EIM-409: DEVELOPMENT OF AN INTELLIGENT SYSTEM FOR MONITORING BIRDS' HEALTH IN THE NEW MANDAI BIRD PARK

LIM YONG SONG

INNOVATION & DESIGN PROGRAMME

NATIONAL UNIVERSITY OF SINGAPORE

2020

EIM-409: DEVELOPMENT OF AN INTELLIGENT SYSTEM FOR MONITORING BIRDS' HEALTH IN THE NEW MANDAI BIRD PARK

LIM YONG SONG

A THESIS SUBMMITED FOR THE DEGREE OF

BACHELOR OF ENGINEERING

(COMPUTER ENGINEERING)

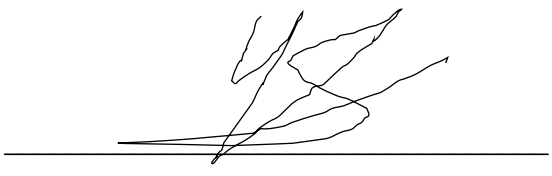
INNOVATION & DESIGN PROGRAMME

NATIONAL UNIVERSITY OF SINGAPORE

2020

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in this thesis.



LIM YONG SONG

29 March 2020

ACKNOWLEDGEMENT

With the progress of technology and pursuit of rights for animals in the recent years, technology has increasingly been utilised to make work more efficient while helping to ensuring the rights for animals are considered. This project aims to utilise emerging technology to improve working conditions for employees in the animal industry, especially bird parks and zoos, while also alleviating the living conditions of the animals in captivity in the process.

First of all, I would like to thank Dr. Tang Kok Zuea for initializing and supervising this project and giving me a chance to take advantage of the progresses in technology to make an impact in the current animal industry. Secondly, I would like to thank Nicole, Marc and Juan, from the Mandai Nature Reserve, who were ever so passionate about the welfare of the animals and keepers by providing us with unprecedented access, resources and collaboration into the Mandai Project. Thirdly, I would like to thank Dr Khoo, Glenn, Jonita and Ben, who have also been providing valuable insights, advice and help throughout the duration of the project. Last but not least, I would like to also sincerely thank my juniors, all the Mandai staff members, vendors, NUS lab technicians and staff members without whom the project would not have been possible.

I have no valuable words to express my thanks, but my heart is full of the favours received from every person involved in the project. It is with your help that the project has been a successful one and I believe the future is bright in the collaboration.

Table of Contents

I. Summary

As part of the Draft Master Plan, Mandai Nature Reserve had planned the Mandai Project which involves the Singapore Bird Park getting a major facelift with hopes of improving show-bird numbers and greater interactivity with more walk-in aviaries. To create a close replication of conditions in the wild, the aviaries will be filled with multiple bird species with appropriate vegetation.

Since the birds would be allowed greater freedom to roam freely in the aviaries, bird keepers would need to invest more time and effort to monitor the birds' health on top of their currently tight schedules. However, injured and unwell birds could be overlooked if close and proper monitoring were not given to the birds in a big aviary environment. On top of this, frequent catch and handling of birds to check on their health could cause unnecessary stress and damage to the birds.

Thus, in this project, a system would be designed to feed, monitor and record critical bird data with minimal interaction between the bird keeper and the birds. It would also be able to identify and secure injured birds for immediate treatment while alerting the keeper. This would improve the efficiency of the keepers while maintaining quality of life for the birds. In this report, user-centred design was utilised to meet the needs of the stakeholders.

The final integrated system included radio-frequency identification (RFID) readers and tags, customised parts using 3D printing and software components for wireless communication and data recording, with ethical considerations and careful laboratory testing.

Index terms *– User-centred design, radio-frequency identification (RFID) technology, sensors, wireless communication, data handling, circuit design, animal-testing*

II. List of Tables

III. List of Figures

IV. List of Illustrations

1. Introduction

The Singapore Bird Park planned a major revamp and relocation to Mandai region in Singapore's SGD$1 billion effort to boost Singapore's eco-tourism (URA SG, 2019). Expected to house one of the world's largest collection of birds, the new bird park will encompass 17-hectares at the planned 126-hectare mega-nature attraction in Mandai (Siau, 2019).

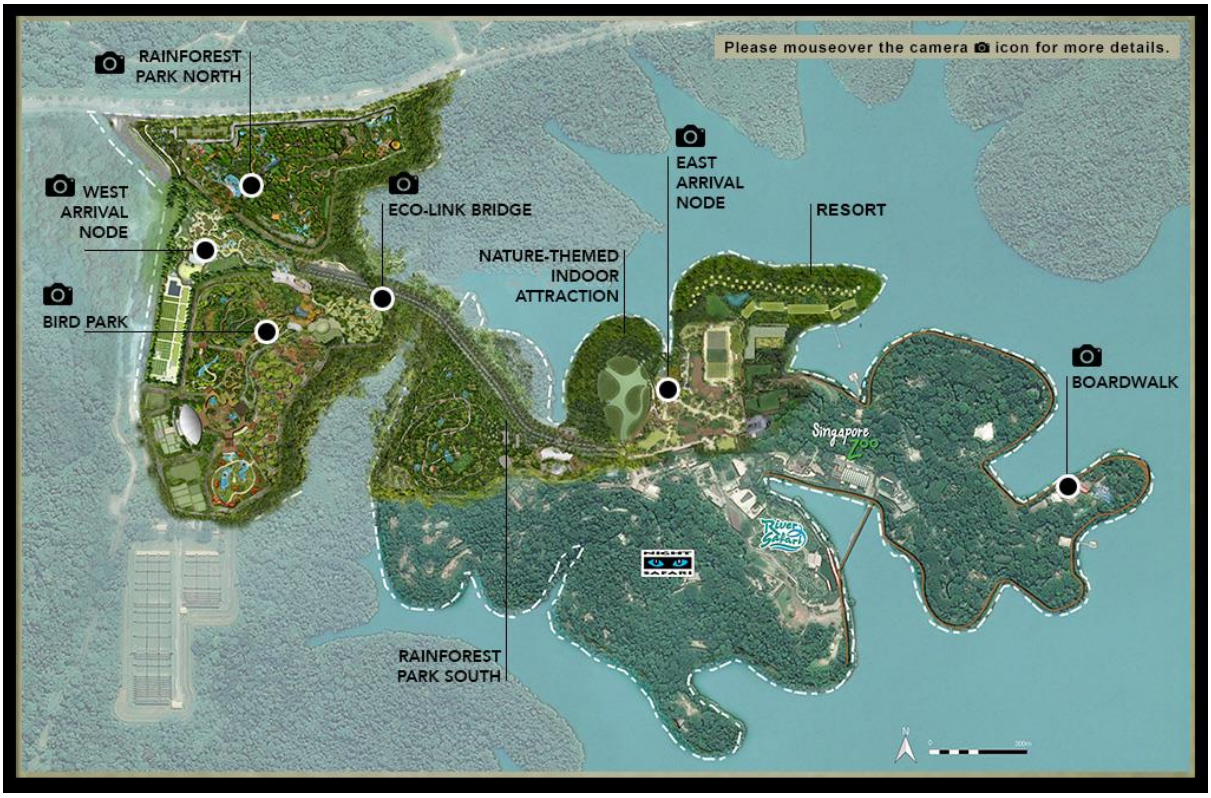


Figure 1.1: Map of Planned Mandai Nature Park by Mandai Park Holdings (MPH)

The new bird park would be moving away from traditional enclosures and expanding on their existing aviaries, from 3 to 9 different walk-in aviaries modelled depending on habitats, which would allow the birds to freely roam in the area (Siau, 2019). This would allow the birds to have more freedom and experience less stress as their environment would be a close replication of what they might experience in the wild.

As the bird park would be introducing more birds while expanding on their free roaming concepts in large-scale aviaries, bird keepers would find it harder to maintain logistics activities and perform tasks in the park. This resulted in difficult manual monitoring of the birds on top of feeding, cleaning, organising park activities, etc, when the birds were frequently out of sight. On top of this, frequent catching and handling of the birds could cause extreme stress within the birds even if they did not display them outrightly (Maho, 1992) and should be avoided if possible. Thus, in this project, a solution would be designed to help streamline data collection while also classifying injured birds so that bird keepers can tend to them promptly, without the need for it to command a large portion of their time.

2. Research

In order to better understand the problem at hand, the three main stakeholders were identified, and they were the bird keepers, birds and visitors

From this outset, primary and secondary research were done so as to come up with insights into the issue at hand and the value proposition to solve the problem.

2.1. Primary Research

The primary research was to establish context based on first-hand knowledge either from oneself or an established individual experienced in the craft. This was to get a deeper understanding of the details and current state of affairs of the problem at different levels. As such, an interview with Nicole, a chat with Glen and a visit to the bird park was done. Nicole was the bird park chief keeper while Glen was the former junior nutritionist assistant. This allowed the project to get a deeper insight into the management, ground worker and current situation of the problem. The summarised

interviews and visit findings were detailed below, while the full information can be found in the Appendix A and B respectively.

Nicole, Current Chief Keeper from Jurong Bird Park

From the interview with Nicole at the bird park, she highlighted that the current way data collection/storage was time intensive and needs improvements. They currently rely on manual bird counting keep up with logistic data while catching the birds for health checks, which happen once a year, to keep up with their health status. However, all these data are stored using paper-based data logging which makes it hard to retrieve and keep up with birds' health information. It is also brought up that the data might also be unreliable as it is hard to see or catch all the birds during data collection attempts.

On top of this, she also highlighted that they want to minimise unnecessary handling as it is hard to find and catch all the birds, especially smaller flying birds. This results in injured birds only being treated when they display major health issues and are seen by the bird keepers on happenstance. Since birds' health deteriorate very fast, late treatment can lead to detrimental circumstances.

Moving on to current state of the park, Nicole mentioned that activities are constantly planned at different parts of the park to help increase visitor engagement and improve visibility of the birds, where birds have specific feeding timings. Most birds are also tagged on their feet or torso with a passive radio-frequency identification (RFID) tag, which is a Trovan FDX-A based tag. The RFID tag is used to identify bird individuals from their counterparts and aid in logistic data collection.

After which, she mentioned some characteristics on birds where they tend to shed feathers every few weeks making it difficult to differentiate between individual birds within a species. Also, bird have a preference on where and what they want to eat, but

when they are sick or injured, they tend to eat less or even stop eating, which results in massive falls in their weight.

Glen Tan, Former Junior Nutritionist Assistant from Jurong Bird Park

From the interview with Glenn, he highlighted that the new bird park will have a heavy focus on the aviaries. For aviaries, bird keepers have many things to do including logistics like cleaning, bird counting, visitor activities and it can get very hectic as some of them needs to be done multiple times a day, so they are unable to finish all their tasks everyday consistently. Feeding can also be very taxing because food needs to be changed multiple times in a day as they spoil very fast. Also, to make things harder some aviaries do not have paths throughout the aviary which makes it difficult to check on birds that prefer stay at the edges (e.g. Lory Loft Aviary) where they are rarely seen.

When talking about the importance of bird counting and keeping track of their health, he said that it is important to keep their numbers constant because birds compete frequently for space and food so to make sure less aggressive birds are well-fed, aggressive birds need to be kept in check, in reference to the less aggressive birds.

Currently bird keepers rely on simple cages to introduce, re-introduce, quarantine or trap specific birds, while more custom cages are used to separate aggressive birds or to improve visibility, so that visitors can see them more easily. When using the cages to trap injured birds, bird keepers will set a trap and wait out to attempt to catch the bird although it typically will take multiple attempts and a lot of time.

Visit to Current Jurong Bird Park

A visit was made to the current Bird Park to better understand the situation and step into the shoes of the visitors. Key findings were that the feeders were placed near

visitor routes to lure birds closer to the routes but they also need to be inconspicuous or blend with their surroundings, while 2 main type of cages was also seen, one which allow people to move in to handle the bird and one which traps birds from a distance, as depicted below.

Figure 2.1: Feeding System

Figure 2.2: Large Cage

Figure 2.3: Small Cage

2.2. Secondary Research

To supplement insights gained through primary research by talking to the stakeholder and direct observation through ethnography, secondary research was done to get variable understanding to supplement any gaps in knowledge. This was done by reviewing articles, government plans and wildlife videos.

Straits Times Article (Tan, 2019)

Reading multiple articles on what the new bird park would be like, it can be summarised that the new bird park will consist of large meshed aviaries to prevent birds from escaping into the nearby nature reserve where they might be harmful to the native ecosystem. It was also noted that the meshes along with other manmade structures should blend into the nature environment so that the visitors do not even notice them.

Wildlife Reserve Singapore (WRS, 2019)

Videos produced from Wildlife Reserve Singapore was reviewed, where different species was found to have different colouration/ sounds, and some are vastly different in male and female counterparts. It was also emphasised that frequent handling was avoided as it can cause unnecessary stress to the birds, as such some birds can be/are trained to submit to medical procedure, where they can be trained to stand on a weighing scale and examined for physical defects. Other notes include, things bird keepers need to look out for when handling the birds where biometrics are typically done in the early morning when it is cooler to prevent overheating the birds during handling, as birds do not have sweat glands.

Corvid Isle Blog (TalkingBird, 2019)

Several popular wildlife blogs covering birds was also studied, where birds were found to become more wary of humans/their surroundings when injured, so traps should be properly concealed. Injured birds can also become stressed very easily and should be kept under warm and dark environment. Typical type of traps used are box traps, where birds can be baited and caught, or a net trap, where a net is used to catch injured but still flying birds. One important thing noted during research was that any person handling birds should note for panting, as it might be an indication of high stress and signs of heart attack. On top of this, the recommended minimum floor space per bird is $0.5m^2$.

2.3 Insights

From the above primary and secondary research, the findings were refined into key insights for each of the stakeholders, who were the bird keepers, birds and visitors.

From these insights, it was also acknowledged that the bird keepers were the key stakeholder that affects the needs of the birds and visitors.

Bird Keepers

From the findings, bird keepers were found out to need to be able to identify and secure injured or sick birds for treatment, since it is especially hard to find and capture small flying birds. They also need to streamline their current bird health monitoring procedure/ data collection while revamping their feeding system to be more specialised for each bird species. Another important need is to be able to keep track and account for all the different birds in the park efficiently.

