Improved Weight Measurement System for Monitoring Kalaharian Fork-Tailed Drongo Well-being: A Scale Design Proposal

EEE4113F
Engineering System Design

Prepared by:
Jonathan Apps, Khelan Hari, Nikhara Naidu and Natasha Soldin

Prepared for:
Dr Stephen Paine and Dr Francois Schonken
Department of Electrical Engineering
University of Cape Town

August 14, 2023

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

August 14, 2023

Jonathan Apps — Date

August 14, 2023

Khelan Hari — Date

August 14, 2023

Nikhara Naidu — Date

August 14, 2023

Natasha Soldin — Date

Abstract

This study proposes a weight measurement system for easily and accurately monitoring the well-being of the Kalaharian forked-tail Drongo population. Weight data provides insight into the feeding habits of Drongos, serving as an indicator of their welfare. Existing methods for obtaining bird weight data in desert-like environments lack accuracy, reliability, and ease of use.

Our solution consisted of a perch on a load cell, connected to an ESP-32 microcontroller and a power system, housed in a protected container. A data processing algorithm was designed to extract an accurate weight reading from data samples collected when a bird is present on the perch. The weight value is then transmitted via the ESP-32's AP (Access Point) Wi-Fi network to a webserver that can be accessed remotely via a device's browser.

The system performed as intended and satisfied the stakeholders. The perch and hardware enclosure were prototype designs, but were found to be effective at protecting the circuitry and enabling the collection of weight measurements. The power module was effective and able to support the system for a day at a time. The data processing algorithm used a modal technique which was found to accurately record weight measurements to 0.1 grams. The transmission and reception of data via Wi-Fi and a webserver were found to be accurate and reliable for transmission distances of over 10 meters. The entire system cost less than 2000 ZAR, which was the defined budget for the project.

We concluded that the solution produces improved measurement accuracy, ease of use, and cost-effectiveness. Thus it provides researchers with an enhanced tool to facilitate improved data collection for their studies of the Drongo population, and potentially other avian species, and should be considered as an alternative to current methods.

Contents

List of Figures

Chapter 1

Introduction

The following report details the design of a scale for capturing weight data of forked-tail Drongos in order to monitor the bird's overall well-being. This is a proposed improvement on the current weight scale implementation used by Ben Murphy of the Fitzpatrick Institute of African Ornithology. Ben's research is on the weight changes of the fork-tailed Drongo specifically during breeding periods as well as the correlation between a bird's weight and environmental temperature fluctuations. This design focuses specifically on improving the accuracy of the data captured and the means by which the data is transmitted.

1.1 Background

1.1.1 Relevance of Bird's Weight

As early as 1938, Nice pointed out that the value of recording birds' weights lay in being able to measure different species accurately and shed light on biological problems they encountered [12]. She called for more regular weighing of various bird species in order for the data to be representative and provide the greatest value in achieving useful information about bird populations. Hummingbirds are amongst the lightest birds at around 2 grams and the heaviest living, flying birds weigh up to 12 kilograms and include bustards and condors [13]. Handling wild birds in order to weigh them or weighing recently demised birds is likely to introduce inaccurate measurements and can be avoided by using weighing perches [13].

Bird weights may assist with species identification as different species have particular average weights. However, weights fluctuate within species due to multiple factors including 1) daily feeding patterns - with an increase of 5-10% towards late afternoon in comparison to the early morning; 2) seasons - with higher weights during harsh winters and dry seasons to reduce the chances of starvation in severe weather; 3) during reproductive periods and caring for their young - with decreasing weights as the young reach maturity; 4) prior to migration - when birds flying long distances over inhospitable habitats accumulate large fat reserves; 5) age - with juveniles usually weighing less than adults; 6) sex - with females being minimally lighter than males except at the time of egg formation; 7) year on year variation - as a result of climate and food availability; and 8) geographic location [12, 13]. Considering this variability, it is evident that recording factors such as age, sex, time of day, temperature and year that weight measurement was performed, breeding season, geographical location and linear dimensions (for example, wing length), are useful for bird weight analysis. Additional obstacles include the fact that adequate weight samples for common bird species have not yet been documented particularly

outside Europe and North America [13].

Weights can be used in trying to understand the bird's physiology or predict outcomes and behaviour. Tracking the weight of a population of birds over time can assist with monitoring their overall health with changes in weight indicative of illness, modifications in diet (potentially due to loss of habitat) or stress that may affect the population's survival [13]. Weights have been used to understand the amount of food eaten in relation to the bird's weight. The conclusion of this research was that food consumption increases in cold weather, resulting in weight gain in winter, and that smaller birds eat proportionately more than larger birds [12]. The breeding season is a time associated with increased energy expenditure. Monitoring increasing body weight as a result of increasing nutrient reserves before the breeding season, has been used to determine the number of offspring and the likelihood of the bird surviving the breeding season [13]. The extent of weight gain prior to migration can be used to predict the flight range and/or severity of the climate at the bird's destination with higher weight gains signalling higher distances to be flown and/or more severe climates to survive at arrival.

Longitudinal monitoring of adequate sample sizes of bird weights of a variety of different species is required to fill the knowledge gaps so that the data can be used to monitor the survival and well-being of bird populations globally.

1.1.2 Relevance of Bird's Weight for Ben's Research

As part of Professor Amanda Ridley's research team at the Fitzpatrick Institute of African Ornithology at the University of Cape Town, PhD student Ben Murphy is studying fork-tailed Drongos on the Kuruman River Reserve in the southern Kalahari Desert [14]. Ben's research interests include measuring the effect of increasing temperature and nesting on bird weight (personal communication: Ben Murphy).

The fork-tailed Drongo (Dicrurus adsimilis) is a medium sized bird weighing around 45 grams (range 38-55g) and is 25cm (range 22-26cm) in length, including the deeply forked tail, with distinctive red eyes. It feeds almost exclusively on insects but has been known to eat nectar and small nestlings. It is a feisty bird and a kleptoparasite (steals food or prey from other animals by mimicking the sounds of their predators) specialist that is widely distributed throughout South Africa.

A temporary decrease in body weight of adult birds rearing their young may be adaptive rather than detrimental since it allows for the parents to live off the energy released during weight loss and use less energy in flying to find food, because they are carrying less weight [15]. This translates into the adult birds requiring proportionately less of the food they can collect in order to provide more food for their nestlings, gifting them a better chance at survival. Nesting research performed in India noted that the chicks are fed by both parents; one parent guards the nestlings whilst the other forages and feeds [16]. The nesting season was determined by temperature and food availability, amongst other factors but no weight or temperature measurements were taken. No documented evidence of the effect of nesting on the weight of local fork-tailed Drongo parents was found, pointing to the unique nature of Ben Murphy's research.

The effect of high temperatures, associated with climate change, are unknown on avian species and desert conditions (like the Kalahari) where temperatures are already high, provide good environments

in which to study these effects. High temperatures cause physical stress to birds that can lead to higher rates of disease, dehydration and mortality as well as reduced reproductivity culminating in reduced populations [17].

Effects of high temperatures that have been described by the Fitzpatrick Institute of African Ornithology at the University of Cape Town in fork-tailed Drongos include a reduction in the ability to forage for food for themselves and their offspring and a propensity to consume a higher proportion of the food foraged seemingly to protect their reproductive ability above the fitness of their offspring (although no decrease in offspring growth was demonstrated). These observations lead to Ben's hypothesis that unlike other birds, Drongos may maintain their weight during nesting.

1.2 Objectives

The objective of this project is to plan and build a system that measures the weight of fork-tailed Drongos in the Kalahari in collaboration with the ongoing research currently being done by Ben and Sam of the Fitzpatrick Institute of African Ornithology. This project however focuses on two key aspects of the research, namely the accuracy of the data processing and the ease of the data transmission. The objective of this project is to build the system while placing emphasis on the aforementioned aspects.

1.3 System Requirements

1.3.1 User Requirements

Table 1.1 below details the user requirements and their corresponding specifications and acceptance test protocols (ATPs). These requirements are derived from what the user wishes the end product to achieve.

User Requirements	**Specifications**	**Acceptance Test Protocols**
UR1	US1.1	UATP1.1
The system must obtain accurate bird weight values from a scale	The load-cell implementation of the scale must produce a weight accuracy of up to 0.1g	Place items of known weight onto the scale and check that the measured value is correct to one decimal point
	US1.2	UATP1.2
	The data processing must ensure that from the sampled data it produces a weight accuracy within a margin of 1.5%	Execute the data processing algorithm on sampled data of known true weight value and examine the accuracy of the produced value
	US1.3	UATP1.3
	The weight value must retain accuracy over transmission	Examine the transmitted data to ensure the same value that was output by the processing algorithm was received by the researcher

UR2	US2	UATP2
The system must include a perch appropriate for a forked-tail Drongo	The perch much be load bearing enough to support the weight of the bait and the forked-tailed D rongo - which at maximum of 60g	Place an item of 60g onto the perch to see if it supports this weight
UR3	US3	UATP3
The system must be approachable by forked-tailed Drongos	The entire system should be camoflauged in colours of the given environment	Observe the bird's behaviour around the system to determine if the camoflauge is successful
UR4	US4.1	UATP4.1
The system must be deployable in an outdoor environment	The power module must include protection circuitry to ensure operation in a harsh climate (reverse polarity protection, thermal shutdown, under voltage lockout, current protection)	Test that the protection circuitry is operational as intended
	US4.2	UATP4.2
	The hardware enclosure of the circuitry must be made of water and dust resistant material	Test that the hardware enclose when exposed to water and dust exposure
UR5	US5	UATP5
The system must be portable	The hardware of the system must not exceed 5kg	Weigh the final system to determine its overall weight
UR6	US6	UATP6.1
The system must be operational for a full day at a time	System must have a constant 5V input to the ESP32 microcontroller for 12 hours	Measure the power module's output to ensure 5V is achieved
		UATP6.2
		Keep the power module operational for 12 hours to check that its 5V output is kept constant
UR7	US7	UATP7
The system must be re-usable (deployable on consecutive days)	Solar Panels will be used to re-charge batteries and ensure power module can always output 5V	Test that the solar panel is able to charge the lithium ion battery
UR8	US8	UATP8
The bird weight data must be obtained from the hardware in a unobtrusive manner	Implement remote access via Wi-Fi that allows for data transmission across at least 10m	Test that the user can access data sent from the ESP-32 from at least 10m away
UR9	US9	UATP9
The system must indicate the external termperature	A temperature sensor will be used to indicate external ambient temperature at the time that a weight value is measured	Test the temperature in known temperature environments to inidcate proper operation
UR10	US10	UATP10
The system must be cost efficient	All together, all hardware acquired should be below 2000 ZAR	Add up the price of all component's purchases

Table 1.1: User Requirements, Specification and Acceptance Test Protocols

1.3.2 Functional Requirements

Table 1.2 below details the functional requirements and their corresponding specifications and acceptance test protocols (ATPs). These requirements are derived from what the system must be able to achieve.

Functional Requirements	**Specifications**	**Acceptance Test Protocols**
FR1	FS1	FATP1
Data processing must be applied to sufficient amounts of data to achieve adequate accuracy	The sampling rate must be fast enough such that even in the worst case of bird's presence on the scale (3s), enough data is collected - a sampling rate of around 10 samples/s	Test the accuracy of the processed weight value output for the worst case (3s) with the current sampling rate
FR2	FS2	FATP2
Data storage must be sufficient to store all sampled data	The data structure must be able to accommodate the maximum number of samples of 250 (25s at a rate of 10 samples/s). These values are real/float (4bytes) and therefore the maximum storage required will be >1kB	Ensure that the chosen data structure can accommodate 1kB
FR3	FS3	FATP3
Transmitted Data must be presented to the user in a legible manner	User interface must be presented in a legible manner and clearly display the weight, temperature and timestamp	Execute the set up web server to determine if the presentation is legible and includes the necessary information
FR4	FS4	FATP4
The data processing must interface with the communication submodule	The data processing must produce a value of correct variable type/ structure that can be transmitted via the web server	input the output of the data processing algorithm (i.e. calculated weight) to the communication module and ensure there are no errors as it is properly displayed on the web server's user interface

Table 1.2: Functional Requirements, Specification and Acceptance Test Protocols

1.4 Scope & Limitations

1.4.1 Scope

The scope of the project is the observance and weight measurement of fork- tailed Drongos in the Kalahari. The current practise is a portable kitchen scale set-up with a perch attachment and the presence of a researcher to lay the bait and visually take readings off the scale using binoculars so as to not disturb the bird's environment. The surveyor also logs the weight data as well as other visual data into a data logging app called Cyber Tracker which is a platform for data logging specifically pertaining to environmental conservation. This scope requires in person, real time surveillance. The design proposed in this report does not diminish the need for an in person surveyor, it does however supply the surveyor with a more accurate weight measurement without their needing to read it off a display and manually record it thus decreasing transcription errors [18]. The design also adds value to their research as it accurately records additional variables such as time and temperature at the instance of weight measurement. Having these data points (weight, time, temperature) automatically recorded allows them the ability to focus on and collect additional variables (for example: age, sex, behaviour, appetite) that can add value to their research. An example of useful observational data, pertaining specifically to Ben's research, that can now be captured is how much of the bait is eaten by

the perched bird and how much of it is taken by that bird to feed his/ her chicks.

1.4.2 Limitations

The following act as limitations to the project:

- Environmental Exposure: the system will be positioned outdoors and therefore exposed to the harsh environmental elements. These are undesirable conditions for any electrical system as it can cause it to malfunction thus producing inaccurate or incorrect measurement data. These conditions can include: sun, wind, rain or dust exposure. The Kalahari has annual temperatures ranging anywhere between -12° and 46° depending on the time of year [19].
- Unpredictable Behaviour of Animals: the system aims to measure the weight of birds but cannot predict, without testing, that the bird's will behave in a manner suitable for the system's operation.
- Engineering Students Expertise: this system is being built by engineering students who lack, to some extent, the experience in building fully functional deployable systems. This includes their inexperience to fully understand the objectives of a non-engineering discipline, such as ornithology. Despite being in communication with Ben on the topic, face-to-face and via email, there is the possibility that due to the differing professions and associated terminology, nuances of the true objectives of the project may be misunderstood.
- Choice of Components: the project's design has opted to use a ESP32 microcontroller instead of a raspberry-pi. This is limiting as python programming has a greater programming ability and grants access to a variety of libraries whereas C++, the language used by the Arduino IDE, is more low level and more restricting in terms of what algorithms can be implemented.

1.5 Report Outline

This report acts as the final report for the project and includes the following sections:

- A Literature Review: where relevant literature and existing application of the same or similar projects are explored and analysed.
- Subsystem Design: where the four main subsystems - namely Hardware, Power, Data Storage and Processing, and Software and Communication - are identified and explored in terms of their compatibility and feasibility for the project's scope. Each subsystem chapter will include an introduction, a design process where different approaches are considered and their advantages and disadvantages outlined, an implementation where one of the design choices is implemented, a testing and results section and finally a conclusion.
- Final Design: the accumulation of the different subsystems and the final proof of concept or actual possible implementation.
- Conclusion: closing statement regarding the final design produced.

- Recommendations: where future development is commented on specifically building upon the subsystem's current implementation.

1.5.1 Contributions

The following table 1.3 indicates the students responsible for the sections of the project's report.

Contributions			
Section	**Section Number**	**Page Number**	**Student/s Involved**
Abstract	N/A	iii	Jonathan Apps
Introduction	1	2	Natasha Soldin
Literature Review	2	9	Jonathan Apps, Khelan Hari, Nikhara Naidu and Natasha Soldin
Hardware	3	18	Khelan Hari
Power	4	29	Nikhara Naidu
Software and Communication	5	39	Jonathan Apps
Data Storage and Processing	6	51	Natasha Soldin
Final System	7	61	Jonathan Apps, Khelan Hari, Nikhara Naidu and Natasha Soldin
Conclusion	8	63	Jonathan Apps
Recommendations	9	64	Nikhara Naidu

Table 1.3: Contributions Table

Chapter 2

Literature Review

2.1 Introduction

This review critiques the relevant literature on the technical subsystems used to design and develop a system to assist Sam with her research in ornithology by obtaining accurate measurements of the weights of drongos. The hardware section introduces the setup and construction of different scaling systems, the power section compares existing power solutions to determine which is most suitable, the memory and storage section emphasises the importance of data retention, the communication section highlights the various protocols that can be applied to the system, and finally a data accuracy and retrieval section to further investigate Sam's need for obtaining precise measurements to derive meaningful conclusions.

2.2 Hardware

Historically, various hardware implementations have been used to measure and record the mass of wild birds. These methods have evolved from simple weighing scales to more sophisticated electronic devices that can capture mass data in real time. Each implementation has its own advantages and limitations, and the choice of method depends on factors, such as the size of the bird, the nature of the study, and the resources available. The overall physical setup of past bird weighing solutions will be discussed in this section.

In 1995 Tibor Szép et al. [20] wrote a paper which describes a technique for using electronic balances in nests of the hole-nesting Bank Swallow to monitor their mass to provide insight into their foraging characteristics. The researcher's hardware implementation was carried out by first finding and digging out a current nest of the Bank Swallow and moving all material used to build the nest along with nestlings into their own specially designed nest box which they named a Riparibox. The Riparibox was a simple wooden box with a removable lid and a plastic tube sticking out of the side to connect to the tunnel that the birds use to enter and exit the nest. The Riparibox was then placed on a Mettler PM4600 scale (4100g capacity and 0.01g accuracy) in the original dugout nest. The scale was connected to a computer hidden 50m away by cables using the RS232-C serial interface and produced eight weight readings per second.

The advantages of this solution to weighing birds are that it is installed as part of the nest so it can give constant readings of weight changes at any time of the day and it only requires disturbing the birds once at the time of installation due to it being monitored at a distance. According to Megan

Emmet in [21], Hornbill nests are usually a cavity 20cm in width by 10cm in height with an entrance hole of about 2.5cm in width which is large enough to implement a similar scale in nest solution as was used for the Swallows. The main difficulty would be trying to set up the scale in the nest without overly distressing the female Hornbill and nestlings due to the fact that their nests are partially sealed with mud. On the other hand, Emmet [21] mentions that Hornbills are so specific about their nest requirements that they often reuse their nests year after year. Therefore, in anticipation of a new mating season, if a nest is known to be used by Hornbills beforehand, a nest scale can be set up before the female Hornbill builds and seals the nest.

In 1999, Reid et al. [22] wrote a paper which describes using an automated weighing system to measure chick provisioning in Antarctic Prions. The Antarctic Prions are a burrow-nesting species similar to the Bank Swallow mentioned above and the hardware implementation also broadly consists of putting an automated scale into the burrow such that the bird nests on top of it and the mass data is relayed to a data logger a distance away from the nest.

The scale comprised of a single load cell with a 2 kg capacity to transform load into voltage. The load cell was mounted on a 150 mm diameter aluminium base plate with 3 adjustable vertical limit stops to provide protection from overloading. The whole assembly was placed in a shallow aluminium dish and sealed to protect the electronics against water and debris by a thin neoprene membrane. A plastic cage was mounted on the platform to keep debris from obstructing the movement of the scale up and down and make sure that the chick's whole mass was on the platform. The amplifier unit provided a 4-20mA current loop output that is proportional to the load and was contained in a small water-resistant case connected to a data logger. Inside the data logger, a 100 ohm resistor was used to convert the current to a voltage value which was then measured and recorded in binary form. This binary data would then be downloaded to a computer where it would be converted to mass data using an integrated software package supplied by Francis Scientific Instruments, Cambridgeshire, UK.

The hardware just described by Reid et al. [22] is very similar to the approach taken by Tibor Szép et al. [20] with the key differences being how Reid et al. takes care in their design to make sure the scale is protected from moisture and debris by concealing the electrical components in water-resistant containers as well as the use of the cage like frame which ensures that no debris will get in the way of the movement of the scale platform. Also notable is that instead of requiring a computer to be recording and processing the data in real-time at a distance away from the nest, a data logger stores all data over a period of time and it is collected by connecting a computer to the data logger periodically [22].

The debris and water-resistant solutions described would be applicable in the case of a device to weigh a Hornbill in its nest since there is likely to be debris such as sand, mud and feathers and moisture from the occasional rains in the Kalahari desert. Some sort of general barrier like the cage around the scale would be important to ensure the Hornbills entire mass is positioned on the scale and that nothing in the nest can get between the moving part of the scale to obstruct its movement. The idea of recording all data on site would be beneficial to allow for data to be captured without the researchers having to be there which would enable them to gather more continuous readings and at times which they otherwise would not be able to.

A study in 2009 made use of a compact, weatherproof, and relatively inexpensive electronic scale for birds at perches [1]. Their scale takes advantage of the fact that many birds use habitual perches and will often shift unknowingly to artificial perches if these are placed at their habitual sites. This is particularly relevant in the case of Hornbills which, according to Benjamin Murphy (interview, February 10, 2023), easily move into the artificial nest boxes built to keep the Hornbills cooler than they would normally be in their self-built nests. It was found that the scale's accuracy was improved when the perch length was kept short, so a bird remains centred over the plunger when it landed on the perch - such centering minimised friction between the plunger and its guide. The perch unit base can also be attached to a perch site using C-clamps or duct tape, which is similar to the current implementation for the Hornbills. It helps to modify a perch site so that the perch base rests on a firm, flat surface which keeps the perch steady and perpendicular. It was therefore found that camouflaging the equipment would minimise the stress and movement of the bird. A diagram of this setup can be found in the figure 2.1 below.

