
Portable LoRa Cat Tracker

Technical Report

1922268

Authored by: Juheb Habib

Supervised by: James Covington

Contents

1	Definition	1
1.1	Abstract	1
1.2	Introduction	1
1.2.1	Justification	1
1.2.2	Aim	2
1.2.3	Objectives	3
1.3	Literature Review	3
1.3.1	Cat behaviour	3
1.3.2	LoRa	4
1.3.3	Alternative products	4
2	Development	6
2.1	Design	6
2.1.1	Specification	6
2.1.2	System overview	8
2.1.3	Hardware	9
2.1.4	Purchasing	12
2.2	Development	12
2.2.1	Initial function	12
2.2.2	Power tester	16
2.2.3	Intermediate steps	18
2.2.4	Battery test	19
2.2.5	Error investigation	23
2.2.6	Case design	24
2.2.7	Distance test	27
3	Conclusions	34
3.1	Assessment	34
3.1.1	Assessment of objectives	34
3.1.2	Assessment of aims and specification	34
3.2	Improvements	36
3.2.1	Issue resolvers	36
3.2.2	Quality improvements	37

3.3 Additional remarks	37
3.4 Evaluation	37
3.4.1 Difficulties	38
3.4.2 Admittances	38
4 References	39
Appendix A Further Reading	I
Appendix B Pinout Diagrams	II
Appendix C Case design	III
Appendix D Purchasing lists	VII
Appendix E Data	X
E.1 Power Consumption	X
E.2 Distance	XII
Appendix F Code	XIV
F.1 Power scripts	XIV
F.2 Distance scripts	XXI
F.3 Microcontroller sketches	XXII
F.4 SBC scripts	XXIX
Appendix G Errors	XXXVIII

List of Figures

2.1	System overview flowchart	9
2.2	Hardware block diagram	9
2.3	GPS hardware serial output	13
2.4	LoRa communications with GPS location	16
2.5	Power meter results	17
2.6	Damaged strip antenna.	19
2.7	Battery charge and discharge curves	22
2.8	Board and battery at the beginning of case development	24
2.9	Failed print	25
2.10	Distance test location and setup	28
2.11	Setup used for distance testing	28
2.12	Test data recordings	29
2.13	Distance test one overlay	31
2.14	Distance test two overlay	32
B.1	Raspberry Pi pinout	II
B.2	LoRa Feather M0 pinout	II
B.3	GPS Featherwing pin images	II
C.1	First case print	III
C.2	Case 1 dimensional drawings	IV
C.3	Second case print	V
C.4	Case 2 dimensional drawings	VI

List of Tables

2.1	SBC considerations	10
2.2	Microcontroller considerations	11
2.3	Non-standard format example - 230127.csv sample	21
2.4	Non-standard location data format	29
2.5	Desired location data format	29
3.1	Objectives evaluation	34
3.2	Aims evaluation	35
3.3	Specification evaluation	35
D.1	Preliminary hardware list	VII
D.2	Updated hardware list	VII
D.2	Updated hardware list (cont.)	VIII
D.3	Antennae list	IX
E.1	Power meter recordings	X
E.2	Discharge data sample - 230213.csv	X
E.4	Charge data sample - 230214.csv	XI
E.6	Distance Test 1 sample - first_test.csv	XII
E.8	Distance Test 2 sample - second_test.csv	XIII

Glossary

ABS Acrylonitrile butadiene styrene, a common thermoplastic. 26

BFI Big Five Inventory, grouping of personality traits into Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism. 4

FDM Fused Deposition Modeling, a method of 3D printing. 24, 25

G-code Motion controlling code for an automated (CNC) machine tool, aka RS-274. 25

GPGLA Global Positioning System Fix Data. 20

GPRMC Recommended minimum specific GPS/Transit data. 13

KML Keyhole Markup Language, an extension of XML for encoding geographical location data. 29

LoRa Long Range (Radio). 1, 2, 3, 4, 5, 6, 9, 10, 11, 12, 14, 15, 16, 18, 19, 21, 34, 37, II

LoRaWAN LoRa radio based Wide Area Network. 2, 5

PC Polycarbonate, a family of thermoplastics. 26

PGTOP Antenna advisor. 13, 14

PLA Polylactic acid, a thermoplastic often used for 3D printing, aka polylactide. 26

PMTK Proprietary data transfer protocol. 14

U.FL Miniature RF coaxial connector, aka I-PEX, AMC, AMCC, UMCC. 19, 26, 27

Acronyms

CRC Cyclic Redundancy Check. 36

CSS Chirp Spread-Spectrum. 4

CSV Comma Separated Value. 20, 21

DHCP Dynamic Host Configuration Protocol. 27

FEC Forward Error Correction. 36

GIS Geographic Information System. 29

GNSS Global Navigation Satellite System. 20

GPIO General Purpose Input Output. 10, 12

GPS Global Positioning System. 2, 6, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 26, 29, 30, 32, 34, 37

GUI Graphical User Interface. 10

HAT Hardware Attached on Top. 12

I²C Inter-Integrated Circuit. 11

IDE Integrated Development Environment. 13, 15

IP Internet Protocol. 27

LED Light Emitting Diode. 32

LPWAN Low Power Wide Area Network. 2, 5

NMEA National Marine Electronics Association. 13, 14

OLED Organic Light-Emitting Diode. 14, 37

PCB Printed Circuit Board. 3, 19, 24, 27, 34, 38

PPS Precise Positioning Service. 20

RF Radio Frequency. 2, 33, 38

RTK Real Time Kinematic. 20

SBC Single Board Computer. iv, 9, 10

SIM Subscriber Identity Module. 5

SoC System-on-Chip. 10, 37

SPI Serial Peripheral Interface. 11, 15

SSH Secure Shell. 20, 27

USB Universal Serial Bus. 16, 23, 37, V

VAT Value Added Tax. 37

Definition

1.1 Abstract

Many pet owners, particularly cat owners, allow their pets to freely roam the outdoors. However, it is impossible to determine what the pet is up to, or more pertinently, where they are and what they are up to. It is also not unusual for pets to go missing, and without some method of tracking location, it is impossible to determine where they are until they show up by themselves. Should a pet be lost or stuck, returning home becomes increasingly unlikely.

To counteract this, solutions for location tracking pets exist. Many of these solutions encounter one drawback or another, for example reliance on infrastructure or high maintenance costs. This thesis concerns investigating and developing a novel solution to the problem of pet tracking using LoRa radio modulation.

This technique allows for highly efficient radio transmissions, gaining large distances with minimal power. In effect, it allows the coverage of the roaming range of a cat. The difficulty lies in ensuring the radio module can be powered and protected from the elements.

Development of this module concerns first basic function, then testing its remote capabilities are suitable, and finally giving it a case, so it can be carried around. To follow on this project, this design will be refined with a focus on minimising its size by moving to a PCB and harness-attachable casing, and running it under a vigorous suite of tests until it has been entirely characterised and is suitable for consumer use.

1.2 Introduction

This thesis records the research, analysis, and development of a LoRa radio based cat tracker. LoRa is a radio modulation technique that allows for great transmission distances and high resistance to noise and interference. The purpose of such a tracker is to allow pet owners, particularly cat owners, to allow their pet to roam freely outdoors, while still providing location reporting. This may be particularly useful for owners of adventurous pets who live on high floors in apartments or otherwise restrictive dwelling. As such, there will be a greater level of emphasis placed on suitability in the average home of a dense, urban environment.

1.2.1 Justification

Many pet owners allow their pets to roam outdoors. None are perhaps as notable as the feline for their free-roaming prowess and sense of independence. 26% of UK households own a total of eleven million cats, of which 63% spend time outdoors (1).

Some of these pets may go missing for long periods of time - in some cases, even years before returning (2) - that is, if they return at all. The anxious pet owner is catered to by the \$2 billion global pet wearables market (3), providing them many options to alleviate their worries.

Pet wearables that concern tracking location fall into two broad categories in how they communicate location data back to the owner: network based, and radio frequency based.

- Network - leveraging the availability of an existing network, these solutions can have high fidelity, with quick and accurate updates covering a large area. They suffer where a network connection cannot be made.
- RF - usually forming a point-to-point connection between the user and the wearable. The range is usually limited, however no reliance on a pre-existing network is required, reducing operational costs.

LoRa radio is a novel technology for radio communications that is beginning to find its niche. Most attention is focused on LoRaWAN, a network layer built upon the physical layer provided by LoRa.

LoRaWAN and LPWANs in general are beginning to proliferate and find common use, particularly with the Internet of Things (4). However, such networks are still not widely available for easy use, and may suffer security issues at their current stage of development (5). Furthermore, while the radio and network definitions have specifications from the LoRa Alliance, there is as yet no formal infrastructure that can be relied upon.

However, LoRa radio by itself as an RF communication method is perfectly capable as a transmission medium without the network overlay to rely on. As such, it will be the focus technology in the development of a new solution to the problem of pet tracking.

1.2.2 Aim

The ultimate goal is a portable GPS tracker that makes use of LoRa to update location information on a periodic basis. The solution will need to monitor the current location of a moving target for a significant portion of the day and work within a certain radius of a stationary base.

The focus is on cats in particular, so care may need to be taken to ensure the tracker is ergonomically suited and safe for outdoor use. Due to the wider possible uses of the tracker, portability may also need to be considered.

As a product designed for use by the average pet owner, the setup must be suitable for use in a home and be relatively easy to set up. A solution that works 'out of the box' is better suited. Location reporting needs to be simple to view so that it can be done quickly and easily.

- Fundamentals - minimum requirements to demonstrate the viability of the product.
 1. Obtain GPS location.
 2. Transmit location to a receiver via LoRa.
 3. Continue transmission remotely for a significant period of time.
 4. Transmit at a regular interval.
- Improvements - upgrades to the product to improve the user's quality of life when using the product.
 1. Ergonomics for owner and pet (fitting and wearing).
 2. Stability of the system (high uptime).
 3. Installable by an average person.

4. Location information easily viewable.
- Production - requirements to ensure the product is suitable for manufacture.
 1. General safety and suitability to be worn.
 2. Resistance to damage and weather.
 3. Certification requirements and law compliance.
 4. Scale considerations, especially manufacturing.

1.2.3 Objectives

The following objectives list will provide guidance for how development will progress, and provide a clear set of criteria against which the success of the project can be examined.

This project will follow the numbered steps, and aim to meet each objective specified within.

1. Research cat characteristics, like roaming distances and time spent outdoors.
2. Research LoRa radio requirements for use, and competitive products.
3. Design the system on a macroscopic level to inform hardware and software requirements. This may include defining a specification of requirements the tracker must achieve.
4. Develop the tracker such that it meets the outlined aims.
5. Perform tests to verify the efficacy of the tracker¹.
6. Develop a PCB for the tracker to minimise form-factor, and produce adequate casing for it to be fitted to a pet collar or harness. This includes verifying basic functions again, in addition to suitability for intended use on an animal.
7. Gain required certifications for the product to be moved to manufacture, and design a manufacturing plan.

1.3 Literature Review

1.3.1 Cat behaviour

Reviewing cat behaviour will allow for a concrete basis upon which to write a specification that the tracker will meet. The foremost questions are how far a tame cat may roam, and for how long.

1.3.1.1 Roaming distance

Hanmer et al. suggests that the maximum distance of 278m, with a median maximum of 99m (6). It furthermore elaborates on how this decreases with the level of urbanisation, with urban areas seeing the smallest home range (79m). Barratt and Meek further corroborate the general value, with a mean range of 2.54ha (159m radius) and 2.9ha (170m) respectively (7, 8). Notably, these both concern felines in a rural area, and therefore the range is expectedly larger. Furthermore, the range can vary greatly on the cat's personality, with 'wandering' cats having a roaming range (5.1ha) an order of magnitude greater than 'sedentary' cats (0.4ha) (8).

Notably, feral and unowned cats have a much greater home range (easily 1.5km) than the average tame cat (9–11). Accommodating both feral and tame cats would greatly in-

¹Live testing using animals will require ethical approval to be obtained in advance.

crease the scope of the project, especially considering the roaming distance required to be covered increases by a factor of six. Therefore, accommodations for feral cats will not be specifically catered for. While the project focus is on tame cats, it is understood that tracking any animal or moving object may be applied outside this focus.

1.3.1.2 Time spent outdoors

As with home ranges, this varies greatly with personality. Interestingly enough, a significant contributing factor to this value was the owner characteristics. Clancy et al. discusses the relationship between owner and cat, amongst which includes time owners allow their cats outdoors. 40% of owned cats had some level of outdoor access, and of this, 88.4% spent less than eight hours outside. 97.1% were not permitted outdoors at night (12). Zhang et al. saw peak activity for cats between 6:00-10:00 and 17:00-21:00 (13).

Altogether, this suggests a required operating capability of eight hours to cover the 90th percentile, and that typical outdoor activity likely ranges three to four hours. The latter will be the critical minimum operating capability.

A marked finding by Finka et al. indicates that owners who score highly in neuroticism on the BFI were associated with lower likelihood of permitting “*ad libitum*” access to the outdoors” to their cat (14). This suggests that the product may be well suited to cater to owners who would like to permit their cat more time outdoors.

1.3.2 LoRa

LoRa radio modulation is a technique derived from CSS, a combination of chirp signals and spread-spectrum technology. Spread-spectrum is a method by which a radio signal has a wider bandwidth due to spreading in the frequency domain, which brings with it benefits such as resistance to noise and reduced spectral flux density (making it more power efficient). Chirp signals are a method by which the signal frequency is increased (up-chirp) or decreased (down-chirp) with time (15). LoRa in particular modulates data with instantaneous changes in the starting frequency to indicate symbol borders (16). Altogether, this gives LoRa modulation the capability of reliable transmission of over 3km in dense suburban areas (17).

The specifics of how modulation is achieved are not necessarily relevant to this project, as the focus is on utility. The theoretical breakdown of frequency shift chirp modulation used by LoRa is provided by Vangelista (18). The patent for the technology is currently held by Semtech (19).

LoRa uses license-free radio bands. In the UK, this is covered by Directive 2011/829/EU (20), transferred over to Ofcom IR 2030 (21). This will be important for when selecting hardware, the radio will need to be able to transmit at 868.0MHz.

1.3.3 Alternative products

1.3.3.1 Patents

Investigating existing patents, there appears to be a device that performs a similar function published in China (22). However, it makes no mention on how the tracker operates be-

sides the use of LoRa and discusses more the construction of the casing. A similar patent can be found (23), which again primarily discusses the construction of the tracker casing. However, from the title it can be deduced that the tracker is Bluetooth-based.

1.3.3.2 Similar products

The problem of pet, or more broadly, animal, tracking is a question that has only gained greater focus in our ever more well-connected world. Tracking farm animals, for example, is already an issue being directly tackled (24). Crucially, this leverages the LPWAN of LoRaWAN specifically, and relies less on LoRa as a point-to-point communication tool. This is a sensible approach due to the large number of incoming connections, which is less applicable here.

A review of modern pet tracking advances bases itself on existing mobile network infrastructure (25). However, one of the key drawbacks it mentions is difficulty of implementation, especially amongst agricultural societies. This emphasises the importance of both suitability away from infrastructure, and simplicity and ease of setup for the consumer.

This problem is present with products the tracker of this project may compete with. One of the foremost such, Tractive, requires an ongoing subscription due to the usage of an inbuilt SIM card, costing as much as £12 per month in addition to the upfront cost of £44.99 (26, 27). With reliance on a third party for this service, issues like privacy² and service availability where the company folds are brought into question.

The alternative is to forego network reliance, and communicate on a point-to-point basis. This is the approach this project will follow. One product, the Tabcat V2, follows this approach. While the product capabilities vary depending on environmental conditions, the quoted usable range is 152m in typical conditions (29). This does not cover the total possible roaming range as discussed in section 1.3.1.1. Their approach is also specifically limited to one tracker registered to one tag. This may be an issue in a multi-cat household as 35% of cat-owning households own multiple cat (1).

The final style discussed here is that employed by products like Tile and AirTags. These are Bluetooth devices (i.e. very short range only) and have additional reliance on Tile app and Apple device owners respectively, meaning the network is even more restricted (30). This also raises additional security concerns, especially regarding unwanted tracking (31, 32).

²Tractive may collect “pet related data that allows to draw conclusions about the pet owner”, which may be of concern for some people (28).

Development

2.1 Design

This section covers design considerations to inform subsequent development. System specifications will be defined, and hardware purchasing will be informed. The requirements of development will then be outlined before development commences.

2.1.1 Specification

To inform design decisions, a specification of requirements will need to be defined, which themselves are informed by the aims in section 1.2.2 and research in section 1.3. This specification will cover a limited scope of the defined fundamentals and a selection of improvements and production, thereby limiting the reach of the project to ensure it remains feasible. This may be revisited if development is ahead of schedule.

2.1.1.1 Reporting

The product will require a LoRa radio modem and GPS receiver in order to obtain and transmit location.

Timing A minimum transmission period of 30s will be defined, however this likely can be greatly improved upon. Any gaps in live location should not be any greater than 300s.

Distance The tracker should be suited for, at worst case, a dense urban environment, and sufficiently capable to report information up to at least 300m, given the roaming range of a cat¹ and the quoted values by competitors products².

2.1.1.2 Portability

With the primary target being cats, the product cannot be a significant burden to wear. Portability in this case means four things:

1. Small form-factor.
2. Lightweight.
3. Battery powered.
4. Attachable.

Small form-factor The product needs to be small enough to fit on a cat. Given the likelihood for an outdoor cat to be squeezing through tight spaces, the product needs to be small enough to not get caught and become a restriction.

The smallest form-factor will be aimed for, and designed in such a way that it can integrate in a seamless fashion with the cat's wearable

As reasonable figure to start with would be dimensions of 50mm × 70mm × 10mm.

Lightweight There is no clear answer for what an appropriate weight for the product would be given great range in the weight of a cat. The range easily covers 3kg to 6kg for healthy pet cats (33).

¹Section 1.3.1.1.

²Section 1.3.3.2.

The answer to how much weight a cat can carry for long periods is even less clear. Anecdotal evidence suggests a value of around 10% is easily manageable. With that in mind, an aim for half that at, 5% of the minimum likely weight (3kg) is a reasonable value, such that the weight is appropriate for any cat. This puts the desired weight for the tracker at $\leq 150\text{g}$, subject to testing to validate.

Battery powered The key factor of portability is that the product can operate without a direct connection to an outlet for power. This means that the product requires some form of portable power.

With convenience, environmental, and energy-density (and thereby weight) considerations in mind (34), the most appropriate type of battery will be rechargeable lithium-ion. This may raise some safety concerns due to the high likelihood of flammability of rechargeable battery cells (35).

The most pertinent question is how long should such a battery allow the tracker to operate. An outdoor domestic cat spends an average of 8.4h outdoors (36), this puts a minimum value of 9h of supply necessary.

Attachable Some method to attach the product to a cat needs to be devised. If it is small enough, this may be on a collar. Alternatively, attached to a vest may also work. This is very dependent on the dimensions the product takes in the end. Furthermore, if the product takes a prominent, protruding shape, it needs to detach easily so as not to be a restriction for the animal.

The primary focus will be on ensuring it cannot be caught in anything to begin with, as the tracker coming loose defeats its purpose.

2.1.1.3 Suitability

This product is designed for use by the average pet owner. Therefore, no specialised knowledge on how to operate any equipment can be expected. Furthermore, the system should be installable and maintainable using skills and hardware the average household can be expected to own, with some allowances for a small amount of DIY work.

Installable If any hardware requires installation, it must be in as simple a manner as possible, with only basic (manual) tools. The expectation will be that installation should be, at most, as difficult as a simple piece of furniture from a DIY store.

Compliance Compliance for consumer products needs to be met. If certification is not explicitly sought (for example, in the case third-party verification is required), the product must still meet the standards outlined such that it can be certified when necessary.

As a product developed in the UK, UKCA guidelines will be used.