Birds

From the findings, the needs of the birds are that they need different types of food and need their different feeding patterns and behaviour accounted for. They also need to be handled as minimally as possible as stress can cause major health issues. Birds' health also tends to deteriorate very fast when they are injured and should be identified as soon as possible.

Visitors

From the findings, the needs of the visitors are that they need to see the different types of birds on display easily. On top of this, visitors do not want to be within close contact of spoiling food that could attracted pests and diseases.

2.4 Value Proposition

From these key insights, the value proposition is then to design a system that can feed, monitor and record critical data of the birds with minimal interaction between the bird keeper and the birds. It should also be able to identify and secure injured/sick birds for immediate treatment while alerting the keeper, so as to improve the efficiency of the bird keepers while maintaining quality of life for the birds.

3. Design Specifications

The project aimed to design a system that could feed, monitor and record the logistic data of the birds with minimal interaction between the bird keeper and the birds. It should also be able to identify and secure injured/sick birds for immediate treatment while alerting the keeper, so as to improve the efficiency of the bird keepers while maintaining quality of life for the birds.

3.1. Function Analysis

To help better partition and understand the design components for a purposed design, the function analysis diagram in Figure 3.1 was created. This would highlight the core functions that the system required and allowed for diverse concept designs that still solved the needs of the stakeholders.

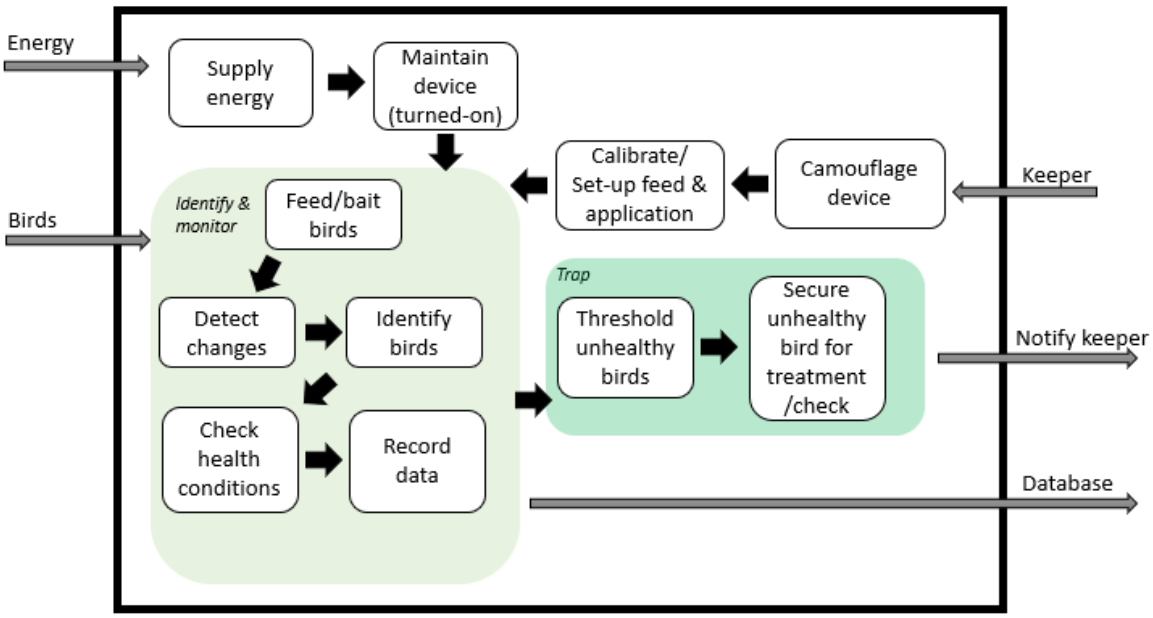


Figure 3.1: Function Analysis Diagram

3.2. Morphological Chart

After analysing and partitioning the design into the basic necessary functions, the morphological chart was then conceived to identify potential solution principles that could be implemented. Table 3.1 served as a guide/outline to develop initial concepts that were in line with the function analysis. This methodical approach followed the double diamond design process which allowed for more diverse and novel design ideas.

Table 3.1: Morphological Chart

Function	**Solution Principles**			
Supply Energy	Grid (mains)	Solar	Battery	
Caging	Box	Wire/metal cage	Mesh nets	Open (no cage)
Feeding	Removable feeding tray	Fixed (to cage) feeding tray	Automatic dispensing	
Detect movement (to power up)	Proximity Sensor	Pressure sensor	None	
Identify bird (RFID + ?)	Camera/ Webcam	Sound (bird call)	None	
Check weight	Weighing scale	Load cell	Trap and weigh	
Record data	Log file in local storage	Database on server	Excel sheet on computer	
Trap bird	Sedate	Manual (by predicting feeding pattern)	Seal/close the cage	
Notify Keeper	SMS	Email	Telegram/Whats App etc.	Sound (beeping)

4. Concept Solutions

Once the morphological chart had been conceived, a combination of solution principles was used to develop a purposed design that could be novel, feasible and meet the needs of the stakeholders. Multiple sketches were devised which resulted in 4 main concepts being chosen for consideration, which could be seen below.

4.1. Initial Concepts

Concept 1 (Reference): This concept aimed to create a portable feeding system that could be place anywhere in the aviary. The system would be battery powered and booted up from standby mode (to be more power saving) when the weight sensor was compressed. The system would then weigh and identify (using rfid) the bird and saved the details into a local log file (where it can be extracted through usb to a computer). If it detected any discrepancies from previous data (i.e when the weight of a bird drops too much), the trap would be triggered to seal the bird in the system and beeped to notify

the bird keepers. The design could be charged at the end of the day and washed easily while all materials and parts chosen are as light weight as possible.

Table 4.1: Concept 1 Principles Selection

Function	Solution Principles			
Supply Energy	Grid (mains)	Solar	Battery	
Caging	Box	Wire/metal cage	Mesh nets	Open (no cage)
Feeding	Removable feeding tray	Fixed (to cage) feeding tray	Automatic dispensing	
Detect movement (to power up)	Proximity Sensor	Pressure sensor	None	
Identify bird (RFID + ?)	Camera/ Webcam	Sound (bird call)	None	
Check weight	Weighing scale	Load cell	Trap and weigh	
Record data	Log file in local storage	Database on server	Excel sheet on computer	
Trap bird	Sedate	Manual (by predicting feeding pattern)	Seal/close the cage	
Notify Keeper	SMS	Email	Telegram/Whats App etc.	Sound (beeping)

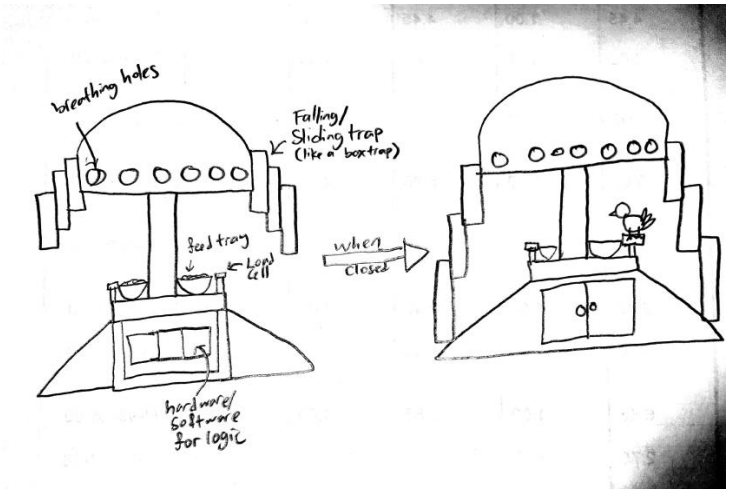


Figure 4.1: Schematic of Concept 1

Concept 2: This concept aimed to build on the current large trap cages in the bird park as the birds should be more used to a familiar system. Being a fixed structure allowed use of heavier elements while leveraging on the grid power supply which allowed for more features. This device would be able to be power on constantly and identify the birds through RFID while also having a backup species identification with a camera if RFID identification failed. It would also send and retrieve data from a database through WIFI to allow consideration of multiple potential injuries types (other than basing solely on weight change). With a small opening to close the cage, it would

not startle the birds as much and would also be able to immediately send emails, wirelessly, to update keepers once it was triggered.

Table 4.2: Concept 2 Principles Selection

Function	Solution Principles			
Supply Energy	Grid (mains)	Solar	Battery	
Caging	Box	Wire/metal cage	Mesh nets	Open (no cage)
Feeding	Removable feeding tray	Fixed (to cage) feeding tray	Automatic dispensing	
Detect movement (to power up)	Proximity Sensor	Pressure sensor	None	
Identify bird (RFID + ?)	Camera/ Webcam	Sound (bird call)	None	
Check weight	Weighing scale	Load cell	Trap and weigh	
Record data	Log file in local storage	Database on server	Excel sheet on computer	
Trap bird	Sedate	Manual (by predicting feeding pattern)	Seal/close the cage	
Notify Keeper	SMS	Email	Telegram/Whats App etc.	Sound (beeping)

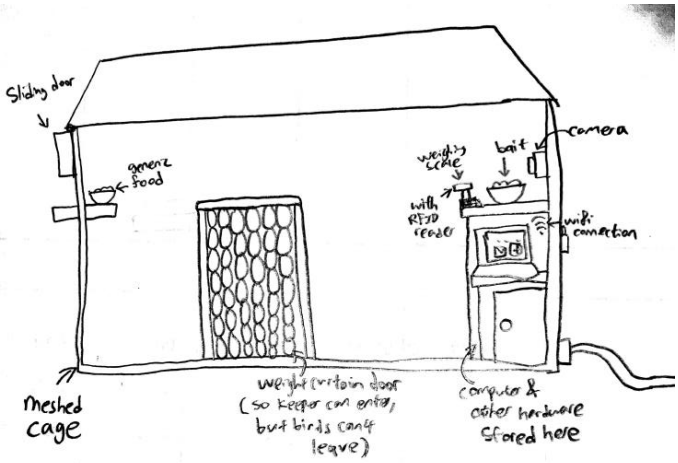


Figure 4.2: Schematic of Concept 2

Concept 3: This concept aimed to create a light-weight device that could be hoisted near treetops to feed/ monitor birds living mainly in those areas. It consisted of a mesh net that had several openings for bird to fly into and one opening for the keeper to move the bird feeder (through a pulley). The mesh contained a hanging weighing scale with RFID reader placed near the feeder that would detect and record bird weights. It

would unfurl the mesh nets at the opening to close them and alert the keeper through SMS to lower the mesh net when an injured bird was detected.

Table 4.3: Concept 3 Principles Selection

Function	Solution Principles			
Supply Energy	Grid (mains)	Solar	Battery	
Caging	Box	Wire/metal cage	Mesh nets	Open (no cage)
Feeding	Removable feeding tray	Fixed (to cage) feeding tray	Automatic dispensing	
Detect movement (to power up)	Proximity Sensor	Pressure sensor	None	
Identify bird (RFID + ?)	Camera/ Webcam	Sound (bird call)	None	
Check weight	Weighing scale	Load cell	Trap and weigh	
Record data	Log file in local storage	Database on server	Excel sheet on computer	
Trap bird	Sedate	Manual (by predicting feeding pattern)	Seal/close the cage	
Notify Keeper	SMS	Email	Telegram/Whats App etc.	Sound (beeping)

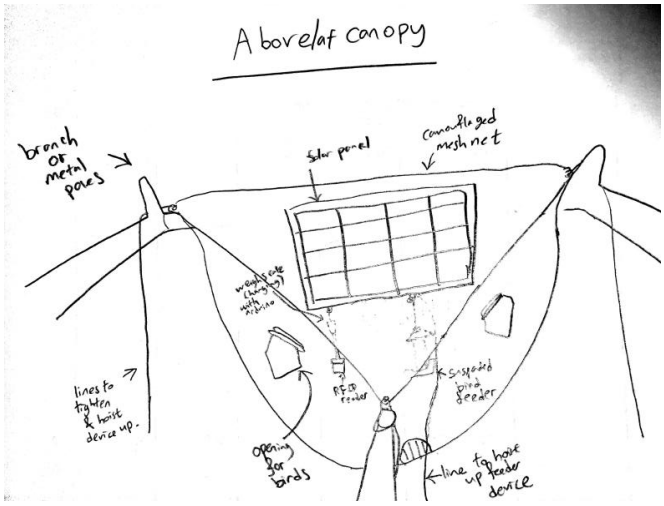


Figure 4.3: Schematic of Concept 3

Concept 4: This concept aimed to create a bare bones device that could be placed near existing feeding spots or to replace current feeding trays. It was a battery powered device that would read the RFID of the birds before weighing and recording the data into a log file. This log file could then be retrieved through USB to any computer available. Since this device solely collects data, a computer program would be written in conjunction with it to process the data which would determine what bird was injured and suggested a time for keepers to trap it by predicting their feeding patterns.

Table 4.4: Concept 4 Principles Selection

Function	Solution Principles			
Supply Energy	Grid (mains)	Solar	Battery	
Caging	Box	Wire/metal cage	Mesh nets	Open (no cage)
Feeding	Removable feeding tray	Fixed (to cage) feeding tray	Automatic dispensing	
Detect movement (to power up)	Proximity Sensor	Pressure sensor	None	
Identify bird (RFID + ?)	Camera/ Webcam	Sound (bird call)	None	
Check weight	Weighing scale	Load cell	Trap and weigh	
Record data	Log file in local storage	Database on server	Excel sheet on computer	
Trap bird	Sedate	Manual (by predicting feeding pattern)	Seal/close the cage	
Notify Keeper	SMS	Email	Telegram/Whats App etc.	Sound (beeping)

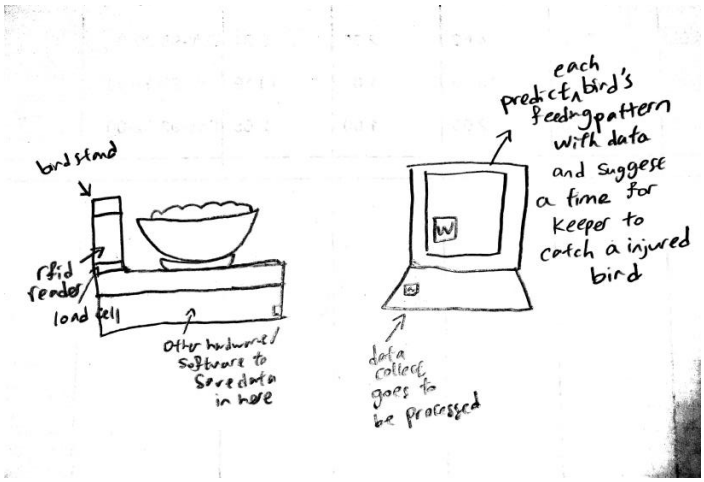


Figure 4.4: Schematic of Concept 4

4.2. Concept Screening

Once enough and varied designs had been devised, concept screening was done to identify the concept that had the most potential with respect to our problem statement. This could be seen in Table 4.5, where concepts 2 and 4 were deemed to be the best fit for the project.