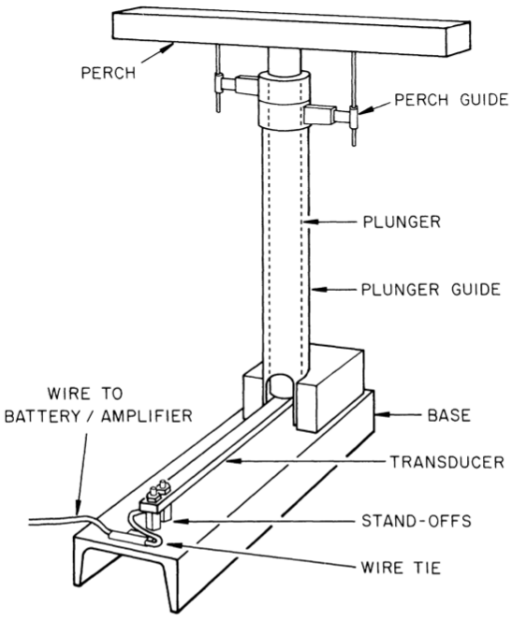


Figure 2.1: Perch Diagram [1]

An artificial perch setup like the one described above [1], would be particularly advantageous because Emmet [21] mentions that Hornbills are so specific about their nest requirements that they often reuse their nests year after year and this setup can therefore be reused either by the same Hornbills in the following year or by new ones.

2.3 Power

For the project's scope, a power source is required to supply power to the system in the context of a remote area to ensure uninterrupted operation. This power system will be evaluated in terms of its longevity and environmental impact as this project is in the field of remote observation and wildlife conservation.

The power source can be supplied in a number of ways, the most practical and commonly used of

which would be battery power or power derived from a renewable energy source. The latter can be of three main types: wind, micro-hydro and solar. These off-grid [23] sources provide direct current, unlike an electrical grid's alternating current. Given the location of the project is in the Kalahari desert, which in certain areas has limited tree shelter and temperatures reaching up to 40^oC [19], solar would appear to be an obvious choice of a renewable energy source. To make an informed decision, the two most compatible power source options for this project - battery and solar - are to be examined and compared based on previously implemented systems.

Firstly, looking at a battery-powered source, there are two types of batteries to be considered in this context: lead-acid or lithium-ion. A lead-acid battery is 'durable, efficient and recyclable' [24]. However, they are heavier by comparison and thus limit the scope of a system's implementation. A lithium-ion battery can supply a greater amount of energy but is more costly to implement. The pitfall of batteries in general is their harmful impact on the environment, whether it be from their material composition or the resources required in battery production.

Poole and Shoikimas's [1] implemented perch scale (as shown in figure 2.1 above) utilized two 6V lithium-ion lantern batteries encased in an ammunition box as its power source. It was noted that the battery life's longevity was around two weeks during which the system was in continual use. However, the nest and scale system was visited every 3-4 days and as such the batteries were changed as often as needed. The system was never allowed to reach the battery's maximum limitations due to continued replacement.

In comparison, Reid et al. [22] implemented an automated weighing platform that was powered by rechargeable lead acid batteries amounting to an average power consumption of 0.83mA at 12V per platform.

In considering the renewable energy source alternatives, Al-Bahadly explored the idea of a portable renewable energy system for small-scale remote locations [25]. In this case, the application was for a reliable energy source required to power basic electrical necessities after the occurrence of natural disasters. These necessities included water purification, basic satellite communication and first responder's usage. The solution in place at the time was the use of diesel generators which in addition to being environmentally damaging are not a reliable solution due to the restricted access to diesel following these damaging events. Al-Bahadly proposed a hybrid system in which the use of renewable energy power generation works in with conjunction with battery energy storage. A DC-DC converter would be configured such that a controlled energy supply can be obtained from an uncontrolled renewable energy source. This application's scope reaches further than just solar and includes wind and hydro turbines which would not be applicable to this project in a desert environment that is best suited to using solar energy as explained above. Regardless of the differences in their application and environment, the solar module proposed in this paper does provide insight as a possible application. This comprises of three 55W solar panels connected together in parallel to create a 12V, 165W solar array which in turn is connected to the power storage unit via an extendable 12-gauge cable. The solar panels can be configured such that they track the sun's movement and obtain maximum sunlight exposure and thus maximum energy absorption. This type of solar application is a consideration in the context of this project.

A power system used in a similar project setting to ours, in that it to deals with wildlife monitoring, was examined by Chaithra and Ajay's [26] proposal of a smart farm monitoring system that utilises solar power and internet of things (IoT). This system aimed to monitor livestock's conditions for better care and intervention. The proposed solution involved a Star Solar D165X165 monocrystalline solar panel with an output of 6V at a peak of 3,65W. A power converter would be used to deliver this generated power to the various sensor nodes, namely temperature, soil moisture and humidity. A Lithium Polymer (LiPo) battery storage component was used to deliver a constant 5V or 3V output depending on the sensor's input voltage requirements.

Although solar energy would be suitable for this project because of the desert environment in which it is situated and being environmentally friendly, its implementation and up-keep costs far out way that of a battery powered option. In many cases, as illustrated by Al-Bahadly [25] as well as Chaithra and Ajay [26] above, the solar power was implemented in conjunction with battery storage emphasizing the common base requirement for the use of batteries regardless.

There are techniques by which a battery's harmful environmental effects can be limited. For example, using rechargeable batteries to eliminate one-off usage as in Reid et al.'s implementation [22] or proper recycling of depleted batteries, which are classified as hazardous waste and thus managed under the Environmental Health and Safety Act [27].

It is also notable that much like in Poole and Shoikimas's [1] implementation, this project site would need to be visited regularly to take readings and/or affix bait to the perch in order to attract birds. Thus, regular battery replacement or recharging would not be a problem.

2.4 Memory And Storage

The importance of having enough memory and storage in the case of data capture for research is that it ensures all the necessary computations will always be able to be performed and that no data will be missed.

The paper by Bouten et al. [28] details the design of an ultra-lightweight Global Positioning System (GPS) tracker and transmitter for studying bird behaviour at multiple scales. Due to the need for an ultra-lightweight design, every piece of hardware had to be made as small and light as possible with no room for excess. This meant that even down to the storage and memory components had to be minimal. Their storage solution used a 4-MB flash memory to store data, which was implemented as a ring buffer. This type of memory allows new data to overwrite old data when the buffer is full, ensuring that data is continuously being recorded. The flash memory had a storage capacity of almost 60,000 GPS records, which included information on the GPS positions, timestamp, and battery status. In the case where the tracker also needs to record speeds (x, y, and z) and 3 seconds of accelerometer data at 20 Hz with every GPS measurement, the storage capacity will decrease significantly to only be able to store about 4,000 GPS entries along with the corresponding accelerometer data. This is because accelerometer data requires more storage space due to its high sampling rate and the amount of data generated per sample. This highlights the importance of managing data effectively to avoid losing important information.

In the Antarctic Prions weight measuring implementation of Reid et al. [22] , each scale data logger was only able to store 16 000 readings in binary form and the readings from these data loggers would have to be taken every 10 min on a computer. This would be an unfavourable solution to the Hornbill research due to it having to be far too hands-on and therefore hardly an improvement over the current method where they have to add bait to the perch and check each weight reading from the scale individually.

Benjamin Murphy (interview, February 10, 2023) also mentioned that they make use of Secure Digital (SD) cards in the Gigabyte range for their cameras and as mentioned above, they tend to the Hornbill sites on a regular basis which would allow them to download data during those visits and avoid the need for a mass storage solution. When storing mass data for the Hornbills, the most complex it could likely be is simple mass values and timestamps which can easily be stored using a low-cost SD card. Additionally, most off-the-shelf micro-controllers should be able to convert and store analogue data into mass and time values.

2.5 Software and Communication

The deployment of technologies which utilize remote data transmission (RDT) necessitates the use of communication, whether wired or wireless. Selecting a communication method and protocol that are suited to the problem at hand is of paramount importance to the success of the design at large. A setup to receive the transmitted data (such as a web server or the like) completes this subsystem.

In terms of the methods, protocols, and interfacing techniques that are available, the complexity of communications is largely inevitable. Despite this, a boom in network development over the past decades has led to near 'worldwide coverage', with nearly 95% of the world's population now living in areas covered by mobile networks [29]. The International Telecommunications Union (ITU) uses the term 'global interconnectedness' to describe this reality [30].

Despite this, rural areas such as the Kalahari desert remain distant from land based networks, and thus long-range connectivity in these regions is oftentimes limited to satellite communications, deployed using a 5G architecture. Although many satellites require interfacing with small cell networks on land, the modern and low-latency Low Earth Orbit (LEO) satellites enable uplink and downlink communication in remote areas that are not covered by cell networks [31]. However, LEO satellites are privately deployed, and are thus generally inaccessible or too costly for small remote applications.

The issues of inaccessibility and deployment cost as mentioned above often lead to potential 'remote' sensing technologies being entirely monitored by humans. Although Holmstrom and Beckham [32] correctly note that 'data collection methods are swinging from paper based methods and are turning to online data collection tools', the fact that online data collection tools often require user presence in the remote environment reveals the deployed technologies are not truly 'remote' after all. However, many researchers and scientists are willing to adventure into remote regions to collect data with 'semi-remote' (wireless communication devices requiring human presence) technologies in hand.

For semi-remote applications, short-distance wireless communications are generally sufficient depending on the exact application. Radio access technologies (RATs) such as WiFi, Bluetooth, NFC (Near Field Communication), and Zigbee present well trusted and high performing wireless connectivity options.

Bluetooth is the most widely used and well-loved short-distance wireless communication RAT/protocol [33, 34]. Due to its simplicity and multiple operating modes, it is appropriate for low-power applications that require prolonged battery life or low power consumption [33]. It is additionally low-cost and widely available on development boards such as Raspberry Pis and Arduinos (In both Station (STA) and Soft Access Point (AP) modes). Certain medical researchers [33] found that Bluetooth was an effective means of developing smart healthcare systems. They were able to add security features to their applications through the use of a deep neural network to determine if communication interceptions or cyber security attacks were occuring. Bluetooth is indeed an insecure transmission protocol due to its use of the 2.4GHz unlicensed spectrum because anyone can use the same frequency band. As Collotta et al. [34] note, Bluetooth has 3 classes, each with distinct operations. In their words, 'Class 1, the most powerful, can reach up to 100 meters; Class 2, the most common, operates only within 10 meters; Class 3, on the other hand, does not exceed 1 meter and is also the least used, especially recently'.

Near Field Communication (NFC) is a short range communication protocol that utilizes magnetic induction to enable communication between devices [35]. Studies have shown that NFC is useful for IoT applications since it is 'the easiest wireless technology to set up a connection' [35] with. Despite this, it is not good at maintaining a connection over distances or for long periods [35]. Utilizing the 13.56MHz spectrum, it allows data transfer over a few centimetres at best [36]. NFC is supported on development boards and microcontrollers, while it additionally offers complementary support for Bluetooth and Wi-Fi [35].

Zigbee is used in multiple semi-remote sensing applications such as localized GPS trackers for birds [28] and cattle or animal monitoring [37]. In Bouten et al.'s work on flexibly monitoring bird behaviour [28], they attached lightweight and non-disruptive Zigbee communication devices to birds so as to track and monitor their behaviour. They noted that compared to Bluetooth, Zigbee can handle more sensors (in terms of a network), consume far less power, transfer data over longer distances, and be scaled more easily (using a network of antennas). They factually stated that these benefits 'come at the cost of a lower data transfer rate' [28].

If a semi-remote system requires a wide scale deployment, a wireless sensor network (WSN) could come in handy. Use of a WSN is advocated for by Handcock et al. [38] who employed a WSN to monitor animal health in north-eastern Australia. They connected a series of devices to different cattle, with each device acting as a node for other cattle's devices to communicate with. This approach requires a network of nodes distributed throughout an environment such that nodes can communicate with neighbouring nodes [38] . This however requires a complex setup and a protocol selection based on user requirements and needs.

2.6 Data Accuracy and Retrieval

Most common weight measurement systems are not suitable to obtain data of dynamic objects [39]. In general, they only offer a precise measurement when the object placed on the scale is stable. In a dynamic system, accurate measurements do not apply [2].

In addition, if the dynamic object is an animal, and in this case a bird, the system further requires that the bird is not stressed. Moreover, it is difficult to keep the bird in a static position for a long time

and the random movements of the bird do not allow the system to obtain regular weighing patterns. Many scientists, engineers and researchers have thus conducted multiple investigations and studies on various methods used to obtain accurate weight measurements for these dynamic objects.

While researching increasing the accuracy of unstable measured weights of wild birds in Northern Australia [2], a group of engineers developed an Artificial Neural Network algorithm (ANN) that allows estimating a weight value from the obtained unstable values of a weighing event. Their system cannot only obtain patterns of body mass variation throughout the breeding period, but also allows for studying daily weight fluctuations.

Professors at the Central University of Technology conducted an investigation into determining the weight of moving objects by designing a conveyor system with the aid of Digital Signal Processing (DSP) techniques [39]. Their Digital Signal Processing analysis, algorithm design and simulations were done in MathCAD, and implementation on DSP hardware using Hyperception's Visual Application Builder (VAB). This system can measure the weight of moving parts via a sliding scale (Tedea-Huntleigh 1010 load cell), filter out industrial noise from samples, auto-zero after every sample, calculate the friction coefficient and log all of this data accurately.

In February 2013 [2],a study was conducted to increase the accuracy of the unstable weight measurements obtained from wild birds. They called it the 'smart scale system'. It allows obtaining weight estimations from both stable and unstable measurements. This algorithm has been designed to fulfil the following goals: to reduce the amount of useless data, increase the accuracy of the measurements (around 5g), to increase the amount of useful data, estimating a body weight from patterns with non-stable measurements and to allow its execution on devices with low resources.

The proposed system allows increasing the accuracy of the measurements using a tare calibration. This calibration is taken from the weight measurements obtained when there is nothing on the scale pan. The algorithm used to increase the amount of useful data is based on a computational intelligence algorithm. The algorithm used to estimate weight over the dynamic system is summed up in the figure 2.2 below.

Multiple studies show that measuring the weight of moving objects is most accurate when the final value obtained is an average of the weights measured for the duration of the object being on the scale.

To conclude this literature review, the knowledge obtained from the consulted sources has provided great insight into an efficient and functional scale design. Although there was no literature dedicated specifically to the weight monitoring of hornbills, alternative sources were found that provided the necessary information regarding the system's subsections.

From the preceding research, it appears that the most suitable application for Sam's context would be a perch scale, powered by a rechargeable battery source with sufficient memory capacity provided by an off the shelf micro-controller and sufficient SD card storage. In terms of final decisions regarding communication, data accuracy and retrieval further investigation is required. Final decisions on all subsystems will be made during the design and implementation phases of the project and are thus all subject to change if it is determined that they are incompatible.

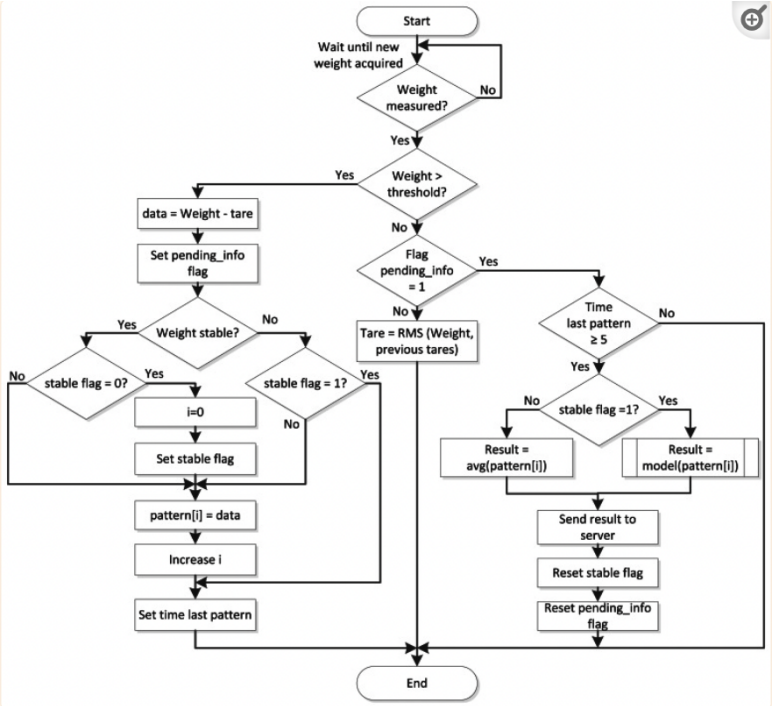


Figure 2.2: Algorithm used to Estimate Weight[2]

Chapter 3

Hardware

3.1 Introduction

The hardware sub-module will deal with the overall form factor of the system. It takes into account the physical, mechanical and analogue electrical components of the system excluding the power supply. The hardware will seek to fulfil the applicable user requirements set out in table 1.1.

This chapter will first discuss the design decisions made in pursuit of fulfilling the user requirements. It will then describe the implementation and finally discuss how the testing results compare with the target specifications.

The design philosophy for the hardware was to adapt the researcher's current implementation by making improvements to it. The reason for adapting the current design rather than performing a complete redesign is so that the system would integrate easily into the researcher's current routine and that it would be familiar to the Drongo's which are habituated to the current setup.

3.2 Design

The design and choosing of the various components seek to satisfy the specifications as laid out in table 1.1. Specifically US1.1, US2, US3, US4.2, US5, US9 and US10.

The following approaches were considered for each aspect of the hardware design.

3.2.1 Load Cell Selection

In order to satisfy the specification for the weight reading hardware to output a weight value that is accurate to 0.1 grams and for that value to be read by a micro-controller, one must consider how the weight of an object can be translated into an electrical signal. The researcher's current implementation makes use of a kitchen scale to produce a weight value when the bird lands on the perch mounted on top of the scale. In principle, the bird sitting on the perch produces a force equal to its weight which is transmitted through the components it is sitting on and causes them to change shape. This change in shape is proportional to the bird's weight and therefore its mass, so measuring a component's change in shape under the bird's load will allow one to determine its mass.

The above-mentioned principle will be used in designing a device that can measure the bird's weight and convert it to an electrical signal so that a micro-controller can read it.

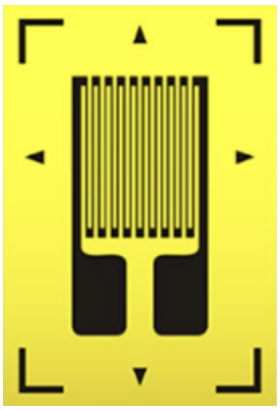


Figure 3.1: Linear strain gauge

[3]

First, a simple cantilevered beam is considered. When a downward force is applied to the end of the beam, it bends such that its top surface experiences a tensile force which causes it to stretch and its bottom surface experiences a compressive force which causes it to shorten. Since the objective is to produce an electrical signal proportional to the change in weight, a linear strain gauge 3.1 can be used to convert the change in length of either of the beams surfaces into a change in voltage. A linear strain gauge 3.1 achieves this by being placed longitudinally on the beams surface such that when the surface stretches or shortens, the longitudinal length of the parallel wires on the strain gauge also stretches or shortens which respectively increases or decreases its resistance. This change in resistance can be measured as a change in the voltage across the strain gauge due to the relationship between resistance and voltage denoted by ohms law $R = V/I$. At this stage, if a bird were to land on the end of the beam, it would bend resulting in its top surface stretching which would stretch the strain gauge longitudinally and increase the voltage drop across it in proportion to the bird's weight.

This solution, however, is not able to account for changes in the temperature of the beam which will also cause it to stretch or contract and would change the resistance of the strain gauge which would produce voltages that are different for the same force on the beam at different temperatures. The effect of temperature needs to be accounted for since the system is used in the Kalahari desert which fluctuates in temperature throughout the year, day and whether in direct sunlight or shade. The effect of temperature can be solved by placing a second strain gauge transversely on the same surface as the first one. The second transverse strain gauge does not get affected by the beam's change in shape due to bending and rather only changes length based on the beam's expansion and contraction due to changes in temperature. When the two strain gauges are used in a half bridge configuration, a change in temperature causes them both to lengthen or shorten by equal amounts so the change in resistance due to temperature is compensated for and only the longitudinal strain gauge changes resistance under a load.

In order to make the strain gauge and beam configuration so far more robust, a configuration shown in figure 3.2 below should be used. By mirroring the placement of strain gauges on the top and bottom surfaces of the cantilevered beam, the sensitivity to bending is decreased, but, the accuracy of the change in voltage relative to the applied force is increased. The mirrored strain gauges compensate for minute imperfections in the overall construction of the configuration.

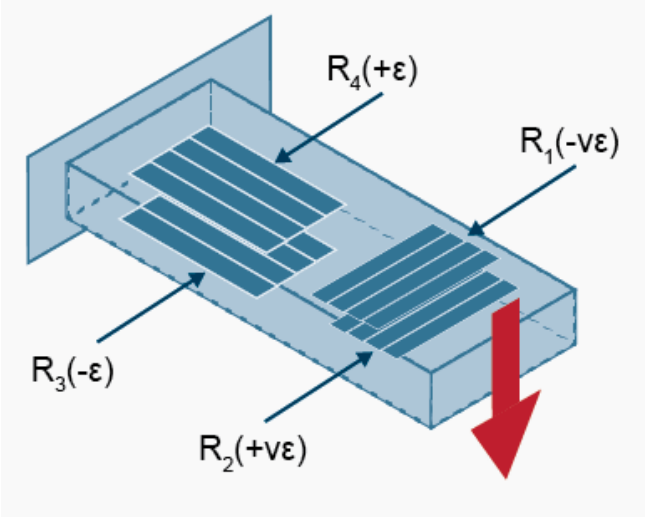


Figure 3.2: Strain gauge configuration for load cell [3]

The resulting electrical configuration of the four strain gauge setup will be a Wheatstone Bridge as shown in figure 3.3 below.