2.1.1.4 Reliability

The reliability of the tracker will be a matter of concern, as it would be especially inappropriate for it to fail in its reporting midway.

Stable Part of the product relies on correct handling and processing of information over code. When interpreting data sets, poorly formatted data has the tendency to cause crashes in unsafe code. The code will need to be built to safely handle errors, both in the above-mentioned case and in any other potential points of software failure.

When an error occurs, it will need to report the error and then continue execution with minimal downtime. Where the fault is fatal for whatever reason, sufficient diagnostic information is required so that fault-finding can be facilitated.

Fault resistant In continuation with safely handling software errors, the hardware will also need to be fault-tolerant. This inclusively includes problems that may occur due to inclement weather or impacts.

Repairable Where such a hardware fault occurs that the product is no longer functional, repairs are to be facilitated in reasonably easy manner. The user should have the facilities to perform simple repairs, like changing an ageing battery. More complex repairs that require tools or specific knowledge will be carried out by a trained individual, and therefore out of scope. However, a user with sufficient knowledge should not be hindered from performing repairs themselves if they are able.

Safety This product introduces flammable electronics to wet environments, and therefore must be designed such that delicate parts are protected from the elements and, should such exposure occur, not be an inherent fire or electrocution risk.

Deduplication Given that all transmissions will be operating on the same frequency, data from different trackers will be received by any given antenna, of which, the unsuitable data will need to be filtered out. This also brings up the question of security and 'listening in' on another's pet.

2.1.2 System overview

The system can be broken down into two independent parts - the 'transmitter' (attached to the cat) and 'receiver' (stationary at the owner's home). Each subsystem has an independent process of operation. The minimum requirements are subsequently outlined in fig. 2.1.

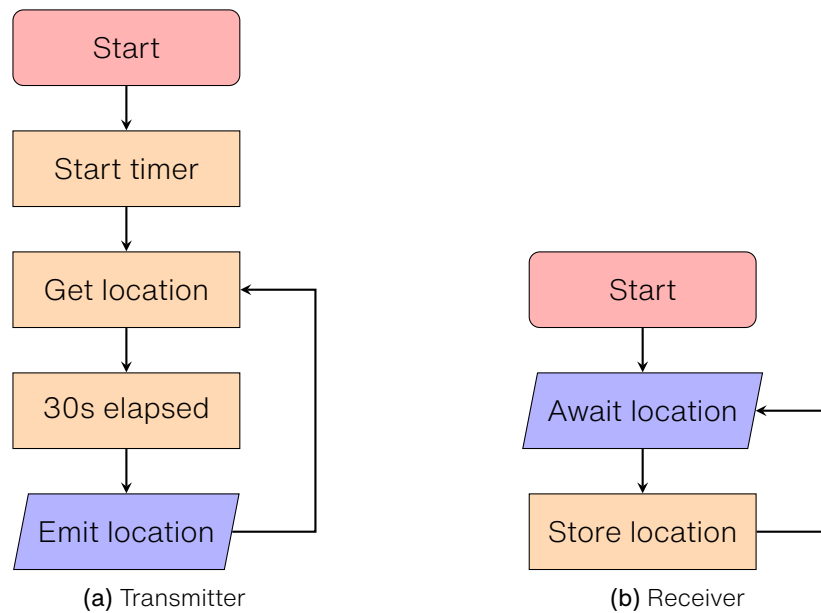


Figure 2.1: System overview flowchart

2.1.3 Hardware

Given the specification list, some hardware requirements can be defined to inform purchasing decisions. The transmitter needs to be portable, therefore each component is size, weight, and power limited. The receiver, on the other hand, is designed to be stationary and has no technical size or power limitations. However, it has to be sized within reason for a product designed for home use.

The following component list will help inform what specific components are required, which has been drawn into a hardware block diagram in fig. 2.2 to aid visualisation.

- Transmitter
 - GPS module for location.
 - LoRa module for transmission.
 - Small form-factor antenna to aid transmission.
 - Microcontroller for general computation.
 - Battery as a remote power source.
- Receiver
 - LoRa module for transmission.
 - Microcontroller or SBC for computation.

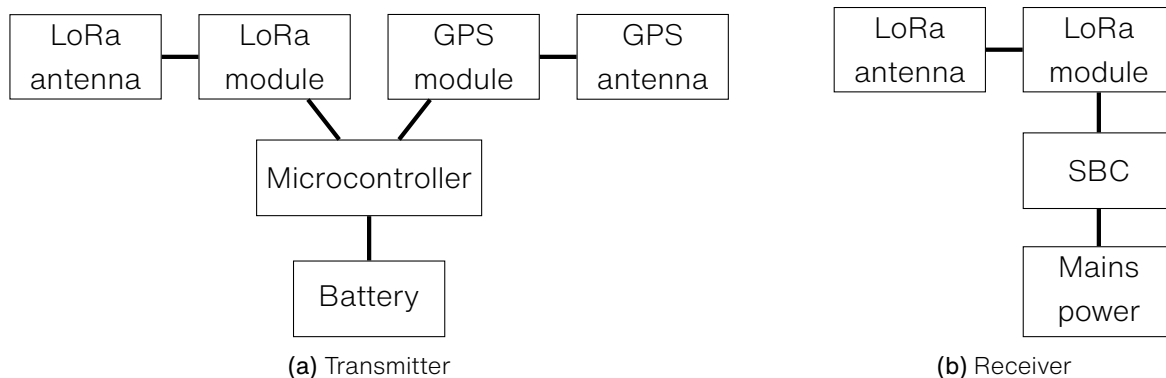


Figure 2.2: Hardware block diagram

There are many possible configurations to produce the sets of hardware required from parts available across different vendors. With this in mind, solutions that incorporate built-in features will be prioritised.

2.1.3.1 SBC choices

Some decisions were made to narrow down the diverse range of options, given hardware availability and technological experience. Regarding the transmitter controller; while this may be as simple as a microcontroller, with extensibility and future-proofing in mind³, an SBC will be the product of focus. Given hardware shortages, should there be none available, this may be revisited. There are a number of SBC offerings available, some of which are listed with a discussion in table 2.1.

Board	Description	Benefits	Drawbacks
Raspberry Pi	A series of general purpose SBCs using Broadcom BCM series SoCs.	<ul style="list-style-type: none"> •Inexpensive •GPIO •Pi 3 available 	
NVIDIA Jetson Nano	AI and neural network development focused, built around the ARM Cortex-A57 and NVIDIA Maxwell.	<ul style="list-style-type: none"> •Powerful •GPIO 	<ul style="list-style-type: none"> •Expensive •More powerful than necessary •Availability
Beaglebone Black	Pi equivalent, using an ARM Cortex-A8.	<ul style="list-style-type: none"> •Inexpensive •GPIO 	<ul style="list-style-type: none"> •Availability
Dell Wyse	Thin client server/PC.	<ul style="list-style-type: none"> •Familiar interface for an end user •Easier GUI programming 	<ul style="list-style-type: none"> •No GPIO •Very expensive

Table 2.1: SBC considerations

With all things considered, the Raspberry Pi is the most obvious choice simply from the standpoint that a Pi 3 is currently available for use. Given the current climate, it is virtually impossible to order computing hardware in this category. Even the Pi Compute Modules, small form-factor Pi boards with no peripherals or breakouts, are often out of stock. This is despite the fact that these boards are unable to function ‘out of the box’.

2.1.3.2 Microcontroller choices

The transmitter does not need to perform anything greater than obtaining GPS coordinates and transmitting them by LoRa radio. For this reason, a microcontroller is well suited. There is an even greater selection available than with the SBC choices. Therefore, as before, the options will be shortlisted and considered in table 2.2.

³For example, serving a user-interface, storing data in a database etc.

Board	Description	Benefits	Drawbacks
Arduino Uno	One of the most popular microcontrollers available, with an ATmega328P at its core.	<ul style="list-style-type: none"> • Widely available • Large peripheral and software support 	<ul style="list-style-type: none"> • Large
Challenger RP2040 LoRa	Embedded computer with integrated LoRa modem using the RP2040, a dual ARM Cortex-M0+ microcontroller.	<ul style="list-style-type: none"> • Integrated LoRa modem 	
Feather RP2040	RP2040 board produced by Adafruit in the feather form-factor.	<ul style="list-style-type: none"> • Feather form-factor • USB-C 	
Wio-E5 mini	Compact LoRa dev board produced by Seeed Studio using an STM32WLE5JC, a bundled ARM Cortex-M4 microcontroller and SX126X LoRa radio modem.	<ul style="list-style-type: none"> • Integrated microcontroller and LoRa modem • USB-C 	<ul style="list-style-type: none"> • Availability • Involved programming requirements
Feather 32u4 with LoRa	Feather form-factor board with LoRa produced by Adafruit, sporting an ATmega32u4.	<ul style="list-style-type: none"> • Integrated LoRa modem • Prior experience • Feather form-factor 	<ul style="list-style-type: none"> • Availability

Table 2.2: Microcontroller considerations

The board of choice here would be the Feather RP2040. While integrated LoRa is certainly beneficial, having the flexibility to customise it during development may prove more helpful⁴. Furthermore, this board shares the advantage of other Feather boards in that they are stackable, meaning Adafruit's large range of peripherals (including LoRa Featherwings and GPS Featherwings) are available to use.

Purchasing two such LoRa Featherwings for use in the transmitter and receiver ensures they are able to communicate with each other without issue. Given the selected form-factor and subsequent ease of use, it follows that the GPS module will be from Adafruit's Featherwing line.

⁴The RFM95W LoRa chipset uses SPI and has, in previous experience, caused difficulties with hardware on the same bus. In case it is necessary, an alternative set of hardware using the I²C bus will be produced.

2.1.4 Purchasing

Some antenna options will be included as the best approach for this has not yet been determined. Ceramic antennae have the smallest form-factor, however they may prove difficult to mount due to small contact points compared to more traditional wire antennae. The receiver will also require a large gain antenna to pick up weak transmissions.

A preliminary hardware set was produced⁵. This list consists of two sets of hardware for the transmitter, hardware required regardless of set, and some antenna options.

The radios selected use the same model modem, the RFM95W (37), ensuring the modules are capable of communicating (i.e. are operating within the same wavelength range).

Order review The list had to be adjusted to primarily target hardware from approved vendors⁶. With the vendor adjustment made, some of the initial hardware had to be substituted for more expensive alternatives. This made board options simpler by limited the feasible solution to a Feather M0 with bundled LoRa modem. Additional accessories were included, along with an extensive set of antenna and battery options to choose from.

This list was further adjusted to omit spare hardware that is available to use (for example, headers, connectors, etc.). Furthermore, battery purchasing will be put on hold until the required capacity can be determined. Finally, the LoRa module for the Pi was changed to a HAT. This uses the same radio hardware, but has the additional benefit of being able to directly interface with the Pi pins. The list was then requested for purchasing.

2.2 Development

2.2.1 Initial function

This section concerns getting the tracker to the point where it can transmit GPS location via LoRa.

2.2.1.1 Hardware

The Feather (M0 LoRa) and GPS Featherwing are designed to be directly stackable⁷, and as such share the same pin placement and 2D dimensions. The pinout diagram for the Feather M0 can be found in fig. B.2, with matching pin placement for the GPS Featherwing in fig. B.3a.

Therefore, each pin can be soldered directly using included headers. For now, additional hardware like batteries and antennae are not necessary, as only basic function will be tested.

The Pi HAT is similarly configured to slot onto the GPIO pins available on the Pi 3 directly. Pinout diagrams for the board can be found in fig. B.1. No additional work is required to interface the hardware.

⁵Table D.1.

⁶Table D.2.

⁷Figure B.3b.

2.2.1.2 GPS

With the boards soldered together, basic operation can be checked by loading the ‘blink’ sketch (38) using Arduino’s IDE. This sketch will simply switch the inbuilt LED on and off every second, and can be used to verify that the board can accept and run sketches.

Having proven basic function, the next test concerns the function of the specific hardware. The GPS module will be focused on to begin with.

The initial test sketch as per the documentation (39, 40) is loaded. This produces output as in fig. 2.3.

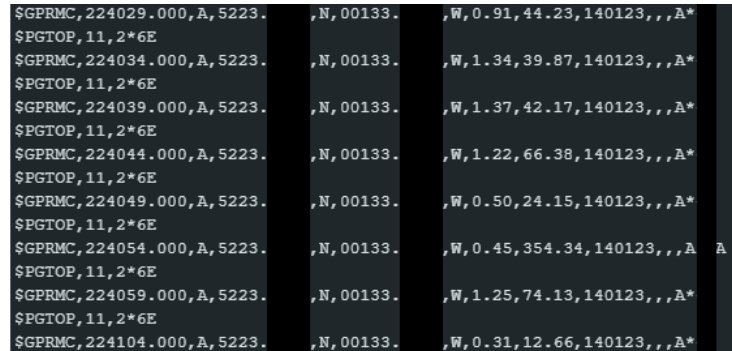


Figure 2.3: GPS hardware serial output with sensitive location data censored, showing GPRMC and PGTOP sentences.

By default, the GPS module outputs many standardised NMEA sentences, the most important of which to begin with is the GPRMC sentence.

```
1 $GPRMC,224029.000,A,5223.00,N,00133.00,W,0.91,44.23,140123,,A*40
```

Listing 2.1: NMEA GPRMC sentence example

The output format is not immediately obvious. The listing 2.1 can be broken down as follows (41, 42):

GPRMC	Message ID
223029.00	Time in UTC (22:30:29.00)
A	Validity - Active/Void
5223.00,N	Latitude (52°23.00" North ⁸)
00133.00,W	Longitude (001°33.00" West)
0.91	Speed (knots)
44.23	Tracking angle
140123	Date (14 th January 2023)
A	Positioning mode indicator
	A Autonomous D Differential E Estimated (dead reckoning) M Manual input N Data not valid
40	Checksum (XOR of all bytes between \$ and *)

The output location was verified to be correct. Obtaining a fix while indoors proved very difficult. The GPS module uses coin cell to save fix data, which would have helped obtain a fix outdoors and then move indoors to observe the data.

⁸The longitude and latitude appear to be in decimal degrees at a glance and will therefore return errors when attempting to plot with this expectation. Furthermore, it is specifically degrees minutes and not degrees minutes seconds as one may expect.

The PGTOP sentence characterises the antenna connection, a non-standard command specific to the FGPMMPA6H module (43).

```
1 $PGTOP,11,2*6E
```

Listing 2.2: NMEA PGTOP sentence example

Listing 2.2 can be broken down as follows:

PGTOP	Message ID
11	Command ID
2	Antenna status
	1 Active antenna shorted
	2 Using internal antenna
	3 Using external (active) antenna
6E	Checksum

The antenna status is not particularly useful, especially so at this stage of the development. Therefore, it can be disabled. Many of the commands to program the device (for example, changing the sentence output frequency) can be found in the PMTK reference document⁹ (44). Alternatively, the Adafruit GPS library (45) has support for some commands. These commands are not documented and can only be found hardcoded into the header file of the library (46).

Regarding the antenna, it can be disabled in two ways, via serial¹⁰ :

```
1 #define GPSSerial Serial1
2 GPSSerial.println("$PGCMD,33,0*6D");
```

Listing 2.3:

or using the library:

```
1 Adafruit_GPS GPS(&GPSSerial);
2 GPS.sendCommand(PGCMD_NOANTENNA);
```

Listing 2.4:

The library provides some useful functions and makes the code less verbose and more readable, however it may prove to include significant enough overhead to impact the battery life. If that happens to be the case the library can be removed as essential function can be programmed to the GPS chip directly via serial.

2.2.1.3 Pi LoRa

The LoRa bonnet library as provided by Adafruit requires Circuit Python to be installed on the Pi (47, 48)¹¹.

With required software, libraries, and fonts installed, the example code (49) can be loaded. This checks that the radio is present and the OLED displays correctly.

⁹With a battery, the PMTK commands are retained and do not need to be presented to the module on power-on. Otherwise, settings must be reconfigured on every boot.

¹⁰This must be the serial connection to the GPS device from the main board, and cannot be accessed directly via serial monitor. In order to send commands on the fly via serial monitor, the code must be configured to then forward those commands to the correct serial connection (in the case of listing 2.3, 'serial1').

¹¹The installation documentation concerns both RFM69 and RFM9X radios. This project uses the RFM95W radio, also referred to as LoRa.

Finally, the radio function script can be loaded (50). This will await incoming LoRa packets, but can also send data using the buttons.

2.2.1.4 Feather LoRa

LoRa transmission with this board relies on the RadioHead library (51) in order to transmit LoRa packets, installation details for which can further be found in the documentation (52).

The transmission test script can be loaded onto the Feather (53) once required software has been installed. This will transmit a packet every second and await a reply. For the time being, the Pi will not automatically send a reply, so the Feather will output the “error text”. This is of little consequence as the Pi primarily only needs to work as a receiver with no two-way communication¹².

2.2.1.5 LoRa with GPS

With one-way communication working as expected¹³, the next step is to integrate LoRa with GPS on the Feather. This stage is where communication collision issues have occurred in previous projects, however as the GPS module uses the second serial connection and the LoRa chip uses SPI, no such issues should occur.

For streamlined testing, the library will be used to strip the data sent to only latitude, longitude, and packet number (incremented every time GPS location is fixed to indicate that the location is up-to-date). The various segments of data then need to be placed together in a structure. The public member function `send()` of the `RH_RF95` class (54) takes two arguments, `data (uint8_t*)` and `len (uint8_t)`. This means that the data to be sent needs to be consolidated into a single structure, with the pointer to the structure and its length sent to the function. The Arduino compiler is built on processing AVR-C and therefore suffers when handling strings and performing conversion and concatenation natively, which can lead to complex and verbose code. The Arduino IDE also supports C++ classes and function, amongst which include the `String` class and `+` concatenating function (55, 56). While the class may prove problematic on embedded hardware due to memory fragmentation on the heap, for the time being it is sufficient where the primary goal is functional and readable code. Two additional benefits are that the length is also easily attainable as the method `.length()` on the class, and also conversion to an array of type `char` is supported. Listing 2.5 shows a snippet of the body code.

```

1 void loop() {
2     //buffer gps location
3     char c = GPS.read();
4     //interpret whether new location obtained and parse
5     if (GPS.newNMEAreceived()) {
6         if (!GPS.parse(GPS.lastNMEA())) {
7             return;
8         }

```

¹²The test script on the Pi does in fact allow output by pressing one of the buttons. This can be used to test communications and will be picked up by the Feather if this is tested.

¹³Both example scripts are set at 915.00MHz and will work by default. This needs to be adjusted to 868.00MHz for use in the UK (27).

```

9      }
10     //execute every 5 seconds
11     if (millis() - timer > 5000) {
12         timer = millis();
13         //increment packet counter if new fix obtained
14         if (GPS.fix) {
15             count++;
16         }
17         //produce concatenated string
18         String output = String(count) + ": " + String(GPS.latitude,4) + "," + String
(GPS.longitude,4);
19         length = output.length();
20         //create variable send to hold packet once converted to type char
21         char send[length];
22         output.toCharArray(send, length);
23         Serial.println(send);
24         rf95.send((uint8_t *)send, length);
25     }
26 }

```

Listing 2.5: Transmission body code

The output can be observed in section 2.2.1.5, showing LoRa and GPS working together.

```

54: 5223. ,133.
55: 5223. ,133.
56: 5223. ,133.
57: 5223. ,133.
58: 5223. ,133.
59: 5223. ,133.
60: 5223. ,133.
61: 5223. ,133.
62: 5223. ,133.
63: 5223. ,133.

```

(a) Feather transmission logs

```

pi@pi:~/Project $ python3 radio.py
RX: 57: 5223. ,133.
RX: 58: 5223. ,133.
RX: 59: 5223. ,133.
RX: 60: 5223. ,133.
RX: 61: 5223. ,133.
RX: 62: 5223. ,133.
RX: 63: 5223. ,133.
RX: 64: 5223. ,133.
RX: 66: 5223. ,133.
RX: 67: 5223. ,133.
RX: 68: 5223. ,133.