Table 4.5: Concept Screening Table

Selection criteria	Reference	Concept 2	Concept 3	Concept 4
Portability	0	-1	0	+1
Camouflage	0	-1	+1	0
Ease of setup/ maintenance	0	0	-1	+1
Ease of use	0	+1	-1	0
Reliability in injury detection	0	+1	0	0
Reliability in trapping bird	0	+1	-1	-1

Amount of stress for birds	0	+1	-1	+1
Speed of notifying keeper	0	+1	+1	-1
Net	0	+3	-2	+1
Rank	3	1	4	2
Continue?	No	Yes	No	Yes

4.3. Concept Selection

After the potential concepts were identified, stakeholder needs were then weighted and rated to determine the best solution that should be developed. In Table 4.6, concept 2 was seen to be able to solve most of the relevant needs, as indicated by the high total score and was further developed as the final concept, albeit with some revisions.

Table 4.6: Concept Selection Table

Selection Criteria	**Weight**	**Concept 2**		**Concept 4**	
		Rating	**Weighted score**	**Rating**	**Weighted score**
Portability	0.04	1	0.04	5	0.2
Camouflage	0.1	2	0.2	4	0.4
Ease of setup/ maintenance	0.15	2	0.3	4	0.6
Ease of use	0.15	4	0.6	3	0.45

Reliability in injury detection	0.25	5	1.25	2	0.5
Reliability in trapping bird	0.1	4	0.4	2	0.2
Amount of stress for birds	0.06	3	0.18	5	0.3
Speed of notifying keeper	0.15	5	0.75	1	0.15
Total score		3.72		2.8	
Rank		1		2	
Continue?		Yes (with some revision/upgrades)		No	

5. Refinement

After deciding on the final concept to develop, Nicole, the key stakeholder in the project, along with key members in the Mandai Project, such as Marc and Juan, were contacted to obtain further insights and comments on the proposed design before the prototype was built. This allowed further refinement and iteration on the concept.

From the meeting, the concept was commented to be feasible as it was similar in shape to current bird cages used at the park, so the birds would be comfortable in entering the cages. However, they shared that it would be better for the cage to be miniaturised so that it could be portable while also being waterproof so that they are able to clean it easily. They also noted that the components should be modular so that the overall design can be customised to different types of bird species. On top of this, they commented that small sized birds should be the main focus as the prototype since it had the most added value.

This user engagement helped to further refine the value proposition such that the system should be portable, easy to clean and modular while being able to feed, monitor

and record the logistic data of the birds with minimal interaction between the bird keeper and the birds. It should also be able to identify and secure injured/sick birds for immediate treatment while alerting the keeper, so as to improve the efficiency of the bird keepers while maintaining quality of life for the birds.

6. Product Specifications

Once the requirement for the final concept had been decided and refined, the product specifications were then developed to help guide the product fabrication process and ensure that the product closely follows the intended outcomes.

Table 6.1: Product Specifications Table

PRODUCT SPECIFICATIONS		
Parameters	**Target Range**	**Unit**
Number of Operators	1-2	People
Primary Identification (RFID)	Between individuals	Birds
Secondary Identification (Camera)	Between species	Birds
Identification Range	>5	Cm
Identification Speed	<10	Seconds
Weight of Device	<100	Kg
Bird weight recording (range)	15-15000	Grams
Bird weight recording (accuracy)	<0.1	Grams
Cleaning	<5	Minutes
Power source	Electricity	Type
Trapping speed	<1	Seconds
Notification speed (after trapping)	<5	Minutes
Automation/ Control	CPU	Type
Bird data storage	Digital	Type
General data transmission	Wired/Wireless	Type
Size of Device	Variable depending on Bird Type	Metres

7. Detail Design

Based on this product specifications, the detailed design of the product was explored to further reveal the components and requirements needed to create a successful product. The bare-bones version, Figure 7.1, showed where each component should be placed and what component needs to be in the system while illustration 7.1 showed a high-level view of how the components should interact with each other.

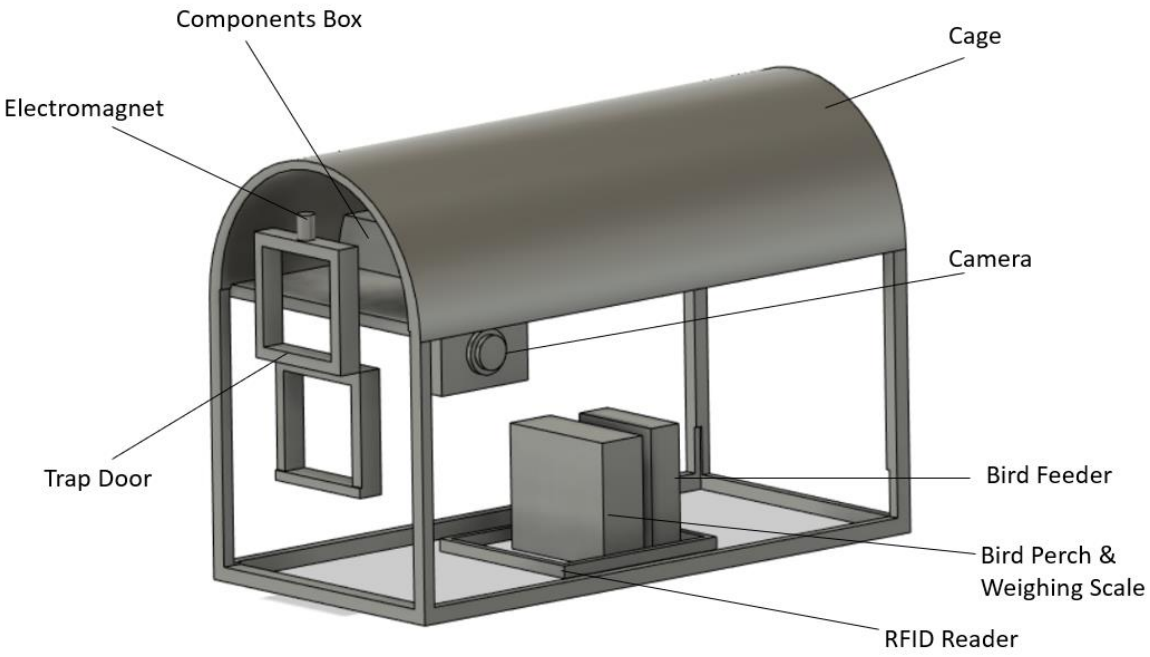
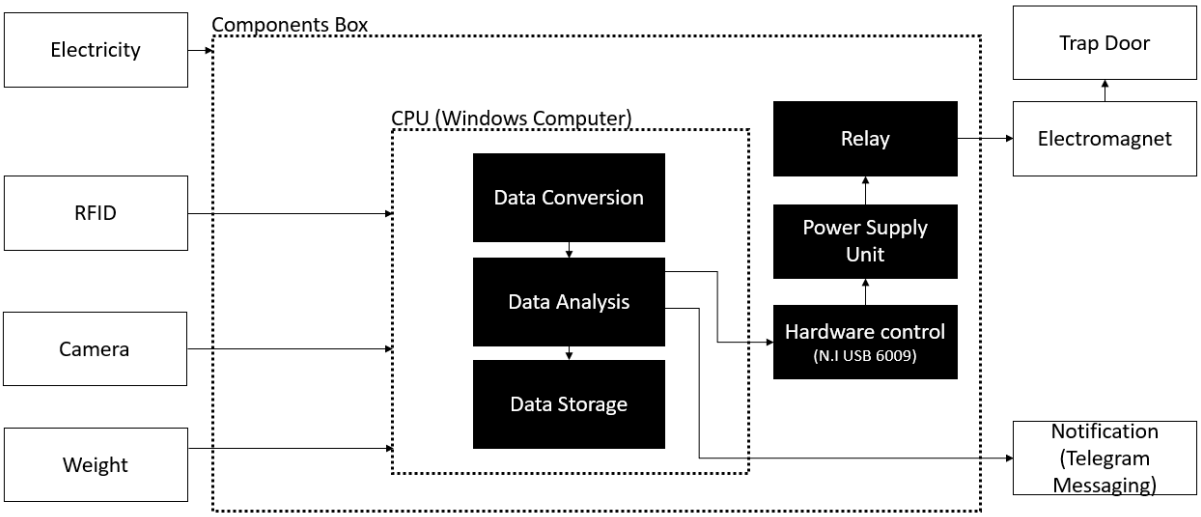


Figure 7.1: Overall Cage Design



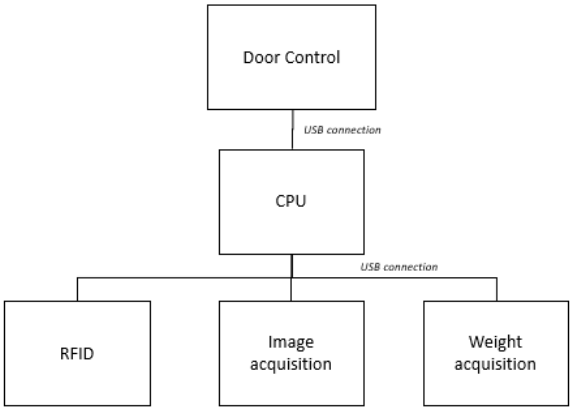
Illustrations 7.1: Overview of System in their Components

8. Prototyping

With this detailed design, a reference was created to which the system could be built upon. This became the skeleton and foundation of the initial prototype. However, to better organise and understand the process from purchase to integration, the design was encapsulated into 2 overarching features, hardware and software.

8.1. Hardware

In terms of hardware, there were 4 major components, surrounding the CPU, that needed to be considered. The 4 components were: the cage door control, radio-frequency identification detection (RFID), image acquisition and weight acquisition. The focus of this section would be to explore the hardware aspects of the project and how the components are wired. The overall wiring, in illustration 8.1, was planned with each component identified as a black box to allow for less dependency being designed into the system. This would allow for each component to be designed as a self-contained entity that could be swapped out or removed depending on the situation it needed to be used in. Current implementations of each component would be expanded upon in the subsection.



Illustrations 8.1: Overview of Overall Wiring

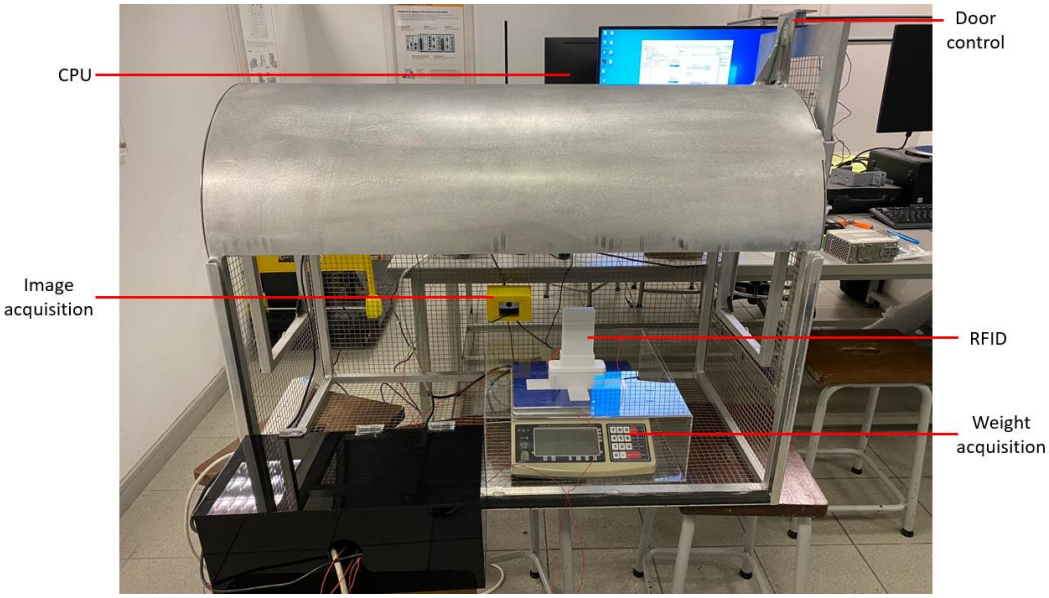
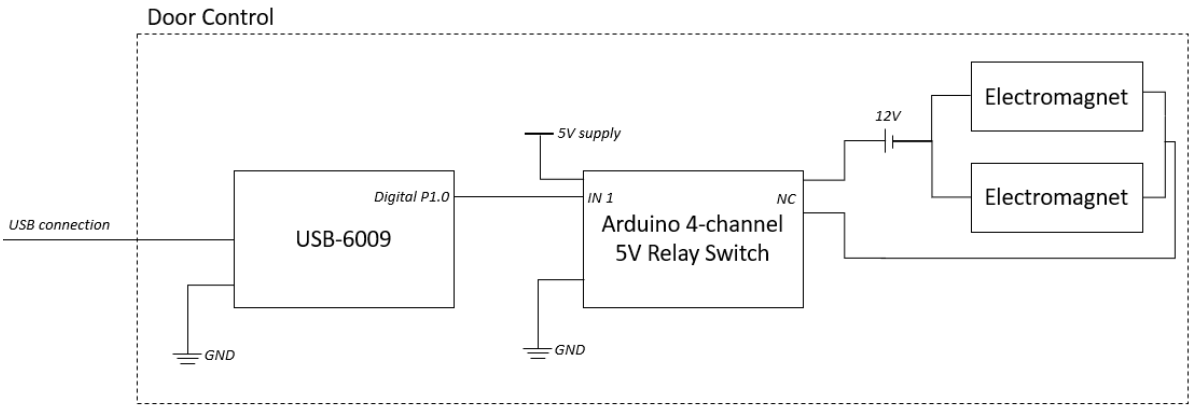


Figure 8.1: Implementation of Overall Wiring

All business logic and data control were housed within the CPU black box, which were controlled by software. Thus, the CPU component, which involved a Windows Computer, would be discussed in the following "Software" section.