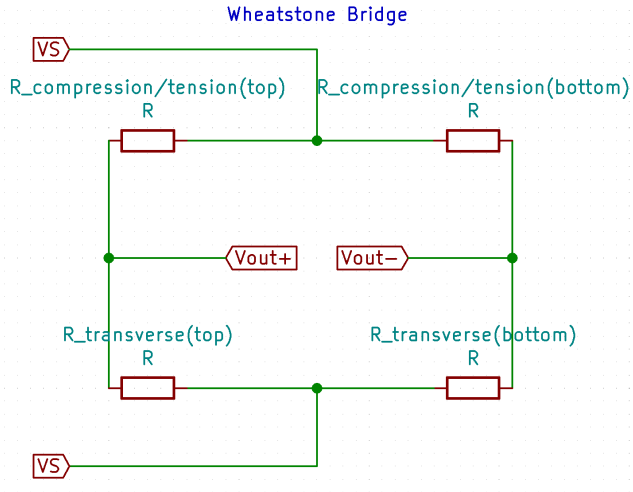


Figure 3.3: Wheatstone Bridge configuration for strain gauges

The Wheatstone Bridge configuration above will produce an output voltage given by the equation:

$$Vout = Vsupply((Rtranstop/(Rtranstop+Rlongtop)-Rtransbottom/(Rtransbottom+Rlongbottom))$$

The full above-mentioned configuration can be made by glueing the strain gauges to the mid-span of a rigid beam. This beam can be mounted such that it is cantilevered and should bend in proportion to a force applied at the end.

This design ends up being a derivation of a typical load cell as shown in figure 3.4 below. Due to load cells like the one shown in figure 3.4, being readily available with all the capabilities including temperature compensation at a relatively low cost of approximately R45, a 1kg load cell was chosen for the system. The load cell was chosen to be 1kg because it is the minimum widely available load cell size that can be purchased and it would only need to support a weight of approximately 500g to

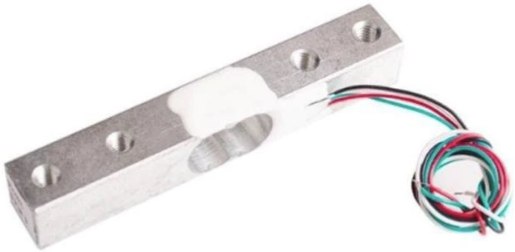


Figure 3.4: 1kg load cell

[40]

account for any perch attached to it as well as the weight of a Drongo which weighs up to a maximum of approximately 60g. The load cell above also comes with threaded holes at either end which makes mounting it in a final system as well as a perch to the opposite end simpler.

3.2.2 Amplifier

The chosen load cell in figure 3.4 above is unable to function and output a voltage signal on its own. Firstly, it requires input power such that there can be a voltage drop over the strain gauges in the Wheatstone bridge 3.3 and secondly, the voltage signal produced needs to be large enough to be read by a micro-controller ADC (analogue to digital converter).

The first problem can be solved by inputting a known stable voltage into the load cell. The second problem requires amplification which can be achieved by the circuit show in figure 3.5 below. The

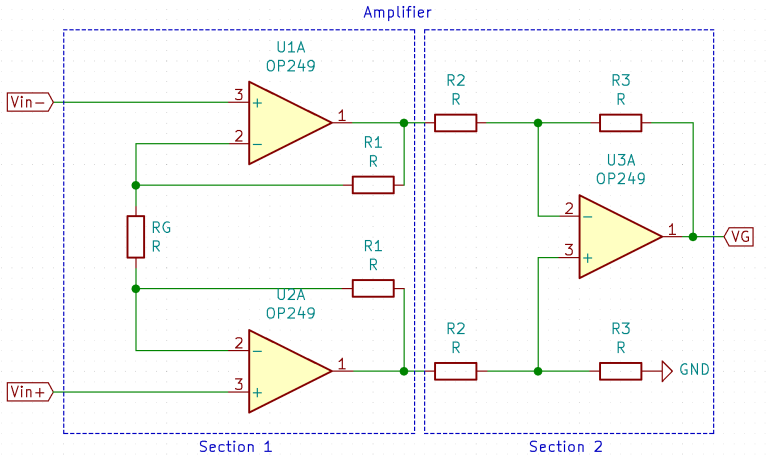


Figure 3.5: Voltage amplifier for load cell

voltage amplifier works in two stages. Section 1 in figure 3.5 above uses two op-amps to amplify the difference in voltage across the Wheatstone Bridge. The gain of Section 1 can be calculated using the following equation:

$$Gain1 = R1/(R1 + RG)$$

Section 2 in figure 3.5 above uses a single op-amp to further amplify the output from section 1 using the following equation:

$$Gain2 = R3/R2$$

The total output gain of the overall amplifier is the gain of each section multiplied and is given by the following equation:

$$Gain = Gain1 * Gain2$$

The advantage of using this amplifier configuration is that the resistors in each section can be chosen in any combination to achieve any factor of amplification. This amplified signal can then be fed into a micro-controllers ADC where a mass value can be calculated from the digital value.

A decision against building the voltage amplifier design shown in figure 3.5 was taken out of interest for reliability. It is crucial that the signal from the load cell is amplified without any distortion as this could result in incorrectly calculated mass values. The design lacks certain features that would help to reduce noise from interrupting the signal.

Instead, the HX711 load cell amplifier and 24-Bit ADC module was chosen for its high precision, reliability, wide availability and relatively low cost of approximately R30. The HX711 features a power supply regulator for the load cell so that the load cell does not need any external power supply circuitry and it can be guaranteed that the load cell's power supply will be controlled and known. Due to the HX711 being designed for use with load cells, it makes them widely available and often sold together with a load cell. This together with its relatively low cost makes it easily replaceable if necessary. The HX711 also has the added benefit of having its own 24-bit ADC which will have been designed with the use of load cells in mind. It negates the need to rely on the general-purpose ADC in a micro-controller. Furthermore, by outputting a digital signal rather than an analogue signal, the signal from the HX711 amplifier to the micro-controller is not susceptible to noise.

3.2.3 Temperature Sensor

In order to satisfy UR9 in table 1.1, a temperature sensor would need to be chosen to be added to the system.

Two temperature sensors will be compared for their suitability to this project, namely the MCP9701 and the TMP117. The MCP9701 is known for its simplicity and relatively low cost of approximately R60. It provides a linear output voltage proportional to temperature changes and has a wide temperature range. Its main downside is its low accuracy as it ranges between ±2°C. On the other hand, the TMP117 sensor offers higher accuracy and precision of up to ±0.1°C. It provides a digital output, low power consumption, high-resolution measurements, and communicates over the I2C interface which simplifies integration into complex systems. The TMP117 does however cost over three times that of the MCP9701 at approximately R200.

Due to the user not requiring highly accurate temperature readings and a need to keep costs as low as possible, especially in the case of replacing a broken component, the MCP9701 was chosen.

3.2.4 Housing and Base

According to table 1.1, UR3, UR4 and UR5 state that the system must be approachable by a forked-tail Drongo, be deployable in an outdoor environment and be portable. The respective specifications derived from these user requirements are that the entire system should be camouflaged in the colours

of the bird's usual environment, the enclosure of the electrical components must be made of water and dust-resistant materials and the total system must weigh less than 5kg and fit in an average backpack.

Figures 3.6 and 3.7 show designs of a box-like housing and a lid for the housing respectively.

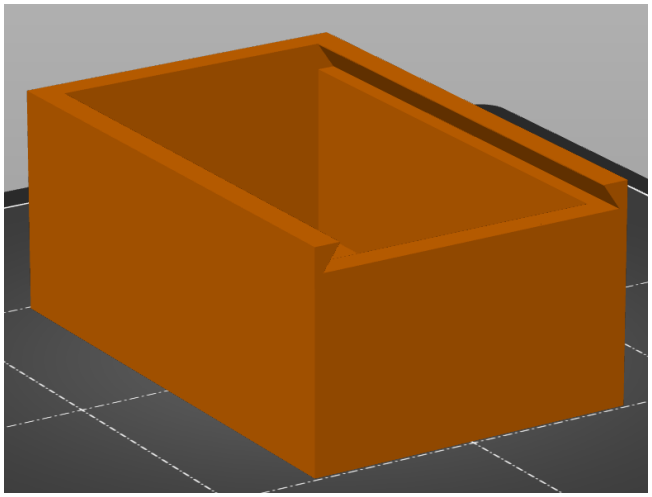


Figure 3.6: Base housing design

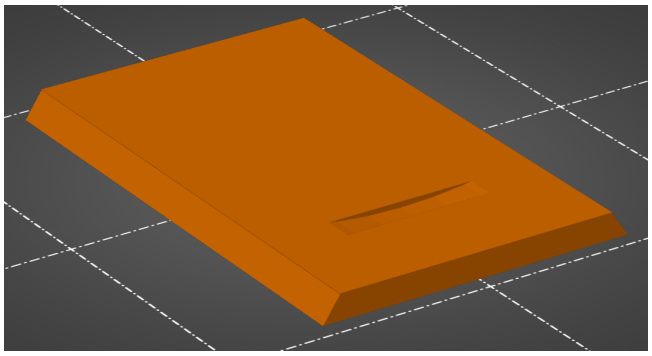


Figure 3.7: Sliding lid design

The sliding lid in conjunction with the base housing seeks to satisfy the user requirements highlighted above. This is done by designing a container for all the electronic components to be housed in a small enough form factor, 18cm by 10cm which is small enough to fit in a backpack. The housing will be 3D printed using Polylactic Acid (PLA) which is water and dust resistant as well as lightweight such that the overall weight will remain under 5kg. PLA was chosen over Polyethylene terephthalate glycol-modified (PETG) because PLA has lower printing temperatures, exhibits less warping and shrinkage during the printing process and it adheres better to the print bed making it a more user-friendly material to work with. While PETG has better impact resistance and can withstand more mechanical stress, this system does not require too much strength as it is only to shield electrical components from the elements and the maximum mechanical stress it will be required to withstand is of a Drongo on perch would weigh in the range of 500g altogether. An added benefit is that PLA is derived from renewable resources such as cornstarch or sugarcane and is biodegradable under certain conditions. It is therefore considered more environmentally friendly compared to PETG, which is derived from petroleum-based sources.

The sliding lid was chosen to allow the user access to the internal electronics in the case where the system needs to be charged or components repaired or replaced. The sliding configuration allows the lid to be secured without any extra moving parts such as hinges or clips which reduces the overall complexity and the amount of things that can potentially break. Lastly, the sliding lid provides a surface onto which the load cell 3.4 with subsequent perch 3.8 can be mounted. The decision to mount these parts on top of the lid is to make sure the load cell is not obstructed by any of the components inside the housing and to give it some extra height.

3.2.5 Perch

According to table 1.1, user requirement two states that the system must include a perch appropriate for a Forked-tail Drongo and the derived system specification for this requirement is that the perch must be able to bear the weight of the bait and the Forked-tail Drongo which would be a maximum of 60g.

The researcher's current setup makes use of a T-shaped perch which the Drongos are habituated to. In pursuit of creating a design that the Drongos would be familiar with and hopefully, therefore, adapt to the new setup, a similar T-shaped perch design will be used.

The first consideration that must be made for the design of the perch is the width of the horizontal part of the perch that the Drongo lands on. A 10mm diameter was chosen for the thickness since this matches closest to that of the current setup.

The second consideration will be the height and length of the perch. The height of the current setup is 40cm so this will be kept. The length of the horizontal part of the perch is currently approximately 15cm to give the Drongo enough space to land next to the bait and for it to bend its head down to pick up the bait. This dimension will also be maintained.

The last consideration that must be made is the orientation that the perch will be fixed to the load cell. Two options are available, the perch can be fixed with the horizontal part sitting parallel to the longitudinal axis of the load cell or it can be fixed with the horizontal part perpendicular to the longitudinal axis of the load cell. The issue with the first option is that if the Drongo does not land directly in the middle of the horizontal part, a torque around the base of the perch and along the bending axis of the load cell will be created. This will either increase or decrease the measured weight of the Drongo depending on which side of the centre of the perch it lands which will not be its true weight. The second option solves this problem because no matter where on the horizontal part the Drongo lands, the torque produced around the base of the perch will not be along the bending axis of the load cell and therefore will not add or detract from the true weight of the Drongo. The torque will still cause some torsion through the cross-section of the load cell, however, this effect on the measured weight is almost negligible in terms of the accuracy required. Figure 3.8 below shows this configuration where the rectangular prism at the base of the perch represents the load cell.

The material that the perch should ideally be made from is 10mm wooden dowel which can be cut and glued or nailed together to form the T-shaped perch. Wooden dowel is strong enough to support the Drongos weight and is light enough to be well under 1kg which is the maximum weight the load cell is capable of measuring. The wooden dowel is also cost-effective to be replaced.

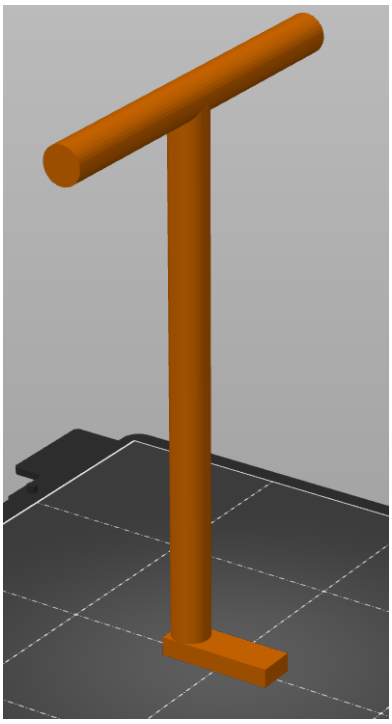


Figure 3.8: Perch design

3.3 Implementation

The implementation of the system closely follows the design laid out in the hardware design section 3.2 above and is shown in figure 3.9 below. The main notable difference is the use of a 3D-printed perch

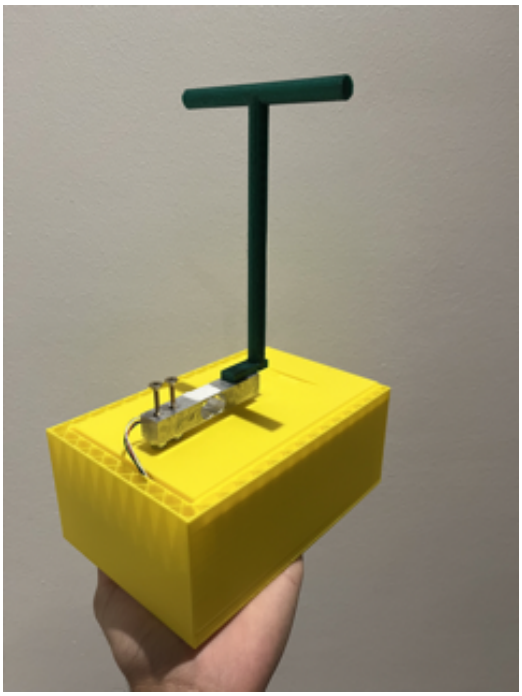


Figure 3.9: Final design

instead of the wooden dowel mentioned in the design. The 3D printed perch was used as a matter of convenience and an exercise in 3D printing. A benefit of it is that it is much lighter than the wooden

dowel would be, however, this came at the cost of it not being as durable and in fact being quite brittle.

The entire system is able to fit in a person's hand as can be seen in figure 3.9 above. A 20% fill factor was used in the 3D printed housing which allows it to be strong enough but also light enough for portability.

The micro-controller, HX711, power module and temperature sensor were installed on a breadboard and situated inside the housing as shown in figure 3.10 below.

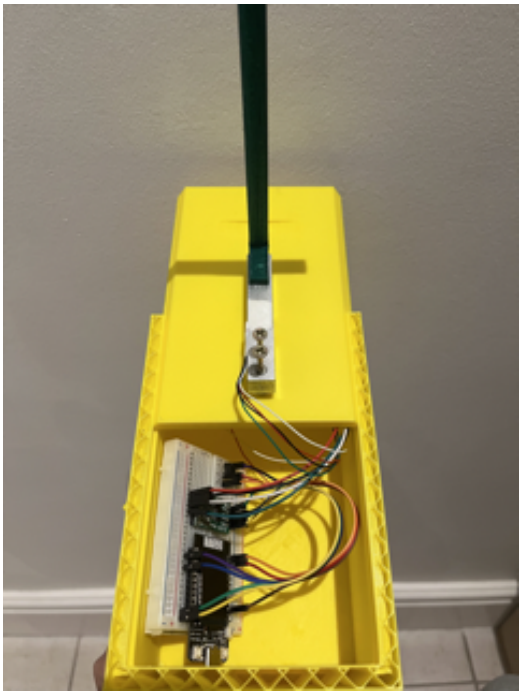


Figure 3.10: Internal configuration of system

3.4 Testing and Results

Referring to table 1.1, specification US1.1 pertains to the most crucial user requirement which is to produce accurate weight measurements. This was tested by putting objects of known weight onto the perch and comparing the measured weight to their known weight. Five test objects were used and the results are shown in figure **??**.

While the system did not meet the requirement of having a 0.1g accuracy, it ended up producing mass values accurate to within 0.5g which shows proof of concept.

3.5 Conclusion

This chapter has comprehensively outlined the hardware subsystem by discussing its five main component design choices: load cell, amplifier, temperature sensor, housing, and perch. The relevant user requirements as laid out in table 1.1 were considered closely when making design decisions to best address them. Each of the components also had to take their effect on one another into account as each specification of each component would influence what could and could not be chosen for the next one.

Test 1: 53 grams:

```
11:56:34.820 -> Scale reading:  0.00    Temperature reading:    19.03
11:56:34.919 -> SENSOR VALUE = 3655
11:56:34.919 -> VOLTAGE = 3.57
11:56:34.919 -> Scale reading:  0.00    Temperature reading:    18.64
11:56:35.019 -> processing block entered
11:56:35.019 -> weight, temp, timestamp = 53.32, 18.39, 00:06:23
```

Test 2: 57 grams:

```
12:07:57.333 -> Scale reading:  -0.30   Temperature reading:    18.30
12:07:57.432 -> SENSOR VALUE = 3644
12:07:57.432 -> VOLTAGE = 3.56
12:07:57.432 -> Scale reading:  -0.30   Temperature reading:    18.10
12:07:57.532 -> processing block entered
12:07:57.532 -> weight, temp, timestamp = 57.26, 18.47, 00:01:04
```

Test 3: 49 grams:

```
12:12:46.701 -> Scale reading:  -0.10   Temperature reading:    20.11
12:12:46.800 -> SENSOR VALUE = 3659
12:12:46.800 -> VOLTAGE = 3.58
12:12:46.800 -> Scale reading:  -0.10   Temperature reading:    18.84
12:12:46.899 -> processing block entered
12:12:46.899 -> weight, temp, timestamp = 49.17, 18.40, 00:05:53
```

Test 4: 44-45 grams:

```
12:14:51.725 -> Scale reading:  0.00    Temperature reading:    19.33
12:14:51.826 -> SENSOR VALUE = 3679
12:14:51.826 -> VOLTAGE = 3.60
12:14:51.826 -> Scale reading:  0.00    Temperature reading:    19.81
12:14:51.926 -> processing block entered
12:14:51.926 -> weight, temp, timestamp = 43.55, 18.40, 00:07:58
```

Test 5: 48 grams:

```
12:16:33.985 -> Scale reading:  -0.10   Temperature reading:    18.84
12:16:34.084 -> SENSOR VALUE = 3650
12:16:34.084 -> VOLTAGE = 3.57
12:16:34.084 -> Scale reading:  -0.10   Temperature reading:    18.40
12:16:34.183 -> processing block entered
12:16:34.183 -> weight, temp, timestamp = 47.53, 18.44, 00:01:41
```

Figure 3.11: Results of five test weights

Once a final design had been set out, it was implemented as close to the intended design as possible and finally, testing was done to determine whether the ATP's as set out in table 1.1 were met. It was

found that UATP1.1 was not met, however, the system showed promise and with some fine-tuning should be able to meet it.

Further improvements would include painting the setup to camouflage it according to the Drongos environment and to extend the height of the perch to a more reasonable height.

Chapter 4

Power

4.1 Introduction

This module consists of the different approaches and designs that were taken to build a power subsystem to power the chosen ESP32 microcontroller. This subsystem is required to be safe, predictable, reliable and is needed to power the ESP32 microcontroller for a full day while the perch is set up and ready to take its readings. There were two circuits designed and implemented to perform this application. The first is a circuit consisting of a circuit initially built with the internal and external circuitry of the L5973AD Integrated Circuit (IC) obtained from its reference sheet [5], and the second is a Power Management Module with the capability to be charged by a solar panel.

The L5973AD is a step-down monolithic power switching regulator with a switch current limit of 2A so it is able to deliver more than 1.5 A DC current to the load depending on the application conditions. The output voltage can be set from 1.235 V to 35 V. This meets the requirements as the input voltage to the ESP32 is 3.3V [41]. The high current level is also achieved due to an SO8 package with exposed frame, that allows the current to reduce the RthJA (Maximum thermal resistance junction-ambient) down to approximately 40 °C/W. The device uses an internal P-Channel D-MOS transistor (with a typical of 200 mΩ) as switching element to avoid the use of a bootstrap capacitor and guarantees high efficiency. An internal oscillator fixes the switching frequency at 500 kHz to minimize the size of external components. Having a minimum input voltage of 4 V only, it is particularly suitable for a 5 V bus, and is available in all computer related applications. The pulse by pulse current limit along with the internal frequency modulation offers effective constant current short circuit protection [5].

This Solar Power Management Module is designed for a 6V-24V solar panel. It can charge a 14500 3.7V Lithium-Ion battery through the solar panel or a USB connection and provides a 5V/1A regulated output through a USB type A port. The module also features MPPT (Maximum Power Point Tracking) function and multiple protection circuits. It is able to keep working with high-efficiency, stability, and safety. It is suited for solar powered, low-power IoT, and other environmental protection projects [8].

These two circuits will now be compared to be able to choose the most effective solution for this design problem. They will therefore be further discussed in this section along with their advantages and disadvantages to assist the decision to conclude on one.

4.2 Design

The main requirements of the Power Module, as can be seen in the User Requirements, Specification and Acceptance Test Protocols table is that it is able to attain its battery life for a full day whilst the birds are being monitored, and it should be safe whilst having to function outdoors. Thus, the two main aspects of Battery Life and Protection Circuitry were considered during the design process. The advantages and disadvantages of various design choices to implement these aspects will be discussed in this section.