```

(b) Pi reception logs

Figure 2.4: LoRa communications with GPS location

2.2.2 Power tester

In order to determine the appropriate battery capacity to use, it is helpful to test the power consumption of the transmitter over the expected period of use. Greater capacity batteries are heavier and larger - two factors that need to be minimised. There is also the potential fire risk, however this is minimal without damage.

By recommendation, a KCX-017 USB tester was purchased to measure the current draw of the transmitter under normal function. The included manual was poorly translated and difficult to follow, however a more informative version was found online (57). No further details about the manufacturer or any other products they may produce were found.

2.2.2.1 Test specification

The outline of the test is to determine the appropriate battery required for the module. The tester has various functions available, however only the capacity recording is relevant. The tester will be plugged into a USB outlet, and the transmitter will be plugged into the tester. The transmitter will be plugged into the tester and will operate from thereon. Recordings

will be taken every thirty minutes for four hours as a preliminary measure of function. The Pi will be running to record that the transmitter is operating as expected.

Two tests were performed:

1. The transmitter was housed indoors so that it is unable to obtain a GPS lock. This will lead to the greatest power consumption.
2. The transmitter was placed near a window so that it is able to update GPS location. This simulates normal operation.

Risks The only identifiable risk is excessive current draw for the tester to handle. This risk is negligible as the tester is quoted to have a current limit of 3500mA (57), which is far beyond the Feather capabilities.

2.2.2.2 Results

Upon commencing the first test, the receiver registered no data. The cause was to simply remove the line `while (!Serial) delay(10);` as it prevents the code executing until the serial interface is connected, so output can be monitored. In the case of this test, serial output was not required, however the line still stalled the code.

With the change made and output occurring as expected, the meter did not appear to register any changes. The meter was checked to be working by observing the current draw of a mobile phone, and was found to be operating as expected. Thereafter, the meter was left and checked every hour. The first change only occurred after three hours.

The consumption incremented only after three hours. It was left overnight and checked again, at this point registering 8mAh. The complete set of recordings can be found in table E.1. The data is plotted below using code in listing F.1.

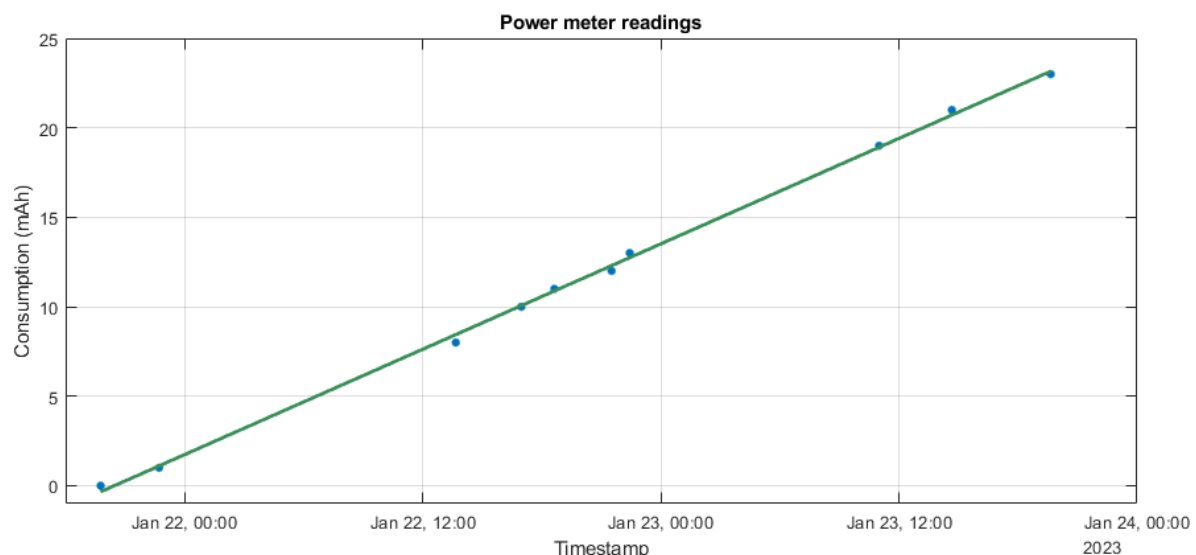


Figure 2.5: Power meter results

The results suggest that, during normal operation, the transmitter requires around only 0.5mA. This is far too low to be accurate, and in turn, likely means the meter does not have sufficient resolution to measure the device. The meter does indicate a linear trend in

power draw suggesting predictability in normal use, however given the poor accuracy of the data, the validity of this cannot be determined.

2.2.3 Intermediate steps

The power meter results were disappointing, however the documentation for the hardware provides useful values with which an estimated battery capacity requirement can be produced. The LoRa board draws 40mA when the radio is actively listening, and draws 130mA for 70ms when transmitting at peak gain (52). The GPS module draws 25mA when tracking and 20mA during navigation (39). This gives a rough total of 60mA typical draw in the average case. Considering the transmitter needs to operate for nine hours¹⁴, this suggests a battery capacity requirement of 540mAh.

This suggests that a repurposed 3.7V 500mAh quadcopter battery, may be used to test the transmitter functionality directly. The LoRa board requires a JST jack (52). By investigating Adafruit's other products, this appears to be a JST-PH connector specifically (58).

The battery header was removed and a JST-PH connector crimped onto the end of the leads. The polarity was then checked before powering the transmitter and ensuring it functioned as before.

2.2.3.1 Code issues

During such a function test, the code on the receiver end crashed twice, producing an error such as in listing 2.6.

```
1 Traceback (most recent call last):
2   File "/radio.py", line 72, in <module>
3     packet_text = str(prev_packet, "utf-8")
4 UnicodeDecodeError: 'utf-8' codec can't decode byte 0xae in position 0: invalid
   start byte
```

Listing 2.6: Program error message

Due to the module being powered up and disconnected abruptly multiple times while setting up the battery, the suspicion was that transmission was interrupted midway, leading to the receiver attempting to parse an unfinished and corrupted packet. However, given the time window of less than 100ms for transmission, the fact this occurred twice suggests this theory may not be likely.

In order to test how long the transmitter is able to function, the code needs to be improved to be more resilient and stable. The example code (50) has been updated to safely handle the exception. While the cause of the error has not been determined or investigated further at this point, the code will now attempt to parse the packet in a `try/except` block. The first handles the `UnicodeDecodeError` directly, while the second handles anything else that may occur, in case there are other issues that have not been uncovered. It will indicate that an unexpected error has occurred and continue execution from thereon. The modified code can be found in listing F.10.

2.2.3.2 Antenna

¹⁴Section 2.1.1.2.

Due to delivery issues, neither module currently is able to use an antenna, and thus have been confined to short range tests only. While part of the purchasing list included ceramic and strip antennae for the transmitter only, the ceramic antennae cannot be used as they require some form of breakout, usually on a PCB and further currently require a U.FL adapter to interface with the board. The strip antenna, on the other hand, use a very delicate cable between the strip itself and the connector, and were therefore easily sheared by one *Felis catus* who found it unattended during operation.



Figure 2.6: Damaged strip antenna.

A spare whip antenna was available to use with the Pi, however its frequency was unknown. Upon casual inspection, it may be appropriately sized. To investigate further, the appropriate antenna length needs to be determined.

The wavelength can be found by:

$$\lambda = \frac{v}{f}$$

where: λ = wavelength

v = velocity, which in this case will be c , the speed of light

f = frequency

For the device operating at 868.0MHz, this appears like the following:

$$\begin{aligned}\lambda &= \frac{c}{868.0\text{MHz}} \\ &= 0.345\text{m}\end{aligned}$$

The antenna whip measures between 21cm and 25cm, although it is impossible to say exactly how long it is without opening it up. If this antenna is centred near 868MHz, it may be a five-eighth whip, given $\frac{5}{8}\lambda = 21.5\text{cm}$. This suggests it is appropriate to use until a more robust set of antennae has requested for ordering¹⁵.

2.2.4 Battery test

The purpose of the test is to determine the length of time the transmitter can continuously operate under normal conditions, i.e. obtaining current GPS location and transmitting it via LoRa. As outlined in section 2.2.2.1, there is an expectation of roughly eight hours of function. This will now be investigated to determine whether this is the case.

The transmitter code will need to be updated to include information about the current status of the battery. The voltage level of the battery indicates whether it is within operating

¹⁵Decided list can be found in table D.3.

range, which can be measured by reading the analogue value of pin A7. At 4.2V, capacity is at maximum, whereas the board will cease function around 3.2V. The exact calculation can be found in the board documentation (52), which has been adapted in listing F.5.

After ascertaining that the calculation was working correctly, the code has been included in listing F.8¹⁶. The output string has also been modified to include the battery data, altitude, and fix quality. GPS fix takes the following values (59):

- 0 No fix (no signal)
- 1 Position fixed with GNSS satellites
- 2 Differential GPS
- 3-5 PPS¹⁷/RTK fix
- 6 Estimated location

The latter two required GPGLL data, which can be included with the library command `PMTK_SET_NMEA_OUTPUT_RMCGGA`. Finally, field separators were replaced with commas as part of preparation for the data to be stored in CSV format.

2.2.4.1 Test parameters

Two tests were conducted.

1. Discharge - The board operates under normal conditions. The battery was charged to maximum capacity (4.2V), before being allowed to discharge completely. Throughout that time, the board was transmitting location data and battery level. The receiver logs the time of packets and stores it in a data structure.
2. Charge - After a complete discharge, the time taken to charge to maximum was recorded similarly.

The way the Pi is operated was changed. Normally, an SSH session is terminated when the client disconnects or times out, and any programs run during that session are closed. This was not a problem with the power meter test as the receiver only needed to indicate that transmitter was working, while the data itself was recorded externally. In this case, the Pi is recording all data and needs to be running without interruption for at least eight hours. Linux comes with a utility named `screen` that allows a virtual terminal session to be started that will not be closed¹⁸.

Risks The greatest risk is related to the battery short-circuiting, however this has been dealt with adequately in section 2.2.3. It is possible the battery may be damaged due to excessive discharge, however the protection circuitry will cut the supply before then.

2.2.4.2 Issues

The tests were executed without issue and a brief study of the results indicated they were within expectation. However, some issues cropped up.

¹⁶This is in fact the third iteration of code. The two former versions can be found in listing F.7 and listing F.6.

¹⁷Encrypted military/government use

¹⁸An alternative would be to run the process and move it to background. The benefit of `screen` is that logging messages are easier to view when printed to terminal.

C treats strings as an array of type `char` and determines the end of the string with the 'null' character¹⁹ `\0`. C++ carries this over. Thereby, when using the C++ `string` class, this behaviour remains. As it is treated as part of the string, it is furthermore included in the length calculation and therefore transmitted over LoRa. The receiver would then interpret the characters of the packet in `utf-8`, and translate the null character as `\x00`. The existence of null characters introduces difficulty when attempting to interpret the file in CSV format.

The eventual solution was to perform packet processing on receipt, as cleaning the files manually had thus far failed. The inbuilt `rstrip()` method on a string type allows a specific character (in this case, `\x00`) to be removed.

Additionally, supplementary information, like errors, were logged to console with extra diagnostic information instead of stored in the file. Beforehand, this information would have to be stripped as the error line was non-standard for the CSV format²⁰. An example is shown in table 2.3.

Date	Time	Packet	Longitude	Latitude	Voltage
23/01/27	20:27:27	3040	0	0	4.15
23/01/27	20:27:32	3041	0	0	4.15
23/01/27	20:27:35	unicode parse error			
23/01/27	20:27:32	3041	0	0	4.15
23/01/27	20:27:37	3042	0	0	4.157
23/01/27	20:27:42	3043	0	0	4.15

Table 2.3: Non-standard format example - 230127.csv sample

This would cause errors for the CSV interpreter due to mismatched field lengths.

Finally, plotting temporal data types proved to be problematic. To simplify this, two changes were made: firstly, the Date and Time fields were collated into one field in the standard format MATLAB expects²¹; secondly, a field was added for recording time relative to Unix epoch (i.e. now completely numeric and linear) in case plotting with such a field proves simpler.

The tests were then performed a second time.

2.2.4.3 Results

Data samples of the results can be found in table E.2 and table E.4. The resulting plot is shown in fig. 2.7 using code in listing F.3²².

¹⁹In two byte hexadecimal representation as an escape character, `\x00`.

²⁰The producing code can be found in listing F.11.

²¹Format: 1st February 2023 at 1:54pm as "01-02-23 13:54:00".

²²This is the second version of plotting code. The first can be found in listing F.2.

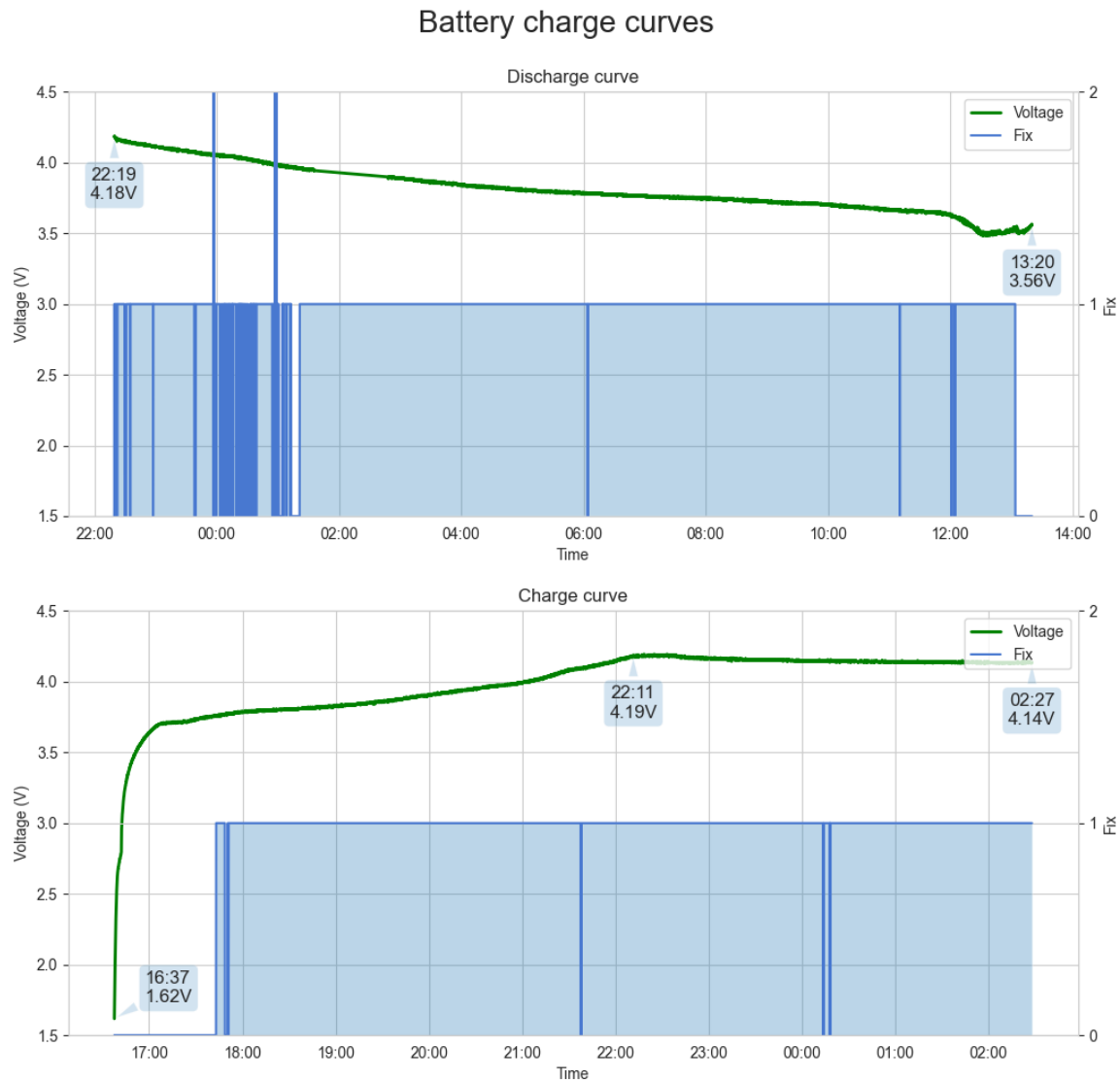


Figure 2.7: Battery charge and discharge curves

The plots show the battery voltage level and whether the GPS has a fix.

Discharge Recording starts with a full battery, indicated by 4.2V. Operation continues as expected, with GPS location fixed for much of that time. On two occasions, the fix was recorded at a value of 6²³. GPS was not unavailable for long enough to indicate whether not having a fix has any significant impact on battery consumption. Overall, the data indicates the total useable span of the device is fifteen hours.

Charge Notably, the voltage at the point of charging is much lower than was last recorded when transmission during discharge cut off. The board does in fact continue to draw power. This ceases when the protection circuitry of the battery cuts supply entirely. This leads to some parasitic draw in the elapsed time. GPS location fix did not have a noticeable effect upon charging. Total time to charge to full capacity was around five hours and thirty minutes.

²³Reference in section 2.2.4.

Remarks The discharge results were surprising. The estimate calculation section 2.2.2.1 returned an expectation of eight hours of function. Despite being a conservative estimate, the experimentally found value was almost double this. Furthermore, additional power saving functions like putting the radio to sleep when not in use were not used, meaning the operating time of the device can be further increased²⁴.

The time required to charge the module to maximum capacity was greater than expected. The slow charging time may come down to a few reasons. The charging regulator built into the Feather board is current limited to protect the circuitry (52). It is further limited by the maximum current a USB 2.0 connector can draw (100mA) (60). This can be negotiated to be greater (61), however, and certainly many modern devices, particularly mobile phones, utilise some form of rapid charging.

The data collected suggests reliability. Any moment-to-moment fluctuations will be smoothed out due to the operating time of the module, so further tests may not be necessary. Furthermore, it is not advisable to repeatedly charge and discharge lithium cells from maximum to minimum as doing so reduces the lifespan of the cell.

2.2.5 Error investigation

Further insight may be gained on the corrupted packets²⁵. The errors recorded before the discharge test was performed can be found in listing G.1. It is notable that there are fewer errors here than in previous tests. The only significant change was the use of antennae during transmission. To elaborate on the error message, many of the bytes in the byte array cannot be decoded as they are all invalid `utf-8`, not just the start byte. The structure should follow 8-bits per character, yet some are greater (`\x90w`, line 13), use invalid hexadecimal (`\xb8B`, line 13), or are entirely not present (`\x89\22Rs`, line 4).

- Invalid hexadecimal - This appears due to the way Python `bytearray` is used. Where the default decoding (`utf-8`) failed, the array is encoded as `ASCII` and printed to console. This leads to non-hexadecimal characters appearing, and likely some of the missing characters also.
- Not 8-bits - If, for example, during transmission part of a byte is lost, the remainder of the packet will fail to decode correctly and appear in different lengths and otherwise non-standard form.

Solution There is no simple solution to what is essentially the question of how to ensure data integrity during transmission in a noisy environment. The RadioHead library supports packet confirmations with the `RHReliableDatagram` class. This does not validate the packet integrity²⁶, but ensures that the packet has arrived. This solution alone would require some reverse engineering of the `recvfromAck` method in order to use it on the receiver end, however can be used to request retransmission (i.e. fail to acknowledge the packet) should it arrive corrupted.

²⁴Transmission was set at 5s, which is better than specified in section 2.1.1.1.

²⁵Section 2.2.3.1.

²⁶A single bit being flipped may still resolve to a valid packet, even if the data is no longer accurate.