8.1.1. Door Control

In order to allow for remote access to close the cage door and trap the injured birds, a door control component was devised, which would allow for software control over when the door should close. The door control was created using a simple circuit connection as seen in Figure 8.2, where the USB-6009 Data Acquisition (DAQ) unit would send a digital output to activate a normally closed 5V Arduino relay switch when a condition is met, which was determined through software. This would open the normally closed circuit, turning off the electromagnets and release the trap door to close the cage.



Illustrations 8.2: Overview of Door Control

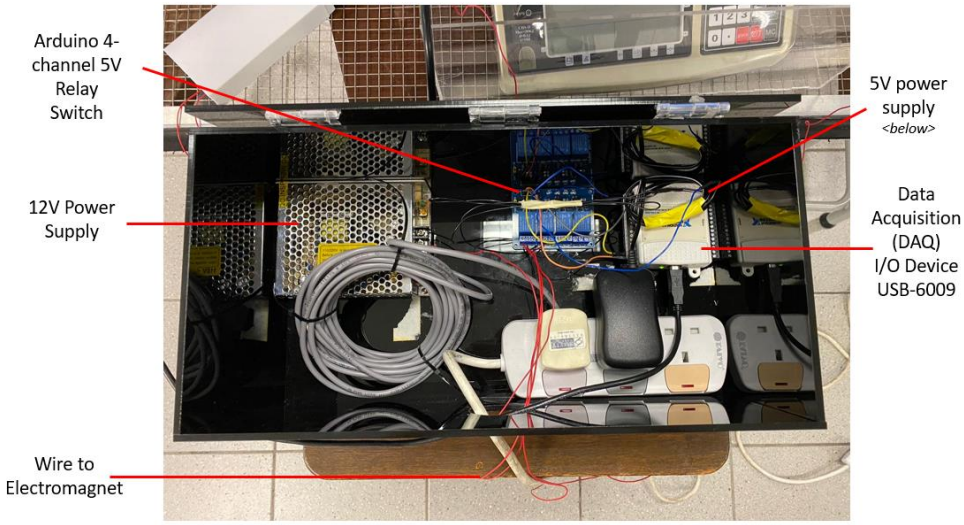


Figure 8.2: Implementation of Door Control

Initially, the design utilised a hinged trap door as seen in Figure 8.3, but after vetting by the stakeholder, Nicole, to ensure that it would the user's needs. It was found that hinged doors had too much swing, which could cause unnecessary stress for the birds while also having a high chance of injuring a bird that is within its swing radius. This round of vetting allowed us to refine the cage design to use sliding doors instead, seen in Figure 8.4, as sliding doors had a smaller and smoother motion.

Figure 8.3: Initial Cage with Hinged Door Design

Figure 8.4: Current Cage with Sliding Door Design

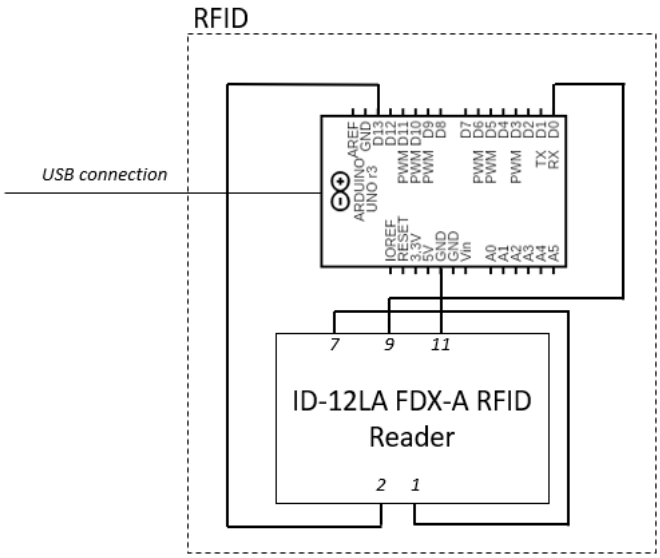
8.1.2. Radio-Frequency Identification (RFID)

According to Nicole, low frequency passive RFID was the main identification method used by the bird park to identify individual birds during bird data collection, where the bird park used Trovan FDX-A transponders, which read at around 125kHz. For the identification data to be received, the RFID reader would send a low frequency wave in the direction of the antenna, following which a passive transponder within the range of the electromagnetic wave would then convert a portion it to energy via backscatter, before using that small amount of energy to send a signal back to the reader to be deciphered.

Thus, the procurement of a suitable RFID reader was of utmost important for the system. However, Trovan RFID readers were costing a minimum of $2500AUD,

according to Mr Varun Uthappa from Trovan, which were far too expensive. Hence, the bird park correspondents were contacted for the loaning of a set of Trovan readers and transponders, which were the ANT-SQR300 reader with a LID650 decoder and ID-100 FDX-A chip, which was of size 7mmL x 1.25mmD, respectively. Apart from this, 2 other readers, that were ISO11784 certified for animal use, were also experimented with. The 2 readers, ID-12LA and Avid MiniTracker 3, were tested so that comparisons and limitations could be drawn from the experiment. This helped better determine an optimal solution since individual bird identification was a major aspect of the system.

ID-12LA (125kHz), an FDX-A RFID reader which comes with an Arduino code, and a transponder of size 12.25mmL x 1.93mmD was procured from sgbotic. Although it worked with an Arduino to decipher the RFID tag, instead of being built into the reader, the significantly lower costs of around $50 made it possible candidate.



Illustrations 8.3: Overview of RFID (ID-12)

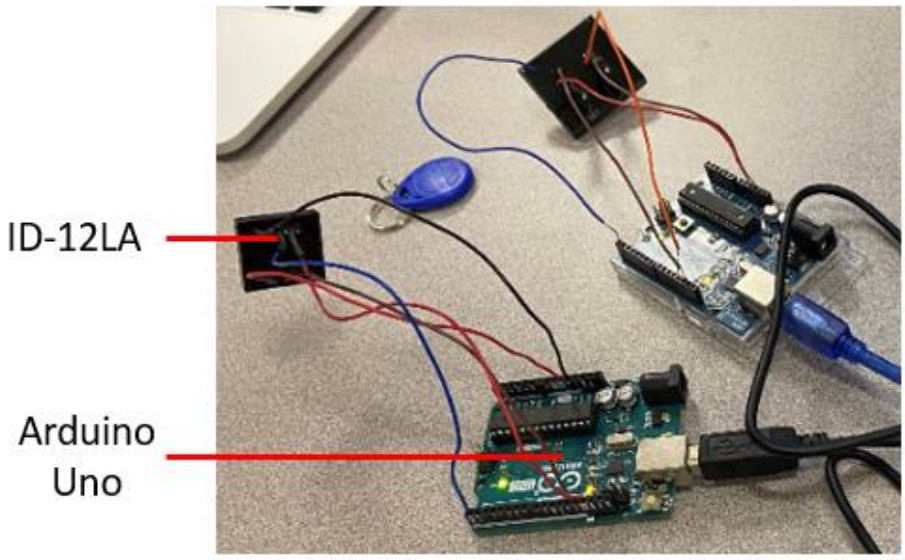
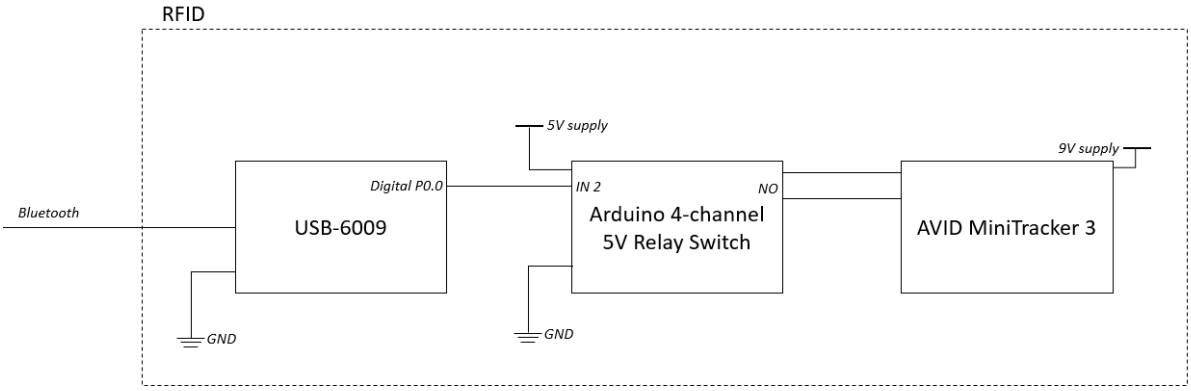


Figure 8.5: Implementation of RFID (ID-12)

Avid MiniTracker 3 was another possible candidate that was touted as a general reader that could read any low-frequency animal tag and was used in several research papers, such as in Alvarez's paper on the assessment of fire salamander populations (Alvarez, 2015). The reader had the decoder built in and cost around $400USD, which was still an acceptable price, albeit on the expensive side. However, the AVID MiniTracker 3 was slightly different, from the other 2 readers, where it required a button to be pressed before it could start reading and also used Bluetooth communication instead of USB. Thus, the buttons had to be wired to a relay switch to be turned on remotely. This was done by connecting the button switch of the reader to the relay switch through a free channel with a normally open connection. This would allow the reader to be controlled remotely and close the switch to start reading when a condition is met and for the software to accept the RFID number from the Bluetooth transmission.



Illustrations 8.4: Overview of RFID (AVID MiniTracker 3)

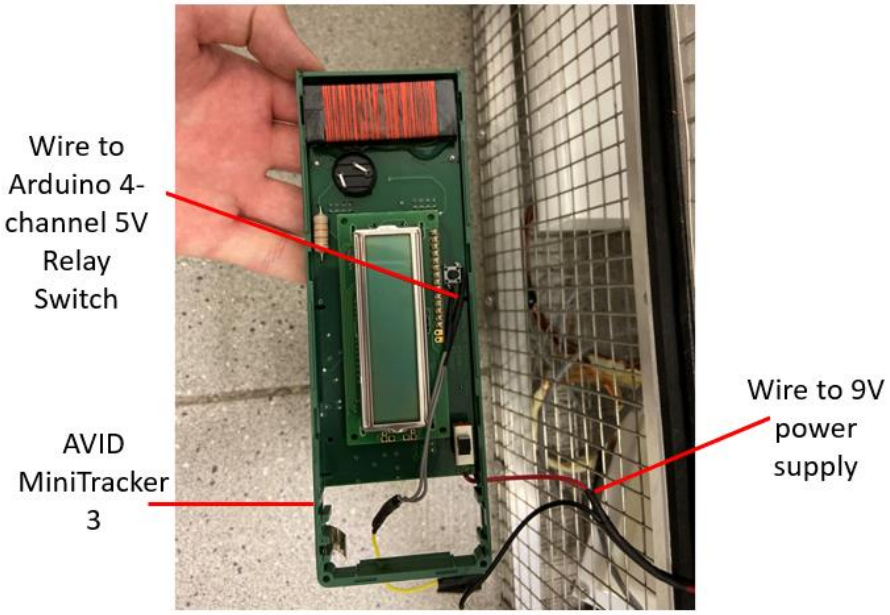


Figure 8.6: Implementation of RFID (AVID MiniTracker 3)

With all 3 readers prepared for testing, a set of tests were then made to determine the approximate read distance of the readers using both the Trovan ID-100 FDX-A chip and the sgbotic FDX-A chip. Based on the tests, it was found that the ANT-SQR300 reader from Trovan was unable to yield favourable results for both tags while ID-20LA and AVID MiniTracker 3 was only able to read either one of the tags. It was also noted that none of the readers could read any tag when a piece of metal was between the RFID tag and reader, while they could still function normally when air, a hand, cloth or plastic was the medium between the tag and reader. The results for the comparison were displayed in Table 8.1.