4.2.1 Battery Life

The battery is required to power the system for the entire day while the birds' weights are being measured, which Ben said takes approximately 8 to 12 hours each day. A 3.7V rechargeable lithium-ion battery usually lasts between 8 and 9 hours without recharging whilst functioning on high power mode [42]. This is therefore only sufficient to keep the system powered for a partial duration of the period specified without needing to be recharged.

In the designed circuit, an IC will be used to do the battery charging. There are two options: the BQ2954PN or the MCP73831. The MCP73831 is a miniature single-cell, fully integrated Li-Ion, Li-Polymer charge management controller [7] and the BQ2954PN uses a flexible pulse-width modulation regulator to control voltage and current during charging [43]. The MCP73831 has 5 pins and the BQ2954PN has 16, therefore the MCP73831 has the advantage of designing a more compact circuit, along with a greater temperature operating range and five more voltage regulation options.

In order to save battery life, it was also suggested to put the ESP32 into sleep mode when not in use. However, this was not possible as there needs to be a time period specified with which to put the microcontroller into sleep mode and during this time, it is possible that a bird could potentially sit on the perch and the readings will not be taken if in this mode. Therefore, putting the ESP32 into sleep mode when not in use to save battery life was not possible.

With regards to the Power Management Module, a 3.7V lithium-ion battery is placed within a 14500 battery holder. There are two options of charging the battery – either through a Micro USB input voltage of 5V or through a solar panel input voltage of between 6V and 24V.

Thus the Power Management Module has the advantage of using a solar panel as it will continuously charge the battery so long as it is charging itself and thus provides backup in the unlikely situation that the battery does completely discharge during the period it needs to be in use. This continuous charging of the battery is also the best way of protecting the energy density and longevity of the battery by keeping it from being fully discharged. There's excessive stress on a battery when it's fully discharged on a regular basis- it will end up lasting for a shorter time period than it should [44]. Another added advantage of the Power Management Module over the designed circuit is that there is an LED indicator to show when the battery is fully charged either through the USB input or the solar power input. This is beneficial for someone waiting for the system to charge fully before using it.

4.2.2 Protection Circuitry

In order to keep a safe system and minimize the potential for any hazards, protection circuitry is needed to be implemented within the integrated and external circuits. There are multiple functional circuits within the L5973AD and the Power Management Module which will be discussed below.

- Voltage Monitoring and Under Voltage Lockout

  Many power devices are designed to operate with low supply voltages, but they still need a certain minimum voltage to operate correctly. This is especially important in battery-powered applications, where the available voltage decreases as the battery discharges. When the supply voltage is too low, a number of things can happen, including:

  (a) The bandgap reference may generate the wrong voltage

  (b) Logic functions may generate the wrong control signals

  (c) Power transistors may be switched only partially on or partially off.

  An undervoltage lockout (UVLO) circuit makes sure that a device does nothing until the supply voltage is high enough for predictable behaviour, giving rise to robust system performance [4]. In the L5973AD, an internal block continuously senses the Vcc, Vref and Vbg shown below. If the voltages go higher than their thresholds, the regulator starts to work. Additionally, there is also a hysteresis on the Vcc for Under Voltage Lockout. This is shown in figure 4.1 below.

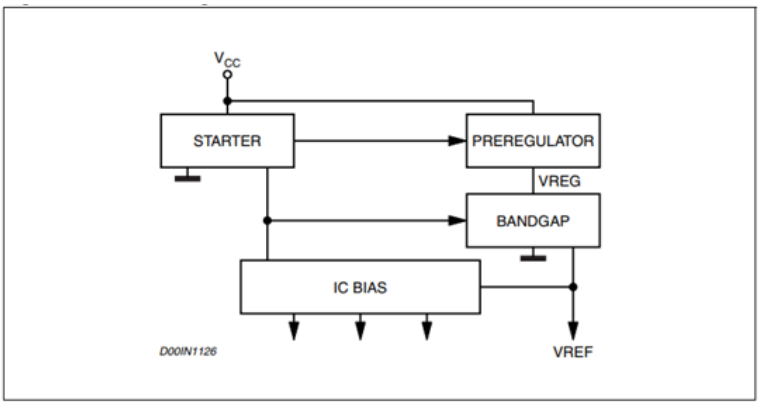


Figure 4.1: Block Diagram showing the UVLO function[4]

- Reverse Polarity Protection

  Reverse polarity protection is an internal circuit that ensures that the device is not damaged if the power supply polarity is reversed. The reverse polarity protection circuit cuts off power to the sensitive electronic circuits in the transmitter or transducer. It's an important feature, because significant power can be available in DC power supply systems in many industrial installations and besides permanent damage to the connected device there can be risk of fire if the device is not protected against reverse polarity [45].

  Schottky diodes are used for their low turn-on voltage, fast recovery time and low-loss energy at higher frequencies. These characteristics make Schottky diodes capable of rectifying a current

by facilitating a much quicker transition from conducting to blocking state [46] than that of a normal diode. Another advantage of the Schottky diode is that it's voltage drop is significantly lower than that of a normal diode.

- Current Protection

  Overcurrent in a circuit could lead to electrical fires, shock or electrocution [47]. Overcurrent protection is thus necessary for every electrical circuit. If a circuit does not have overcurrent protection, these severe repercussions would potentially be reached. As such, all electrical circuits and equipment should have overcurrent protection devices to interrupt and open circuits when overcurrent events occur. As a result of proper protection, one can significantly reduce the risk of damage and electrical hazards.

  The L5973AD has two current limit protections, pulse by pulse and frequency fold back. The schematic of the current limitation circuitry for the pulse by pulse protection is shown in figure 4.2 below. The output power PDMOS (P-channel Depletion Metal Oxide Semiconductor) transistor is split into two parallel PDMOS's. The smaller PDMOS has a resistor in series, Rsense. The current is sensed through Rsense and if the threshold is reached, the mirror is unbalanced and the PDMOS is switched off until the next falling edge of the internal clock pulse. Due to this reduction of the ON time, the output voltage decreases. Since the minimum switch ON time (necessary to avoid false overcurrent signal) is not enough to obtain a sufficiently low duty cycle at 500 kHz, the output current in strong overcurrent or short circuit conditions, could increase again. For this reason, the switching frequency is also reduced so as to keep the inductor current under its maximum threshold. The Frequency Shifter depends on the feedback voltage - as the feedback voltage decreases due to the reduced duty cycle, the switching frequency decreases too.

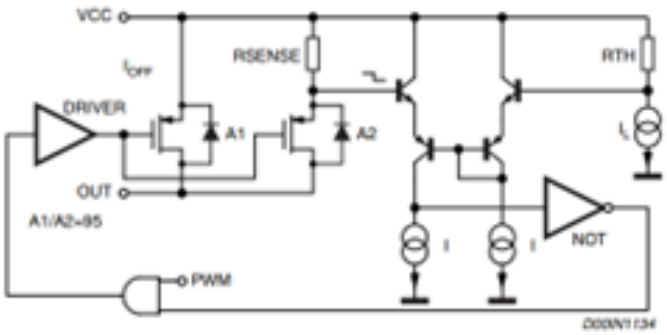


Figure 4.2: Current Limitation Circuitry of the L5973AD[5]

- Thermal Shutdown

  In the environment that this system is going to be used, the temperature can sometimes reach extremes of 46°C [48]. Such temperatures can cause serious damage to an electrical device. For instance, if a single lithium-ion battery overheats, it can quickly create an incident of thermal runaway occurring in batteries that are stored nearby. This domino effect will amplify the likelihood and impact of a fire or explosion, causing severe damage to the surrounding environment [49]. In this scenario, a fire or explosion could also potentially harm the surrounding

wildlife and cause great interference to the ecosystem.

The circuit in figure 4.3 below is the thermal shutdown circuitry within the L5973AD IC and is also the thermal shutdown circuitry used in the designed circuit. It employs a voltage comparator to monitor the difference between a temperature-independent reference voltage, VREF, and a voltage with a complementary-to-absolute-temperature variation, VNTC, generated by a bipolar transistor. The circuit is sized so that in normal operation the voltage VNTC is higher than VREF. When the die temperature increases the value of the VNTC voltage decreases, so that at a certain temperature it becomes smaller than VREF, causing the comparator to trip, thus turning on transistor T1. In turn, this enables a current source with two outputs: the first output injects a current into resistor R1, thus pushing up the voltage level at the node TSDOUT. Finally, the output voltage, Vout, goes high and this flag activates the circuitry within the system (not shown here) that changes the operating condition of the protected circuit [6].

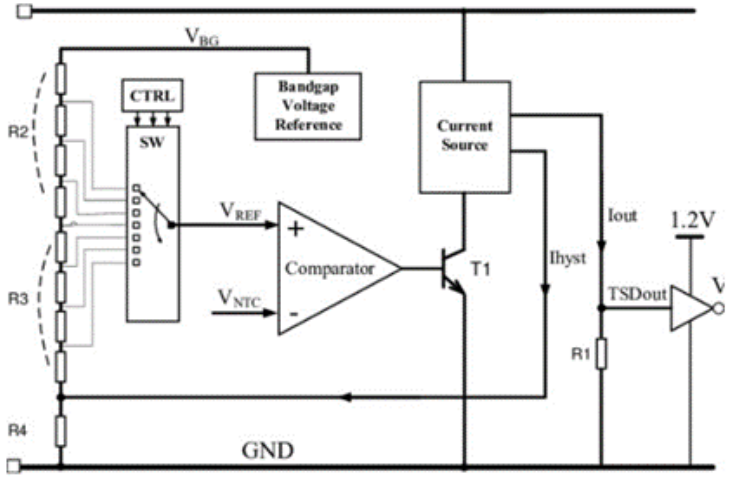


Figure 4.3: Thermal Shutdown Circuit[6]

In comparison, the Power Management Module also includes thermal shutdown at 150°C and thus there are no added features with respect to this function block.

4.3 Implementation

During implementation, it was initially decided to build the circuit using the internal circuitry of the L5973AD as a guide. This circuit was thus initially built on a breadboard as per the schematics described in this section, however it was not able to be completed due to the late arrival of components and time constraints because of this. Thus, the final design used was the Power Management Module and will therefore also be further discussed in this section.

4.3.1 Battery Life

A battery charging circuit was designed using the MCP73831 IC. A 3.7V Lithium-Ion battery was charged using this IC and was connected as per 4.4 below, according to the IC datasheet [7].

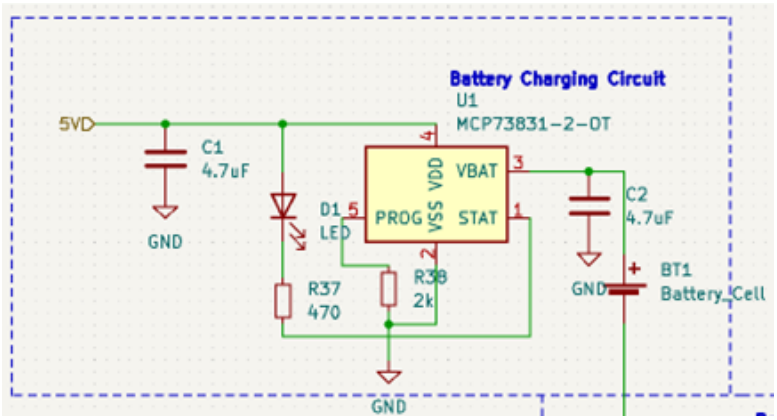


Figure 4.4: Battery Charging Circuit[7]

4.3.2 Protection Circuitry

- Reverse Polarity Protection

  Because of the advantages associated with the Schottky diode, it was chosen for reverse polarity protection and a 0Ω resistor placed was placed in series with it to short circuit the battery in case of failure. In the circuit, reverse polarity protection was achieved by adding a Schottky diode in series with the battery as shown in figure 4.5 below. The diode becomes forward biased when the polarity is correct and the load's normal operating current flows through the diode. When the battery is installed backwards, the diode reverse–biases and no current flows passed it.

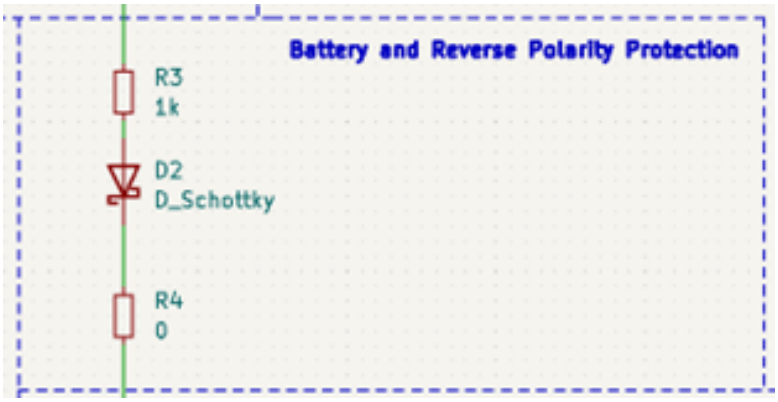


Figure 4.5: Schematic of Reverse Polarity Protection Circuit

- Under Voltage Lockout

  The circuit shown in figure 4.6 is the chosen UVLO circuit that was implemented.

The overall circuit is an operational amplifier used as a comparator to control the switching of a p-channel MOSFET. Resistors R1 and R2 are used to provide a divided input voltage to the negative terminal of op-amp. This voltage should remain higher than the voltage at the positive terminal of the op-amp when the battery input voltage is within acceptable levels. The voltage at the positive terminal of the op-amp drops at a slower rate than the voltage at the negative terminal of the op-amp as the battery discharges. This is due to the added zener diode voltage reference. When the voltage at the negative terminal of the op-amp drops lower than the voltage at the positive terminal, the output

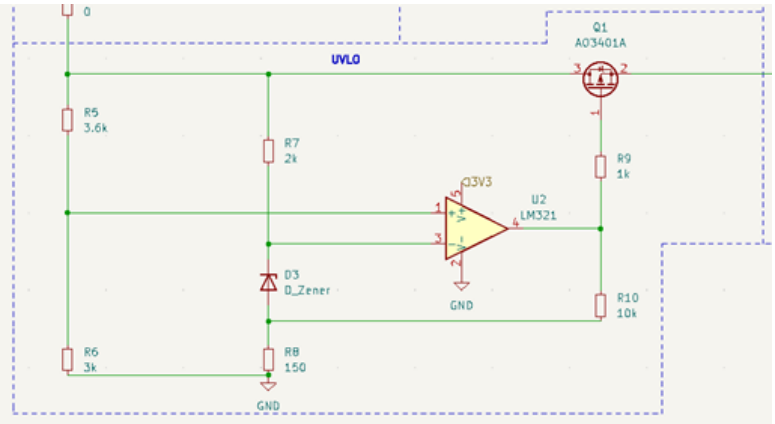


Figure 4.6: Schematic of Under Voltage Lockout Circuit

of the op-amp swings high and turns off the PMOS switch. The addition of resistor Rhys facilitates hysteresis in the switching operation. This works by injecting a small amount of feedback current into the node connected to the cathode of the zener diode when the battery voltage is low. This extra current lifts the voltage at the positive terminal of the op-amp making it such that the voltage at the negative terminal must increase to a higher level before reaching the critical crossover point and closing the PMOS switch.

4.3.3 Final Designs

The final circuit designed for the circuit is shown below in figure 4.7.

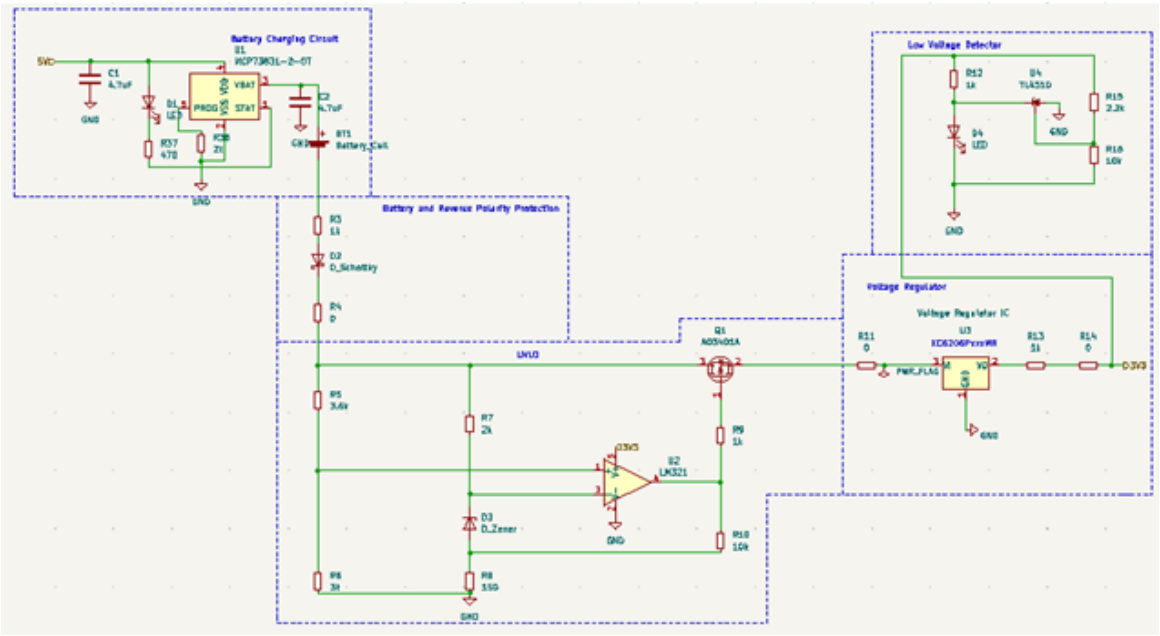


Figure 4.7: Schematic of the final designed circuit

As stated above, this circuit was unable to be implemented due to time constraints and the late arrival of components.

Thus, the Power Management Module was chosen to be the Power System and is shown below in figure 4.8 , along with the solar panel that can be used to charge it.

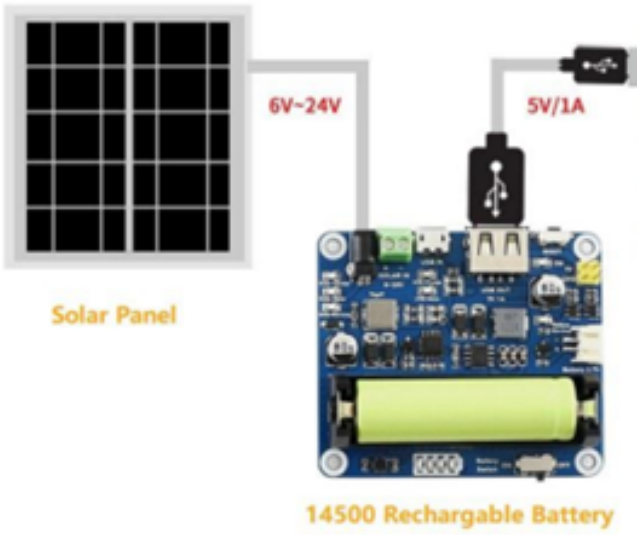


Figure 4.8: Power Management Module [8]

The schematic for the final Power Management Module can be found below in figure 4.9.

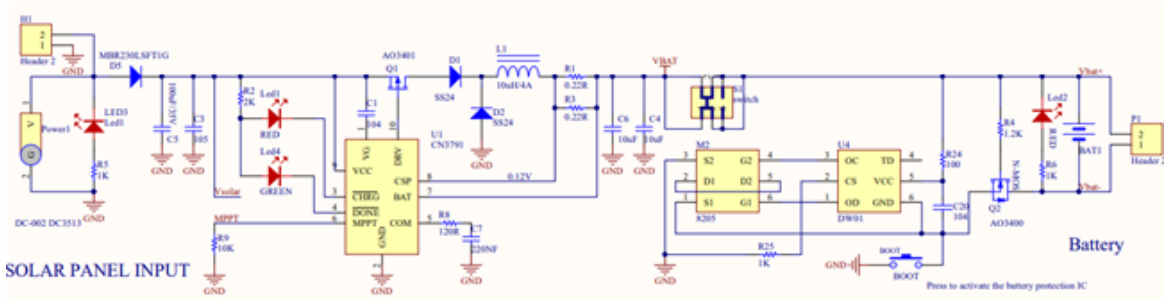


Figure 4.9: Schematic of Power Management Module [9]

4.4 Testing and Results

This section will describe the testing methods used to achieve the requirements briefly discussed in section 1.3, and will document the testing methods used to meet the specifications in the User Requirements, Specification and Acceptance Test Protocols table relating only to the Power subsystem.

The first user requirement pertaining to this subsystem is UR6 in table 1.1 and mandates that the system needs to be powered for a full day at a time. This was attained by the Power Management Module and the 3.7V Lithium Ion battery outputting a constant 5V. This was tested by measuring the output of the pinheads and the USB output of the board using a multimeter. It was found that this was a constant 5V as can be seen in figure 4.10, which is the voltage required to power the ESP32.

The above test is the UATP6.1 in table 1.1. For UATP6.2, another test was conducted to check that the 5V was constant for 12 hours. During testing, the power module without the solar panel was realised to run for approximately 8 hours, outputting a constant 5V which was checked every hour for these 8 consecutive hours using a multimeter. With the solar panel connected, there was a second test conducted and it was found that the system was able to be powered indefinitely so long as the solar panel was charging and still outputting a constant 5V to supply the ESP32.