An alternative would be to utilise some form of forward error correction, like parity checking. This has the limitation of increased complexity and computational load on the transmitter, and is further limited by being able to correct only one error. To adequately cover the entire transmission, the packet size (and subsequently, radio power draw and processor time) would increase.

The corrupted packets are few and infrequent enough (only one at a time) that they can simply be omitted. As reporting is set to 5s, up to five packets can fail in a row for the specification of 30s reporting to be met. Moving the device to production would require a revisit of handling data integrity.

2.2.6 Case design

In preparation for the distance test, ensuring the electronics were, at minimum, protected from damage increased in importance. The case will be attached to a cat harness or collar as a complete unit (i.e. sewn in), however this level of integration will be paused until the hardware is moved onto a PCB with smaller form-factor. Instead, the purpose of this case design will be to provide a protective enclosure for the period of function testing the tracker.

Figure 2.8 shows the board when development of the case started (especially showing the stacked height). As can be seen, the connecting wires between the battery and board were a potential point of failure. Furthermore, in the process of disengaging the battery repeatedly to remove power and halt battery drain (due to no current off-switch available), the wire sheath had pulled back to expose the stranded core²⁷.

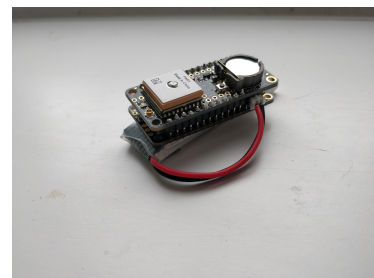


Figure 2.8: Board and battery at the beginning of case development

Two key factors drove considerations for the case development:

- Due to mismatched sizing between the battery and board, both had to be enclosed, so they would not rock about in the casing.
- Given the stage in development and likelihood of frequent easy access, the components were separated to best allow easy access, even at the expense of the smallest volume possible.

2.2.6.1 First case

Design The case was modelled in Autodesk Fusion 360 after measurements were taken of the components. A raised battery platform with rounded edges was created to hold the battery. The board takes up noticeably less space and therefore a cage was modelled to house it securely. An arch was included to allow the battery connection port to remain available. The lid was designed to fit in the casing recess. Due to the tendency for FDM parts to widen when laid, a fillet was included in the lid lip to allow room for this. Extra space for the power cabled to come out of the front of the battery was provided. Finally,

²⁷ This became bad enough that a makeshift cover made from insulating tape was added to ensure the wires did not short circuit - as can be seen most clearly in fig. C.1c.

holes were added on all vertical sides to allow an antenna to be routed through wherever most convenient.

Dimensional drawings for the case can be found in fig. C.2.

Manufacture The manufacturing method of choice was 3D printing using an Ultimaker 3 FDM printer. In order to print the casing, it would first have to be ‘sliced’ and encoded into G-code. The software chosen to do this was Ultimaker Cura, as it supported Ultimaker printers natively. Default ‘fast’ settings were used as the casing was a prototype and therefore dimensional precision is of less importance. Due to large overhangs in the board enclosure, especially at the battery port, the head movement speed was increased by 15% in order to bridge the gaps. Doing so increases the risk of small features like the pillars to fail due to insufficient time for the layer to cool. The increased bridging capability meant that supports could be disabled, however the gap for the battery port would have been too large to bridge without supports, and was therefore modified into an arch to support itself.

Result The first print failed due to an extruder blockage towards the end of the first layer being completed. As this occurred after the period of observation to ensure successful printing, the issue was only apparent the next day. The result can be seen in fig. 2.9.



Figure 2.9: Failed print

Immediately, a second print was started, but with the print moved to the second extruder. Images of the board in the successful second print can be found in fig. C.1.

Review The case turned out to exceed expectations in quality. The battery platform held the battery in place surprisingly well (likely largely owing to the material of the battery casing). The board enclosure also worked very well to secure the board. The lid also fit quite securely, indicating the fillet

worked effectively.

As expected, the pillars widened out somewhat. Despite this, the enclosure top rail was in very good condition, especially including over the battery port - the largest gap. Relatedly, the arch turned out to be a hindrance and too troublesome to connect the battery wires through, and was therefore cut off instead. It was also not centred on the battery port, due to a mistake during the design process where the measurement was to the centre of the port, whereas the design was dimensioned to the edge of the port only. Enough space was granted either side of the port to account for such an error, and therefore came in use. The cage as a whole did not prove useful, and only made removing the board from the casing more tedious. It also was a blockage when routing the power cables, leading to no adequate way to place them without straining them.

2.2.6.2 Second case

Design Considering the issues raised in review of the first case design, some changes were made. The overall style of the first design was retained, with the battery and board

placed side-by-side. While resoldering the board for a flatter profile was preferred, this was avoided to prevent any unforeseen issues in advance of the distance test and would be revisited once testing was complete.

Extra space provided for front-facing egress of power cables from the battery was removed from the design, amounting to 204mm² saved. To compensate this, a channel was cut into the battery platform for the cables to run.

A major omission from the first design was the charging port, which was now included in this revision. The antenna hole diameter was doubled to 5mm.

The board enclosure was removed, and a small fence was used to replace it.

The dimensional drawings can be found in fig. C.4.

Manufacture The same manufacturing process was used as in the first case²⁸. A finer print setting was used as the print was being left overnight, and therefore the greater resolution would have no impact on turnaround time. This setting is part of Cura's default presets, with no modifications made to fine-tune it.

Result The print turned out extremely well. Images can be found in fig. C.3.

Review The first case was not tested with any antennae, so two issues occurred that were not accounted for. Despite the antennae holes being enlarged, they were still too small. This may have been due to collapse of the layers as there were no supports (leading to oval holes that were too small for the U.FL connector). The board 'enclosure' interfered with the GPS antenna wire. This can be seen more clearly in fig. C.3c, and overall means that the board cannot sit flat. The cable will require a relief channel of its own like the battery wires.

This case is sufficient for testing with, even if the antennae have to be routed out the lid. The casing with all components weighs in at 55g.

There are some aspects to be mindful of when designing a third iteration. If a flat form factor layout can be used by resoldering the boards, there is potential to allow for a curved, and thereby more ergonomic design, however strain across the connection must be accounted for.

Attaching to the harness was not considered at all during either of these designs, and will be a major factor for how guiding the design. Similarly, general weatherproofing was not included. A simple and effective method of waterproofing would be to use an o-ring as a seal, however this may require more pressure than the plastic (in this case, PLA, which is extremely flexible) is able to provide. This may be mitigated somewhat by using a more rigid material (for example, ABS or PC) and using screws to secure the lid and apply the required amount of pressure to form an adequate seal.

Another omission was adequate antenna support. For that reason, during test a flat support material will be used to protect the antennae. However, this will be an important factor

²⁸Section 2.2.6.1.

with further iterations of case design, especially considering interference sources for the antennae being so close to a PCB edge.

2.2.7 Distance test

This test forms a crucial part of determining the suitability of the tracker to perform its key function - tracking a remote target. It is only possible as the culmination of every step before it.

2.2.7.1 Preparation

This test required more preparation than any performed previously.

Location Determining a suitable location for testing was of heightened importance. The general area to perform the test was decided across the campus of the University of Warwick. As a publicly accessible space, setting up the receiving antenna anywhere with thoroughfare would not have been appropriate, however easy access to the Pi to download data would be needed.

In short, the location needed to be easily accessible, but not in a location with many people. This ruled out many of the building rooftops due to potential safety issues and authorisation requirements. The final location decided was room A3.27, located in the School of Engineering²⁹. This location was sufficiently high up and had a clear view of Car Park 9 (CP204) and Lord Bhattacharyya Way. Unfortunately, there was no logistical method with which the antenna could be mounted outdoors from this location, and therefore was confined to being indoors. This would severely impact the capability of the antenna.

Work flow For ease-of-access, being able to view the data log remotely would be most ideal. It also means that, should the connection drop out, this can be noticed immediately by viewing current incoming packets. Remote access requires some manner in which the devices accessed appears in the 'same location'. In this case, it would, in its simplest form, require a static IP assignment. Due to the university network security requirements and address assignment protocol (likely at best a DHCP assignment from a limited pool), there was no ideal way in which the Pi could be assigned a static IP address.

The alternative was to set the Pi and antenna up to record data, and retrieve data afterwards. This comes with the downside that the test can no longer be dynamic. Any issues that occur can only be investigated after the fact.

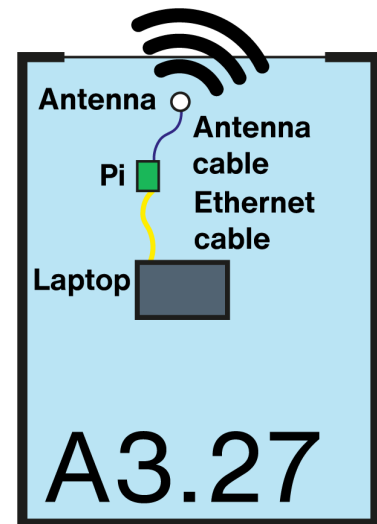
With the work flow determined, the setup has the antenna next to a window while connected to the Pi. A laptop is then connected with an Ethernet cable to the Pi over SSH. Figure 2.10b shows a diagram of this hardware setup.

Container While the transmitter now has appropriate casing, the antenna holes were too small to run the U.FL connector through. Due to time constraints, this could not be remedied, and instead, the antennae were exposed through the lid. This also left them

²⁹Location on map: fig. 2.10a.



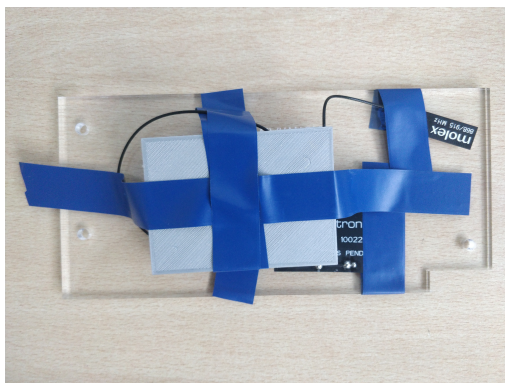
(a) A3.27 location, showing Car Park 9 (CP204)



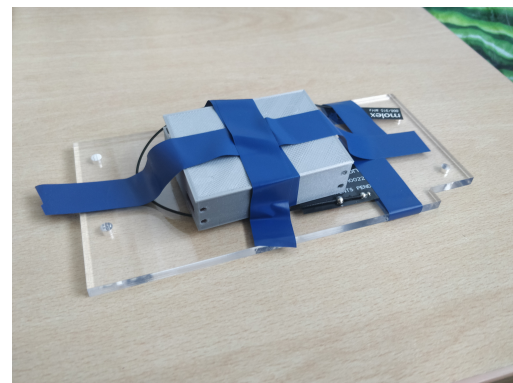
(b) Hardware setup

Figure 2.10: Distance test location and setup

with no support, which may prove a point of failure if left unattended³⁰. To secure the antennae, a scrap sheet of acrylic was used to tape them down, thereby providing them a rigid support surface with a low chance of leading to interference. The final result of the casing as used in testing can be seen in fig. 2.11.



(a) Overview



(b) Angled view

Figure 2.11: Setup used for distance testing

Data An example of data recordings can be seen in fig. 2.12. As mentioned previously³¹, the output data is in a non-standard format, an example of which is given in table 2.4. In order for this data to be plotted, it needs to be converted into a standardised format, as per table 2.5.

³⁰Figure C.3b shows how the antennae splay.

³¹Section 2.2.1.2.

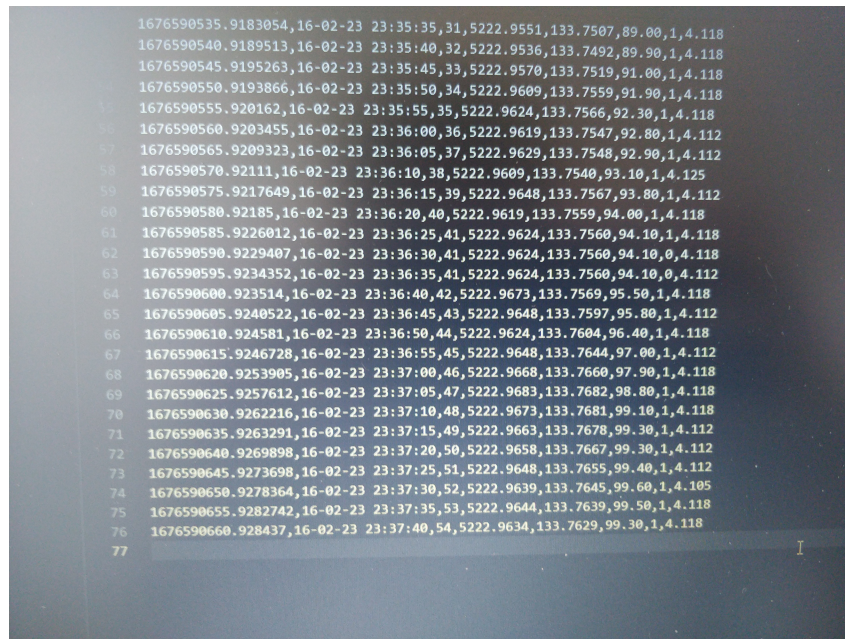


Figure 2.12: Test data recordings

The first step is to add direction indicators. This supports portability, but in this case will simply be used to determine whether the direction is positive (North or East) or negative (South or West). While the GPS library is able to provide directional information (45), in this instance, the output files were simply modified to include a column of the direction, since only locations North and West were recorded for any tests. The modified columns in table 2.4 are highlighted in blue.

Latitude	Direction	Longitude	Direction
5222.9409	N	133.7407	W
5222.9409	N	133.7407	W
5222.9409	N	133.7404	W

Table 2.4: Non-standard location data format, with manually added columns highlighted

Latitude	Longitude
52'22.958	-1'33.76
52'22.959	-1'33.7594
52'22.9565	-1'33.7566

Table 2.5: Desired location data format

With sufficient information for processing, a script was then written to convert the given format into standardised degrees-minutes format, as per table 2.5, the code for which can be found in listing F.4. The script works by processing the location columns of each row. It performs a regular expression match to extract the minutes from the degrees, and return both in an array [lines 6-7]. It then determines the polarity by the relevant direction column, and adds the degree separator (') as necessary [lines 12-14]. Finally, it writes the new format data to a new file, which can then be used for plotting.

Plotting The plotting software of choice was QGIS, a free and open-source GIS software package. While QGIS has many advanced features, one of the most useful in this case is the ability to lay GPS coordinates over a map. It furthermore accepts many formats of location data, including KML markup, which will be important as part of the verification

step.

2.2.7.2 Test specification

Due to blind recording, the test is carried out with no knowledge as to how successful it is. The transmitter was powered on by connecting the battery, and the script on the receiver³² was started. The tracker was then carried around the predetermined route. At the same time, an app based GPS tracker was started, which records location data independently. When the route was complete, the data was downloaded for later processing and comparison to the externally recorded data.

Risks Inclement weather may cause a problem for the circuitry, as the casing is not waterproofed. Otherwise, only typical precautions when travelling outdoors near roads are necessary.

2.2.7.3 Test one

The route was determined to include walking behind the School of Engineering and around Lord Bhattacharyya Way while holding the transmitter. This route covered a maximum straight line distance of 175m and involved parts with direct line of sight to the receiver, and large sources of interference in other areas (e.g. buildings, antennae, girders, etc).

This test was performed with perhaps what could be considered the worst possible conditions, and success from hereupon would determine whether an altered setup was required. To check suitability, after the test was run, the data logs would be checked to see whether there was consistent location information. If so, a second test at a greater distance would be performed. If not, the setup would be revisited.

Results After the test was performed, the logs appeared to indicate successful transmission, although this could not be analysed at the time. As such, a second test at a much greater distance was performed immediately afterwards.

Figure 2.13 shows the recorded points in red³³ and the externally recorded route in blue³⁴ after processing. As can be seen, the results are well-matched. Despite the extremely poor testing conditions, the tracker operated surprisingly well.

A cluster of points next to Physical Sciences, is likely points recorded in A3.27, but offset due to inaccuracies in location while indoors. A smaller cluster next to Car Park 10 is due to a slightly longer period of time spent in that location.

There are no noticeable errors or omissions present. The location data recorded by the tracker also appears to be more accurate to the precise route followed than is indicated by app-based tracker.

2.2.7.4 Test two

Given the apparent positive results of the first test, the second test was devised to cover a much greater distance. In this case, the route consisted of Kirby Corner Rd. and Charter

³²Listing F.12.

³³Data sample in table E.6.

³⁴Data sample in listing E.1.



Figure 2.13: Location recordings with route overlay

Ave. in a large triangle, with the greatest distance 1.2km away. This test was devised in this manner to cover the most extreme capabilities of the tracker, since it is impossible to determine live whether a range is too great.

The route includes a number of points that would be in 'line of sight' (were it not for foliage), in addition to areas of dense foliage and building obstructions.

Due to the distances involved, the mode of transportation chosen was bicycle. This would at least affect the distance between reported locations. It is also possible that transmissions would be affected due to the speed. Furthermore, the tracker could no longer be held and would have to be carried in a pocket, which may affect transmission capabilities.

Results After testing, the logs appeared to have much fewer records than expected³⁵. Furthermore, the app-based tracker failed to work. In fig. 2.14, the travelled route was drawn on afterwards. Noticeably, there are far fewer points plotted, and of that, the furthest was 436m away with the majority of the route missing. This position did not have clear sight, however it was a position with the 'least' obstruction at that distance. The points were spaced out a greater distance, which was expected.

Focusing on the points recorded on Lord Bhattacharyya Way and Library Rd., assuming these points were sequential (5s) transmissions, this indicates an approximate travel speed of 40km/h to 55km/h. This is practically impossible to be the case, therefore suggesting there were either missing points during transmission, or the recorded location is not precise enough to calculate the speed from (where it is 'large').

³⁵Data sample in table E.8.

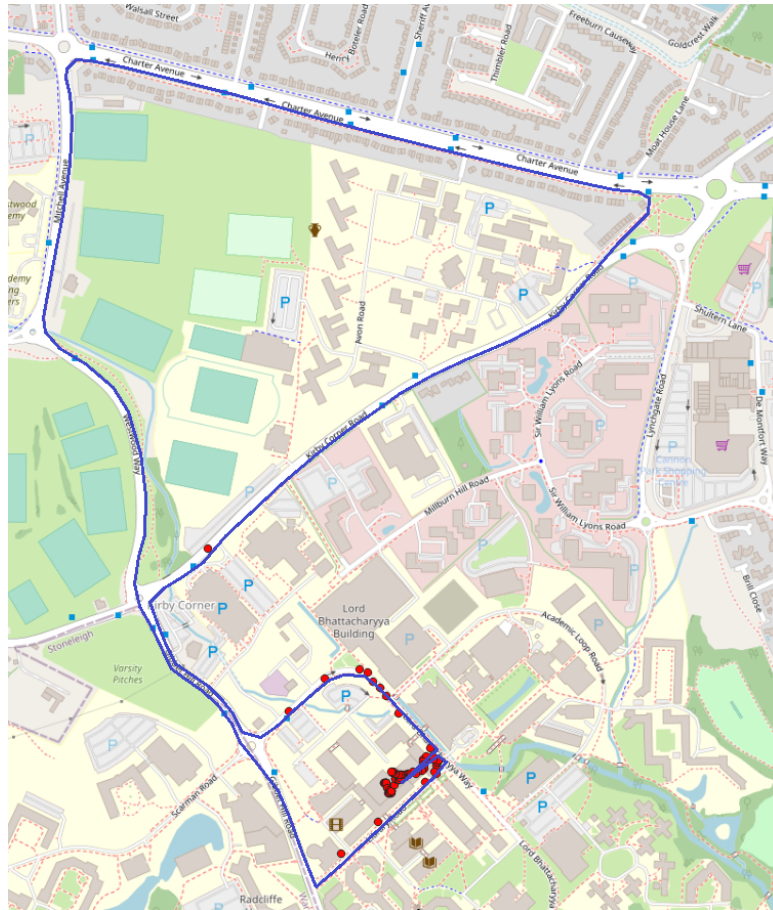


Figure 2.14: Location recordings with route overlay

Due to insufficient reporting, it is impossible to determine whether GPS location was lost at any point. Periodic checks on the device suggest this did not happen³⁶.