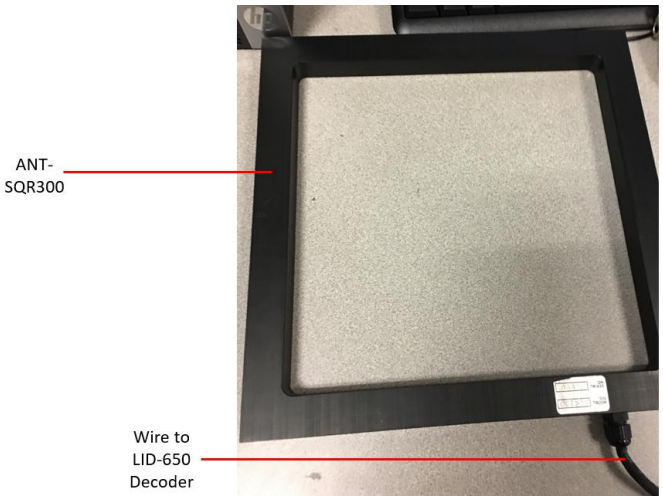


Figure 8.7:ANT-SQR300 with LID-650 Decoder

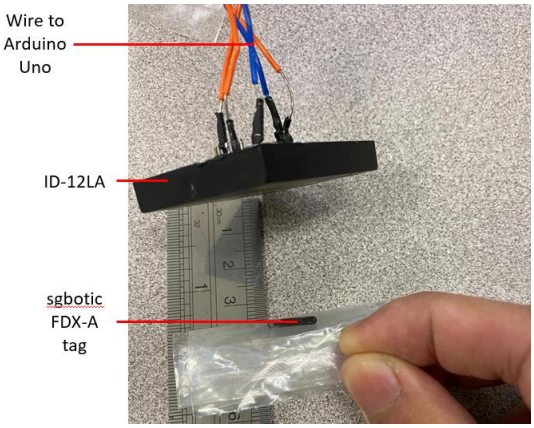


Figure 8.8: ID-12LA with Arduino Uno

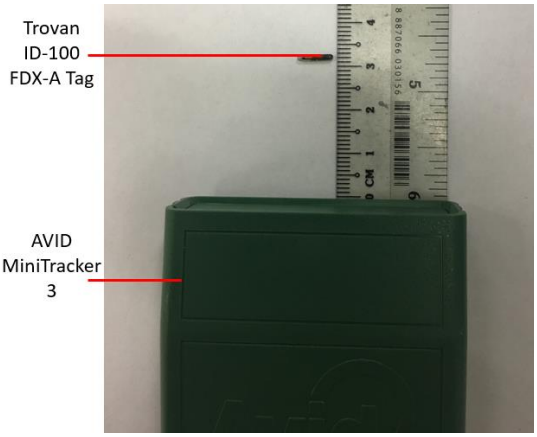


Figure 8.9: AVID MiniTracker 3

Table 8.1: FDX-A RFID Reader Comparison

RFID Reader	**Cost**	**Cost** *(In SGD)*	**Advertised Read Range** *(in mm)*	**Read Range** *(in mm)*			
				Trovan ID-100 FDX-A chip		sgbotic FDX-A tag	
				Through metal	Through other medium	Through metal	Through other medium
ANT-SQR300	$2500 AUD	$2100	90	*Unable to Read*	<10*	*Unable to Read*	*Unable to Read*
ID-12LA	$50 SGD	$50	120	*Unable to Read*	*Unable to Read*	*Unable to Read*	~35
AVID MiniTracker 3	$400 USD	$600	80	*Unable to Read*	~30	*Unable to Read*	*Unable to Read*

**Unable to read most of the time, and when it could read it was only able to detect the tag at certain locations and less than 1cm from the antenna*

Upon the findings from the experiment, the stakeholder, Nicole, was once again contacted to obtain feedback and recommendations to ensure the design stays on path to

meet their needs. From the meeting, it was commented that an optimal read range should be as high as possible and since they were going to re-tag the birds at their feet, it was suggested that the ID-12LA could be a possible reader for use in the tests.

8.1.3. Image Acquisition

In order to acquire a quality image that could be used for postprocessing when a condition was met, the camera needed to be compatible with a computer with high enough bit depth. Thus, a Logitech webcam was selected. The webcam had 24-bit RGB bit depth which allowed for 16.7 million colours. When saved as a PNG image with lossless compression, the webcam could take high quality images that could be used to determine fine gradients and edges if computer vision and machine learning was to be utilised. This selection of the camera allowed for future proofing where functions can be easily added to the system, while also providing a high-quality image for bird keepers to identify injuries or confirm if logistics data match their findings.

Figure 8.10: 3D Printed Camera Housing with Webcam

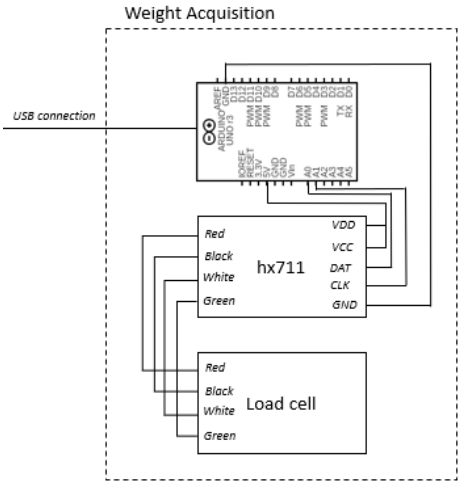
Although not within the scope of the current system, the idea of using machine learning to identify certain species, and potentially individuals, was a promising prospect to the stakeholders and they commented that it could be useful in a variety of scenarios, applying to even the zoos and night safaris.

8.1.4. Weight Acquisition

For the weight to be acquired, a bait and perch needed to be able to lure the bird to land on the weighing balance. According to the stakeholders, the weight of the bird was a determining factor of an injured bird as they tend to feed less during serious injuries.

To keep the weight acquisition inconspicuous while being able to acquire accurate results, 2 methods were experimented with, one being the hx711 with a load cell and Arduino Uno, while the other being a High Precision Weighing Scale by GMM Technoworld.

Hx711 was a precision 24-bit analog-digital converter, designed to be used with a straight bar load cell, commonly used in the do-it-yourself (DIY) community for a multitude of projects. It had a dimension of 34.3mmL x 20mmW x 3mmH, with a corresponding load cell that is 80mmL x 13mmW x 12mmH. Paired with an Arduino Uno, the footprint of the design was small and allowed for inconspicuous weight measurement. This setup worked when the load cell flexes when a weight is acted upon it, which changes the electrical resistance at the connections with the hx711 depending on the weight. The hx711 module would then convert the changes into a digital signal, which was deciphered by Arduino code through a calibration factor that was altered to obtain the correct weight number in the correct measurement unit, in this case, grams.



Illustrations 8.5: Overview of Weight Acquisition

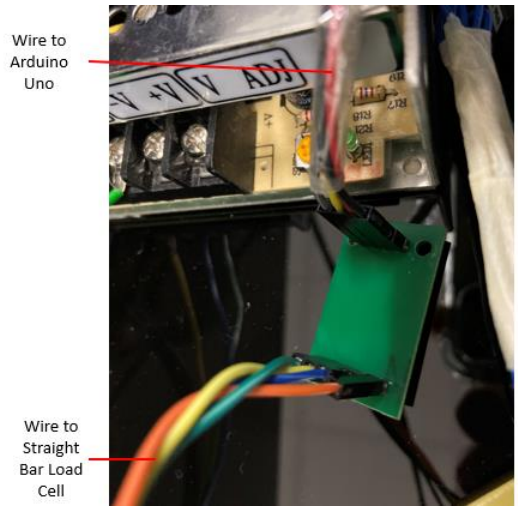


Figure 8.11: Implementation of Weight Acquisition

High Precision Weighing Scale by GMM Technoworld, was an ultra-high precision scales that was accurate to 0.1g and was typically used to measure a multitude of objects from mechanical parts to plastics and chemicals with a dimension of 340mmL x 297mmW x 110mmH. Although it was on the larger side, it could still be made inconspicuous through proper concealment or masking. The measuring balance had built-in functions that send weight measurements directly to the computer through a USB connection.

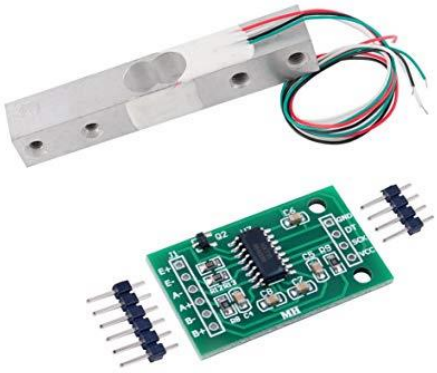


Figure 8.12: Loadcell w/ Hx711

Figure 8.13: High Precision Weighing Scale by CMM Technoworld

With both weight acquisition units setup, a set of tests were then made to determine the accuracy and reliability of the units using different sets of weights. Both the units were able to get decent accuracy, although the high precision balance fared a lot better. It was also noticed that both units had varying weighing, the hx711 had a much larger variation especially when the experiment lasts a period of time.

Table 8.2: Weighing Scales Comparison

RFID Reader	**Cost** *(In SGD)*	**Size**	**Advertised Accuracy**	**Weight**			
				75g	125g	175g	225g
Hx711 with load cell*	$10	Small	-	~63.00-92.00g	~117.00-132.00g	~163.00-190.00g	~218.00-231.00g
High Precision Counting Weighing Scale Balance with PC interface	$400	Large	0.1g	~75.0g	~125.0g	~175.0g	~225.0g

**When left turned on for some time, the hx711 load cell can drift at least 10grams or more from the correct weight and continue to drift despite removing the weight. See Appendix C*

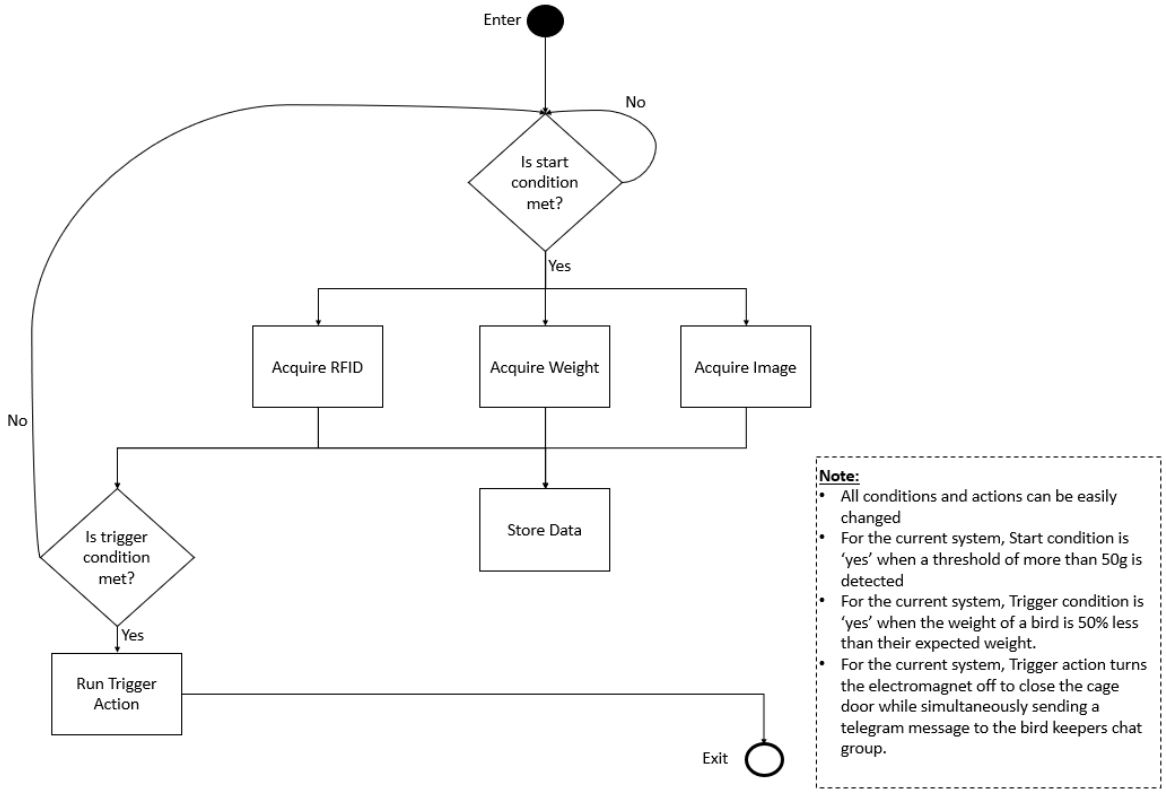
From the testing, it could be seen that the high precision balance had much better accuracy and reliability. The unreliability of the hx711 was found out to likely be due to differences in humidity and temperature, which was affecting the resistance of the load cell but the hx711 algorithm could not account for. Since it was an issue that could not be resolved and the margin of error was too large, considering the average weight of small birds were around 100-200g, the hx711 with load cell unit was shelved in favour for the high precision balance, despite its size.

8.2. Software

In terms of software, there were 7 major functions that needed to be considered, being the start condition, weight acquisition, RFID acquisition, image acquisition, data

storage, trigger condition, which triggered when birds are deemed unhealthy or injured, and trigger action. The program should run in a loop which allowed the system to be perpetually turned on and awaiting conditions to start its processes. Illustration 8.6 showed the flowchart of how the logic should flow in the program. Each component function would be abstracted and encapsulated so that the overall system would still be able to function, with minor changes, upon removal of certain functions or swapped out for certain substitutes.

The software that was used is the Laboratory Virtual Engineering Workbench (LabVIEW) and Python. LabVIEW was a system-design platform and a development environment using visual programming made by National Instruments. It allowed for quick visualization of the application and fast iteration of macro level hardware and software integration. On top of this, LabVIEW also allowed for Python code integration, which allowed for better micro level control of the system. The combination of LabVIEW and Python gave the perfect combination for macro and micro control of the system that could be rapidly iterated upon. They were both also software that are easy to pick up and allow for easy maintenance and upgrade without a steep learning curve.



Illustrations 8.6: Flowchart of System Logic

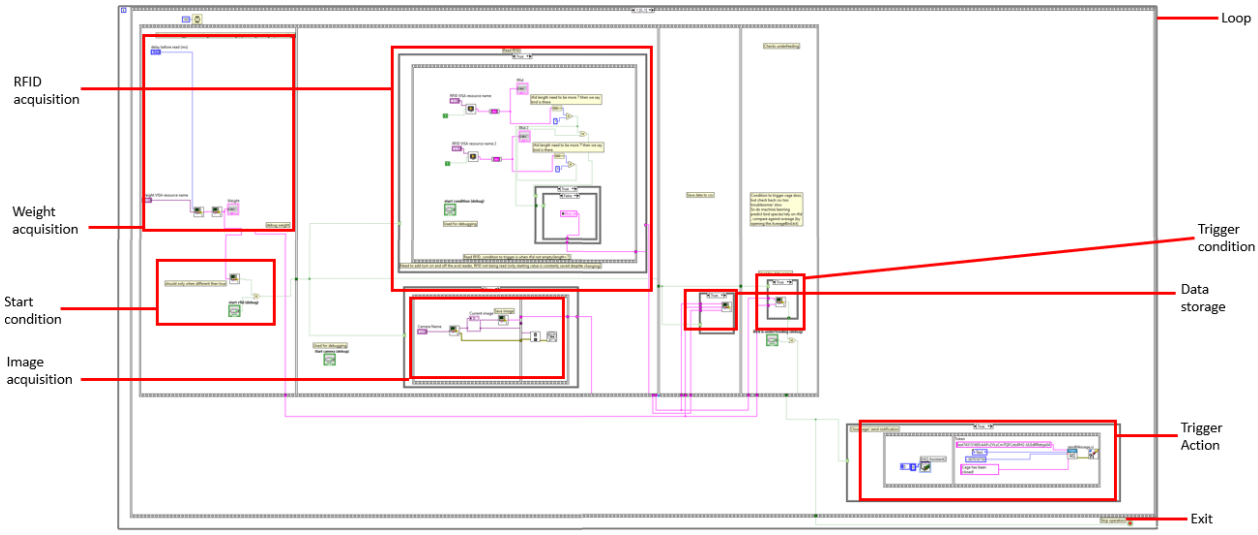


Figure 8.14: Implementation in LabView (a Visual Programming Language)

8.2.1. Start Condition

The start condition of the program was used to determine the presence of a bird on the platform. In the current system, the value on the weighing balance was used to determine whether a bird was on the platform or not. Since the birds that were being

tested would be around the range of 120-200g, a threshold of 50g was set as the weight where any weight reading over 50g would start the program. A LabVIEW function was written to fulfill this purpose.

