2007.

[15] Link Labs. (2015) What is lora? Link Labs. [Online]. Available: http://www.link-labs.com/what-is-lora/

[16] I. Poole. (2016) Lora physical layer & rf interface. [Online]. Available: http://www.radio-electronics.com/info/wireless/lora/rf-interface-physical-layer.php

[17] W. Sun, M. Choi and S. Choi, "IEEE 802.11ah: A Long Range 802.11 WLAN at Sub 1 GHz," vol. 1, pp. 83–108. [Online]. Available: http://riverpublishers.com/journal/journal_articles/RP_Journal_2245-800X_115.pdf

[18] Federal Standard 1037C. Spread spectrum. Glossary of Telecommunication Terms. [Online]. Available: http://www.atis.org/glossary/definition.aspx?id=1028

[19] Semtech. (2016) Lora frequently asked questions. [Online]. Available: http://www.semtech.com/wireless-rf/lora/LoRa-FAQs.pdf

[20] Technical Marketing Workgroup, "A technical overview of LoRa and LoRaWAN," *LoRa Alliance*, 2015.

[21] M. McCauley. (2016) Radiohead packet radio library. [Online]. Available: http://www.airspayce.com/mikem/arduino/RadioHead/

[22] K. Dems. (2010) Understanding almanac and ephemeris data. [Online]. Available: http://www.brighthub.com/electronics/gps/articles/73766.aspx

[23] W. Huber. (2011) Measuring accuracy of latitude and longitude. [Online]. Available: http://gis.stackexchange.com/questions/8650/measuring-accuracy-of-latitude-and-longitude/

[24] A. Sengpiel. (2014) Decibel levels and perceived volume change. [Online]. Available: http://www.sengpielaudio.com/calculator-levelchange.htm

[25] G.J. Bishop-Hurley, D.L. Swain, D.M. Anderson, P. Sikka, C. Crossman, P. Corke, "Virtual fencing applications: Implementing and testing an automated cattle control system," *Computers and Electronics in Agriculture*, vol. 56, pp. 14–22, 2007.

[26] K. Winer. (2016) Mpu9250 arduino library. [Online]. Available: https://github.com/kriswiner/MPU-9250

[27] DeLaval. (2007) Cow comfort: Resting. [Online]. Available: http://www.milkproduction.com/Library/Scientific-articles/Housing/Cow-comfort-9/

10 Appendices

All code created, Schematics, PCB Gerbers, CAD files can be found on the project GitHub and is released as open source, licensed under GNU General Public License version 3, which is a copyleft license.

10.1 Code

Collar Code: https://github.com/plyalex/OpenFence/tree/master/software/collar/CollarSoftware
Base Station Code: https://github.com/plyalex/OpenFence/tree/master/software/repeater/BaseStation
Web Interface Code: https://github.com/plyalex/OpenFence/tree/master/software/web

10.2 Schematics

https://github.com/plyalex/OpenFence/raw/master/hardware/pcbs/MainCollar/MainCollar.pdf

10.3 PCB Design

https://github.com/plyalex/OpenFence/tree/master/hardware/pcbs/MainCollar

10.4 Solidworks Files

https://github.com/plyalex/OpenFence/tree/master/hardware/solidworks