2.2.7.5 Comments

Overall, the conclusion to be drawn from this suite of tests is that the tracker is very clearly viable for its purposes, however requires some points of refinement.

Test one went extremely well, with no particular major issues identified. It is most useful in comparison to test two, where almost all the route was not sufficiently tracked. This difference likely comes down to two things - the distance travelled being obviously much greater, and the tracker being held in a pocket.

Regarding distance, the area overlapped by both tests was reported on with acceptable clarity. This suggests the greatest issue was simply difficulty transmitting over range, or the increasing number of interference sources as distance (and perhaps speed of travel) increased.

It is particularly noteworthy that the test conditions were the poorest possible, possible aside from the addition of rain. The receiving antenna was placed indoors where it would preferably be outdoors, and the tracker was held in a pocket where it would also be prefer-

³⁶While searching for a fix, the GPS module will blink its LED every second. If a fix is found, this will increase to every fifteen, and can be used to easily indicate if there is a fix (39).

ably exposed. The general area was heavily built with many sources of interference, inexhaustibly including trees, buildings, and metal structures. While elevated, the surrounding area was heavily built up, with only a small area of space between buildings where the ground (and thereby, tracker) could be seen³⁷. The location in particular, a university campus, suggests that there was also a higher likelihood of RF interference sources.

Finally, the data, while indicative of function, is far from complete. Performing more tests, especially in different conditions, to accurately determine the limits of the tracker conclusively would be advised. The information gleaned from these tests is meaningful, yet incomplete in thoroughness.

³⁷This area is where Car Park 9 is located, however it is still has much foliage in the way.

Conclusions

3.1 Assessment

The project success will be assessed against specified the aims¹ and objectives².

3.1.1 Assessment of objectives

The objectives outline the steps of the project that needed to be followed for success. Analysis of met objectives is available in table 3.1, where success of a give point is determined by whether the objective was met ■, unmet ■, or somewhat met ■.

Objective	Met?
Research cat characteristics	
Research LoRa radio requirements	
Design the system and specification	
Develop the tracker according to specification	Further details can be found in table 3.3.
Perform verification tests	Tests were performed, but were limited in completeness. More were required.
Develop PCB, fit to harness, and live-test	
Certification	

Table 3.1: Objectives evaluation

Colour progression indicates that, while the project started off well, completion began to suffer as time passed.

3.1.2 Assessment of aims and specification

Aim	Met?
Fundamentals	
Obtain GPS location	
Transmit via LoRa	
Transmit remotely for a significant period of time	Up to fifteen hours
Transmit at a regular interval	Every five seconds
Improvements	
Ergonomics	The product is not designed for use with a pet
Stability	Only the receiver handles errors
Installable	No specific tools required for installation, however it does not have appropriate casing to be mounted
Location information easily viewable	Information is viewable, but requires familiarity with Bash
Production	
General safety and suitability	No safety measures in place

¹Section 1.2.2.

²Section 1.2.3.

Aim	Met?
Resistance to damage and weather	Some resistance to damage
Certification requirements and law compliance	
Scale considerations	

Table 3.2: Aims evaluation

Table 3.2 shows that the Fundamental aims have been met, and shows a similar colour progression to table 3.1 in that primary points were met, but later ones were not. By meeting the Fundamental aims, the viability of the product has been demonstrated, and much from thereon is to improve and refine the design for consumer suitability.

3.1.2.1 Assessment of specification

This will provide a more granular overview of which elements of the specification in section 2.1.1 were met.

Specification	Met?
Reporting	
Timing	Performs better than 30s, but no way of minimising transmission gaps to under 300s
Distance	Subject to more thorough testing to find the reliable limit. Current data shows transmission is possible from over 400m, however this is not consistent
Portability	
Small form-factor	Dimensions measure 52mm × 68mm × 17mm ³
Lightweight	Weight is 55g, a 63% improvement
Battery powered	Battery life is 15h, a 67% improvement
Attachable	
Suitability	
Installable	
Compliance	
Reliability	
Stable	Fault handling in receiver code
Fault resistant	Some shock resistance in the casing
Repairable	No proprietary parts or complex tools are used, however technical knowledge is required
Safety	
Deduplication	

Table 3.3: Specification evaluation

The greater detailed view in table 3.3 indicates some points met in the aim may not quite meet the specification, with many such reasons overlapping.

³Figure C.4.

3.2 Improvements

Using this project's findings as a basis, there are many points of specific improvements that can be explored in future projects. These will be broken down into two types: improvements that resolve an issue that was faced during the project, and improvements that will increase the quality of the product.

3.2.1 Issue resolvers

Battery connection There was no way to control the power to the transmitter. As soon as the battery was connected, it would begin to transmit, and more crucially, drain the battery. Repeated disconnections of the power port led to the cables exposing and almost shorting at one point. This was resolved using insulting tape⁴, however such a method is only a temporary measure.

A better way of managing this would be to use a switch across one of the battery lines, allowing power to be killed gracefully. This has a drawback itself in that it would require the user to power on the device in order to recharge the battery. A smarter, user-friendly approach would require discrete charging circuitry to handle charging and power, which is moving towards bespoke circuit design and out of the scope of this project. However, this would be vital towards making this a viable consumer product.

Transmission errors Transmission errors only appeared to occur while the system was used on a close-range basis. However, it is entirely possible for such errors to occur on a general basis. To minimise this, a CRC may be used. Another method of transmission error handling would be to utilise FEC (for example, parity checks), however this may greatly increase the complexity of the project, and likely is not necessary given the frequency rate of errors, packet size, and hardware capabilities. The Radiohead library supports client-server acknowledgement with the `RHReliableDatagram` class, which allows checking that a packet has been received correctly, and if not, retransmit. This can be additionally used to notify when the tracker is no longer able to transmit to the server (as it would fail to acknowledge the server's checks).

The `RHDatagram` superclass additionally allows for addressing, meaning nodes (receivers) are numbered with an address 0 to 255 and only packets addressed to that specific node are accepted, thereby filtering out packets from 'other' transmitters. This meets the de-duplication specification point⁵, however suffers the obvious drawback of being limited by address pool. An alternative method would be to encode addresses into the packet information manually rather than relying on a provided class, thereby allowing the address pool criteria to be virtually any size. Whatever method is chosen, the transmitters will also be able to encode a unique address, allowing an owner to track multiple pets.

Charging time While the transmitter may boast an impressive battery life, the charging time leaves much to be desired. Protocols like Qualcomm's Quick Charge may mitigate

⁴The taping can be seen in fig. C.1.

⁵Section 2.1.1.4.

this, however it is limited to Qualcomm SoCs. This works by negotiating a higher current over the USB connection, as the default is limited to 500mA (60).

3.2.2 Quality improvements

Display The Pi LoRa bonnet comes with an OLED display that was completely unutilised. This could be used to display warnings to the user to indicate when GPS is lost or the battery levels are low.

Advanced power management The Feather M0 supports underclocking, meaning that battery life could be greatly improved. The LoRa modem also supports commands to sleep, as otherwise it stays in a passive ‘listening’ mode that draws some power (around 2mA (52)). Since it is only required to transmit, the radio can be turned off to save power in the meantime.

3.3 Additional remarks

Some aspects have not been discussed in any great detail. Here, such aspects will be focused upon.

Environmental impact The project does not have an explicit environmental impact. Manufacture of electronics in general is an environmental burden, as is safe disposal, and this project certainly falls within the purview of such elements. Such considerations will be vital should this project be taken to manufacture, as safe disposal of electronics and batteries in particular will be necessary.

Cost The overall cost of the project is not entirely straightforward due to the scattered and chaotic nature of the ordering. All parts used in the project ordered through approved vendors totals to £198.11⁶. This cost does not include some components used (i.e. the overall cost of the project is greater), the most significant of which is the Pi 3. While this retails for £40⁷, it was often seen listed on Amazon or eBay with a markup over 120%.

3.4 Evaluation

The project was, overall, reasonably successful. The fundamental aims were met, and many of the objectives outlined were followed through. Many of the advanced aims were not met, however. In retrospect, some of these were rather ambitious for the timescale available, however are certainly worth revisiting and exploring in some depth in future projects built upon this one.

One of the key takeaways is the importance of planning around expected and unexpected issues. While hardware delivery delays certainly held the project up significantly, having a mitigation plan in place such that progress in some area, whether that be more advanced design work, or a greater depth of research, would have been beneficial. This also assumes that alternative hardware cannot be used at all, whereas in many cases, it potentially may be.

⁶Before VAT.

⁷Available at The PiHut [↗](#)

One of the greatest outcomes was the successful function of the tracker despite the tight time scales. With a small amount of refinement weatherproofing the case, the tracker is perfectly sufficient for function, even if it is not at the point where it is suitable for consumer use. This project can be used as an excellent springboard upon which to explore building a genuinely usable product.

3.4.1 Difficulties

The overall trend indicated by the assessments performed is that many of the early points for the project were met, however later and more advanced points were not. The primary limiting factor to progression was time constraints, with the biggest contributor being delivery delays. As a product development focused project, there was a limited amount of work possible without any hardware to develop on. When hardware did finally arrive, it was incomplete, leading to haphazard development missing components. Delays in antennae delivery (including the receiver antenna in particular), shifted distance testing to the final step performed, and limited the possibility of additional tests. This has the knock-on effect leaving an incomplete picture of the true capabilities of the tracker, and also shifting every subsequent step (like PCB design) to the point where they were no longer viable to tackle

In general, development for this project moved rather quickly, with the bulk of programming to produce a minimum viable product complete in the span of a week. Hardware delivery issues were expected, given global semiconductor shortages, however the true extent of delays were not adequately accounted for and thus led to severe delays in the project itself.

3.4.2 Admittances

The process of code development progressed extremely smoothly. Prior experience with the relevant libraries helped in this regard. Development could have proven extremely difficult, given the unfavourable nature of diagnosing RF issues. This project made use of many more scripts than anticipated, most of which were for data handling or graphing than direct functionality. The programmatic nature made this all very straightforward to design and use.

References

1. Cats Protection, Online; accessed 20 October 2022, (<https://www.cats.org.uk/media/11908/cats-report-2022-uk.pdf>) (2022).
2. C. Davis, *BBC News*, Online; accessed 20 October 2022, (<https://www.bbc.co.uk/news/uk-england-birmingham-51738246>) (6th Mar. 2020).
3. Research and Markets, (<https://www.researchandmarkets.com/reports/5615579>) (May 2022).
4. J.-P. Bardyn, T. Melly *et al.*, presented at the ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference, pp. 25–30.
5. M. Eldefrawy, I. Butun *et al.*, *Computer Networks* **148**, 328–339, (<https://www.sciencedirect.com/science/article/pii/S1389128618306145>) (2019).
6. H. J. Hanmer, R. L. Thomas *et al.*, *Journal of Urban Ecology* **3**, (<https://doi.org/10.1093/jue/jux014>) (Dec. 2017).
7. D. G. Barratt, *Ecography* **20**, 271–280, (<https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1600-0587.1997.tb00371.x>) (1997).
8. P. D. Meek, *Australian Mammalogy* **25**, 51–60 (1 2003).
9. A. J. Bengsen, D. Algar *et al.*, *Journal of Zoology* **298**, 112–120, (<https://zslpublications.onlinelibrary.wiley.com/doi/abs/10.1111/jzo.12290>) (2016).
10. J. A. Horn, N. Mateus-Pinilla *et al.*, *The Journal of Wildlife Management* **75**, 1177–1185, (<https://wildlife.onlinelibrary.wiley.com/doi/abs/10.1002/jwmg.145>) (2011).
11. R. Molsher, C. Dickman *et al.*, *Wildlife Research* **32**, 587–595 (2005).
12. E. A. Clancy, A. S. Moore *et al.*, *Journal of the American Veterinary Medical Association* **222**, 1541–1545, (<https://avmajournals.avma.org/view/journals/javma/222/11/javma.2003.222.1541.xml>) (2003).
13. Z. Zhang, Y. Li *et al.*, *Animals* **12**, 1141, (<http://dx.doi.org/10.3390/ani12091141>) (Apr. 2022).
14. L. R. Finka, J. Ward *et al.*, *en, PLOS ONE* **14**, ed. by I. A. S. Olsson, e0211862, (<http://dx.doi.org/10.1371/journal.pone.0211862>) (Feb. 2019).
15. D. Loukatos, A. Fragkos *et al.*, in *Information and Communication Technologies for Agriculture—Theme I: Sensors*, ed. by D. D. Bochtis, M. Lampridi *et al.* (Springer International Publishing, Cham, 2022), pp. 101–120, (https://doi.org/10.1007/978-3-030-84144-7_4).
16. J. C. Liando, A. Gamage *et al.*, *ACM Trans. Sen. Netw.* **15**, (<https://0-doi-org.pugwash.lib.warwick.ac.uk/10.1145/3293534>) (Feb. 2019).
17. A. Augustin, J. Yi *et al.*, *Sensors* **16**, (<https://www.mdpi.com/1424-8220/16/9/1466>) (2016).
18. L. Vangelista, *IEEE Signal Processing Letters* **24**, 1818–1821 (2017).
19. O. B. A. Seller, US9647718B2 (2015-09-09).
20. EU, (https://eur-lex.europa.eu/eli/dec_impl/2011/829/oj) (8th Dec. 2011).
21. Ofcom, *IR 2030 - UK Interface Requirements, Licence Exempt Short Range Devices*, Apr. 2021, (https://www.ofcom.org.uk/_data/assets/pdf_file/0028/84970/ir-2030.pdf).
22. J. Pengtao, CN210090660 (2019).
23. C. Xifa, CN209911563 (2019).
24. D. Davcev, K. Mitreski *et al.*, presented at the 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS), pp. 1–4.
25. S. Sivaraman, A. A. Zainuddin *et al.*, presented at the 2021 International Conference on Green Energy, Computing and Sustainable Technology (GECOST), pp. 1–4.
26. Tractive, *Pick your Tractive plan*, Online; accessed 20 October 2022, (<https://tractive.com/en/c/plans>).
27. Tractive, *GPS Tracker for Cats*, Online; accessed 20 October 2022, (<https://tractive.com/en/pd/gps-tracker-cat>).
28. Tractive, *Privacy Policy of Tractive GMBH*, Online; accessed 20 October 2022, (<https://assets.tractive.com/static/legal/en/privacy-policy.html>).
29. Tabcat, *Tabcat Cat tracker device*, Online; accessed 7 October 2022, (https://tabcat.com/shop/tabcat-v2/?sscid=a1k6_6pjj5).
30. Tile Inc., *How It Works*, Online; accessed 02 March 2023, (<https://uk.tile.com/how-it-works>).
31. P. Haskell-Dowland, *Remember, Apple AirTags and 'Find My' app only work because of a vast, largely covert tracking network*, Online; accessed 02 March 2023, 16th May 2021, (<https://theconversation.com/remember-apple-airtags-and-find-my-app-only-work-because-of-a-vast-largely-covert-tracking-network-160781>).
32. B. Lovejoy, *AirTag stalking 'frighteningly easy'; multiple problems identified*, Online; accessed 02 March 2023, 6th May 2021, (<https://9to5mac.com/2021/05/06/airtag-stalking/>).
33. E. Kienzie, K. Moik, *British Journal of Nutrition* **106**, S113–S115 (2011).
34. G. E. Blomgren, *Journal of The Electrochemical Society* **164**, A5019, (<https://dx.doi.org/10.1149/2.0251701jes>) (Dec. 2016).
35. Y. Chen, Y. Kang *et al.*, *Journal of Energy Chemistry* **59**, 83–99, (<https://www.sciencedirect.com/science/article/pii/S2095495620307075>) (2021).
36. R. Bischof, N. R. Hansen *et al.*, *Scientific Reports* **12**, (<https://doi.org/10.1038/s41598-022-09694-9>) (25th Mar. 2022).

37. HOPERF, *RFM95/96/97/98(W) - Low Power Long Range Transceiver Module V1.0*, HOPE MICROELECTRONICS CO.,LTD (2/F, Building 3, Pingshan Private Enterprise Science and Technology Park, Lishan Road, XiLi Town, Nanshan District, Shenzhen, Guangdong, China, 2006).
38. S. Fitzgerald, A. Guadalupi *et al.*, *Blink.ino*, Commit: 4bbbf5c13a3c40e4435c9e659247e5fcbad2c3e6, 3rd Oct. 2022, (<https://github.com/arduino/arduino-examples/blob/main/examples/01.Basics/Blink/Blink.ino>).
39. L. Fried, *Adafruit Ultimate GPS featherwing*, Online; accessed 24 October 2022, (<https://learn.adafruit.com/adafruit-ultimate-gps-featherwing?view=all>).
40. M. G, L. Fried, *GPS_HardwareSerial_EchoTest.ino*, Commit: 22c229d7bf5eac49bb4d6b296cf451b72f33e8df, 8th July 2022, (https://github.com/adafruit/Adafruit_GPS/blob/master/examples/GPS_HardwareSerial_EchoTest/GPS_HardwareSerial_EchoTest.ino).
41. G. Baddeley, *GPS - NMEA sentence information*, Online; accessed 2023-02-24, 20th Aug. 2001, (<http://aprs.gids.nl/nmea/>).
42. NovAtel, *GPS, GPS specific information*, Online; accessed 26 February 2023, Dec. 2022, (<https://docs.novatel.com/OEM7/Content/Logs/GPRMC.htm#NMEAPositioningSystemModeIndicator>).
43. G.top, *FGPMMPA6H GPS Standalone Module Data Sheet*, Online; accessed 23 October 2022, GlobalTop Technology Inc. (No.16 Nan-ke 9th Rd, Science-Based Industrial Park, Tainan, 741, Taiwan, R.O.C., 2011), (<https://cdn-shop.adafruit.com/datasheets/GlobalTop-FGPMMPA6H-Datasheet-V0A.pdf>).
44. G.top, *PMTK command packet*, Online; accessed 26 October 2022, GlobalTop Technology Inc. (No.16 Nan-ke 9th Rd, Science-Based Industrial Park, Tainan, 741, Taiwan, R.O.C., 24th Sept. 2012), (https://cdn-shop.adafruit.com/datasheets/PMTK_A11.pdf).
45. R. Sellens, L. Fried *et al.*, *Adafruit_GPS*, Online; accessed 26 February 2023, 2022, (https://github.com/adafruit/Adafruit_GPS).
46. R. Sellens, *Adafruit_PMTK.h*, Commit: 4bfbd03b90b1d019f8a371e0093a46239265691f, 19th Jan. 2020, (https://github.com/adafruit/Adafruit_GPS/blob/master/src/Adafruit_PMTK.h).
47. M. LeBlanc-Williams, *CircuitPython on Linux and Raspberry Pi*, Online; accessed 27 October 2022, (<https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/installing-circuitpython-on-raspberry-pi>).
48. K. Rembor, *Adafruit Radio Bonnets with OLED Display - RFM69 or RFM9X*, Online; accessed 24 October 2022, (<https://learn.adafruit.com/adafruit-radio-bonnets?view=all>).
49. B. Rubell, E. Herrada, *rfm9x_check.py*, Online; accessed 27 February 2023, 30th Sept. 2021, (https://github.com/adafruit/Adafruit_Learning_System_Guides/commits/main/pi_radio/rfm9x_check.py).
50. B. Rubell, E. Herrada, *radio_rfm9x.py*, Online; accessed 27 February 2023, 30th Sept. 2021, (https://github.com/adafruit/Adafruit_Learning_System_Guides/blob/main/pi_radio/radio_rfm9x.py).
51. M. McCauley, *RadioHead Packet Radio library for embedded microprocessors*, Online; accessed 28 October 2022, (<https://www.airspayce.com/mikem/arduino/RadioHead/>).
52. L. Fried, *Adafruit Feather M0 Radio with LoRa Radio Module*, Online; accessed 20 October 2022, (<https://learn.adafruit.com/adafruit-feather-m0-radio-with-lora-radio-module?view=all>).
53. L. Fried, S. Barnes *et al.*, *Feather9x_TX.ino*, Online; accessed 27 February 2023, 8th Mar. 2022, (https://github.com/adafruit/RadioHead/blob/master/examples/feather/Feather9x_TX/Feather9x_TX.ino).
54. M. McCauley, *RH_RF95 Class Reference*, Online; accessed 28 October 2022, (https://www.airspayce.com/mikem/arduino/RadioHead/classRH_RF95.html).
55. Arduino, *What is Arduino?*, Online; accessed 27 February 2023, (<https://www.arduino.cc/en/Guide/Introduction>).
56. Arduino, *Language Reference*, Online; accessed 27 February 2023, (<https://www.arduino.cc/reference/en/>).
57. KCX, *Power Bank capacity tester User Manual*, Online; accessed 20 January 2023, KCX, (<https://www.solrobotics.com/wp-content/uploads/kcx-017-power-bank-testing.pdf>).
58. Adafruit, *JST PH 2-Pin Cable - Female Connector 100mm*, Online; accessed 30 January 2023, (<https://www.adafruit.com/product/261>).
59. ArcGIS, *GpsFixStatus Enumeration*, Online; accessed 03 March 2023, 2011, (<https://help.arcgis.com/en/arcgismobile/10.0/apis/arcgismobile/api/ESRI.ArcGIS.Mobile-ESRI.ArcGIS.Mobile.Gps.GpsFixStatus.html#:~:text=The%20fix%20status%20indicates%20the,of%20the%20location%20being%20reported.>).
60. J. Axelson, *USB Complete: The Developer's Guide, Fifth Edition* (Lakeview Research LLC, 2015).
61. FTDI, *Simplified Description of USB Device Enumeration*, Online; accessed 03 March 2023, Future Technology Devices International Ltd., (https://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_113_Simplified%20Description%20of%20USB%20Device%20Enumeration.pdf).
70. E. Herrada, L. Fried *et al.*, *Adafruit Feather M0 RFM9x Pinout.pdf*, [Online; accessed 11 March 2023], 19th Apr. 2022, (<https://github.com/adafruit/Adafruit-Feather-M0-RFM-LoRa-PCB/blob/master/Adafruit%20Feather%20M0%20RFM9x%20Pinout.pdf>).