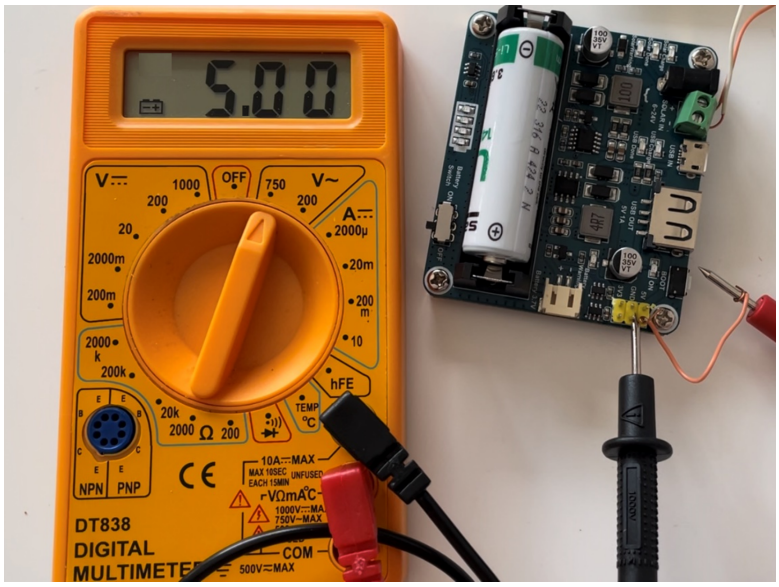


Figure 4.10: Multimeter reading of the output voltage of the Power Management Module

The second user requirement is UR4 and stipulates that the system needs to function seamlessly and safely in a harsh environment, which was implemented through protection circuitry. Below is a description on how the different circuits were tested to meet the requirement and specification US4.1.

- Reverse Polarity

  When the battery's polarity was reversed, an LED switches on on the module board indicating a warning signal that it needs to be reversed. This was then tested by reversing the polarity of the battery, and witnessing the indication of the LED as shown in figure 4.11 below. The LED is labelled 'Battery Warning'.

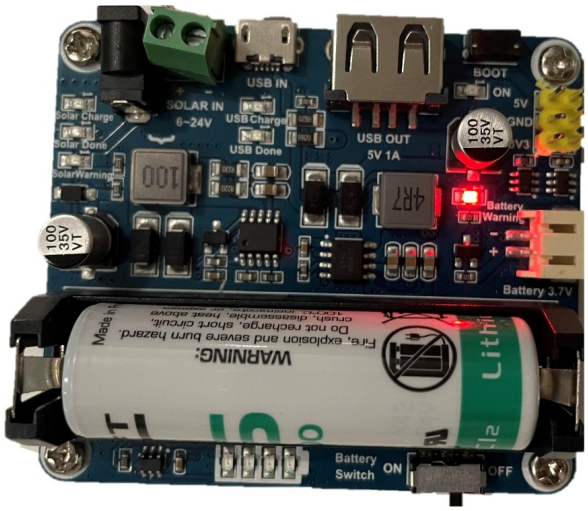


Figure 4.11: Image showing the Warning LED ON when there is reverse polarity

- Thermal shutdown

  The power module was placed in direct sunlight for 8 hours and there was no overheating of the

battery, therefore meeting the safety requirement.

- Under voltage lockout

  Voltages less than 5V were applied to the terminals of the module board in place of the battery using a power supply machine. During this test, it was found that the module only started to switch on and operate once the input voltage reached 5V, satisfying the under voltage lockout specification.

The final user requirement, UR7, requires the system to be re-usable on consecutive days. This was implemented by using the rechargeable battery and the solar panel.To test specification US7, a test was conducted to monitor if the battery was charging.

The Lithium-Ion battery is rechargeable either through a USB input cable on the Power Management Module or via a connection from the solar panel. When the solar panel is charging both itself and the battery, there are two separate LED's on the module board to indicate this as well. These LED's were found to be ON during charging. The charging of the battery was then tested as well through the USB input. It took approximately 1 hour for the battery to charge fully, after which an LED on the module board indicated that it was fully charged.

4.5 Conclusion

This chapter involved the comparison of two different circuits that were both designed to meet the user requirements and derived specifications.

Despite both circuits being designed to meet the user requirements in theory, only the Power Management Module was practically implemented and tested. This was because of the major advantages of the module that outweighed those of the designed circuit that used the L5973AD IC.

The Power Management Module was chosen as the power system due to its advantages of being able power the system indefinitely so long as the solar panel is charging itself, the thermal shutdown protection circuitry which is extremely beneficial considering the climate in the Kalahari, the system being able to be recharged through a USB input and the multiple added features of LED's indicating when the battery is fully charged, when the solar panel is charging, and when there are warnings in terms of temperature hazards and reverse polarity.

This subsystem is able to power the system solution for the period specified by Ben whilst being re-usable, portable and safe and was therefore able to meet all the user requirements outlined in section 1.3.

Chapter 5

Software and Communication

5.1 Introduction

Software and communication methods form essential parts of RDT (remote data transmission) designs. In the context of the weight measurement design being discussed, the data that the system aims to retrieve (e.g. drongo weight, temperature, etc.) gives important insights into how Kalaharian drongos can be protected under changing environmental conditions. It is imperative that the software and communication methods are concretely defined, detailed, and implemented, so that trends in drongo behaviour and health can be comprehensively analysed. This will give the ornithologists of the Fitzpatrick Institute of African Ornithology details that enable them to ensure the protection of the vulnerable drongo species.

This chapter examines the completed software and communication subsystem. Initially, the technical design choices made for the software and communication methods will be overviewed and justified. A detailed explanation of the subsystem's implementation will then be given. Subsequently, the tests performed on the subsystem, as well as the corresponding results, will be presented. These tests aim at validating the software and communication subsystem against the requirements given for the subsystem in Section 1.3.

5.2 Design

This section will discuss software and communication related design choices made throughout the life-cycle of the project before the implementation is presented. The microcontroller and development board selected will first be discussed along with the selected development environment. Subsequently, the communication method selection, the programmatic design, and the webserver design will be considered.

5.2.1 Microcontroller and Development Board

It should be made clear at the outset of this section that the ESP-32 TTGO board was selected for use in the final design. There were two readily available development boards at our disposal - the ESP-32 TTGO and the Raspberry Pi Zero W. The decision to choose the ESP-32 TTGO board over the Raspberry Pi was driven primarily by software considerations, hence the topic is being discussed in this chapter.

The Raspberry Pi was initially chosen for use, given its powerful abilities and its support of the Python

programming language. Python has libraries and functionality that prioritize simplicity and ease of use, making it appropriate for complex and multi-layered systems. For example, the data processing algorithm (cf. Chapter 6) could be implemented seamlessly with few import statements and the use of dynamic arrays. The benefits of the Pi do however come with drawbacks when compared to the ESP32, such as slower code execution due to higher level programs. A brief comparison of the Raspberry Pi Zero W and the ESP-32 TTGO boards is provided in Table 5.1 below. These comparisons are based on datasheets [50] and [51]:

	Raspberry Pi Zero W	ESP-32 TTGO
CPU Frequency	1GHz	240MHz
Memory (RAM)	512MB	16MB
Comm Modules	WiFi, Bluetooth	WiFi, Bluetooth
Max Comm Distance	100-250m	300m
Working Voltage	3-5V	3-5V
Working Current	160mA	67mA
Active Mode Power	800mW	335mW
Approximate Cost	R2000	R200

Table 5.1: Raspberry Pi Zero W vs. ESP-32 TTGO

Table 5.1 above shows that the Pi outperforms the TTGO in processing speed and memory capacity. The TTGO however outperforms the Pi in communication distance, power consumption, and cost (by 10 times!). Both development boards are appropriate for low power remote projects such as the one being considered in this paper, although each choice comes with its own strengths and weaknesses.

The Raspberry Pi setup initially went smoothly, although issues were encountered throughout the process. Flask was then selected as the micro-framework in which the Python webserver would be contained. However, the imported Flask modules threw errors when the webserver was being created. These errors reported that the Flask modules contained null bytes and characters, prompting the editing of certain configuration files. Despite editing these erroneous files as directed, wiping data on the Pi, and repetitively reinstalling modules, the errors remained unresolved.

After careful research and consideration of alternative Python frameworks such as Django, the decision to use the ESP32-TTGO instead of the Pi was taken. This was not disadvantageous due to aforementioned reasons and the fact that the TTGO is far more replaceable than the Pi, due to its cost. Other development boards were considered, but none stood out as being a uniquely good fit for the design. The time constraints for the project additionally prompted the use of the ESP-32 TTGO, as it was promptly accessible. After selecting the TTGO, a development environment was briefly researched and decided upon. This will be discussed in the next section.

5.2.2 Development Environment

Selecting the ESP-32 TTGO board limited the number of potential development environments, since only select IDEs and platforms can be used for ESP-32 embedded development. Two development routes were considered: PlatformIO and the ArduinoIDE. PlatformIO is a collaborative platform for embedded development with powerful tools such as a unified debugger and unit testing capabilities. It

allows for static code analysis, which gives insights into the status of the programmed embedded device in terms of RAM usage, on-board defects, and so on [52]. It can either be setup on the PlatformIO IDE, or it can be used as an extension from within Visual Studio Code. Both options provide a clean and effective user interface.

Due to its simplicity, the ArduinoIDE is often preferred to PlatformIO. For simple designs and beginner developers, the ArduinoIDE works effectively. The PlatformIO website itself states that the ArduinoIDE is best suited to simple applications, whereas PlatformIO is designed specifically to offer 'a more powerful and feature rich development environment' [52].

All being considered, the ArduinoIDE seemed more suited to the project than PlatformIO, despite PlatformIO's additional features. The simplicity of the ArduinoIDE and our prior experience with it made it the better choice. Furthermore, the software that defines the logic of the perch/scale design is relatively simple, and thus a single .ino script suffices for the project. In short, PlatformIO seemed unreasonable as its advanced features would bloat the project's software by adding unnecessary complexity.

5.2.3 Communication Method

When operating in AP (Access-Point) mode, the ESP-32 TTGO offers connectivity through both Bluetooth and Wi-Fi modules. BLE (Bluetooth Low Energy) is the standard used by Bluetooth on the TTGO, enabling it to transmit data while consuming minimal power. The Bluetooth module on the TTGO can function as a class 1, class 2, or class 3 emitter, indicating that it can transmit data at different power levels for different communication ranges (approximate ranges of 100m, 10m, and 1m respectively). The Bluetooth module can also act as a receiver, with the ability to sense signals as weak as -97dBm [51]. These features allow the TTGO to communicate effectively with an external device.

The Wi-Fi module on the TTGO is effective, simple, and somewhat secure. It communicates upto a maximum distance of 300m, using the unlicensed 2.4GHz spectrum with the 802.11b protocol [51]. Although power consumption is higher for the Wi-Fi module than for the Bluetooth module, the Wi-Fi's signal transmission power levels offer greater reliability for remote connections. The Wi-Fi module additionally demands user-authentication via password provision. In other words, if a user wants to join the ESP-32's Wi-Fi network, they must provide a password, defined in the source code. This does not safegaurd the system from all forms of information leaks, but it does provide an elementary level of security which is beneficial.

The Wi-Fi module was selected for use in the design due to its implementation simplicity, its long range and reliable communication abilities, and its security features. Due to time constraints, the Bluetooth module was not tested - this would be potentially helpful to experiment with in future design iterations.

5.2.4 Programmatic Design and Coordination

The programmatic design of the software subsystem was split into the following sections: webserver handling, sensor handling, Wi-Fi/communication, and data processing. This is illustrated in Figure 5.1.

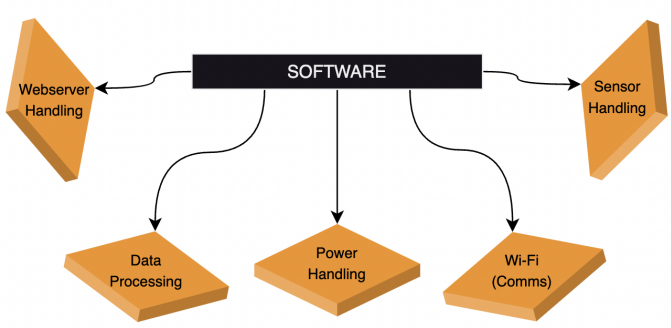


Figure 5.1: Diagram of Software Subsections

The data processing section warranted its own subsystem in the broader context of the design, due to its importance and complexity (cf. Chapter 6). Each of the other sections will be further outlined in the rest of this chapter.

The most difficult problem that was encountered during the design of the software subsystem relates to timely coordination between the subsystem's sections (illustrated in Figure 5.1). The coordination between these sections depends heavily on the methods used to handle events on time. Two methods exist for the handling of events: interrupts and polling.

Initially, it was thought that interrupts could be used to resolve timing issues related to web-server handling, data processing, and power handling (in terms of the ESP-32's different power modes). Interrupts trigger the immediate execution of code when a specified event occurs. These present an alternative to polling, in which checks are iteratively made to see whether a certain condition has been met. This means that polling can cause system lags or timing issues. Where interrupts can be used, they are generally preferred to polling methods, although some have argued that polling is better for many applications [53].

In terms of the design being discussed, interrupts would ideally be set to trigger execution of the data processing block as soon as a weight increase is sensed on the load cell. They would additionally be set to handle the web server whenever a client makes a request, while handling the power mode of the TTGO whenever the load cell transitions from an active to an inactive state (and vice versa).

Researching the use of interrupts in the ArduinoIDE revealed that interrupts, to be triggered, require either a pre-defined waiting period or a change in voltage on a particular pin. For example, to interrupt to a code block to handle a client's request from the webserver would require either knowing when the user would want to make the request, or sensing voltage changes on the TTGO's pins that indicate an incoming request. Due to the complexity of these factors along with the time constraints on the design, interrupts were not used in the software design as polling was found to suffice.

5.2.5 Webserver UI Design

The webserver acts as the user interface (UI) for the entire system. It is therefore essential that its design is intuitive and seamless in terms of its logic, navigation, and presentation. The UI was designed to conform as much as possible to Nielsen's 10 famous usability heuristics of mobile design and development (cf. [54]).

The main components needed on the web server UI were a home page and a results page. The home page would be used to show the mode of the ESP-32, to indicate whether it is actively processing data due to a bird's presence on the perch or not. The results page would be used to show the weight of the bird and the corresponding temperature and timestamp after the data processing algorithm had extracted these variables. This functionality is demonstrated with rough sketches and transitions as shown below:

Figure 5.2: UI Design Sketches

As illustrated above, the home page can redirect to the results page, and vice versa, upon a button press. The home page will show the mode of the ESP-32 and disable access to the results page (by removing the redirect button) if the ESP-32 transitions from sleep mode to processing mode. If the results page was open when processing mode was entered, the results would remain accessible (no redirection would occur). Once the new dataset has been processed, the home page will return to its original state, enabling access to the results page and indicating sleep mode. The results page will then have a row of data added to the table, collected from the hardware and sent via Wi-Fi to the UI. Up to 10 results will be tabularly shown on the results page, before the first row of data added to the table is removed. This means the results table will only include up to the 10 most recently collected datasets.

Although other pages could have been added to provide more functionality to the UI, simplicity and ease of use were prioritized. An additional page was considered which would provide graphs of weight and temperature versus time, but the graphing tools considered require an active internet connection to display the results, which is clearly not appropriate for the Kalahari.

Reloading pages for updates and new information is necessary for updated information to be made available on the UI. This is due to the functionality of the 'server.handleClient()' function that is used

in the ArduinoIDE to process client requests. Dynamic webserver updates were considered as there are libraries which support this in the ArduinoIDE. However, the functionality provided by the webserver and the data processing algorithm was prioritized in the design. A dynamic interface was thus not implemented although it would be beneficial.

5.3 Implementation

This section will discuss the implementation of the software and communication subsystem, with a specific focus given to the subsections shown in Figure 5.1. Attention will also be given to the webserver UI, following on from section 5.2.5 above.

5.3.1 Overall Subsystem

The flowchart in Figure 5.3 below broadly overviews the entire functionality of the implemented software and communication subsystem:

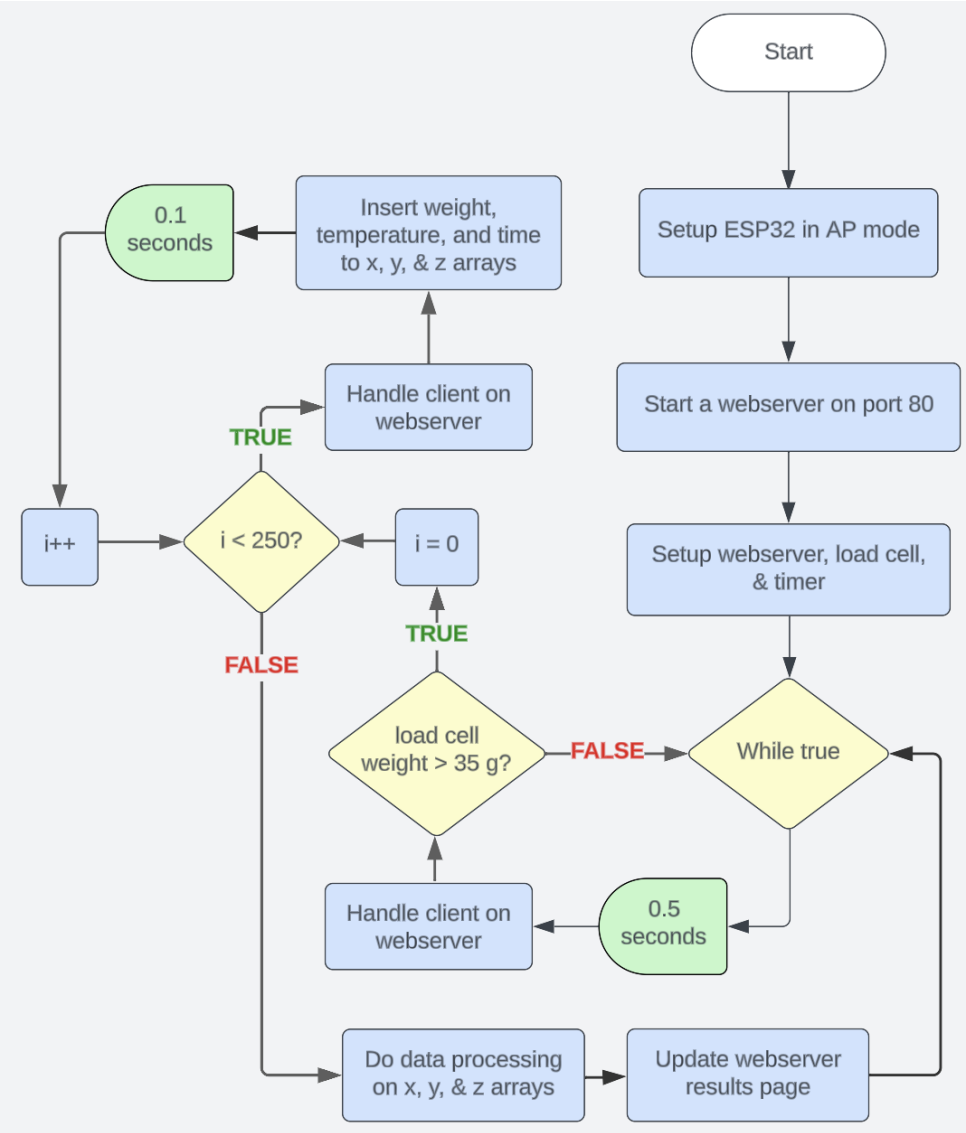


Figure 5.3: Software Flow Chart

The setup of the the ESP32's Wi-Fi module enables it to act as an access point (AP). The webserver, sensors, and timer are then initialized. Within the loop() function (the while loop in Figure 5.3), the weight on the load cell is read. If the weight is above 35g - considered as the minimum drongo trigger weight - 250 iterations are taken at a frequency of 10Hz (10 readings are taken per second). This means that 250 readings are added to arrays which are passed to the data processing function (cf. Chapter 6).

5.3.2 Wi-Fi (Communication) and Webserver

The communication (Wi-Fi) and webserver subsections are implemented at program startup as follows:

```
#include <WiFi.h>
#include <WebServer.h>
// SSID & Password for AP communication
const char* ssid = "ESP32";
const char* password = "12345678";
// Define IP Address details
IPAddress local_ip(192,168,1,1);
IPAddress gateway(192,168,1,1);
IPAddress subnet(255,255,255,0);
WebServer server(80); // Start webserver on port 80
```

Listing 5.1: Wi-Fi and Webserver Initialization

This code defines the name and the password of the network while setting up the network's IP addresses. The WebServer object then starts a server on port 80, which can be accessed at the address defined by the 'local_ip' variable. The communication and webserver subsections are further developed in the setup() method:

```
setup() {
    // ...
    WiFi.softAP(ssid, password);
    WiFi.softAPConfig(local_ip, gateway, subnet);
    delay(100);
    server.on("/", handle_OnConnect);
    server.on("/results", get_results);
    server.onNotFound(handle_NotFound);
    server.begin();
    Serial.println("HTTP server started");
    // ...
}
```

Listing 5.2: Wi-Fi and Webserver Setup

The code above configures the ESP-32 in AP mode and defines the webserver states. Webserver

states refer to the home and results pages, accessible via unique URLs. It attaches methods (handle_onConnect, get_results, and handle_NotFound) to each URL, such that when a page is accessed, a linked method is called to render the appropriate HTML and CSS.