The function was done by first acquiring the weight before passing it into the Start condition function, as seen in Figure 8.15. The string input of the weight would then be passed into a python node, where the python node would search for a python file, check_weight.py, within its current directory. After which, the python node starts up the python environment, in this case v3.6.0, and runs the python script, as seen in Figure 8.17. The python script would then determine whether the condition was met and return a boolean, true or false value, as seen in Figure 8.15. If the output was true, the program will start.

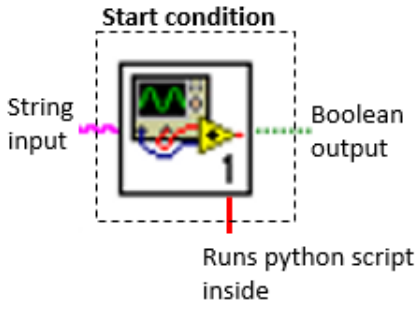


Figure 8.15: Start Condition Function

```
1
2  def main(str_val):
3      threshold = 50 #in grams
4      weight = float(str_val)
5      if (weight > threshold):
6          return True
7
8      return False
```

Figure 8.16: Python Snippet of Start Condition Function

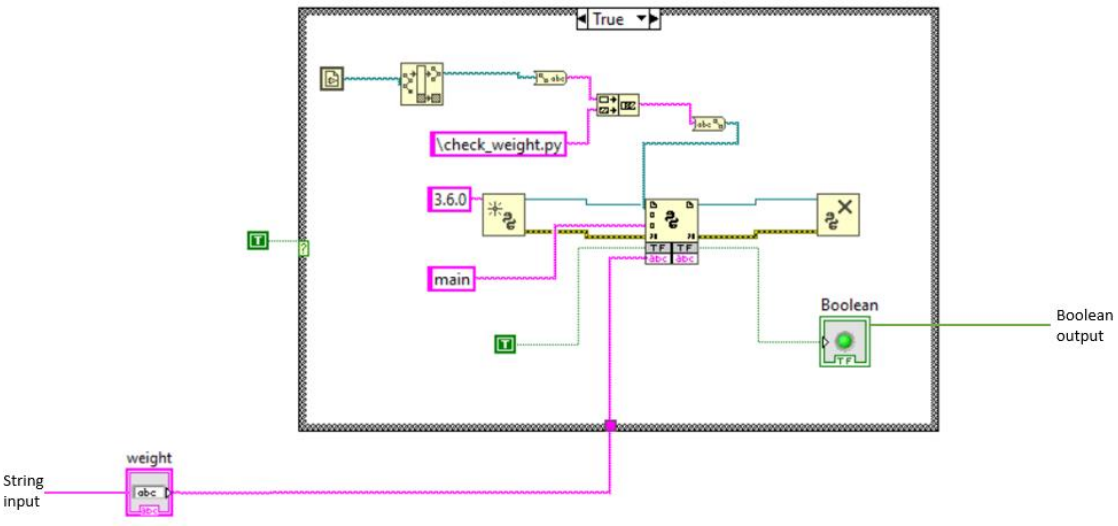


Figure 8.17: Deeper Analysis of Start Condition Function in LabVIEW

8.2.2. Weight Acquisition

The weight acquisition function was used to obtain the weight value from the weighing balance through a serial connection, as seen from Figure 8.18. It was done by first using the leveraging on inbuilt LabVIEW functions which allowed for serial reading of as bytes before obtaining a raw and unclean string data, as seen in Figure 8.20. After obtaining this unclean weight data, it was then passed into a python node, which happens similar to Figure 8.17 (explained in 8.2.1.). The python code was then written to clean the incoming data, by removing away unnecessary characters like "=" and whitespace, before obtaining the maximum weight value within the reading window, as seen in Figure 8.19. This process of data cleaning allowed the function to return a cleaned and correct weight data, instead of gibberish. This weight value would then be passed further down the pipeline for data storage.

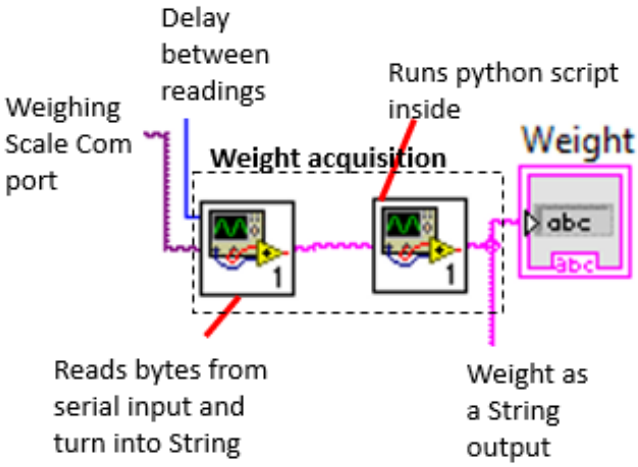


Figure 8.18: Weight Acquisition Function

```python
import re

def main(str_val):

    textlist = re.split('=',str_val)
    textlist = list(filter(None, textlist))
    textlist = list(map(float, textlist))
    text = max(textlist)
    text = str(text)
    return text
```

Figure 8.19: Python Snippet of Weight Acquisition Function

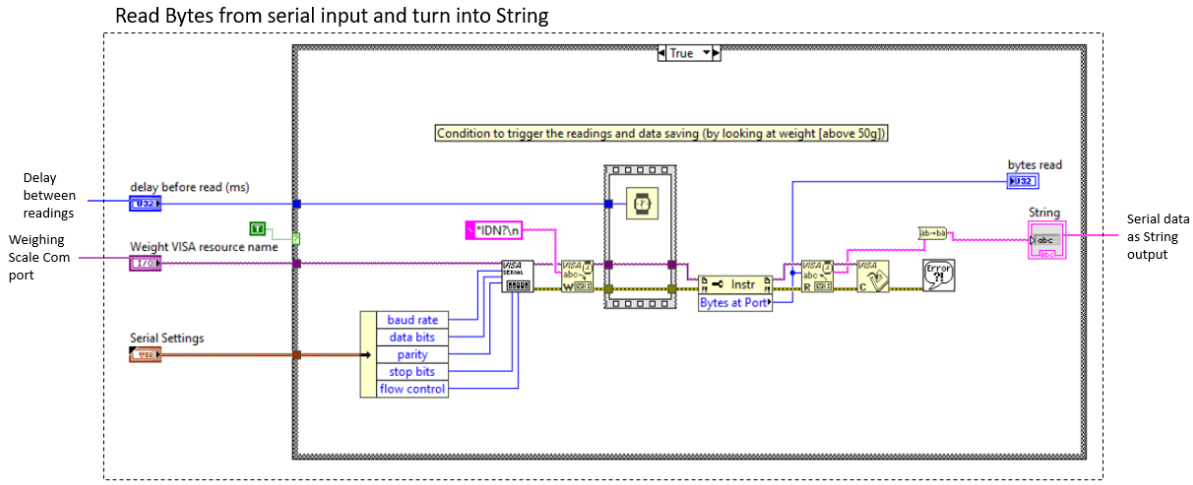


Figure 8.20: Deeper Analysis of Read Serial Function in LabVIEW

8.2.3. RFID Acquisition

The RFID acquisition function was used to obtain the RFID tag number of the birds so that it could be used to identify the individual. The function was started when the start condition is true. It was done by first reading the serial input of the RFID from the com port and cleaning it to get a string, similar to what was done above. Since 2 RFID readers were utilised, the function allowed for 2 possible RFIDs to be read. When the start condition was true, the function would check whether a particular bird's RFID was detected by checking whether the string length was more than 7 or not, where having a string length of more than 7 means that an RFID is detected, since RFID lengths' are typically 7 or more. When either one of the RFID tags was detected, a Boolean value is sent to determine which of the 2 RFID reader it should get the value from. This process would allow the function to return the correct RFID tag if it detected one and return an empty string if no RFID was detected. This RFID tag would then be passed further down the pipeline for data storage, as seen in Figure 8.21.

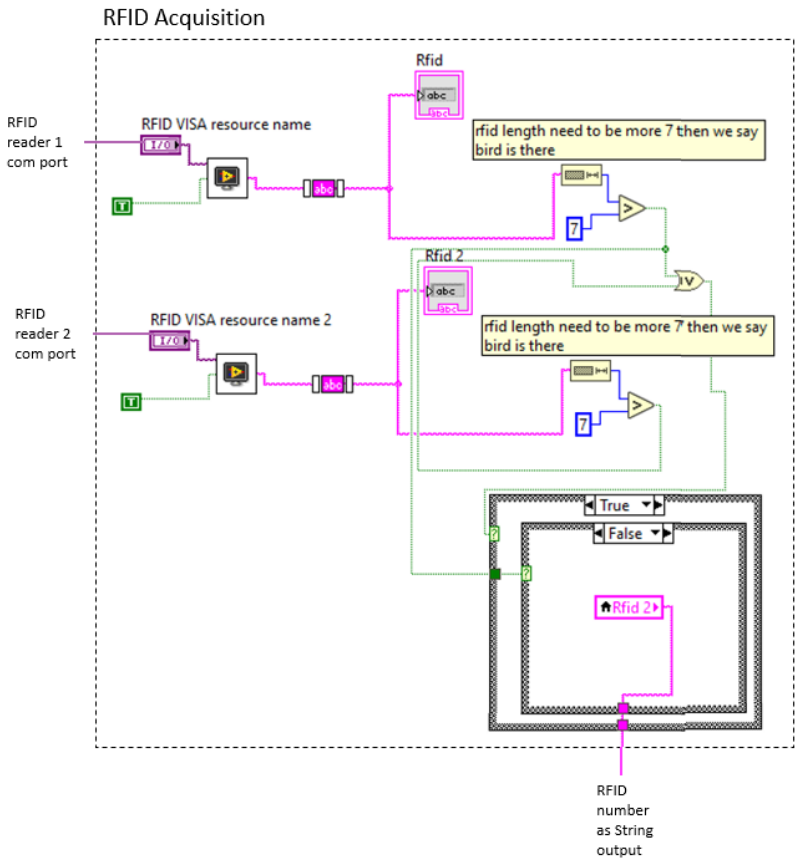


Figure 8.21: RFID Acquisition Function

8.2.4. Image Acquisition

The image acquisition function was used to capture a PNG image and store it at an appropriate location. The function was started when the start condition was true. It was done by first accepting in the webcam com port to get access to the webcam resource. Then, the IMAQ display image function, provided by LabVIEW, was used to turn on the webcam resource and take a single picture. The image was then stored temporarily in a cache before the webcam resource was turned off again. After which, a function was written to save the cached imaged permanently into the image directory. The function assigns the image file name based on the current date and time, before looking for the image directory to store the file as a PNG, as seen in Figure 8.23. This image file name would then be passed further down the pipeline for data storage, as seen in Figure 8.22.