Appendices

Further Reading

62. D. Machado, A. Bragança *et al.*, *Animal Welfare* **30**, 331–339 (2021).
63. R. Foreman-Worsley, L. R. Finka *et al.*, *Animals* **11**, 253, (<http://dx.doi.org/10.3390/ani11020253>) (Jan. 2021).
64. H. W. McGregor, S. Legge *et al.*, *Wildlife Research* **42**, 223–231 (3 2015).
65. A. J. Bengsen, J. A. Butler *et al.*, *Wildlife Research* **39**, 258–265 (3 2012).
66. D. F. Horwitz, I. Rodan, *Journal of Feline Medicine and Surgery* **20**, PMID: 29706091, 423–436, (<https://doi.org/10.1177/1098612X18771204>) (2018).
67. Cats Protection, Online; accessed 20 October 2022, (https://www.cats.org.uk/media/1023/eg12_indoor_and_outdoor_cats.pdf).
68. Ofcom, *Consultation response form, Notice of Ofcom's proposals for changes to the licence exemption for Wireless Telegraphy Devices*, (https://www.ofcom.org.uk/__data/assets/pdf_file/0021/196131/lora-alliance.pdf).
69. Arduino, *Arduino Pro IDE (alpha preview) with advanced features*, Online; accessed 20 October 2022, (<https://blog.arduino.cc/2019/10/18/arduino-pro-ide-alpha-preview-with-advanced-features/>).

Pinout Diagrams

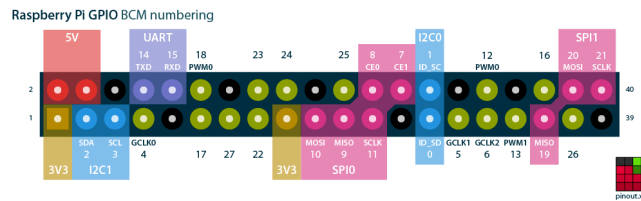


Figure B.1: Raspberry Pi pinout, available at pinout.xyz

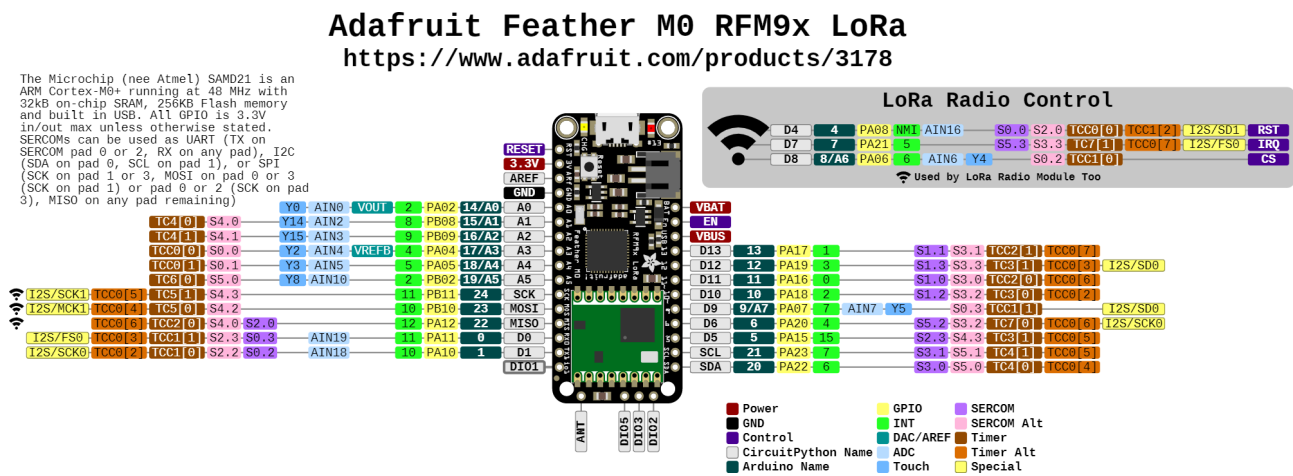
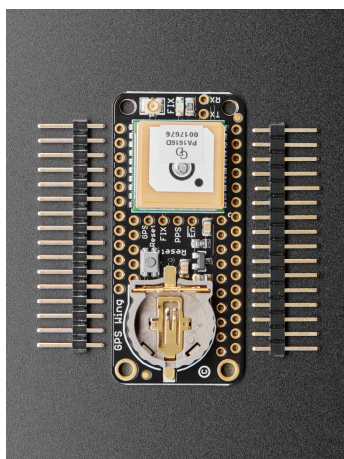
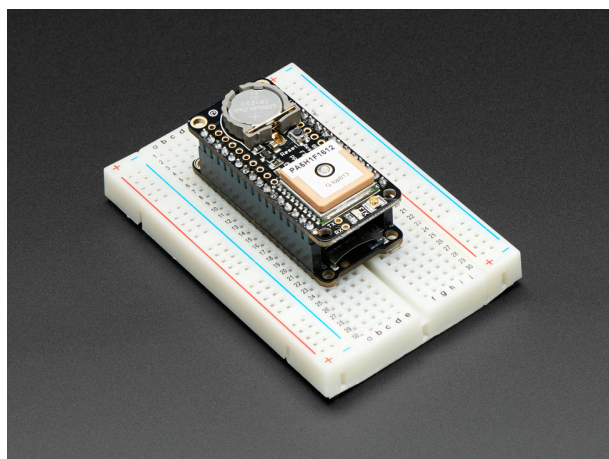


Figure B.2: LoRa Feather M0 pinout (70)



(a) GPS Featherwing pins



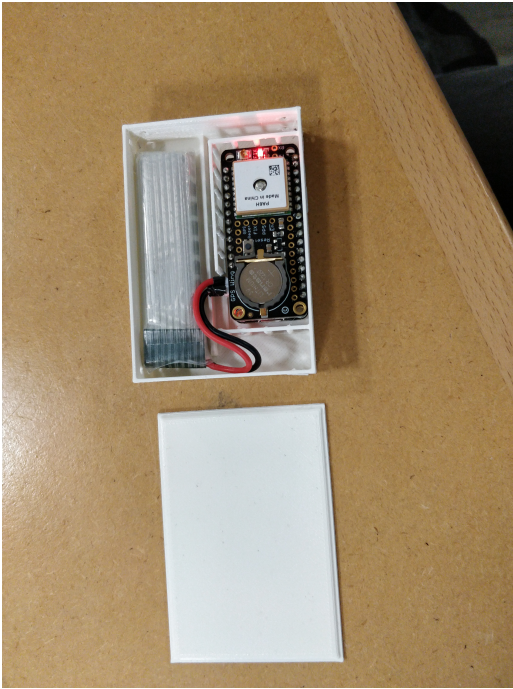
(b) GPS Featherwing stacked

Figure B.3: GPS Featherwing pin images, available at Adafruit [↗](#)

GPS Featherwing pinouts reference link 

Case design

Figure C.1: First case print, showing various angles



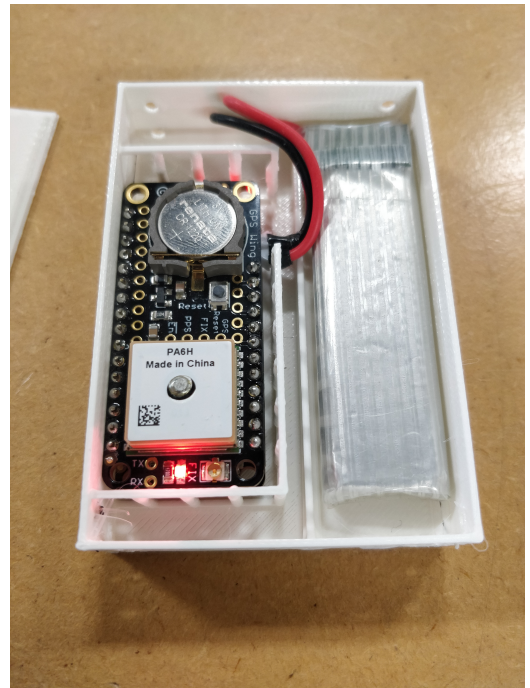
(a) Case overview with lid



(b) Battery port view



(c) Top view



(d) Angled front view

Figure C.2: Case 1 dimensional drawings

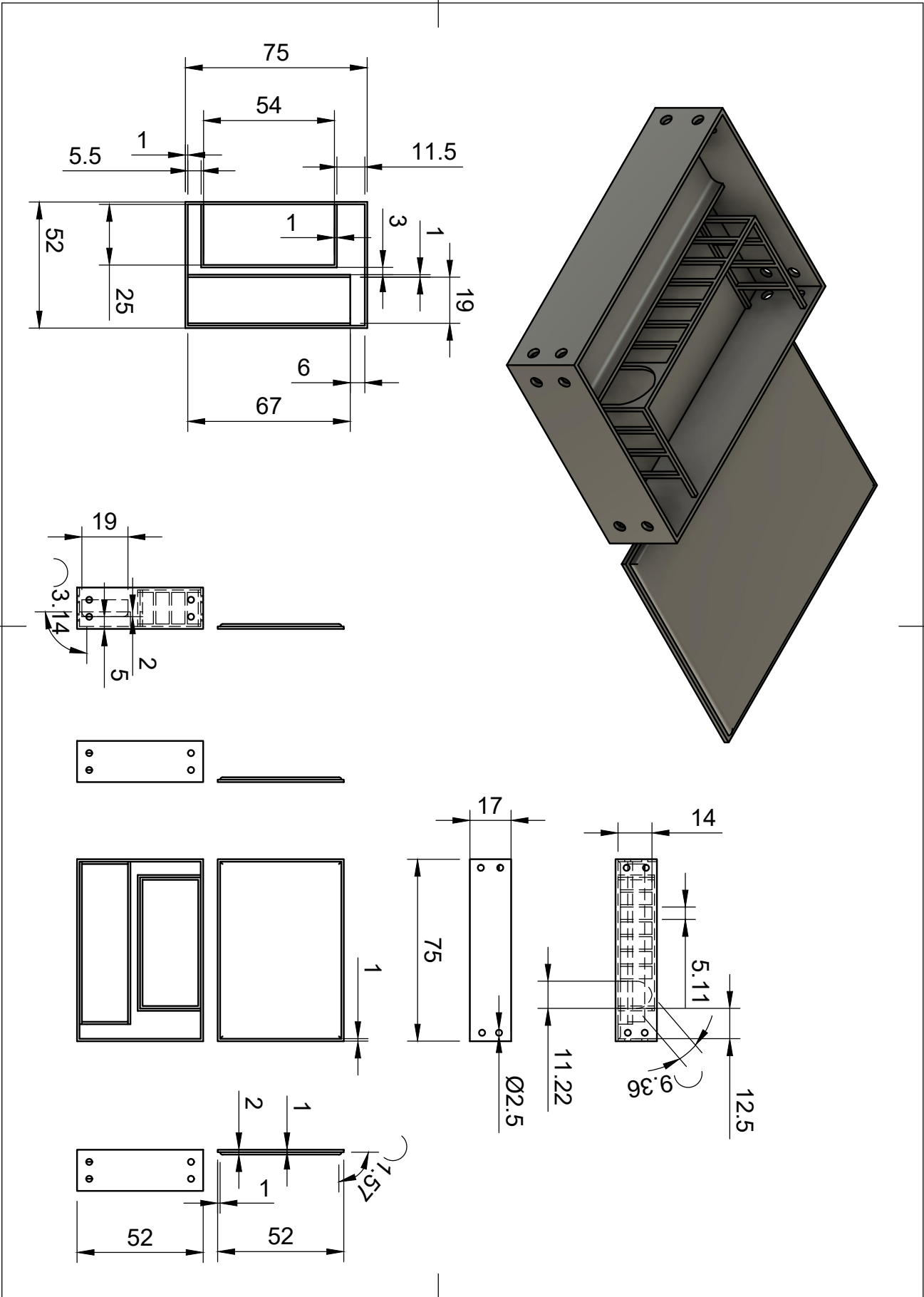


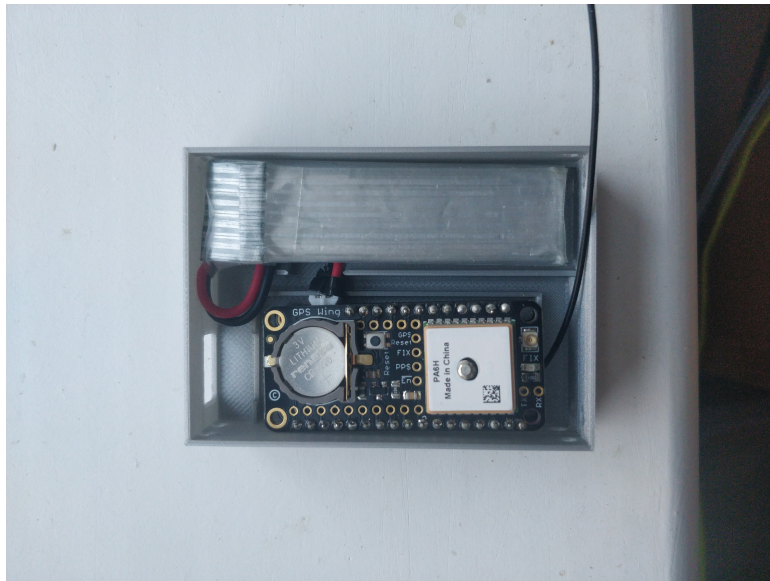
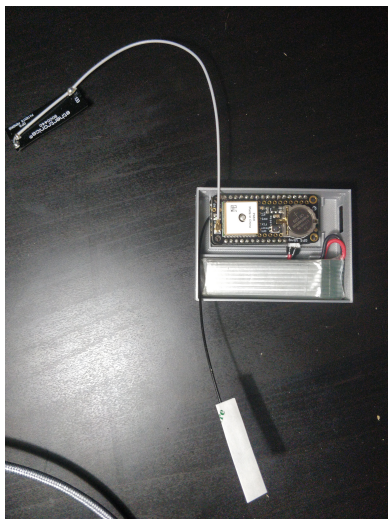
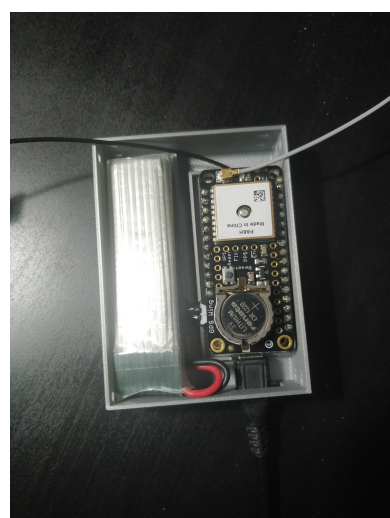
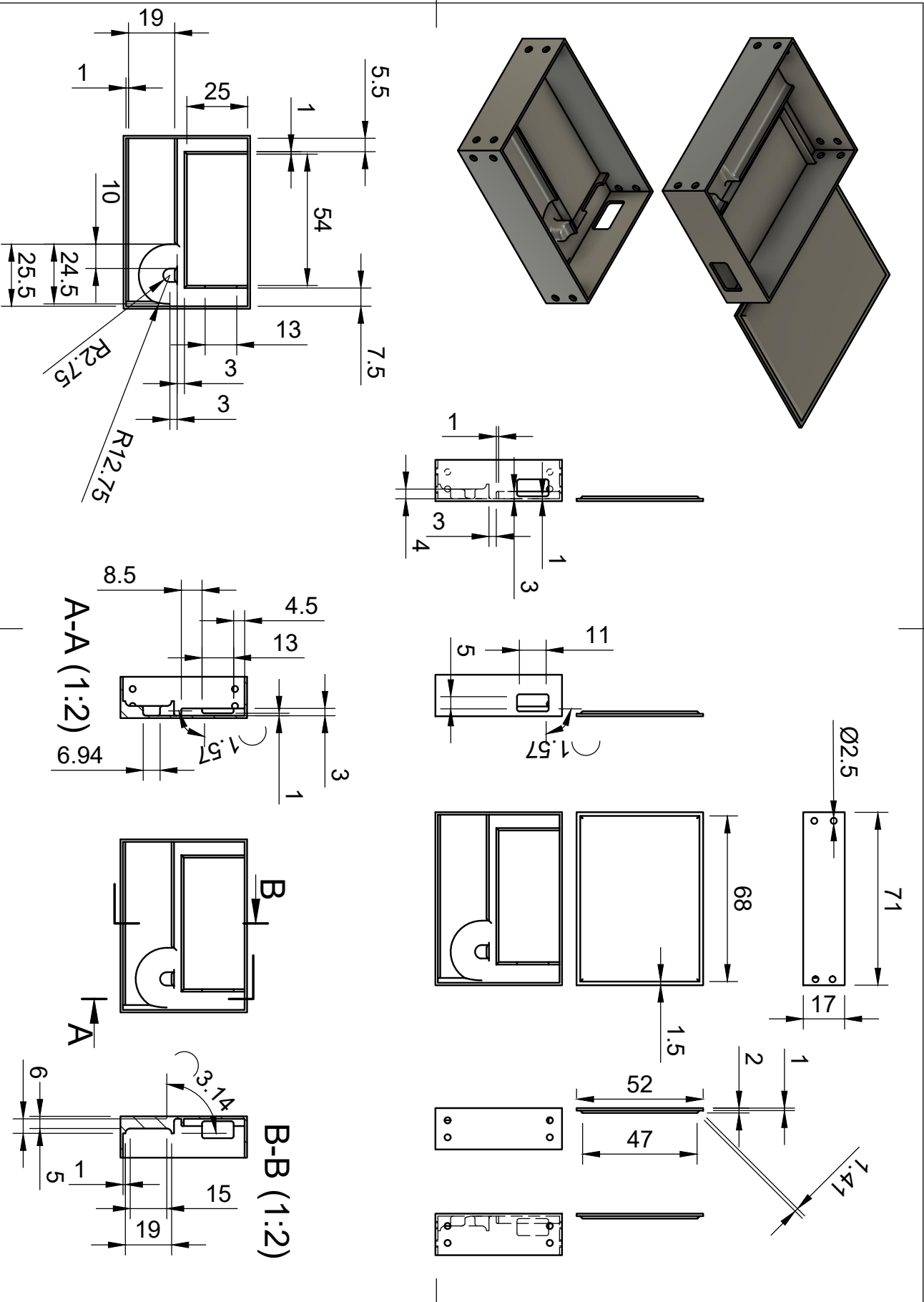
Figure C.3: Second case print, showing various angles**(a)** Case overview**(b)** Case with antennae**(c)** Antenna raising issue**(d)** Angled front view with USB connection**(e)** Top view with USB connection