5.3.3 Sensor Handling and Data Processing

The sensor handling and data processing code is shown together below for conciseness. The sensor code configures the temperature sensor and the load cell, while the data processing code appends data to arrays and calls the data processing function on those arrays. This is abbreviated below:

```
#include "HX711.h"
HX711 scale; // Declare scale variable
// Arrays to hold values for webserver
float previous_temps[10]; float previous_weights[10]; char* timestamps[10];
// Arrays to hold values for data processing
float weight_readings[250]; float time_readings[250]; float temperature_readings[250];
void setup() {
    // ...
    for (int i = 0; i < 10; i++){ // Initialize arrays
        weight_readings[i] = 0; time_readings[i] = 0; temperature_readings[i] = 0;
    }
    scale.begin(27, 22); scale.set_scale(2350); scale.tare(); // Setup load cell
    // ...
}
void loop() {
  // ...
  if (round(scale.get_units()*10.0)/10.0 > 35){ // If the measured weight is over 35 grams
    for (int i = 0; i < 250; i++){ // Loop 250 times
        // ...
        float weight = round(scale.get_units()*10.0)/10.0; // Format weight in grams
        temperature = (analogRead(A0) * (3 / 1023.0) - 0.5)/(0.01*51); // Get temperature
        weight_readings[i] = weight; temperature_readings[i] = temperature; time_readings[i]
    ↪  = i*0.1;
        delay(100); // 100 ms delay
    }
    process_data(); // process data!
    // ...
  }
}
```

Listing 5.3: Sensor Handling and Data Processing Code

On each iteration of the inner loop, the analogRead() method was called on pin A0 to extract the reading from the connected temperature sensor. Each reading was added to the temperature_readings

array for later processing. The weight read from the load cell with the scale.get_units() method was similarly handled, as a new reading was appended to the weight_readings array every 0.1 seconds. Time readings were collected in the same fashion. At the end of the inner loop (once enough readings have been collected while the bird was on the scale), the process_data() method is called. This method is discussed in Chapter 6 and operates on the arrays populated in the loop() method.

5.3.4 User Interface

The user interface was implemented very similarly to how it was designed in section 5.2.5 (cf. Figure 5.2). The final design of the UI is shown in Figure 5.4 where a standard navigation sequence between the pages is presented.

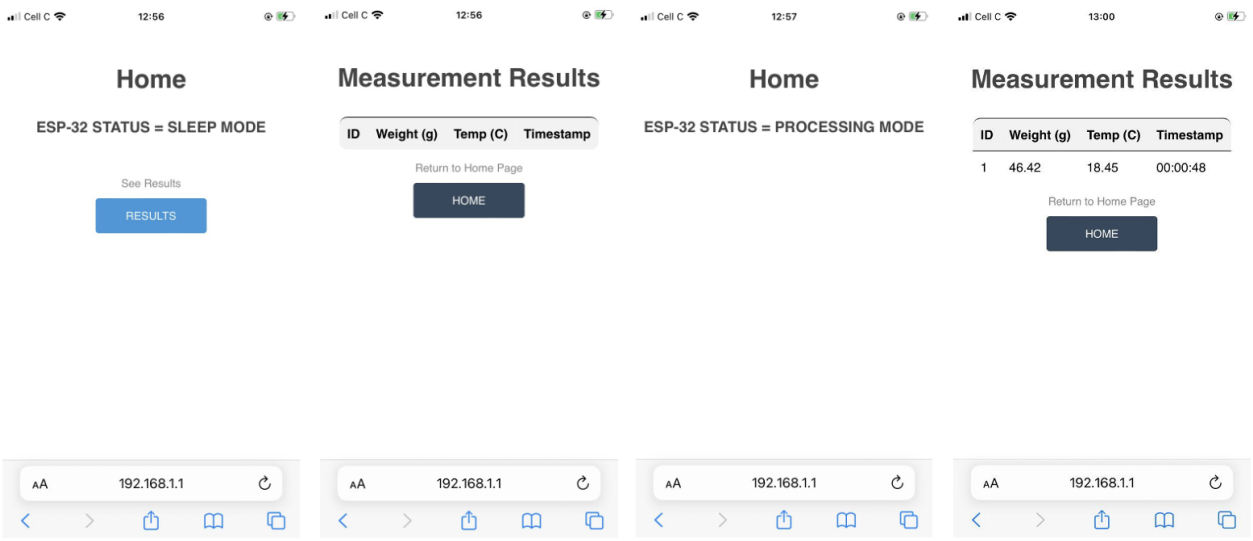


Figure 5.4: Webserver User Interface Implementation

Initially, the home page is accessed at the local IP address (discussed in section 5.3.2). The status of the ESP-32 is initially set to 'SLEEP MODE', merely indicating that it is not actively processing data (sleep mode in not in terms of power usage). The user can then access the results page by pressing the 'RESULTS' button. This page stores tabulated values of readings from the sensors, determined by the data processing algorithm. Initially this table will be empty, as demonstrated above.

The Home page will reveal that the ESP-32 has transitioned to 'PROCESSING MODE' when an object weighing over 35 grams is present on the load cell. Once 250 readings at a sample rate of 10Hz have been collected and processed, the home page will return to its original state - indicating sleep mode and enabling access to the results page. The results page will then display the collected weight, temperature, and timestamp. Note that the timestamp reveals the length of time that the ESP-32 has been powered for. The time of day was not provided as the timestamp, as this would require an internet connection.

The UI was made of HTML, and CSS for styling of the results table. The HTML attached redirect links to each button by citing the URLs to be navigated to upon the button presses. The different URLs were linked to specific functions (discussed in section

```
void handle_OnConnect() {server.send(200, "text/html", SendMainHTML());}
void get_results() {server.send(200, "text/html", SendResultsHTML());}
void handle_NotFound() {server.send(404, "text/plain", "Error: Page Not found");}
```

Listing 5.4: HTML/CSS Rendering Methods

These methods send status codes to the server while calling HTML/CSS rendering methods. These methods both return a single string of HTML/CSS to be rendered on the relevant page.

5.4 Testing and Results

The functionality of the software and communication subsystem was tested using the relevant Acceptable Test Procedures given in section 1.3. These tests along with their results are documented below.

5.4.1 Data Transmission Accuracy and Communication Distance

The first ATP pertaining to this subsystem (UATP1.3) aims at ensuring the accuracy of data transmission and presentation, while the second relates to communication distance (UATP8). These are taken from Table 1.1 and are given again below for ease of reference:

UATP1.3 - 'Examine the transmitted data to ensure the same value that was output by the processing algorithm was received by the researcher.'

UATP8 - 'Test that the user can access data sent from the ESP-32 from at least 10m away.'

These ATPs were tested together as this could be done with relative ease. At a distance of approximately 11m away from the system, a known weight was placed on the load cell, and the data was collected and processed from the sensors. The readings generated by the data processing algorithm (weight, temperature, and timestamp) were then printed out to the ArduinoIDE's serial monitor and transmitted to the webserver. If the values appeared on the webserver (meaning that the first ATP was satisfied), they would be compared to the values printed out to the serial monitor to ensure that the data accuracy was preserved throughout the transmission and presentation processes. This experiment was repeated three times to ensure that the printed data was able to accurately be transmitted to the webserver over a distance greater than 10m. In this way, both ATPs were carried out.

The following three figures show the results printed to the serial monitor for each of the three tests:

```
23:18:05.389 -> processing block entered
23:18:05.389 -> weight, temp, timestamp = 51.62, 18.74, 00:03:58
```

Figure 5.5: Test 1: Printed Output of Processing Algorithm

```
23:19:12.752 -> processing block entered
23:19:12.752 -> weight, temp, timestamp = 44.96, 18.70, 00:05:05
```

Figure 5.6: Test 2: Printed Output of Processing Algorithm

```
23:21:36.744 -> processing block entered
23:21:36.744 -> weight, temp, timestamp = 52.83, 18.72, 00:07:29
```

Figure 5.7: Test 3: Printed Output of Processing Algorithm

The following figure shows a screenshot of the webserver, taken from approximately 11m away from the ESP-32 and connected hardware, after all of the three tests had been performed. The webserver results are ordered from most recent to first, so ID = 1 corresponds to test 3 (Figure 5.7), ID = 2 corresponds to test 2 (Figure 5.6), and ID = 3 corresponds to test 3 (Figure 5.5).

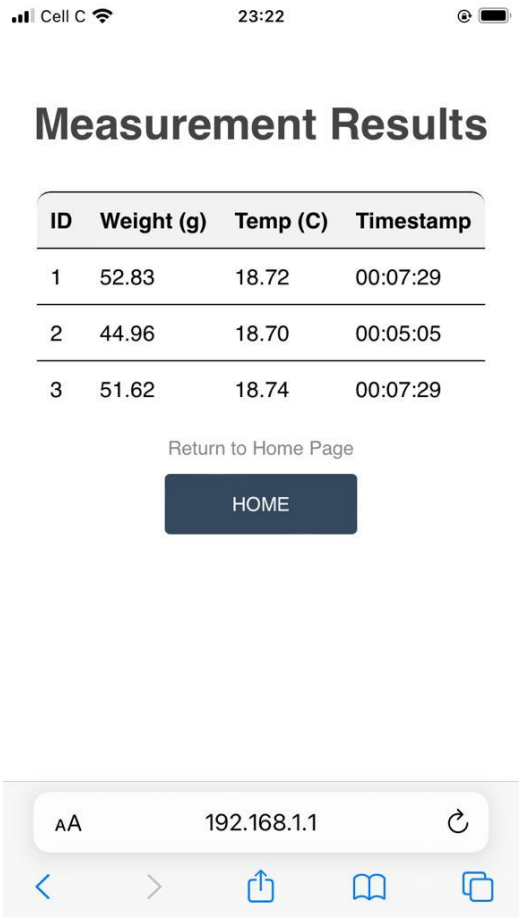


Figure 5.8: Data From Tests 1, 2, & 3 Received on Webserver From 11m

The datasets printed to the serial monitor in figures 5.5, 5.6, and 5.7 are replicated perfectly on the webserver UI shown in Figure 5.8. This means that both ATP1.3 and ATP8 have been satisfied by the software and communication subsystem, since the data is able to accurately be transmitted over a distance greater than 10m and displayed properly on the webserver.

5.4.2 Data Accuracy and Presentation

Functional requirements FATP1 and FATP3 were satisfied. Tests performed showed that the accuracy of the processed weight was within reason when the bird was on the scale for a minimal amount of time (approximately for 3 seconds). 5 gram accuracy was realized for these minimal time periods. Thus, FATP1 was satisfied.

FATP3 was satisfied as shown in section 5.4.1 above. The presentation of the webserver UI allowed for ease of navigation while providing a fully functional interface.

5.5 Conclusion

This chapter has comprehensively outlined the software and communication subsystem by discussing its 5 sections: webserver handling, data processing, power handling, communication (Wi-Fi), and sensor handling (cf. Figure 5.1). The design and implementation of the webserver UI were also detailed. Tests were conducted to evaluate the subsystem's performance on the basis of ATPs 1.3 and 8, and FATPs 1 and 3 that were defined in section 1.3.

The webserver handling section discussed how the user could interact with the entire system. The webserver needed to be accessible and functional so that data could be properly transmitted to the user's device from the ESP-32. The data processing section, examined in more detail in Chapter 6, discussed how data samples were iteratively collected from the sensors, so that singular weight and temperature values could be returned from the data processing algorithm. The sensor handling section discussed how the temperature sensor and the load cell were programmatically configured, while the power handling section was included so that the power modes of the ESP-32 could be transitioned between to lessen the system's overall energy consumption (although this was not implemented). Finally, the communication/Wi-Fi section explained how the Wi-Fi module on the ESP-32 was setup, enabling remote communication between a user's device and the TTGO.

The subsystem performed as expected, as demonstrated in section 5.4 above. Both ATP 1.3 and ATP 8 were satisfied by the final implementation. The significance of the software and communication subsystem in the overall design was evident, highlighting the essential roles that software and communication play in RDT (remote data transmission) designs.

Chapter 6

Data Storage and Processing

6.1 Introduction

This system submodule deals with the appropriate storage and meaningful processing of the acquired weight sensor data. This will aim to fulfill user and fucntional requirements UR1.3, FR2 and FR4 displayed in table 1.1 and 1.2 above.

To illustrate where these two aspects fit into the overall bird weighing process, the following diagram, shown in figure 6.1 below, has been constructed with an accompanying explanation.

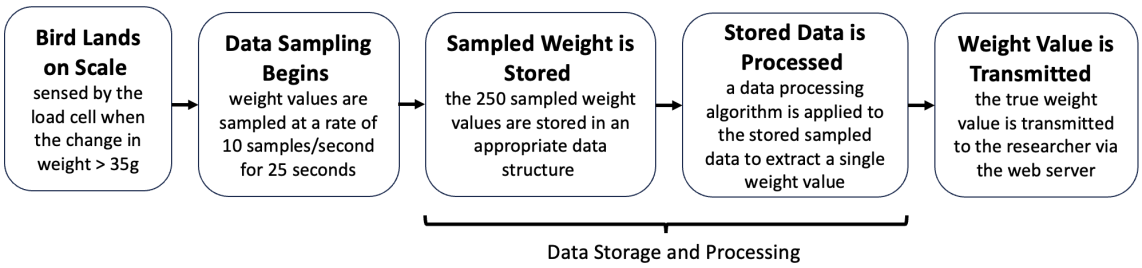


Figure 6.1: Bird Weighing Process

The data collection or sampling is triggered by a change in weight measurement detected by the load cell exceeding 35g. The sampling rate was chosen to be 10 samples/ second to ensure that even in the 'worst case' scenario or shortest time spent by the bird on the scale, there are sufficient samples. The forked-tailed Drongo's stay on the perch can vary anywhere from 3 to 40 seconds [55]. Therefore the worst case scenario of 3 seconds would produce 30 data samples - an adequate number of samples for meaningful processing. The data is sampled for 25 seconds as it was decided that this duration, which would produce 250 data samples, is sufficient to ascertain the true weight of the bird. The sampled data is saved in a data structure and passed from there into a processing algorithm to determine the true weight of the bird. This single value is then transmitted via a web server to the researcher.

Storage and processing will both execute on the ESP32 microcontroller, the chosen microcontroller for the project as explained in section 5.2.1 above and may be executed in a number of ways. This section examines the possible data storage and processing approaches to determine which is most suitable and realistic for this project's application.

6.2 Design

6.2.1 Data Storage

The role of data storage in this project's scope is minimal. This is because the current implementation is focused only on obtaining a single data weight value for each bird that occupies the perch. This value is currently read off a kitchen scale via binoculars used by the researcher who is standing sufficiently far away so as not to disturb the bird as discussed in section 1.4.1 above. This application does not monitor the fluctuating weight as the bird moves around (i.e. the sampled weight data).

As the emphasis of this proposed design focuses on data processing and transmission, the retention of the sampled bird data is less important than the single weight value produced by processing the sampled data. Thus it was decided that the sampled values will not be stored post-processing. The storage of them will be temporary and for the sake of the data processing algorithm. Once the processing is finished, and the final weight value has been obtained and transmitted to the researcher, the storage is disregarded. This means that if a different bird were to land on the perch, the array would be re-written with the new bird's sampled weight data.

The following approaches and implementations were considered for the project's data storage.

Microcontroller Storage

The ESP32 TTGO microcontroller has memory specifications of 4MB flash memory and a 520kB Static Random Access Memory (SRAM) [56], however only the flash memory would be used and thus a 4MB memory constraint is enforced by the microcontroller.

The ESP32 microcontroller would be coded in Arduino IDE in C++. There are many data structures offered in C++ [57] that could be used to store the weight data. Some of these being:

- Arrays: simple, sequential list of data items of the same variable type (i.e. char, int, float etc.).
- Vectors: a variable length array (i.e. same structure as an array but whose length can be adjusted during runtime). Similar to an array in that is stores a single variable type.
- Stacks: containers of objects which implement other forms of data structures (i.e. arrays, linked lists, dynamic arrays, vectors & deque). The manner in which they operate is that the most recently added item is removed first - Last in First Out (LIFO).
- Queues: containers of objects which implement other forms of data structures (i.e. arrays, linked lists, dynamic arrays, & heaps). The manner in which they operate is that the least recently added item is removed first - First in First Out (FIFO).
- Linked Lists: linear connection of nodes that implement a sequence. Each node holds a data item and a pointer to the next node in the sequence. They can be implemented in multiple ways (i.e. Singly-linked list, Doubly-linked list, Circular-linked list & using a sentinel or dummy node).

The lesser common C++ data storage structures are Graphs, Hash Tables, Tries and Heaps. Out of the common structures listed above, the most suitable for this project would be an array or vector. This is due to the fact that weight measurement values will be added sequentially, all will be of the

same variable type, namely float, and that arrays or vectors (which are variations of arrays) are easily traversed and manipulated which will aid in data processing. Although vectors would allow for the array length to be changed during run-time, making data processing easier, they are twice as slow as regular arrays [58] and thus for algorithm efficiency, arrays will be the chosen data structure.

The advantage of using this storage option would be that it simplifies the scope of the project as no external hardware is required, which in turn decreases the budget required because no additional costs are incurred. Having the data stored on a chip will also improve the speed of the system as no latency will be experienced by the system waiting for data to be fetched from external storage. The array will hold 250 weight samples of float variable type which are 4-bytes each in size. Therefore the storage space required will be a maximum of 1kB. This is well within the microcontroller's storage constraint of 4MB of flash memory.

The disadvantage is that a micro-controller has less storage capacity when compared to an external storage option. It would likely not be able to retain a bird's sampled weight data after processing and transmission of the bird's final calculated weight value. Thus, in the unlikely event of system malfunction, a bird's weight data could be lost and no back-up will be available.

External Storage

An external storage element in this case in the form of a microSD card would provide the project a greater storage capability. The microSD card's storage ability can range anywhere between 2GB to 128TB depending on which microSD card is chosen (microSD, microSDHC, microSDXC, microSDUC) [59]. The common choice of microSD card when interfacing with an ESP32 microcontroller is 16GB due to its mid-range but sufficient storage capacity and compatibility with the microSD module. This module is used to connect the microSD card to the ESP32 microcontroller. This configuration can be seen in figure 6.2 below.

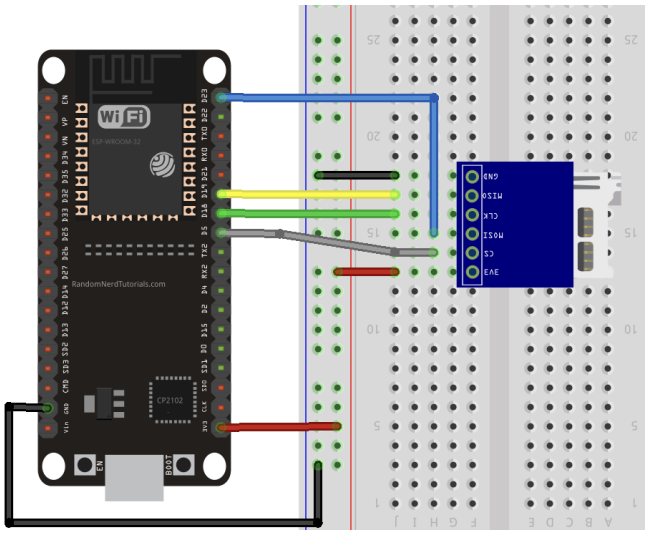


Figure 6.2: ESP32 Microcontroller and microSD Card Module Configuration [10]

The advantage of using this storage option is that a greater amount of data will be able to be stored and in theory this would allow for more meaningful data processing to occur and a more accurate

weight value to be obtained. This option would allow for the utilization of both the microcontroller's on-chip storage as well as an external storage. This larger storage will allow for the retention of sampled weight and final calculated weight for a longer period of time which provides a back-up or fail-safe in case of data loss or system malfunction. The ability to store the sampled data as well could widen the scope of the project as it might lend to further research benefit. This is because the sampled data will not just be used to calculate a final weight value but can also allude to the bird's movement and behaviour while on the scale.

The disadvantage would be the added complexity and cost in order to obtain the microSD card and its associated module, interface it with the system and incorporate it to the existing functionality. Another notable reason against this option is that the project's scope in terms of storage only requires a single weight data value which post-processing, takes up minimal storage space and thus this extended storage capacity would be wasteful.

6.2.2 Data Processing

Data processing plays a significant role in the project's overall proposed improvement as it determines the level of data accuracy. To improve data accuracy, sampling the bird's weight fluctuations whilst on the perch has been introduced to obtain multiple weight values. These values can be processed to obtain an accurate 'true' weight value of the bird. This is a substantial improvement on the current method of data collection which is by watching a kitchen scale via binocular's from a distance. This practice introduces inaccuracies due to a number of features such as human error, dependence on binoculars as well as the attention of the researcher needing to be split as they are not only observing the weight of the bird but other variables beneficial to their research (i.e. behaviour, gender, identity from a tag on the bird).

Possible data processing algorithms were examined in section 2.6 of the Literature Review and those will be further explored below. However, now that physical implementation is being considered, there are practical restraints that must be acknowledged. The use of the ESP32 microcontroller restricts the choice of algorithm because if they are too 'high level' or computationally complex then they cannot be used as this microcontroller lacks the computational capacity to execute them. Even if they can be executed, the complexity will drastically decrease the efficiency of the system as the execution time will be extended.

The following approaches and implementations were considered for the project's data processing

Machine Learning

Machine Learning (ML) is an Artificial Intelligence (AI) technique that 'teaches' computers to learn from experience [60]. An application of ML was highlighted in the Literature Review whereby D. Larios et al. implemented an Artificial Neural Network (ANN), which is a common ML model type, to integrate into their autonomous weighing system of animals. This shows the direct applicability of a ML approach to the weighing of animals.

In this project's application, the machine learning model would be taught to intake sampled weight data and determine the true weight value of the bird. It would be able to do so as it would have been

previously trained on input data samples and known true weight values. This is a form of supervised learning as it is trained on 'labelled' datasets that have a ground truth or known true weight values available. This would equip the model to accurately predict unknown true weight values given unseen input weight data samples.

The advantages of using this data processing method is that the data accuracy will likely be very high. This is due to the advanced nature of Machine Learning models and their ability to drastically improve program performance. The disadvantage, as emphasised above, is that such a computationally complex processing method cannot be implemented on a microcontroller due to its limited computing ability.

Kalman Filter

The Kalman Filter is a recursive state-space algorithm that is applied to noisy input values in an attempt to filter out noise and determine the best estimate for data's true value [11]. It has wide spread applications in navigational systems, object movement tracking and computer vision applications.

The operation of a Kalman Filter, as illustrated graphically in figure 6.3 below, is fairly complex but can be summarised as the iteration through input data and the application of two steps: prediction and updating.