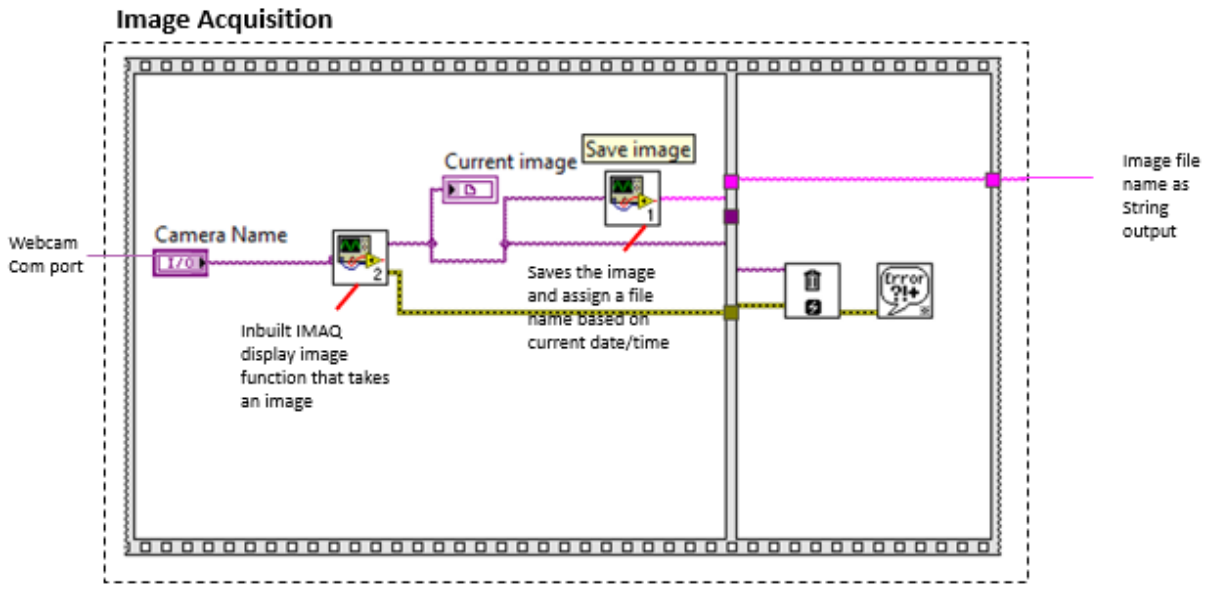


Figure 8.22: Image Acquisition Function

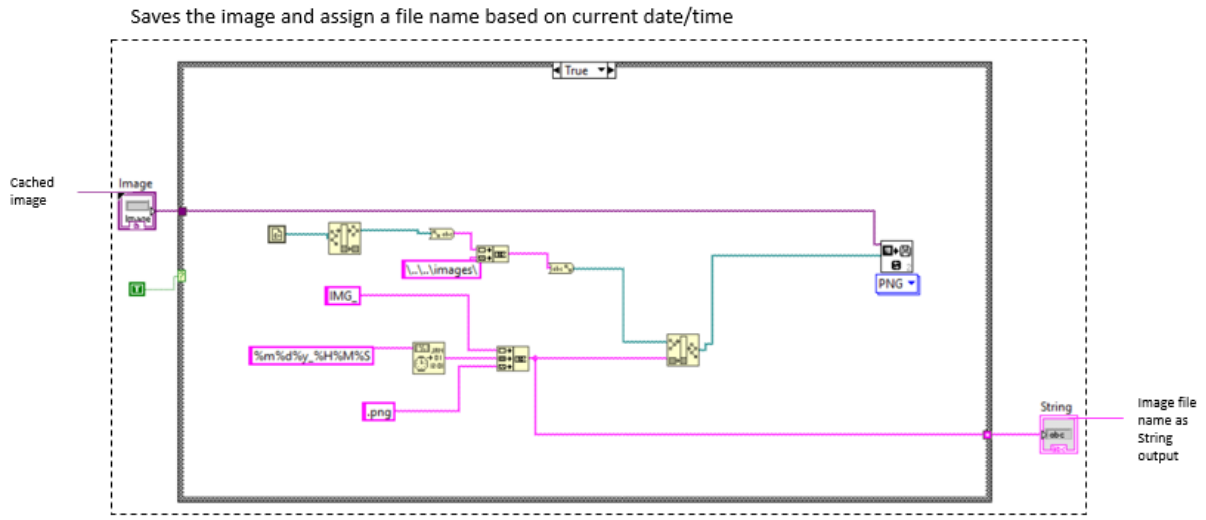


Figure 8.23: Deeper Analysis of Image File Name Assignment and Image Storage Function

8.2.5. Data Storage

The data storage function was used to collate and store the collected data into an appropriate location. In the current build of the system, the function collects the current datetime, weight, RFID number and image file name to be stored in comma-separated values (csv) file. The file type was chosen as such because it could be easily manipulated in Microsoft Excel, which the bird keepers have experience in. This would allow the bird keepers to peruse and manipulate the collected data easily without specialised

training. The function was started when the 3 data (weight, RFID, image) collection functions have been completed, which are accepted as string inputs, as seen in Figure 8.24.

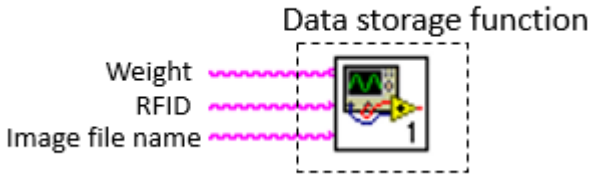


Figure 8.24: Data Storage Function

The data storage function leveraged on the 3 functions (that was coded as mentioned in the previous subsections) to obtain the weight, RFID and image data, after which, a python script was used to prepare the data and create the files for proper storage. The python script was activated in a similar way to Figure 8.17 (explained in 9.2.1.). In this function, the logic was done mostly in python as it allowed for greater micro level controls, where the essential data, file location, file type and styling can be controlled.

The aim of the function was to create a file for each day the data was collected, which is stored in a folder/directory that indicates the year and month it was collected in. This could allow the bird keepers to sieve through pertaining information as they need them. As seen in Figure 8.25, 4 major methods, create_dir(), compile_prev(), update_file() and main(), were created to achieve the outcome. Each method was written to fulfill a certain purpose to ensure principles of abstraction is followed, resulting in more effective and maintainable code.

```python
#This script aims to create folders at the target location every 1st day of the month (or any other day if there are no readings on 1st day)
#Files are created and updated everyday based on the readings (default values are blank), Data stored are time,rfid,weight,imageID
#At the start of the next month (during the 1st reading), all data from previous months will be collated into a single csv file
#End result would have monthly folders that contain daily files
import os
import time
from datetime import date
from datetime import datetime
from argparse import ArgumentParser
import pandas as pd

#every 1st day of the month, we want to create a new directory
def create_dir(today,dirpath): ...

#compile the previous month data into a graph
def compile_prev(today,dirpath): ...

#creates or updates the files everyday when there is a new reading
def update_file(dirpath, today, currentmonth, now, rfid, weight, imageID): ...

def main(rfid, weight, imageID): ...
```

Figure 8.25: Python Snippet of Data Storage Function

The create_dir() method was written to create a directory (for the current month) at the target location whenever a reading is received and the month's directory has not been created yet. This creates the folder where the individual daily files will be stored.

The update_file() method was written to create a csv file within the existing directory whenever a reading is received and the day's csv file has not been created yet. It then obtains the current datetime and consolidates it with the remaining 3 data, which are weight, RFID and image name. It then opens the csv file (for the current day) and, with this consolidated data, create a new entry to store the data.

The compile_prev() method was written to create a new file which consolidates a whole month's data. This method is called at the start of every month and consolidates the previous month's data into a single file. The method was created to allow birdkeepers' to get a monthly data review, to allow for better planning and higher-level view of the situation.

The main() method was written to call the 3 aforementioned methods and handle the python call from the LabVIEW, it accepts the 3 collected data, weight, RFID and image name, before obtaining the current datetime from the operating CPU and locates the target location for data storage.

With these 4 methods, the collected data were then able to be stored in an easy to access and manipulate location, the bird keepers. Seen in Figure 8.26, the result was

a folder that contains multiple monthly folders, with the daily files and image nested within it.

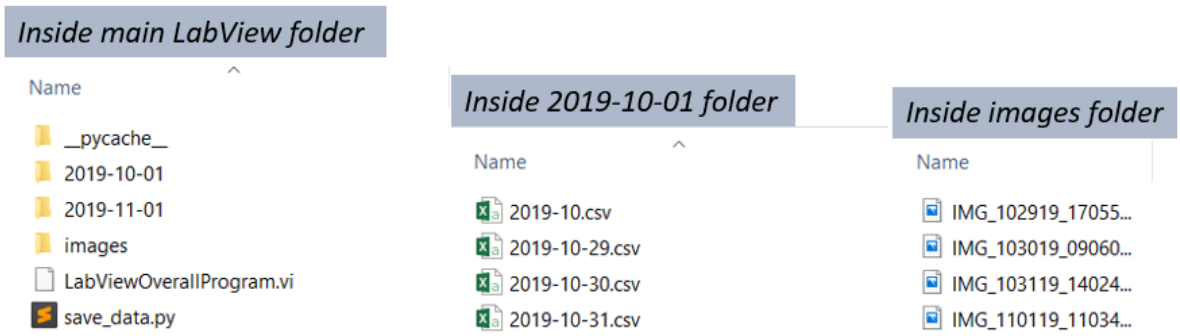


Figure 8.26: Example of Data Storage

8.2.6. Trigger Condition

The trigger condition function was written to determine whether a bird is injured or not. The function was started when the data storage has been completed. It was done by retrieving the RFID number and weight of the bird that was currently on the platform before checking it against a specified condition, as seen in Figure 8.27. The condition was written within a python script because it allowed for greater variation and control over the types of conditions that can be determined. In the current system, the condition was that when a bird is less than 60% of their average weight, the trigger condition would become true and the function would return true to trigger the trigger action. This was done by comparing the current bird's RFID and weight to a pre-existing file that contained all the birds' RFID numbers and their average weight using the RFID to differentiate the individual birds as seen in Figure 8.28. Although it could be any condition, and could still be changed, weight was chosen as the condition in the current system because birds tend to feed less and drop massively in weight when injured, which was shared by the stakeholders. The python node was then activated in a similar way to Figure 8.17 (explained in 9.2.1.).

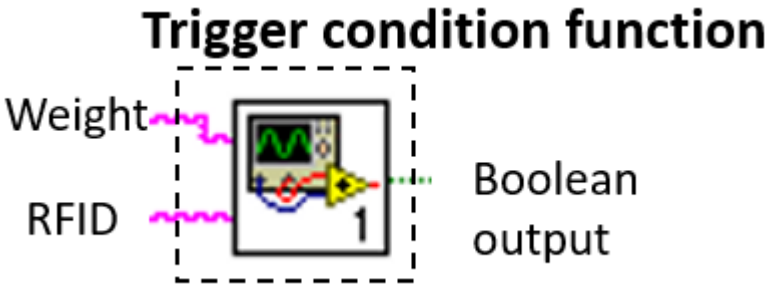


Figure 8.27: Trigger Condition Function

```python
# need to have "\AverageBird.csv" in current directory
import os
import csv

def main(bird_spec, weight):
    THRESHOLD = 0.6 #0.6 = 60% of expected weight
    bird_list = []
    weight_list = []
    avg_weight = 0
    dirpath = os.getcwd()
    f = dirpath + "\AverageBird.csv"
    with open(f) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        #reads the file to create 2 lists
        for row in csv_reader:
            bird_list.append(row[0])
            weight_list.append(row[1])

    if bird_spec in bird_list:
        avg_weight = weight_list[bird_list.index(bird_spec)]
    else
        avg_weight = 0
    avg_weight = float(avg_weight)
    weight = float(weight)
    if (weight < avg_weight * THRESHOLD):
        return True
    return False
```

Figure 8.28: Code Snippet of Trigger Condition Function

8.2.7. Trigger Action

The trigger action function was written to allow for remote trapping and notification for the bird keepers. The function was started when the Boolean input from the trigger condition was true which meant that an injured bird was in the cage and was skipped when the Boolean input was false, as seen in Figure 8.29.

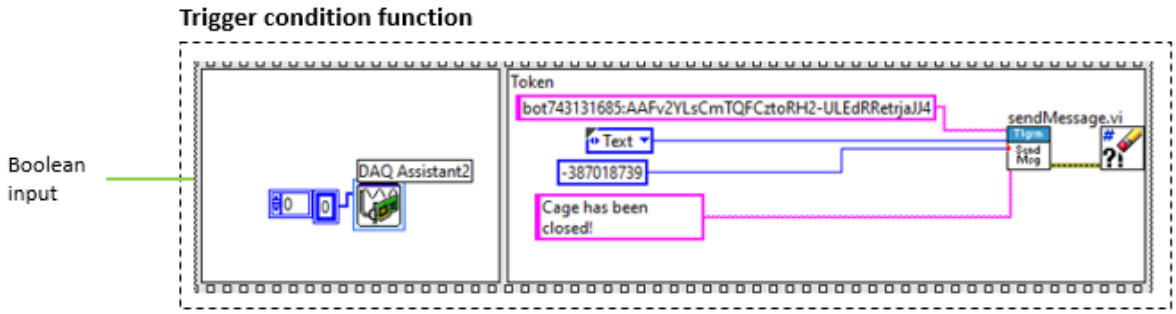


Figure 8.29: Trigger Action Function

To achieve remote trapping, a DAQ assistant within LabVIEW was utilised. This allowed for easy software control over the USB-6009 that was used to command the Arduino 4-channel relay switch, which in-turn was used as an electrical switch to keep the electromagnets powered on (explored in section 9.1.1.). By using the DAQ assistant to send a digital output of 0 to open the relay switch, the circuit for the electromagnet became an open circuit, killing the power to the electromagnet to allow the door to fall down due to gravity and trap the bird within the cage.

After which, the bird keepers needed to be notified so that they are able to tend the injured bird as quickly as possible. This was done using wireless communication coupled with an opensource LabVIEW API, telegram-bot-master, (Vadim, 2018) and a telegram bot, from the free BotFather developed by Telegram (Telegram, 2020). Telegram was chosen because they were the communication platform of choice by the birdkeepers and had an abundance of community support. In order to inform the bird keepers about the injured birds, the sendMessage method from the telegram-bot-master was used. The method was able to accept a telegram bot, through providing its bot ID as the token, the chat group id, and the message it needed to send. This allowed for easy dictation of who should be sending the notification, which telegram group should receive the notification and what types of message needed to be sent. In the current system, a

default telegram bot would send a “Cage has been closed!” message to the bird keepers telegram group immediately and stop the system.

By integrating the door control and wireless communication into the system, the bird keepers can be alerted of any injured birds immediately while requiring minimal interaction.

9. Deployment

To test the full system in an operational setting, the bird park correspondents were contacted to present the full prototype. This allowed the stakeholders to confirm whether the system could fit their needs before it was deployed. From the demonstration of the system, the bird park correspondents were then able to document a functional system that could be vetted by the crucial departments, such as the ethics department, in the bird park before testing. This round of vetting allowed for permission for animal testing and access to a test location at the Zoo, which worked in conjunction with the bird park. The permission allowed the bird park correspondents to capture wild common Mynas for testing, since show birds could not be used for testing.

While the birds were being prepared for testing, access was given to the test site to allow pre-testing of the setup and to gauge the placement and build of the system. Since there was limited time during each visit, help was acquired to help set-up the test environment. During the period, exposed electronics were properly sealed up to ensure that they were waterproof so that the bird keepers could routinely clean the cage easily while the system was set-up and inspected. More photos on the deployment trial can be found in Appendix D.

Figure 9.1: Setting Up at Test Site (in the Zoo)

Figure 9.2: Waterproofed RFID Reader (ID-12LA) with Weighting Scale

Before the birds were released into the cage where the system is held, a camera was also setup at the top of the test site to monitor the condition of the system and get a bigger overview of how well the system handled in a real-world scenario. On top of this since access to the test location was limited and only allowed when a bird park correspondent is with the team, which can be difficult to schedule, TeamViewer, a software application for remote control and file transfer, was used during the testing period so that the test computer could be accessed remotely to check and correct for any software issues.