Figure C.4: Case 2 dimensional drawings



Purchasing lists

Table D.1: Preliminary hardware list

Part	Description	Quantity	Price per unit	Supplier	SKU	Web link
Common requirement						
SX1262 LoRa HAT	LoRa radio for the Raspberry Pi	1	28.10	PiHut	WAV-16806	Link ↗
500mAh LiPo battery		1	6.00	PiHut	105189	Link ↗
Transmitter Set 1						
Challenger RP2040	microcontroller with lora radio	1	21.5	PiHut	104944	Link ↗
GPS Featherwing	gps module (uart)	1	24.6	PiHut	ADA3133	Link ↗
Transmitter Set 2 - I ² C alternative						
Feather RP2040	microcontroller	1	11.4	PiHut	ADA4884	Link ↗
LoRa Featherwing	lora radio (i2c)	1	19.5	PiHut	ADA3231	Link ↗
GPS breakout	gps module (i2c)	1	29.7	PiHut	PIM525	Link ↗
uFL connector	to get an antenna onto the lora board	5	0.8	PiHut	ADA1661	Link ↗
Antenna Options						
W3013	ceramic	2	1.48	Mouser	673-W3013	Link ↗
PIOV008NRAA-100	strip with ufl	1	1.87	Mouser	523-PIOV008NRAA-100	Link ↗
318020612	Outdoor antenna - N male	1	30.8	Mouser	713-318020612	Link ↗
	amazon option with included cable and adapter	1	30.99	Amazon		Link ↗
CAB.951	N female - SMA male, 1m	1	17.13	Mouser	960-CAB.951	Link ↗
	much cheaper amazon option	1	11.98	Amazon		Link ↗
Pigtail Antenna	only if none of the other options work	2	4.9	PiHut	105078	Link ↗
Raspberry Pi	it's okay I actually own a Pi3	1	219.99	Amazon		Link ↗









Table D.2: Updated hardware list

Part	Description	Quantity	Price per unit	Supplier	SKU	Web link
Transmitter						
3178	Feather M0 LoRa	1	30.76	Mouser	485-3178	Link ↗
3133	GPS Featherwing	1	21.96	Mouser	485-3133	Link ↗
	Pimoroni equivalent	1	24.6	Pimoroni	ADA3133	Link ↗
Receiver						
3231	LoRa Featherwing	1	18.68	Digikey	1528-1741-ND	Link ↗

Table D.2: Updated hardware list (cont.)

Part	Description	Quantity	Price per unit	Supplier	SKU	Web link
	Equivalent	1	19.5	Pimoroni	ADA3231	Link
	Bonnet OLED alternative	1	31.8	Pimoroni	ADA4074	Link
	+ headers	1	2.1	Pimoroni	COM0003	Link
Accessories						
2258	Pi Case	1	7	Mouser	485-2258	Link
	Equivalent	1	7.8	Pimoroni	ADA2258	Link
4410	lipo (jst) charger	1	5.24	Mouser	485-4410	Link
2830	stacking headers 1st level	1	1.1	Mouser	485-2830	Link
	Equivalent	1	1.1	Pimoroni	ADA2830	Link
2886	stacking headers base	1	0.836	Mouser	485-2886	Link
	Equivalent	1	1.2	Pimoroni	ADA2886	Link
Battery						
4714	JST PH Female jumper	1	0.836	Mouser	485-4714	Link
RS PRO battery	LiPo 1.8Ah	1	10.86	RS	144-9405	Link
Alternative	1.2mAh, doesn't require jumper or housing	1	9.9	Pimoroni	BAT00044	Link
PHR-2	JST PH Female connector housing	1	0.66	RS	820-1466	Link
Antenna						
ANT-868-CPA	ceramic	1	2.97	Mouser	712-ANT-868-CPA	Link
PIOV008NRAA-100	strip with ufl	1	1.87	Mouser	523-PIOV008NRAA-100	Link
318020612	Outdoor antenna - N male	1	30.8	Mouser	713-318020612	Link
CAB.951	N female - SMA male, 1m	1	17.13	Mouser	960-CAB.951	Link
CONUFL001-SMD-T	uFL connector	5	0.625	Mouser	712-CONUFL001-SMD-T	Link
CASMA-UFL-1	uFL to SMA F	2	8.71	Mouser	125-CASMA-UFL-1	Link
	Equivalent	2	3.9	Pimoroni	ADA851	Link
Additional Antennae						
W3214	ceramic	1	2.08	Mouser	673-W3214	Link
M620720	ceramic	1	1.73	Mouser	581-M620720	Link
W3013	ceramic	1	1.48	Mouser	673-W3013	Link

Table D.3: Antennae list

Description	Part	Name	Price	Supplier	SKU	Web link
Original antenna:	318020612	Outdoor antenna - N male	30.8	Mouser	713-318020612	Link 
Suggested replacement:	318020690	5.8dBi antenna	41.24	Farnell	4060414	Link 
Cheaper alternative:	318020708	3dBi antenna	24.72	Farnell	4060419	Link 
Lora antenna	211140-0100	0.3dBi at 868, 38mm	1.7	Farnell	3498957	Link 
Additional lora antenna	1002289F0-AA10L0200	1.8 dBi	3.95	Farnell	3407000	Link 
gps antenna	9000440		1.79	Farnell	3407009	Link 
additional gps antenna if within budget	APKD1507G2-0100S		6.83	Farnell	3924367	Link 
alternative selected lora	206764-0100		3.31	Farnell	2885764	Link 

Data

E.1 Power Consumption

Table E.1: Power meter recordings

Timestamp	Consumption (mAh)
21/01/2023 19:46	0
21/01/2023 22:43	1
22/01/2023 13:40	8
22/01/2023 16:58	10
22/01/2023 18:38	11
22/01/2023 21:31	12
22/01/2023 22:26	13
23/01/2023 11:00	19
23/01/2023 14:40	21
23/01/2023 19:39	23

Table E.2: Discharge data sample - 230213.csv

Timestamp	Datetime	Packet	Longitude	Latitude	Altitude	Fix	Voltage
1676326772.56411	13-02-23 22:19:32	11	5223.██	133.██	87.10	0	4.183
1676326777.5640159	13-02-23 22:19:37	12	5223.██	133.██	87.10	1	4.183
1676326782.5647342	13-02-23 22:19:42	12	5223.██	133.██	87.10	0	4.189
1676326787.564926	13-02-23 22:19:47	12	5223.██	133.██	87.10	0	4.183
...							
1676380803.0929441	14-02-23 13:20:03	10188	5223.██	133.██	69.10	0	3.564
1676380808.0946279	14-02-23 13:20:08	10188	5223.██	133.██	69.10	0	3.557
1676380813.0988088	14-02-23 13:20:13	10188	5223.██	133.██	69.10	0	3.564
1676380818.1010823	14-02-23 13:20:18	10188	5223.██	133.██	69.10	0	3.564

×

Table E.4: Charge data sample - 230214.csv

Timestamp	Datetime	Packet	Longitude	Latitude	Altitude	Fix	Voltage
1676392659.3032475	14-02-23 16:37:39	0	0.0000	0.0000	0.00	0	1.617
1676392664.3044226	14-02-23 16:37:44	0	0.0000	0.0000	0.00	0	1.695
1676392669.3057065	14-02-23 16:37:49	0	0.0000	0.0000	0.00	0	1.772
1676392674.305263	14-02-23 16:37:54	0	0.0000	0.0000	0.00	0	1.830
...							
1676428057.1960237	15-02-23 02:27:37	6260	5223.██	133.██	87.50	1	4.131
1676428062.1964362	15-02-23 02:27:42	6261	5223.██	133.██	87.50	1	4.131
1676428067.197315	15-02-23 02:27:47	6262	5223.██	133.██	87.60	1	4.131
1676428072.1974156	15-02-23 02:27:52	6263	5223.██	133.██	87.60	1	4.137

E.2 Distance

Listing E.1: Distance Test 1 sample - 2023-02-17_17_Feb_2023_1_41_52_pm.kml

```
<gx:MultiTrack>
  <altitudeMode>absolute</altitudeMode>
  <gx:interpolate>0</gx:interpolate>
  <gx:Track>
    <gx:coord>-1.56132623 52.38261212 62.51</gx:coord>
    <when>2023-02-17T13:44:48Z</when>
    <gx:coord>-1.56123128 52.38268469 127.56</gx:coord>
    <when>2023-02-17T13:45:10Z</when>
    <gx:coord>-1.56129532 52.38279329 126.44</gx:coord>
    <when>2023-02-17T13:45:14Z</when>
    <gx:coord>-1.56138629 52.38286398 127.49</gx:coord>
    <when>2023-02-17T13:45:20Z</when>
    <gx:coord>-1.56153087 52.38289276 127.94</gx:coord>
    <when>2023-02-17T13:45:31Z</when>
    <gx:coord>-1.56166255 52.38283815 128.35</gx:coord>
    ...
  </gx:Track>
</gx:MultiTrack>
```

Table E.6: Distance Test 1 sample - first_test.csv

Datetime	Latitude	Longitude
16/02/23 23:33	52'22.958	-1'33.76
16/02/23 23:33	52'22.959	-1'33.7594
16/02/23 23:33	52'22.9565	-1'33.7566
16/02/23 23:33	52'22.9561	-1'33.756
...		
16/02/23 23:51	52'22.9482	-1'33.7032
16/02/23 23:51	52'22.9463	-1'33.7058
16/02/23 23:51	52'22.9463	-1'33.7058
16/02/23 23:51	52'22.9463	-1'33.7058

Table E.8: Distance Test 2 sample - second_test.csv

Datetime	Latitude	Longitude
17/02/23 0:48	52'22.9409	-1'33.7407
17/02/23 0:48	52'22.9409	-1'33.7407
17/02/23 0:48	52'22.9409	-1'33.7404
17/02/23 0:48	52'22.9409	-1'33.7403
...		
17/02/23 1:16	52'22.9502	-1'33.7284
17/02/23 1:16	52'22.9497	-1'33.73
17/02/23 1:16	52'22.9497	-1'33.7307
17/02/23 1:16	52'22.9502	-1'33.7317

Code

F.1 Power scripts

Listing F.1: Power meter graphing script

```
1 ds = [  
2     datetime("21-Jan-2023 19:46:00")  
3     datetime("21-Jan-2023 22:43:00")  
4     datetime("22-Jan-2023 13:40:00")  
5     datetime("22-Jan-2023 16:58:00")  
6     datetime("22-Jan-2023 18:38:00")  
7     datetime("22-Jan-2023 21:31:00")  
8     datetime("22-Jan-2023 22:26:00")  
9     datetime("23-Jan-2023 11:00:00")  
10    datetime("23-Jan-2023 14:40:00")  
11    datetime("23-Jan-2023 19:39:00")  
12 ];  
13  
14 con = [  
15     0  
16     1  
17     8  
18     10  
19     11  
20     12  
21     13  
22     19  
23     21  
24     23  
25 ];  
26  
27 dsv = datenum(ds); % convert to number type for line of best fit calculations  
28  
29 coeff = polyfit(dsv,con,1); % line of best fit coefficients  
30 lobfx = linspace(min(dsv), max(dsv), 1000); % x data points  
31 lobfy = polyval(coeff, lobfx); % y data points  
32  
33 lobfx = datetime(lobfx, 'ConvertFrom', 'datenum'); % convert back to date type  
34  
35 figure(1);  
36 plot(ds,con, '.', 'markersize', 15);  
37 grid on;  
38 hold on;  
39 plot(lobfx, lobfy, 'LineWidth', 2, 'Color', '#388f58');  
40 ylabel("Consumption (mAh)");  
41 xlabel("Timestamp");  
42 title("Power meter readings");
```

```
43 ylim([-1,25])  
44 hold off;
```

Listing F.2: Power consumption graphing script v1

```

1 import csv
2 import matplotlib.pyplot as plt
3 from datetime import datetime as dt
4 import seaborn as sns
5 sns.set_style("whitegrid")
6
7 blue, = sns.color_palette("muted", 1)
8
9 timestamps_1 = []
10 level_1 = []
11 fix_1 = []
12
13 with open('Tests\\Battery\\data\\230213.csv', 'r') as csv_file:
14     reader = csv.reader(csv_file)
15     for row in reader:
16         timestamps_1.append(dt.strptime(row[1], '%d-%m-%y %H:%M:%S'))
17         level_1.append(float(row[7]))
18         fix_1.append(int(row[6]))
19 with open('Tests\\Battery\\data\\230214-old.csv', 'r') as csv_file:
20     reader = csv.reader(csv_file)
21     for row in reader:
22         timestamps_1.append(dt.strptime(row[1], '%d-%m-%y %H:%M:%S'))    #13-02-23 22:19:32
23         level_1.append(float(row[7]))
24         fix_1.append(int(row[6]))
25
26
27 fig, (ax1, ax2) = plt.subplots(2, 1)
28 fig.subplots_adjust(hspace=0.5)
29
30 ax1.plot_date(timestamps_1, level_1, fmt='g', lw=2, tz='UTC', xdate=True, ydate=False, label = "Battery level")
31
32 # plt.xticks(rotation = 25)
33 # plt.xlabel('Timestamps')
34 # plt.ylabel('Voltage (V)')
35 # plt.title('Battery discharge')
36 # plt.grid()
37 # plt.legend()
38
39
40 ax1.plot(timestamps_1, fix_1, color=blue, label = "GPS fix quality")
41 ax1.fill_between(timestamps_1, 0, fix_1, alpha=.3)
42

```

XVII

```

43
44 # plt.grid()
45
46
47 timestamps_2 = []
48 level_2 = []
49 fix_2 = []
50
51 with open('Tests\\Battery\\data\\230214.csv', 'r') as csv_file:
52     reader = csv.reader(csv_file)
53     for row in reader:
54         timestamps_2.append(dt.strptime(row[1], '%d-%m-%y %H:%M:%S'))
55         level_2.append(float(row[7]))
56         fix_2.append(int(row[6]))
57
58 # plt.subplot(2, 1, 2)
59 ax2.plot(timestamps_2, level_2, color = 'g', lw=2, label = "Battery level")
60
61 # plt.xticks(rotation = 25)
62 # plt.xlabel('Timestamps')
63 # plt.ylabel('Voltage (V)')
64 # plt.title('Battery charge')
65 # plt.grid()
66 # plt.legend()
67
68 ax2.plot(timestamps_2, fix_2, color=blue, label = "GPS fix quality")
69 ax2.fill_between(timestamps_2, 0, fix_2, alpha=.3)
70
71 # plt.grid()
72
73 # plt.suptitle("Battery charge curves")
74
75 plt.show()

```


Listing F.3: Power consumption graphing script v2

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 from matplotlib.dates import DateFormatter
5 from datetime import datetime as dt
6
7 sns.set_style("whitegrid")
8
9 blue, = sns.color_palette("muted", 1)
10
11 def legend_merge(plot1, plot2, loc, axis1, axis2):
12     handle1, label1 = plot1.get_legend_handles_labels()
13     handle2, label2 = plot2.get_legend_handles_labels()
14     axis1.legend(loc=loc, handles=handle1+handle2, labels = label1 + label2)
15     axis2.get_legend().remove()
16
17 #read data into dataframes
18 # merging two csv files
19 df_discharge = pd.concat([pd.read_csv(['Tests\\Battery\\data\\230213.csv', 'Tests\\Battery\\data\\230214-old.csv']),
20                             ignore_index=True)
21 df_discharge['Datetime'] = pd.to_datetime(df_discharge['Datetime'], format='%d-%m-%y %H:%M:%S')
22
23 df_charge = pd.read_csv('Tests\\Battery\\data\\230214.csv', parse_dates=['Datetime'], dayfirst=True)
24
25 # subplot with shared x
26 fig, ax = plt.subplots(nrows=2, ncols=1, figsize=(10, 6))
27 ax00 = ax[0]
28 ax01 = ax[0].twinx() #twin the axis
29 ax10 = ax[1]
30 ax11 = ax[1].twinx()
31
32 #first subplot
33 d_level = sns.lineplot(data=df_discharge, x='Datetime', y='Voltage', color='g', lw='2', ax=ax00, label='Voltage')
34 d_fix = sns.lineplot(data=df_discharge, x='Datetime', y='Fix', ax=ax01, color=blue, label='Fix')
35
36 #styling
37 legend_merge(d_level, d_fix, 0, ax00, ax01)
38 ax01.fill_between(df_discharge['Datetime'], 0, df_discharge['Fix'], alpha=0.3)
39 #limits
40 ax00.set_ylim(1.5,4.5)
41 ax01.set_ylim(0,2)
42 ax01.set_yticks([0,1,2])

```

XX

```

42 ax00.set_title('Discharge curve')
43
44 #second subplot
45 c_level = sns.lineplot(data=df_charge, x='Datetime', y='Voltage', color='g', lw='2', ax=ax10, label='Voltage')
46 c_fix = sns.lineplot(data=df_charge, x='Datetime', y='Fix', ax=ax11, color=blue, label='Fix')
47
48 #styling
49 legend_merge(c_level, c_fix, 1, ax10, ax11)
50 ax11.fill_between(df_charge['Datetime'], 0, df_charge['Fix'], alpha=0.3)
51 #limits
52 ax10.set_ylim(1.5,4.5)
53 ax11.set_ylim(0,2)
54 ax11.set_yticks([0,1,2])
55 ax10.set_title('Charge curve')
56
57 #figure styling
58 ax00.set_ylabel('Voltage (V)')
59 ax10.set_ylabel('Voltage (V)')
60 ax00.set_xlabel('Time')
61 ax10.set_xlabel('Time')
62 fig.suptitle(t='Battery charge curves', ha='center', size='20')
63 fig.tight_layout(pad=2.0)
64 # x time formatting
65 fmt = DateFormatter("%H:%M")
66 ax10.xaxis.set_major_formatter(fmt)
67 ax00.xaxis.set_major_formatter(fmt)
68
69 # annotations
70 ax00.annotate('13:20\n3.56V', xy=(dt(2023,2,14,13,20,18), 3.564), xycoords='data',
71             xytext=(0, -20), textcoords='offset points',
72             size=12, ha='center', va="top",
73             bbox=dict(boxstyle="round", alpha=0.2),
74             arrowprops=dict(arrowstyle="wedge,tail_width=0.5", alpha=0.2))
75 ax00.annotate('22:19\n4.18V', xy=(dt(2023,2,13,22,19,32), 4.183), xycoords='data',
76             xytext=(0, -20), textcoords='offset points',
77             size=12, ha='center', va="top",
78             bbox=dict(boxstyle="round", alpha=0.2),
79             arrowprops=dict(arrowstyle="wedge,tail_width=0.5", alpha=0.2))
80
81 ax10.annotate('02:27\n4.14V', xy=(dt(2023,2,15,2,27,52), 4.137), xycoords='data',
82             xytext=(0, -20), textcoords='offset points',
83             size=12, ha='center', va="top",
84             bbox=dict(boxstyle="round", alpha=0.2),

```