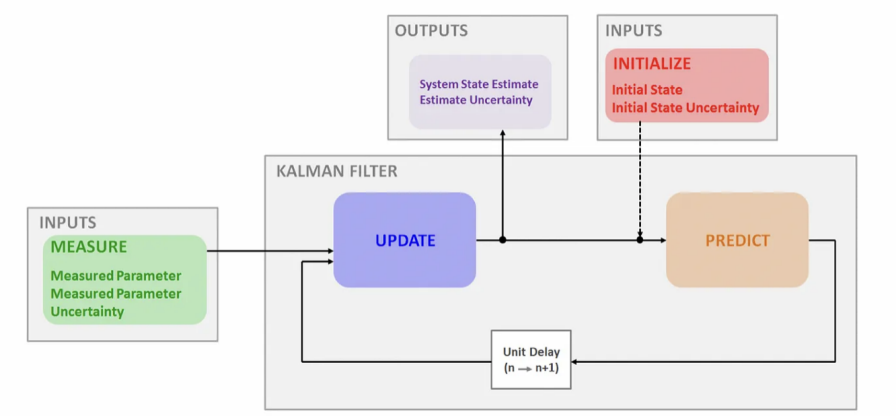


Figure 6.3: Kalman Filter Simplified Operation Diagram [11]

1. **Prediction**: the program has a constructed mathematical model that emulates the system's behaviour, with which it predicts the system's future state based on its previous states as well as inputs supplied to the system. A predicted uncertainty is produced along with the predicted state, as the filter expects there to be an error between its predicted state and the systems actual future state.

2. **Updating**: the system then compares the predicted state with the actual state and updates the estimate value. The updated estimate value takes into consideration and is a combination of both the predicted state and the actual state with an applied gain. This gain is derived from predicted uncertainty as well as a measured uncertainty - which refers to the uncertainty of the actual state having been measured incorrectly.

This process, as stated above, repeats iteratively through the system's operation. Constantly updating

the estimated value in an attempt to minimise the predicted uncertainty and obtain a more accurate predicted value which is not tarnished by noise.

The advantage of using this data processing method is that it is specifically designed to intake data samples with a certain level of uncertainty and produce a final value of adequate accuracy. This aligns perfectly with the scope of the project and could even be applied to the temperature data if it were sampled as well, to obtain a more accurate ambient temperature.

The disadvantage, however is that the above explanation does not appropriately highlight the complexity of the Kalman Filter, it is instead an incredibly simplified version. Generally, this filter deals with systems of multi-dimensional natures and thus would need to be greatly simplified for this application, which has the potential to diminish its effectiveness.

Tailored Algorithm

This would be a simple intuitive program tailored specifically to the project. It would accept as a parameter an array of sampled weight data and apply the following processing steps:

1. Each weight value would have the weight of the bait subtracted from it. Ben stated that they use two meal-worms as bait, which weigh approximately 0.125g each [61] and thus a value of 0.2g will be subtracted from each weight measurement.

2. A band pass filter will be applied to the array such that any values that fall out of an acceptable range will be disregarded. Ben stated that a drongo's weight is usually between 42g and 57g [55] but that the upper limit should be extended to 60g to accommodate the rare encounters of Drongos with a weight larger than 57g. The data is sampled for 25s regardless of how long the bird remains on the perch (ranges from 3s to 40s [55]) which will produce a fixed number of samples and thus if the bird were to depart before all samples were taken, then a weight of 0g would occur repeatedly in the array - the filter will rid the data of these effective null data points. Any weight lower than 42g would indicate that the bird is hovering above the perch and any weight above 60g would indicate the landing force of the bird arriving onto the perch - none of these values are relevant to the true weight of the bird and thus can be filtered out.

3. The mode, or most recurring weight measurement will then be obtained from the remaining data samples as the true weight of the bird. If more than one mode exists for a given dataset then the average of the modes will be taken and outputted as the true weight of the bird.

The advantage to this data processing approach is that it is customised to this project and thus prioritises this specific application. It is very intuitive and thus easily understandable by non-engineering professionals who can add to or modify it at any time. It will be of simple program complexity and thus will not be computationally exhaustive to implement and will execute quickly.

The disadvantages however are that its simplicity may at times produce slightly inaccurate measurements. This approach is reliant on the bird being on the perch long enough to take enough samples and 'still' enough to take stable readings, which cannot be assured. It has the potential to lack robust processing results.

6.3 Implementation

6.3.1 Data Storage

The data storage is a simply array of fixed length (250) that is populated with sampled bird weight data. This can be seen by the code in Listing 5.3 above in the Sensor Handling and Data Processing portion of section 5 above.

6.3.2 Data Processing

The two most promising or relevant options discussed in section 6.2.2 above would be the Kalman Filter or the Tailored Algorithm approach. Therefore both will be implemented and tested on sample data to determine which produces more accurate results. This implementation and testing will occur in Python. Even though the final data processing algorithm will need to be implemented in C++ for ESP32 microcontroller compatibility, Python will be used to create easily coded programs for ease of implementation and sake of comparison.

Tailored Algorithm

The implementation of the tailored algorithm, as described in section 6.2.2 above can be graphically explained by the flowchart in figure 6.4 below. This was implemented in Python for testing purposes and the Python code can be found in Appendix A.1 below or the project's Git Repository.

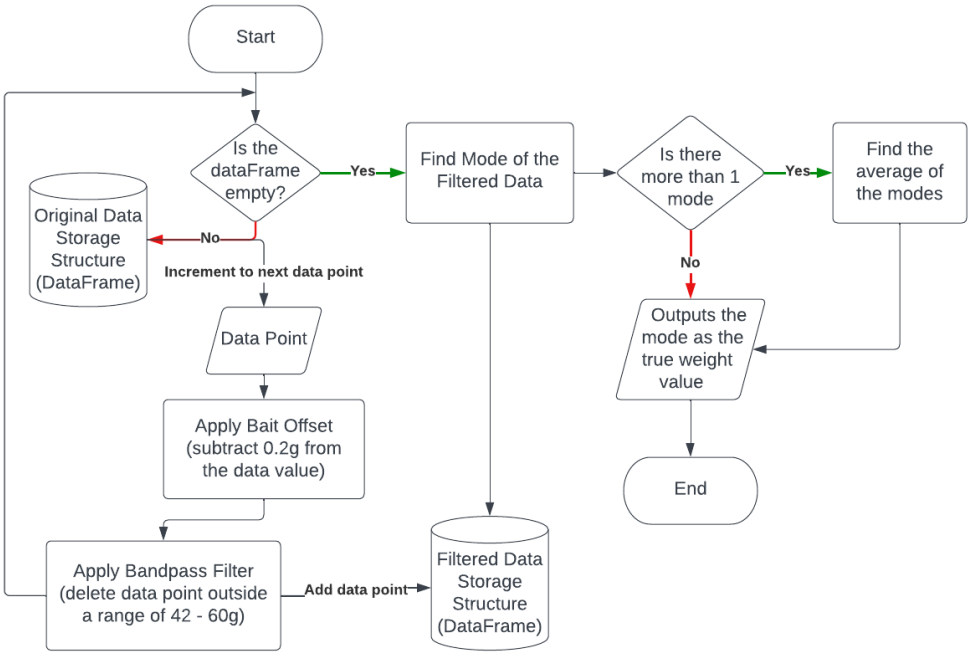


Figure 6.4: Flowchart showing the Operation of the Python Implementation of the Tailored Algorithm

Kalman Filter

The implementation of the Kalman filter, as described in section 6.2.2 above can graphically be explained by the flowchart in figure 6.5 below. This was implemented in Python for testing purposes

and the Python code can be found in Appendix A.2 below or the project's Git Repository.

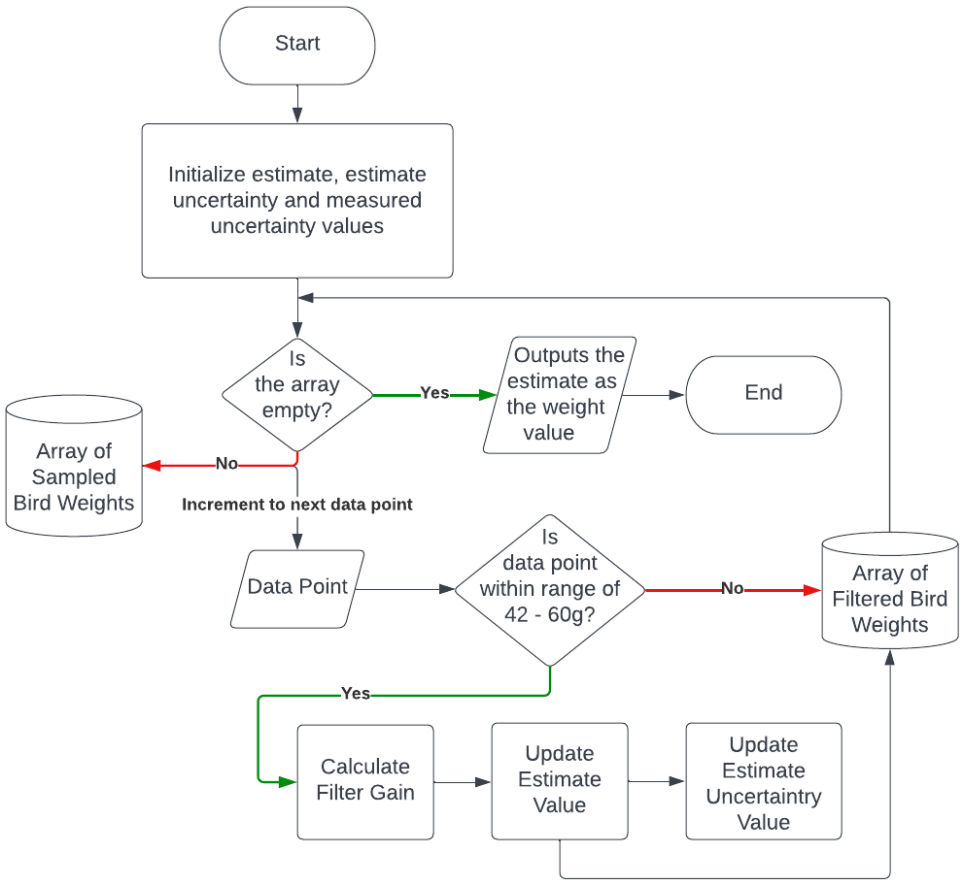


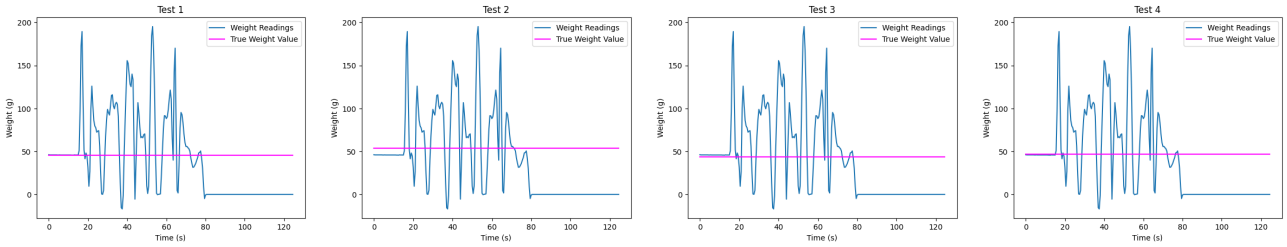
Figure 6.5: Flowchart showing the Operation of the Python Implementation of the Simplified Kalman Filter

6.4 Testing and Results

For testing purposes, an object of known weight was placed on the scale and the sampled values were stored in an array. This will allow for the two aforementioned data processing algorithms to be applied and produce their version of the 'true' weight value. These values can then be compared to the known weight to evaluate the algorithms' accuracies.

Four testing data sample arrays were obtained for different objects of known weight values made to move around the scale in similar manners, such that four experiments could be conducted. To emphasise meaningful testing, these weight values lie within the range of acceptable forked-tail Drongo weights (i.e. 42 - 57g [55]). The four test's data can be seen by the plots in Figure 6.6 below, where the fluctuating weight data samples are shown in blue and the object's true weight value is shown in pink.

Each array was passed through as a parameter into the Python Tailored Algorithm and Python Kalman Filter Algorithm, as shown in Appendix A below, separately and the output final weight value results are tabulated in table 6.1 below. The percentage error will be calculated, using equation 6.1 below for



(a) Test 1 Data (b) Test 2 Data (c) Test 3 Data (d) Test 4 Data

Figure 6.6: Testing Data showing Sampled Weight Data (blue) as well as True Weight Value (pink)

each program in order to test User Requirement 1.2's Acceptance Test Protocol (UATP1) which states that the error must be within a margin of error of 1.5%.

$$error(\%) = \frac{Known\,Weight\,Value - Predicted\,Weight\,Value}{Predicted\,Weight\,Value} \cdot 100 \tag{6.1}$$

Test Number	True Weight Value (g)	Tailored Algorithm Output Weight (g)	Kalman Filter Output Weight (g)	Tailored Algorithm Error (%)	Kalman Filter Error (%)
1	46	45,8	51,26	0,436	0,851
2	52	52	55,15	0	0
3	43	43,5	48,97	-1,149	-2,34
4	47	46,8	49,58	0,427	0,86

Table 6.1: Data Processing Test Results

From table 6.1 it is evident that, albeit through minimal testing, the tailored algorithm proved to be more accurate than the Kalman Filter. This could be due to many factors, as discussed above, the most likely being that this is too small an application for the Kalman Filter to show its true potential of value estimation and that the tailored algorithm has been written specifically for weight ranges and movements as included in the testing sampled data.

Having decided on a data processing algorithm, it now needs to be implemented in a programming language that is compatible to the ESP32 microcontroller and the chosen Integrated Development Environment (IDE) which is Arduino IDE. Therefore the tailored algorithm must be implemented in C++. This will change the algorithm slightly due to C++ not being as high level of a language as Python which makes certain aspects more exhaustive to implement (i.e. requiring more code), one of which is array handling. The C++ tailored algorithm code can be found in Appendix A.3 below or the project's Git Repository.

The C++ code was tested with the same sampled data as the Python algorithm. It produced the same results as its Python implementation. Therefore, the data processing algorithm was deemed to be successfully implemented.

6.5 Conclusion

This chapter examined multiple implementation choices for data storage and processing in this project's context.

The decision was to use data storage only during data processing and therefore only a maximum of 1kB worth of data would be required to be stored. This data did not need to be retained. Therefore, the storage capacity on the chosen ESP32 microcontroller of 4MB, was sufficient. The C++ data storage structure was chosen to be the array due to its simplicity and efficiency within a data processing context.

Data processing plays an integral part of this project's proposed improvement on the current implementation. A tailored algorithm was chosen as it was suited specifically to this project and its context and it is computationally inexpensive enough such that it does not slow the execution speed of the algorithm execution and is able to execute despite the microcontroller's limited computing capacity. During testing it achieved a percentage margin error of no larger than 1,149% which meets user requirement 1.2 (UR1.3) as detailed in table 1.1 above.

This sub-modules' implementation also meets functional requirements FR2 and FR4 as detailed in table 1.2 above in that the data structure can accommodate 1kB of data storage and when interfaced with the web server produces the correct weight value, as seen in section 5.4 above. This signifies that the data processing produces a value compatible with the communication submodule.

Chapter 7

Final System

The final completed system was implemented by connecting up the four aforementioned sub-modules namely: Hardware, Power, Software and Communication and Data Storage and Processing. These sub-modules were implemented and tested separately in the previous chapters in order to meet the user and functional requirements outlined in tables 1.1 and 1.2 above.

The only user requirement not addresses in the above subsystems would be UR10 as this involves the overall price of the project's implementation. This involves all sections, specifically Hardware and Power. The overall price came to be 1707 ZAR which is within the projec't budget of 2000 ZAR.

The following images in figure 7.1 below show the completed system and the following table 7.1 below indicates whether the system met the associated acceptance test protocols. This is indicative of a completed and functional system or at very least proof-of-concept.

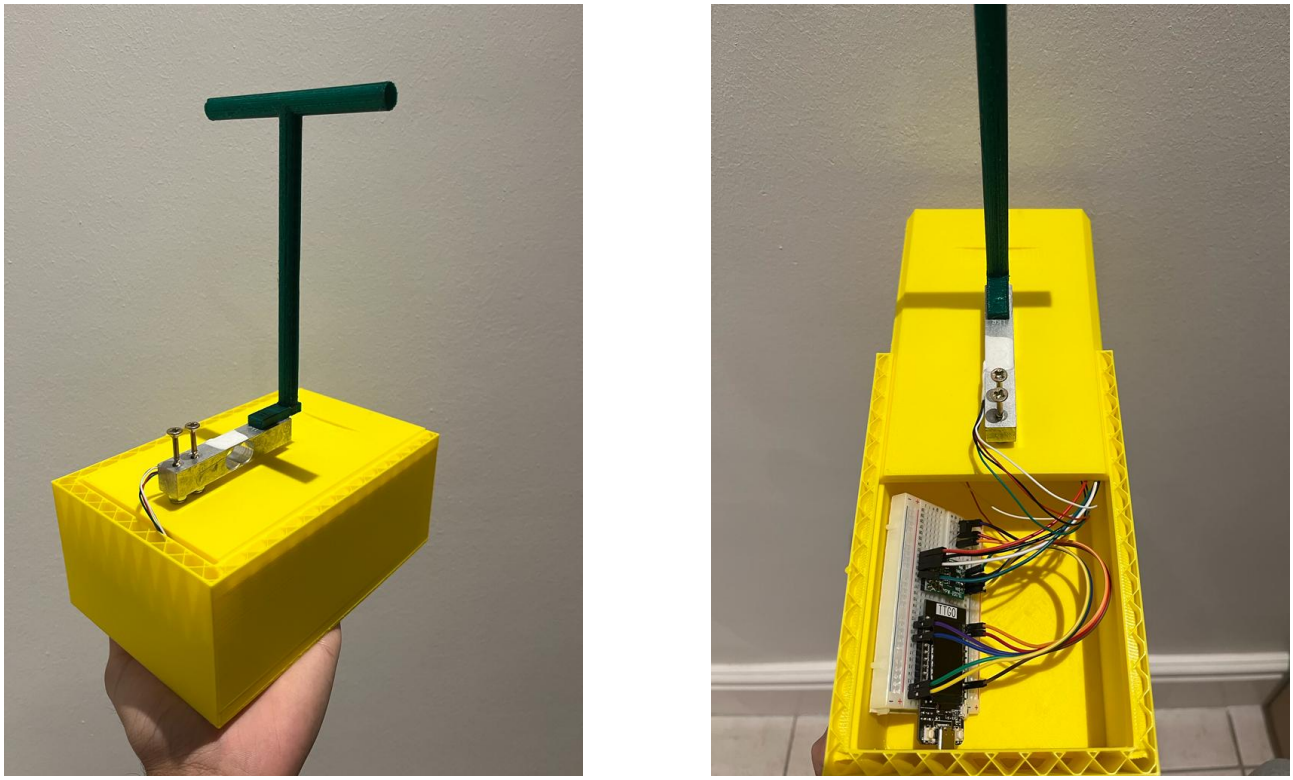


Figure 7.1: Final Proof of Concept System

ATP	Met or Not	Reason
UATP1.1	Met	The output value of the mass from the load cell was accurate to within 0.1g
UATP1.2	Met	The output of the data processing algorithm when applied to known weight sampled data produced results within 1.5% of the known weight
UATP1.3	Met	The output data from the data processing algorithm, printed to the serial monitor, matched the data shown on the webserver
UATP2	Met	When an object of greater than 60g was hung from the perch, it managed to support it without breaking.
UATP3	N/A	It was not possible to perform this test as it was not possible to access the Drongo's nor was this proof of concept system camouflaged.
UATP4.1	Met	When the battery is put into the holder with reverse polarity, a warning light comes on. When the input voltage is less than 5V, the system does not operate.
UATP4.2	Met	When dust or splashes of water was thrown at the housing, little to none made it inside.
UATP5	Met	System weighs less than the average laptop and is small enough to fit in a backpack.
UATP6.1	Met	The output of the power module was measured with an oscilloscope and found to be a constant 5V.
UATP6.2	Met	The output was checked consecutively ever hour to ensure the output was a constant 5V.
UATP7	Met	When the solar panel is charging the battery, an LED switches ON to indicate so.
UATP8	Met	The data sent from the ESP-32 was able to be displayed on the webserver when loaded from a distance of 11m
UATP9	Not	The temperature sensor was not able to be tested effectively as no equipment was present to find a known temperature of the environment.
UATP10	Met	The system cost 1707 ZAR to implement with is less than the ZAR 2000 allocated budget.
FATP1	Met	The accuracy of the data processing algorithm is accurate to within 5 grams when a bird is only present for 3 seconds on the perch
FATP2	Met	The ESP32 Flash mempry is 4MB which is greater than the 1kB minimum storage requirement
FATP3	Met	Presentation on the webserver is legible and includes the necessary information
FATP4	Met	The data processing output true weight value is the same as the one displayed on the web server. This indicates that interfacing between the two submodules works and results in no corruption of data

Table 7.1: Contributions Table

Chapter 8

Conclusions

This report has examined the design and implementation of a weight measurement/scale system, in order to propose a different approach and an improvement to the current systems used for drongo weight measurement in the Kalahari.

We began by introducing the background to the project and detailing requirements and ATPs that the design would need to meet in order to satisfy stakeholders. A survey of relevant literature was then conducted to overview current field methods for weighing birds, while highlighting the benefits and deficiencies of the same. The literature review also examined current approaches to the implementation of hardware, power, data processing, and software and communication subsystems.

The report progressed by analysing each of the subsystems in turn: hardware, power, software and communication, and finally data storage and processing. These chapters presented and justified design choices and corresponding implementations.

The solution we proposed consists of a perch on a load cell, connected to an ESP-32 microcontroller and an efficient and protected power system. Both the power system and the microcontroller would be housed in a protected container. A data processing algorithm was designed to extract an accurate weight reading from data samples collected when a bird is present on the perch. This algorithm was found to preserve simplicity while yielding high accuracies. The weight value would then be transmitted via the ESP-32's AP (Access Point) Wi-Fi network to a webserver that can be accessed remotely via a device's browser.