```
85         arrowprops=dict(arrowstyle="wedge,tail_width=0.5", alpha=0.2))
86 ax10.annotate('16:37\n1.62V', xy=(dt(2023,2,14,16,37,39), 1.617), xycoords='data',
87             xytext=(20, 20), textcoords='offset points',
88             size=12, ha='left', va="center",
89             bbox=dict(boxstyle="round", alpha=0.2),
90             arrowprops=dict(arrowstyle="wedge,tail_width=0.5", alpha=0.2))
91 ax10.annotate('22:11\n4.19V', xy=(dt(2023,2,14,22,11,30), 4.189), xycoords='data',
92             xytext=(0, -20), textcoords='offset points',
93             size=12, ha='center', va="top",
94             bbox=dict(boxstyle="round", alpha=0.2),
95             arrowprops=dict(arrowstyle="wedge,tail_width=0.5", alpha=0.2))
96
97 plt.show()
```

XX

F.2 Distance scripts

Listing F.4: Location format converter

```

1 import re
2 import csv
3
4 # take a coordinate as a string and split it into degrees and minutes
5 def splitter(coordinate_string):
6     match_result = re.search("(\\d{2})\\.\\.*", coordinate_string) #match minutes
7     return [coordinate_string[0:match_result.span()[0]], match_result.group(0)] #return [degrees, minutes]
8
9 #reformat coordinate into standardised system
10 def coord_rewrite(split_coord, direction):
11     d = ''
12     if direction == 'W' or direction == 'S':
13         d = '-'
14     return d + split_coord[0] + " " + split_coord[1]
15
16 with open('Tests\\Distance\\data\\230217-edit.csv') as csv_file:
17     with open('Tests\\Distance\\second_test.csv', 'w+', newline='') as write_file:
18         reader = csv.reader(csv_file)
19         writer = csv.writer(write_file)
20         writer.writerow(['datetime', 'latitude', 'longitude']) #headers
21         line_count = 0
22         for row in reader:
23             line_count += 1
24             #row0      row1      row2      row3      row4      row5      row6      row7      row8      row9
25             #timestamp  datetime  packet  lat      dir      long     dir      alt      fix      volt
26             datetime = row[1]
27             lat = row[3]
28             lat_dir = row[4]
29             long = row[5]
30             long_dir = row[6]
31
32             if line_count == 1: #headers
33                 # print(f'{datetime}\\t{lat}\\t{long}')
34                 continue
35             # elif line_count == 5:
36             #     break
37             else:
38                 # print(f'{datetime}\\t{coord_rewrite(splitter(lat), lat_dir)}\\t{coord_rewrite(splitter(long), long_dir)}')
39                 writer.writerow([datetime, coord_rewrite(splitter(lat), lat_dir), coord_rewrite(splitter(long), long_dir)])

```

F.3 Microcontroller sketches

Listing F.5: Modified example battery voltage

```
1 #define VBATPIN A7
2
3 void setup() {}
4
5 void loop() {
6     float measuredvbat = analogRead(VBATPIN);
7     measuredvbat = (measuredvbat*6.6)/1024;
8     Serial.print("VBat: "); Serial.println(measuredvbat);
9 }
```

Listing F.6: Modified example GPS by library

```

1  #include <SPI.h>
2  #include <RH_RF95.h>
3  #include <Adafruit_GPS.h>
4
5
6  #define GPSSerial Serial1
7  Adafruit_GPS GPS(&GPSSerial);
8
9  #define GPSECHO false
10
11 uint32_t timer = millis();
12
13 void setup() {
14     while (!Serial)
15         ;
16     Serial.begin(115200);
17     GPS.begin(9600);
18     // GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
19     GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);
20     // GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_ALLDATA);
21     // GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);
22     GPS.sendCommand(PMTK_SET_NMEA_UPDATE_200_MILLIHERTZ); // 10 second update time
23     // GPS.sendCommand(PGCMD_ANTENNA);
24 }
25
26 void loop() {
27     char c = GPS.read();
28     // if (GPSECHO) {
29     //     if (c) {
30     //         Serial.print(c);
31     //     }
32     // }
33
34     if (GPS.newNMEAreceived()) {
35         // Serial.print(GPS.lastNMEA());
36         if (!GPS.parse(GPS.lastNMEA())) {
37             return;
38         }
39     }
40     if (millis() - timer > 2000) {
41         timer = millis();
42     }

```

```
43     if (GPS.fix) {  
44         String output = String(GPS.latitude,4) + GPS.lat + ", " + String(GPS.longitude,4) + GPS.lon;  
45         Serial.println(output);  
46     }  
47 }  
48 }
```

Listing F.7: Modified example GPS by serial

```

1 // Test code for Ultimate GPS Using Hardware Serial
2 // (e.g. GPS for Leonardo, Flora or FeatherWing)
3 //
4 // This code shows how to test a passthru between USB and hardware serial
5 //
6 // Tested and works great with the Adafruit GPS FeatherWing
7 // -----> https://www.adafruit.com/products/3133
8 // or Flora GPS
9 // -----> https://www.adafruit.com/products/1059
10 // but also works with the shield, breakout
11 // -----> https://www.adafruit.com/products/1272
12 // -----> https://www.adafruit.com/products/746
13 //
14 // Pick one up today at the Adafruit electronics shop
15 // and help support open source hardware & software! -ada
16
17
18 // what's the name of the hardware serial port?
19 #define GPSSerial Serial1
20
21
22 void setup() {
23   // make this baud rate fast enough to we aren't waiting on it
24   Serial.begin(115200);
25
26   // wait for hardware serial to appear
27   while (!Serial) delay(10);
28
29   // 9600 baud is the default rate for the Ultimate GPS
30   GPSSerial.begin(9600);
31 }
32
33
34 void loop() {
35   if (GPSSerial.available()) {
36     char c = GPSSerial.read();
37     Serial.write(c);
38   }
39 }

```

XXX

Listing F.8: LoRa with GPS

```

1  #include <SPI.h>
2  #include <RH_RF95.h>
3  #include <Adafruit_GPS.h>
4
5
6  #define RFM95_CS      8
7  #define RFM95_INT     3
8  #define RFM95_RST     4
9
10 #define VBATPIN  A7
11
12 #define GPSSerial Serial1
13 Adafruit_GPS GPS(&GPSSerial);
14
15 #define RF95_FREQ 868.0
16
17 RH_RF95 rf95(RFM95_CS, RFM95_INT);
18
19 uint32_t timer = millis();
20 int count = 0;
21 int length = 0;
22
23 void setup() {
24     // make this baud rate fast enough to we aren't waiting on it
25     Serial.begin(115200);
26
27     pinMode(RFM95_RST, OUTPUT);
28     digitalWrite(RFM95_RST, HIGH);
29
30     // wait for hardware serial to appear
31     // while (!Serial) delay(10);
32
33     // manual reset
34     digitalWrite(RFM95_RST, LOW);
35     delay(10);
36     digitalWrite(RFM95_RST, HIGH);
37     delay(10);
38
39     while (!rf95.init()) {
40         Serial.println("LoRa radio init failed");
41         Serial.println("Uncomment '#define SERIAL_DEBUG' in RH_RF95.cpp for detailed debug info");
42         while (1);

```

XX/

XXVII

```

43 }
44
45 if (!rf95.setFrequency(RF95_FREQ)) {
46     Serial.println("setFrequency failed");
47     while (1);
48 }
49
50 rf95.setTxPower(23, false);
51
52 // 9600 baud is the default rate for the Ultimate GPS
53 GPS.begin(9600);
54
55
56 GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
57
58
59 GPS.sendCommand(PMTK_SET_NMEA_UPDATE_200_MILLIHERTZ); // 5 second update time
60
61
62
63
64 }
65
66 void loop() {
67     float measuredvbat = analogRead(VBATPIN);
68     measuredvbat = (measuredvbat*6.6)/1024;
69     char c = GPS.read();
70
71     if (GPS.newNMEAreceived()) {
72         // Serial.print(GPS.lastNMEA());
73         if (!GPS.parse(GPS.lastNMEA())) {
74             return;
75         }
76     }
77
78     if (millis() - timer > 5000) {
79         timer = millis();
80
81         if (GPS.fix) {
82             count++;
83         }
84         String output = String(count) + "," + String(GPS.latitude,4) + "," + String(GPS.longitude,4) + "," + String(GPS.altitude) + ","
            + String(GPS.fixquality) + "," + String(measuredvbat, 4);

```

```
85     length = output.length();
86     char send[length];
87     output.toCharArray(send, length);
88     Serial.println(send);
89     rf95.send((uint8_t *)send, length);
90
91 }
92
93
94 }
```

F.4 SBC scripts

Listing F.9: Example test

```

1 # SPDX-FileCopyrightText: 2018 Brent Rubell for Adafruit Industries
2 #
3 # SPDX-License-Identifier: MIT
4
5 """
6 Wiring Check, Pi Radio w/RFM9x
7
8 Learn Guide: https://learn.adafruit.com/lora-and-lorawan-for-raspberry-pi
9 Author: Brent Rubell for Adafruit Industries
10 """
11 import time
12 import busio
13 from digitalio import DigitalInOut, Direction, Pull
14 import board
15 # Import the SSD1306 module.
16 import adafruit_ssd1306
17 # Import the RFM9x radio module.
18 import adafruit_rfm9x
19
20 # Button A
21 btnA = DigitalInOut(board.D5)
22 btnA.direction = Direction.INPUT
23 btnA.pull = Pull.UP
24
25 # Button B
26 btnB = DigitalInOut(board.D6)
27 btnB.direction = Direction.INPUT
28 btnB.pull = Pull.UP
29
30 # Button C
31 btnC = DigitalInOut(board.D12)
32 btnC.direction = Direction.INPUT
33 btnC.pull = Pull.UP
34
35 # Create the I2C interface.
36 i2c = busio.I2C(board.SCL, board.SDA)
37
38 # 128x32 OLED Display
39 reset_pin = DigitalInOut(board.D4)
40 display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c, reset=reset_pin)
41 # Clear the display.

```

XXX

```

42 display.fill(0)
43 display.show()
44 width = display.width
45 height = display.height
46
47 # Configure RFM9x LoRa Radio
48 CS = DigitalInOut(board.CE1)
49 RESET = DigitalInOut(board.D25)
50 spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
51
52 while True:
53     # Clear the image
54     display.fill(0)
55
56     # Attempt to set up the RFM9x Module
57     try:
58         rfm9x = adafruit_rfm9x.RFM9x(spi, CS, RESET, 915.0)
59         display.text('RFM9x: Detected', 0, 0, 1)
60     except RuntimeError as error:
61         # Thrown on version mismatch
62         display.text('RFM9x: ERROR', 0, 0, 1)
63         print('RFM9x Error: ', error)
64
65     # Check buttons
66     if not btnA.value:
67         # Button A Pressed
68         display.text('Ada', width-85, height-7, 1)
69         display.show()
70         time.sleep(0.1)
71     if not btnB.value:
72         # Button B Pressed
73         display.text('Fruit', width-75, height-7, 1)
74         display.show()
75         time.sleep(0.1)
76     if not btnC.value:
77         # Button C Pressed
78         display.text('Radio', width-65, height-7, 1)
79         display.show()
80         time.sleep(0.1)
81
82     display.show()
83     time.sleep(0.1)

```

Listing F.10: Modified example radio

```

1 # SPDX-FileCopyrightText: 2018 Brent Rubell for Adafruit Industries
2 #
3 # SPDX-License-Identifier: MIT
4
5 """
6 Example for using the RFM9x Radio with Raspberry Pi.
7
8 Learn Guide: https://learn.adafruit.com/lorawan-for-raspberry-pi
9 Author: Brent Rubell for Adafruit Industries
10 """
11 # Import Python System Libraries
12 import time
13 # Import Blinka Libraries
14 import busio
15 from digitalio import DigitalInOut, Direction, Pull
16 import board
17 # Import the SSD1306 module.
18 import adafruit_ssd1306
19 # Import RFM9x
20 import adafruit_rfm9x
21
22 # Button A
23 btnA = DigitalInOut(board.D5)
24 btnA.direction = Direction.INPUT
25 btnA.pull = Pull.UP
26
27 # Button B
28 btnB = DigitalInOut(board.D6)
29 btnB.direction = Direction.INPUT
30 btnB.pull = Pull.UP
31
32 # Button C
33 btnC = DigitalInOut(board.D12)
34 btnC.direction = Direction.INPUT
35 btnC.pull = Pull.UP
36
37 # Create the I2C interface.
38 i2c = busio.I2C(board.SCL, board.SDA)
39
40 # 128x32 OLED Display
41 reset_pin = DigitalInOut(board.D4)
42 display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c, reset=reset_pin)

```

XXX

XXXX

```

43 # Clear the display.
44 display.fill(0)
45 display.show()
46 width = display.width
47 height = display.height
48
49 # Configure LoRa Radio
50 CS = DigitalInOut(board.CE1)
51 RESET = DigitalInOut(board.D25)
52 spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
53 rfm9x = adafruit_rfm9x.RFM9x(spi, CS, RESET, 868.0)
54 rfm9x.tx_power = 23
55 prev_packet = None
56
57 while True:
58     packet = None
59     # draw a box to clear the image
60     display.fill(0)
61     display.text('RasPi LoRa', 35, 0, 1)
62
63     # check for packet rx
64     packet = rfm9x.receive()
65     if packet is None:
66         display.show()
67         display.text('- Waiting for PKT -', 15, 20, 1)
68     else:
69         # Display the packet text and rssi
70         display.fill(0)
71         prev_packet = packet
72         try:
73             packet_text = str(prev_packet, "utf-8")
74         except UnicodeDecodeError:
75             print('parse error')
76         except:
77             print('other error')
78         display.text('RX: ', 0, 0, 1)
79         print('RX: ' + packet_text)
80         display.text(packet_text, 25, 0, 1)
81         time.sleep(1)
82
83     if not btnA.value:
84         # Send Button A
85         display.fill(0)

```

```
86     button_a_data = bytes("Button A!\r\n","utf-8")
87     rfm9x.send(button_a_data)
88     display.text('Sent Button A!', 25, 15, 1)
89     elif not btnB.value:
90         # Send Button B
91         display.fill(0)
92         button_b_data = bytes("Button B!\r\n","utf-8")
93         rfm9x.send(button_b_data)
94         display.text('Sent Button B!', 25, 15, 1)
95     elif not btnC.value:
96         # Send Button C
97         display.fill(0)
98         button_c_data = bytes("Button C!\r\n","utf-8")
99         rfm9x.send(button_c_data)
100         display.text('Sent Button C!', 25, 15, 1)
101
102
103     display.show()
104     time.sleep(0.1)
```

XXXXIII

Listing F.11: First test

XXXXV

```

1 import busio
2 from digitalio import DigitalInOut, Direction, Pull
3 import board
4 import adafruit_rfm9x
5 from datetime import datetime
6
7 # Create the I2C interface.
8 i2c = busio.I2C(board.SCL, board.SDA)
9
10 # Configure LoRa Radio
11 CS = DigitalInOut(board.CE1)
12 RESET = DigitalInOut(board.D25)
13 spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
14 rfm9x = adafruit_rfm9x.RFM9x(spi, CS, RESET, 868.0)
15 rfm9x.tx_power = 23
16
17 prev_packet = None
18
19 # timestamped log file for incoming packets
20 file = open((datetime.now()).strftime("%d%m%y.txt"), 'w+')
21 print("Start")
22
23 try:
24     while True:
25         packet = None
26         packet = rfm9x.receive()
27         if packet is not None:
28             prev_packet = packet
29             try:
30                 # create timestamp
31                 now = datetime.now()
32                 date_time = now.strftime("%y/%m/%d,%H:%M:%S")
33
34                 packet_text = str(prev_packet, "utf-8")
35                 text = (date_time + ", " + packet_text)
36             except UnicodeDecodeError:
37                 # log timestamp of invalid packet
38                 print(date_time + ", " + "unicode parse error", file=file, flush=True)
39             # log accepted packet
40             print(text, file=file, flush=True)
41 # gracefully exit on ^C command
42 except KeyboardInterrupt:

```

```
43 print("\nTerminating")
44 file.close()
45 raise SystemExit
```

XXXXV

Listing F.12: Battery test

XXXXVI

```

1  # Import Blinka Libraries
2  import busio
3  from digitalio import DigitalInOut, Direction, Pull
4  import board
5  # Import RFM9x
6  import adafruit_rfm9x
7  from datetime import datetime
8  import time
9
10
11 # Create the I2C interface.
12 i2c = busio.I2C(board.SCL, board.SDA)
13
14 # Configure LoRa Radio
15 CS = DigitalInOut(board.CE1)
16 RESET = DigitalInOut(board.D25)
17 spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
18 rfm9x = adafruit_rfm9x.RFM9x(spi, CS, RESET, 868.0)
19 prev_packet = None
20 file = open((datetime.now()).strftime("%y%m%d.csv"), 'w+')
21 print("Start")
22
23 try:
24     while True:
25         packet = None
26         packet = rfm9x.receive()
27         if packet is not None:
28             prev_packet = packet
29             try:
30                 now = datetime.now()
31                 date_time = now.strftime("%d-%m-%y %H:%M:%S")
32                 timestamp = str(time.time())
33
34                 packet_text = (str(prev_packet, "utf-8")).rstrip('\x00')
35                 text = (timestamp + "," + date_time + "," + packet_text)
36
37             except Exception as e:
38                 print(date_time)
39                 print(e)
40                 print("data" + str(prev_packet))
41                 print(text, file=file, flush=True)
42 except KeyboardInterrupt:

```

```
43 print("\nTerminating")
44 file.close()
45 raise SystemExit
```

Listing G.1: Decoding error

```

1 Start
2 13-02-23 19:50:11
3 'utf-8' codec can't decode byte 0x9f in position 0: invalid start byte
4 bytearray(b'\x9f\x88\x11\xe\x9c\xf5\x89\\22Rs\n\x95\xa\x95')
5 13-02-23 20:12:48
6 'utf-8' codec can't decode byte 0xe3 in position 1: invalid continuation byte
7 bytearray(b'=\xe3\xc1\x90\xef\x8b\x04.\xc4i\xa9\xbe\xab\x07\xb2q\xc24\xbd\xec\xc1\xb54H`\x9c#k\xaeG\x92\x0f\x17\xdd\xafr\xd0\
   xa5V \x7f\xb0\xaa\xb0\xbt6;d\x8c\xc1\xd8\xe6]\xef\x15d\xb9\x0c\xaeT\x89R\xdbTF\x0b\xa5%=o\x96')
8 13-02-23 20:49:27
9 'utf-8' codec can't decode byte 0xc2 in position 0: invalid continuation byte
10 bytearray(b'\xc2\xed\xa9\xff\x8c\xcf\xfa`X\xcf\x88I\xa5\xa6\x15\x7f\x9b\xa7^c\xbf\xac')
11 13-02-23 20:51:02
12 'utf-8' codec can't decode byte 0xb2 in position 0: invalid start byte
13 bytearray(b'\xb2r\x9aLqR\x90w\xb8B\xfd\b8\xd8\x9b\xc2\xb5U\xla\x9e;\xda\x02v\xdb\x95`-&\xae+\x92\xd2\x99\xd5\xa2\x16\x04\x1b
   \xa0<|t2\x9e\xdc3g$\xf5I\xe6\x13\xaJ\xfa\x85\xb3v+\xa9\x03He\x16\b8\xad\xdf')

```

XXXXV|||