The system performed as intended and satisfied the stakeholders. The perch and hardware enclosure were prototype designs, but were found to be effective at protecting the circuitry and enabling the collection of weight measurements. The power module was effective and able to support the system for a day at a time. The data processing algorithm used a modal technique which was found to accurately record weight measurements to 0.1 grams. The transmission and reception of data via Wi-Fi and a webserver were found to be accurate and reliable for transmission distances of over 10 meters. The entire system cost less than R2000, which was the defined budget for the project.

In summary, the report achieved its goal, in that it proposed and outlined a Kalahari-appropriate drongo weight-measurement solution that improves measurement accuracy, ease of use, and cost-effectiveness. The system thus provides an improved study tool that will enhance research activities into the well-being of the fork-tailed Drongo population and potentially other avian species. We suggest that this design be viewed as a reference prototype that can be scaled to design similar systems that outperform existing ones.

Chapter 9

Recommendations

While the overall system functioned as intended by meeting all the user requirements and ATPs, there could have been more designing and implementations put into place had there been less of a time constraint. Thus, this chapter will briefly discuss the future recommendations that could be put into effect.

- The hardware i.e. the perch and scale, could be camouflaged to seem less obtrusive to the birds and would potentially allow them to be more comfortable, thus attracting more drongos to sit on the perch more frequently.
- Circuitry could be added to the power module to flash an LED when the battery life is getting low, and the battery life percentage could also be displayed on the web interface.
- With regards to the project's software design, interrupts could be readily incorporated to handle the power modes on the ESP-32 (cf. Chapter 4).
- The Bluetooth BLE module could be tested as a class 1, class 2, and class 3 emitter. The power performance analytics of these options could be compared to the power performance of the Wi-Fi module.
- A 'Graphs' page could be added to the webserver's UI so that plots of weight and temperature vs time could be rendered and stored on the webserver.
- With more time and field testing, a more complex and applicable data processing algorithm described in section 6.2.2 could be implemented to calculate the weight of the bird more accurately.

It is believed that the above recommendations would assist in making the current system more efficient, accurate, and valuable to the research done by Ben and Sam. Thus if there had been more time or if there is a possibility for the system to be further experimented with and improved, these recommendations should be implemented to enhance the current system.

Bibliography

[1] A. Poole and J. Shoukimas, "A scale for weighing birds at habitual perches," *Journal of Field Ornithology*, vol. 53, no. 4, pp. 409–414, 1982. [Online]. Available: http://www.jstor.org/stable/4512767

[2] D. Larios, C. Rodríguez, J. Barbancho, M. Baena, M. Leal, J. Marín, C. León, and J. Bustamante, "An automatic weighting system for wild animals based in an artificial neural network: How to weigh wild animals without causing stress," *Sensors (Basel)*, vol. 13, no. 3, pp. 2862–2883, February 2013.

[3]

[4] [Online]. Available: https://www.ti.com/lit/an/slva769a/slva769a.pdf

[5] [Online]. Available: https://export.farnell.com/stmicroelectronics/l5973ad/ic-buck-reg-smd-5973-hsop8/dp/1077143?gclid=Cj0KCQjwxYOiBhC9ARIsANiEIfaDiGQgju4JNh8NCvjNd-Ouf5qmD9ubxz2rCj4rXzPeq6wBDDYbt1IaAtCmEwcB&mckv=s_dc%7Cpcrid%7C653915408603%7Ckword%7Cl5973ad%7Cmatch%7Cp%7Cplid%7C%7Cslid%7C%7Cproduct%7C%7Cpgrid%7C146008773565%7Cptaid%7Ckwd-11598475781%7C&CMP=KNC-GEXP-GEN-SKU-MDC-Semiconductors

[6] [Online]. Available: https://www.researchgate.net/publication/314127933_Design_Options_for_Thermal_Shutdown_Circuitry_with_Hysteresis_Width_Independent_on_the_Activation_Temperature

[7] [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/MCP73831-Family-Data-Sheet-DS20001984H.pdf

[8] [Online]. Available: https://www.waveshare.com/w/upload/6/6c/Solar_Power_Manager_user_manual_en.pdf

[9] [Online]. Available: https://www.waveshare.com/w/upload/d/d2/Solar_Power_Manager_Schematic.pdf

[10] I. Beaver, J. Baars, J. Loveman, S. Santos, Rose, D. Evans, B. Navarro, E. B., A. Brody, C. Seysen, and et al., "Esp32: Guide for microsd card module arduino," Sep 2022. [Online]. Available: https://randomnerdtutorials.com/esp32-microsd-card-arduino/#:~:text=To%20interface%20the%20microSD%20card,programmed%20using%20the%20Arduino%20core.

[11] R. Khandelwal, "An easy explanation of kalman filter," Mar 2022. [Online]. Available: https://arshren.medium.com/an-easy-explanation-of-kalman-filter-ec2ccb759c46

[12] M. M. Nice, "The biological significance of bird weights," *Bird-Banding: A Journal of Ornithological Investigation*, vol. 9, no. 1, p. 1, 1938.

[13] G. A. Clark, "Body weights of birds: A review," *The Condor*, vol. 81, no. 2, p. 193, 1979.

[14] [Online]. Available: http://www.fitzpatrick.uct.ac.za/fitz/research/programmes/understanding/pied_babbler&fork-tailed_drongo

[15] R. A. Norberg, "Temporary weight decrease in breeding birds may result in more fledged young," *The American Naturalist*, vol. 118, no. 6, p. 838–850, 1981.

[16] G. Kaur and T. Kaur Kler, "Breeding biology of black drongo (dicrurus macrocercus) in relation to nest-site selection and habitat characterization in agricultural landscape," *Journal of Entomology and Zoology Studies*, vol. 6, no. 5, Jul 2018.

[17] R. Olinger, 2017.

[18] H. Jackson, "What is a transcription error amp; who does it better - humans or ai?" Jan 2022. [Online]. Available: https://www.rev.com/blog/resources/what-is-a-transcription-error

[19] "Climate of the kalahari desert," accessed: March 17, 2023. [Online]. Available: https://www.britannica.com/place/Kalahari-Desert/Climate

[20] T. Szép, Z. Barta, Z. Tóth, and Z. Sóvári, "Use of an electronic balance with bank swallow nests: a new field technique," *Journal of Field Ornithology*, vol. 66, no. 1, pp. 1–11, 1995.

[21] M. Emmet, "Hornbill nests," Available: https://southafrica.co.za/hornbill-nests.html, accessed: March 17, 2023.

[22] K. Reid, G. M. Liddle, P. A. Prince, and J. P. Croxall, "Measurement of chick provisioning in antarctic prions pachyptila desolata using an automated weighing system," *Journal of Avian Biology*, vol. 30, no. 2, p. 127, 1999.

[23] R. Maxwell, "Best off-grid power systems for your cabin," Aug 2022, accessed: March 17, 2023. [Online]. Available: https://www.familyhandyman.com/article/off-grid-power-systems-solar-wind-micro-hydro/

[24] "What is the environmental impact of a battery?" accessed: March 17, 2023. [Online]. Available: https://greenly.earth/en-us/blog/ecology-news/carbon-footprint-battery

[25] I. Al-Bahadly, "Portable multi-inputs renewable energy system for small scale remote application," *Journal of Power and Energy Engineering*, vol. 06, no. 02, pp. 59–73, 2018.

[26] P. N. A. T. Chaithra, "Smart farm monitoring system using solar-power and internet of things devices," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 02, no. 07, 2022.

[27] "Battery recycling," accessed: March 17, 2023. [Online]. Available: https://bhgpower.co.za/battery-recycling/

[28] W. Bouten, E. W. Baaij, J. Shamoun-Baranes, and K. C. Camphuysen, "A flexible gps tracking system for studying bird behaviour at multiple scales," *Journal of Ornithology*, vol. 154, no. 2, pp. 571–580, 2013.

[29] "Over half world's population now using mobile internet," accessed: March 17, 2023. [Online]. Available: https://www.gsma.com/newsroom/press-release/over-half-worlds-population-now-using-mobile-internet/

[30] "2030 agenda for sustainable development," accessed: March 17, 2023. [Online]. Available: https://www.itu.int/en/ITU-D/Statistics/Pages/intlcoop/sdgs/default.aspx

[31] "5g from space - the role of satellites in 5g," accessed: March 17, 2023. [Online]. Available: https://www.nokia.com/thought-leadership/articles/5g-space-satellites/#:~:text=For%20years%2C%20satellite%20communication%20has,in%20remote%20and%20rural%20areas.

[32] L. Holmstrom and T. Beckham, "Technologies for capturing and analysing animal health data in near real time," *Revue Scientifique et Technique de l'OIE*, vol. 36, no. 2, p. 525–538, 2017.

[33] M. Zubair, A. Ghubaish, D. Unal, A. Al-Ali, T. Reimann, G. Alinier, M. Hammoudeh, and J. Qadir, "Secure bluetooth communication in smart healthcare systems: A novel community dataset and intrusion detection system," *Sensors*, vol. 22, no. 21, p. 8280, 2022.

[34] M. Collotta, G. Pau, T. Talty, and O. Tonguz, "Bluetooth 5: A concrete step forward toward the iot," *IEEE Communications Magazine*, vol. 56, 07 2018.

[35] G. Lawton, "What is near-field communication (nfc)?" Mar 2022, accessed: March 17, 2023. [Online]. Available: https://www.techtarget.com/searchmobilecomputing/definition/Near-Field-Communication

[36] Shawn, "Introducing pn532 nfc rfid with arduino guide," Dec 2019, accessed: March 17, 2023. [Online]. Available: https://www.seeedstudio.com/blog/2019/12/09/introducing-pn532-nfc-rfid-with-arduino-guide/#:~:text=PN532%20is%20an%20NFC%20RFID,pairing%20with%20your%20Arduino%20projects!

[37] E. Nadimi, H. Sø gaard, T. Bak, and F. Oudshoorn, "Zigbee-based wireless sensor networks for monitoring animal presence and pasture time in a strip of new grass," *Computers and Electronics in Agriculture*, vol. 61, pp. 79–87, 05 2008.

[38] R. Handcock, D. Swain, G. Bishop-Hurley, K. Patison, T. Wark, P. Valencia, P. Corke, and C. O'Neill, "Monitoring animal behaviour and environmental interactions using wireless sensor networks, gps collars and satellite remote sensing," *Sensors*, vol. 9, no. 5, p. 3586–3603, 2009.

[39] N. Luwes and P. Hertzog, "Weight determination of moving objects, in a conveyor system with the aid of digital signal processing (dsp) techniques," *Interim: Interdisciplinary Journal*, vol. 4, pp. p. 102–112, 01 2005.

[40] Communica, "Hkd electronic load cell 1kg," Available: https://www.communica.co.za/products/hkd-electronic-load-cell-1kg, accessed: May 20, 2023.

[41] Admin, "Power supply for esp32 with boost converter amp; battery charger," Oct 2022. [Online]. Available: https://how2electronics.com/power-supply-for-esp32-with-boost-converter-battery-charger/#:~:text=or%209V%20Battery.-,ESP32%20Power%20Requirement,pin%20(External%20Supply%20Pin)

[42] A. Lerma, "Lithium-ion vs lead acid battery life." [Online]. Available: https://www.fluxpower.com/blog/lithium-ion-vs.-lead-acid-battery-life#:~:text=A%20typical%20charge%20or%20use,another%208%20hours%20of%20use.

[43] "Texas instruments bq2954pn, battery charge controller ic, 5 v 16-pin, pdip." [Online]. Available: https://za.rs-online.com/web/p/battery-management/6622319P?cm_mmc=ZA-PPC-DS3A-_-google-_-DSA_ZA_EN_Semiconductors_Index-_-Battery%2BManagement%7C%2BProducts-_-DYNAMIC%2BSEARCH%2BADS&matchtype=&dsa-1651367915998&gclid=Cj0KCQjwsIejBhDOARIsANYqkD05-MmX9y3N-1dcICXwLRxoO-F0i-6OrngMDZlloEQw8s4QAV28keIaAuIJEwcB&gclsrc=aw.ds

[44] "What's the lifespan of your lithium-ion battery?" Feb 2021. [Online]. Available: https://northeastbattery.com/whats-lifespan-lithium-ion-battery/

[45] "What is reverse polarity protection?" [Online]. Available: https://sens4.com/reverse-polarity.html#:~:text=Reverse%20polarity%20protection%20is%20an,in%20the%20transmitter%20or%20transducer.

[46] [Online]. Available: https://www.wolfspeed.com/knowledge-center/article/schottky-diode-characteristics-and-applications/#:~:text=Schottky%20diodes%20are%20used%20for,from%20conducting%20to%20blocking%20state.

[47] B. Power, "What is overcurrent protection?" Sep 2021. [Online]. Available: https://www.baypower.com/blog/what-is-overcurrent-protection/#:~:text=Vital%20Role%20of%20Circuit%20Overcurrent%20Protection&text=As%20such%2C%20all%20electrical%20circuits,of%20damage%20and%20electrical%20hazards

[48] "Climate of the kalahari desert." [Online]. Available: https://www.britannica.com/place/Kalahari-Desert/Climate

[49] M. Hampton, "Lithium-ion batteries and thermal runaway," Jul 2013. [Online]. Available: https://blog.storemasta.com.au/lithium-ion-batteries-and-thermal-runaway#:~:text=If%20a%20single%20lithium%2Dion,your%20staff%20and%20your%20surrounds.

[50] S. Electronics, "Raspberry pi zero v1.3," p. 1, May 2023.

[51] S. X. Yuan, "Lilygo® ttgo t-display esp32 wifi and bluetooth module development board for arduino 1.14 inch lcd," p. 1, 2020.

[52] PlatformIO, "Professional collaborative platform for embedded development," p. 1, 2023.

[53] J. Yang, "When poll is better than interrupt," p. 1, February 2012.

[54] J. Nielsen, "10 usability heuristics for user interface design," p. 1, November 2020.

[55] B. Murphy, "Ms teams interview with ben murphy," May.

[56] "Ttgo t-display esp32 1.14 inch wifi module." [Online]. Available: https://www.robofactory.co.za/esp-controllers/176-ttgo-t-display-esp32-114-inch-wifi-module.html

[57] E. Vartanian, "9 c++ data structures you need to know for your coding interview," Dec 2021. [Online]. Available: https://www.educative.io/blog/cpp-data-structures-interview-prep

[58] M. York, "Is std::vector so much slower than plain arrays?" May 1957. [Online]. Available: https://stackoverflow.com/questions/3664272/is-stdvector-so-much-slower-than-plain-arrays

[59] "A guide to sd and microsd card types." [Online]. Available: https://www.kingston.com/en/blog/personal-storage/microsd-sd-memory-card-guide

[60] "What is machine learning? | how it works, tutorials, and examples - matlab amp; simulink." [Online]. Available: https://www.mathworks.com/discovery/machine-learning.html#:~:text=Machine%20Learning%20is%20an%20AI,predetermined%20equation%20as%20a%20model.

[61] "Live worms." [Online]. Available: https://jurassicjungle.com.au/products/livemealworms#:~:text=and%20100%20gram).-,Each%20mealworm%20weighs%20approximately%200.125%20grams%20each%2C%20which%20equates%20to,week%20at%20normal%20room%20temperature.

Appendix A

Code

A.1 Python Tailored Algorithm

The following python code implements the tailored algorithm described in section 6.2.2 above and graphically displayed in the flow chart shown in figure 6.4. Pandas data-frames were used for ease of array manipulation.

```
import pandas as pd
# constant values
BAIT_WEIGHT_OFFSET = 0.2
DRONGO_WEIGHT_LOWER_BOUND = 42
DRONGO_WEIGHT_UPPER_BOUND = 60

# function that determines true measurement of birds weight from sampled data
def determine_drongo_weight(sampled_data: list[float]) -> float:

    # accounts for the weight of the bait
        # 0.2g is subtracted from each sampled data element
    df = pd.DataFrame(sampled_data, columns=['total_weight'])
    df['weight_drongo'] = df['total_weight'] - BAIT_WEIGHT_OFFSET

    # disregards any value that falls outside of the acceptable range of a drongos weight
        # effectively applies a band pass filter with lower limit 42g and upper limit 60g
    df_filtered = df[(df['weight_drongo'] > DRONGO_WEIGHT_LOWER_BOUND) | (df['weight_drongo'
    ↪ ] < DRONGO_WEIGHT_UPPER_BOUND)]

    # obtaining the mode of the data
    drongo_weight_modes = df_filtered.mode()['weight_drongo'].tolist() # accounting for the
    ↪ possibility of multiple modes
    drongo_weight = sum(drongo_weight_modes)/len(drongo_weight_modes) # averaging the modes
    ↪ if there is more than 1

    return drongo_weight
```

Listing A.1: Python Implementation of Tailored Algorithm

A.2 Python Kalman Filter Algorithm

```python
# function that applies the Kalman Filter to an array of samples
def kalman_filter(signal):
    estimate = 42.0                     # initial estimate of the bird's weight in grams
    estimate_uncertainty = 50.0         # initial uncertainty of estimate - will be updated
    measured_uncertainty = 50.0         # initial uncertainty of measurement - will be
    ↪ updated
    output = np.zeros(len(signal))

    for i in range(len(signal)):
        if 42.0 <= signal[i] <= 60.0:

            gain = estimate_uncertainty / (estimate_uncertainty + measured_uncertainty)
    ↪     # obtaining the filter gain
            estimate = estimate + gain * (signal[i] - estimate)
    ↪     # updating the estimate

            estimate_uncertainty = (1 - gain) * estimate_uncertainty
    ↪     # updating the estimate uncertainty
            output[i] = estimate

        output[i] = estimate

    return estimate
```

Listing A.2: Python Implementation of Kalman Filter Algorithm

A.3 C++ Tailored Algorithm

```cpp

// online C++ compiler to run C++ program online
#include <iostream>
#include <cmath>

// constant values set for the bait weight offset and the upper and lower limits of a
    ↪ drongos possible weight
float BAIT_WEIGHT_OFFSET = 0.2;
float DRONGO_WEIGHT_LOWER_BOUND = 42;
float DRONGO_WEIGHT_UPPER_BOUND = 60;   // this was 57 but as added precaution increased to
    ↪ 60

```

```
11 int main() {
12
13   int samples_to_remove = 0; // number of samples that fall out of the acceptable range of
       ↪ drongo weight
14
15   // apply the bait weight offset and counts the number of samples that fall out of the
       ↪ acceptable range of drongo weight
16   for (int i = 0; i < 50; ++i) {
17     weight_readings[i] = weight_readings[i] - BAIT_WEIGHT_OFFSET;
18     if (weight_readings[i] > DRONGO_WEIGHT_UPPER_BOUND || weight_readings[i] <
       ↪ DRONGO_WEIGHT_LOWER_BOUND) { // checks that the weights are outside of the range
19       samples_to_remove += 1; // increments counter
20     }
21   }
22
23   int final_arr_size = 50 - samples_to_remove; // number of samples that fall within the
       ↪ acceptable range of drongo weight - size of final array
24
25   float final_arr[final_arr_size];        // array to store filtered data
26   int final_arr_rounded[final_arr_size];  // array to store rounded filtered data
27
28   int k = 0; // index counter for rounded values
29
30   // adds the usable weight data to a final array - data within the acceptable range of
       ↪ drongo weight and creates a rounded (i.e.integer) version of the final array
31   for (int i = 0; i < 50; ++i) {
32     if ((weight_readings[i] < DRONGO_WEIGHT_UPPER_BOUND) && (weight_readings[i] >
       ↪ DRONGO_WEIGHT_LOWER_BOUND)) { // checks that the weights are within the range
33       final_arr[k] = weight_readings[i];
34       final_arr_rounded[k] = round(weight_readings[i]);
35       k += 1;
36     }
37   }
38
39   // calculates the mode of the data
40   bool mode_bool = false;   // boolean that indicates if the sampled data has a mode
41   int mode = 0;             // initilaises the mode to 0
42   int counter = 1;          // set to 1 because every value has at least 1 repeat
43   int maxCount = 0;         // number of occurances of the mode
44
45   // loops through all values in rounded array and compare to all other values in rounded
       ↪ arary to determine mode (i.e a repeated value)
46   for (int i = 0; i < final_arr_size; ++i) {
```

```
47      counter = 1; // resets the counter
48      for (int j = i+1; j < final_arr_size; ++j){
49        if (final_arr_rounded[i] == final_arr_rounded[j]){ // increments counter if there is a
        ↪  repeated value and sets boolean to true as there is a mode
50          counter += 1;
51          mode_bool = true;
52        }
53      }
54      if (counter > maxCount){ // if there is a more repeated value, it becomes the mode
55        maxCount = counter;
56        mode = final_arr_rounded[i];
57      }
58    }
59
60    // loops through the final array to find the average of all decimal values that occur
        ↪ around the mode determined from the rounded final array
61    float weight_temp = 0.0;      // stores the total weight for average calculation
62    for (int i = 0; i < final_arr_size; ++i) {
63      if (final_arr[i] > float(mode - 1.0) && final_arr[i] < float(mode + 1.0)){
64        weight_temp += final_arr[i];   // adds all the values together to obtain the total
65      }
66    }
67
68    float weight = float(weight_temp/float(maxCount)); // calculates the average of the values
        ↪  occuring around the rounded mode value
69
70    // if the sampled weight does not have a mode then the average of all the values will be
        ↪ used as the final weight value
71    if (mode_bool == false) {
72      float total = 0.0;      // stores the total weight for average calculation
73      for (int i = 0; i < final_arr_size; ++i) {
74        total += final_arr[i];    // adds all the values together to obtain the total
75      }
76      weight = total/float(final_arr_size); // calculates the average of the values in the
        ↪ final array
77    }
78
79      return 0;
80  }
```

Listing A.3: C++ Implementation of Tailored Algorithm