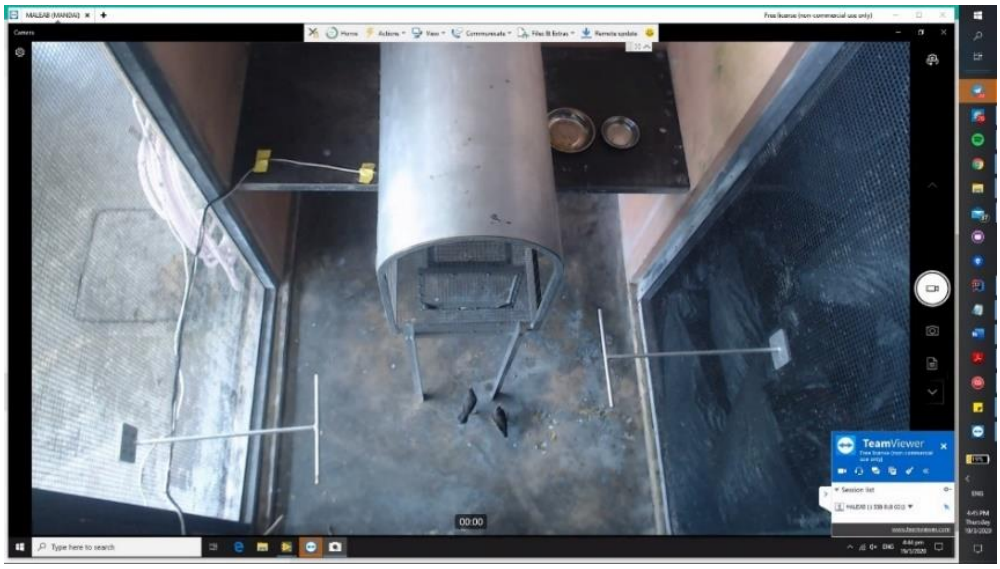


Figure 9.3: Remote Access to Check on System (with TeamViewer)

During the testing period, it could be seen that the system was working well for the most parts and data was constantly being collected, as seen from Figure 9.6. However, it was seen that there was no RFID information being collected, despite a weight, image and datetime information being collected and stored in the system.

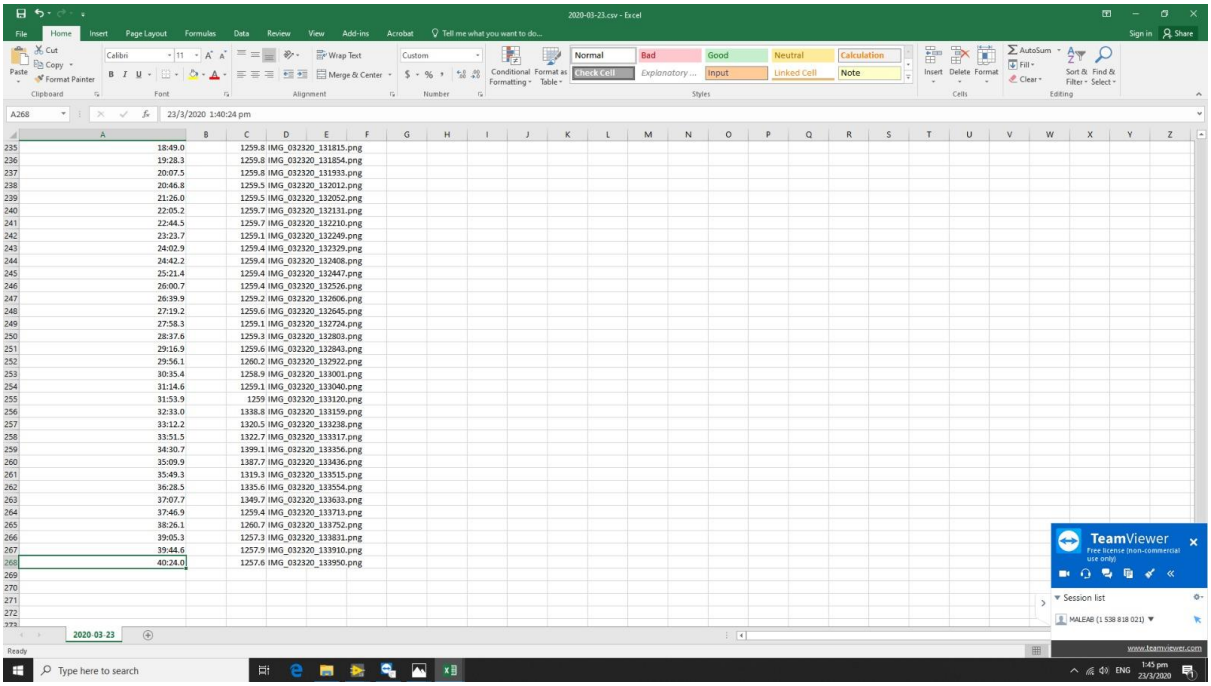


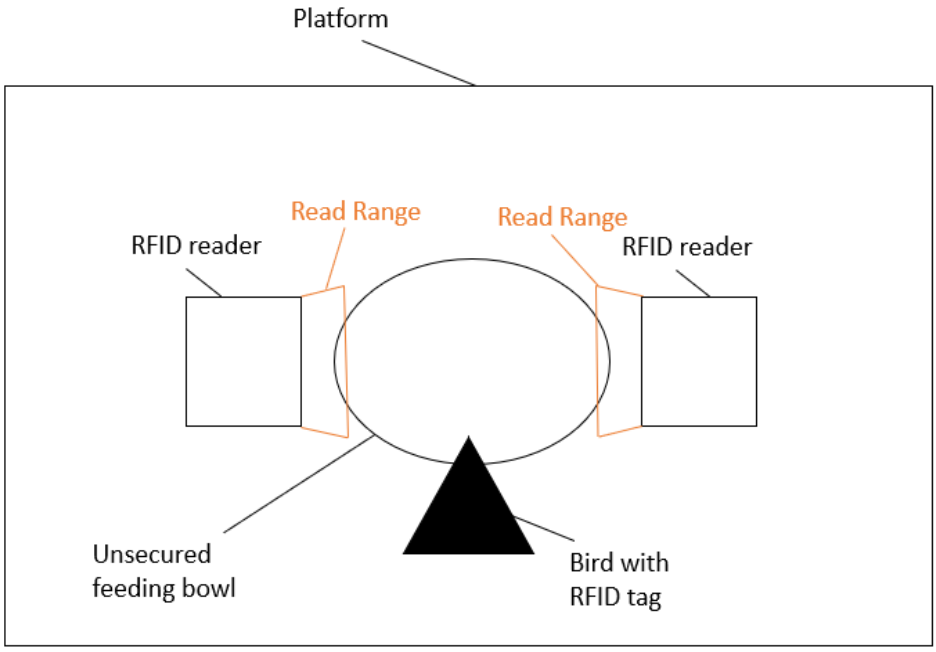
Figure 9.4: Issues where RFID was not read, despite datetime, weight and image being stored

10. Discussion

Based on the on-site testing, it was seen that the RFID issues was a problem and more information needed to be revealed. Thus, the images that were taken by the system were reviewed. As highlighted in Figure 10.2, it was discovered that the birds were never within the reading range of the RFID reader and the read range of the reader was not wide and far enough to capture the RFID transponders.

Figure 10.1: Baited Platform

Figure 10.2: Test Birds Feeding on Platform



Illustrations 10.1: Overview of Figure 10.2 where bird was never in RFID range

However, as the bird park correspondents mentioned that they were planning to use encrypted Trovan FDX-A RFID tags for future iterations, RFID issues were put on-hold while the bird park obtained their new supply of encrypted RFID tags and readers. The reason for this was that further tests on off-the-shelf RFID readers would be meaningless as they would eventually be replaced as the encrypted tags would require proprietary readers made by their manufacturers to be read (AZA, 2010). On top of this,

the target reader was also explained to have a further read distance, with an advertised read distance of 95mm and has a dimension of ~300mmL x 300mmW x 20mmH.

Although, the current testing period did not yield a positive data collection result for the RFID component, emphasis was made during design to ensure that the current system would be able to save the RFID number from any type of reader, as long as the transponder was within range and are decoded. Thus, the system could still obtain the RFID information smoothly if the bird park were to obtain proprietary Trovan decoders and antennas that could read the encrypted tags.

Apart from the RFID acquisition component, the decision to create generalisable code and easily customised system allowed the system to be resistant against changes while allowing for easy substitutions of components. With each component being separate from each other, different data could also still be collected when other components fail, as seen in Figure 9.4.

11. Future Work

Since the system had been designed with abstraction in mind, it was very modular and had no dependencies between irrelevant components. The components and features could be easily added to fit different customizations that the stakeholders might need as each component could be easily changed or substituted.

As alluded to in section 8.1.3., computer vision coupled with machine learning was an interesting topic that was discussed during meetings with stakeholders. The bird park correspondents, Nicole, Marc and Juan were excited to share about the potential use cases of computer vision being used across the attractions, Bird park and Zoo at the Mandai Nature Reserve. The captured images could potentially be used to determine animal species or individuals within a small area, by using the different colourations, patterns or eye scans, with the help of python libraries. It was mentioned by Marc that it

could help the keepers keep track of where dangerous species, such as Cassowaries or Tigers, typically are within the enclosure and avoid them without constant surveillance over the area.

Another potential work could be done to streamline data storage and retrieval processes by including a database and server to store them and a user interface, like a web application, to retrieve them. This could also be done by setting up a server on the CPU while simultaneously using python scripts to save data to the server using LabVIEW and retrieve data through requests from a user interface when needed. The user interface could also display statistics and macro level data. This would allow bird keepers to more easily review data, without even needing to handle the excel files.

Last but not least, another potential improvement was to miniaturise the components. Currently the computer takes up a large portion of the system and might require long cabling to deploy at more remote locations while keeping the computer safe. Since LabVIEW also has a Raspberry Pi compiler, a potential substitution for the CPU could be to replace the computer with a Raspberry Pi. Although this would result in slower computation and less potential for functionalities, a miniature system could be magnitudes more likely to be implemented in different scenarios.

Each of these potential works could provide a feasible way of customisation and solve the needs of the stakeholders in different scenarios, and thus, could be considered since the functionalities could be built upon the current system without much need for amendment of core logic and component designs.

12. Conclusion

A solid foundation and viable system had been built that is able to monitor birds' health remotely, which can vastly help improve the efficiency of the bird keepers and increase the welfare of the birds involved by minimizing their interaction. By being able to identify, feed, weigh and count birds that visit the system, it could also effectively alleviate menial but intensive job tasks that current bird keepers have to go through. On top of what has been done, the system has a lot of potential to continue to grow and adapted to different scenarios based on how the design was concepted.

By taking advantage of technology in design, human efficiency can be drastically improved while simultaneously increasing the welfare of animals involved. This can help vastly improve the animal industry by both reducing costs in manpower and making advances in the ethical department. As technology continues to grow, engineers should continue to find ways to integrate new and emerging technology to fill the gaps and limitations of current system which are lagging behind. This project showed the potential of applying technology within the bird park and the promise of expanding into the wider animal industry.

References

Urban Redevelopment Authority, URA SG. (2019, June 20) Draft Master Plan 2019, Regional Highlights. Retrieved from: https://www.ura.gov.sg/Corporate/Planning/Draft-Master-Plan-19/Regional-Highlights/North-Region

Siau Ming En (2019, August 29) Bird park, new rainforest to be added to Mandai wildlife attractions. Retrieved from: https://www.todayonline.com/singapore/new-bird-park-rainforest-park-mandai

Maho, Y. L., Karmann, H., Briot, D., Handrich, Y., Robin, J. P., Mioskowski, E., … Farni, J. (1992). Stress in birds due to routine handling and a technique to avoid it. American Journal of Physiology-Regulatory, Integrative and Comparative Physiology, 263(4). doi: 10.1152/ajpregu.1992.263.4.r775

Audrey Tan (2017, October 11) Bird Park to cover up show venue at new home to curb escapes. Retrieved from: https://www.straitstimes.com/singapore/environment/bird-park-to-cover-up-show-venue-at-new-home-to-curb-escapes

Wildlife Reserves Singapore, WRS (2019, July) Keeping Up with the Keepers S2. Retrieved from: https://www.facebook.com/wrs.sg/videos/761123327636569/?v=761123327636569

Corvid Isle Blog, TalkingBird (2019, January 31) First Aid for Birds – How to Catch a Poorly Bird. Retrieved from: https://corvid-isle.co.uk/first-aid-catch-poorly-bird

Association of Zoos & Aquariums, AZA (2010, March 31) Guidelines for Transponder Placement and Recording Retrieved from: https://assets.speakcdn.com/assets/2332/guidelines_for_transponder_placement_and_recording_aza_branded1.pdf

Álvarez, D., Lourenço, A., Oro, D., & Velo-Antón, G. (2015). Assessment of census (N) and effective population size (N e) reveals consistency of N e single-sample estimators and a high N e /N ratio in an urban and isolated population of fire salamanders. Conservation Genetics Resources, 7(3), 705–712. doi: 10.1007/s12686-015-0480-0

Ladik Vadim (2018) Open-source LabVIEW API for telegram bots. Retrieved from: https://github.com/ladikvadim/Telegram-Bot

Telegram (2020) Bots: An introduction for developers Retrieved from: https://core.telegram.org/bots

Appendix

Appendix A: Interviews

Nicole, bird park chief keeper

- Data collection:
    - Currently bird keepers write down bird conditions onto a form/paper and store them into an excel sheet from time to time (Noted that digital storage is not frequently done and there are very little data saved in this format)
- Health check and accounting:
    - Do not happen very often as it is difficult to gather the birds, especially smaller flying birds
    - Large or non-flying birds/ in enclosures are typically checked once a year
    - Injured birds are mostly captured and treated only when keepers chance upon physical defects or during their rare check-ups (small flying birds suffer in this case as they do not have routine check-ups)
    - Currently more valuable birds are tracked using the Trovan FDX-A RFID tags (which are also commonly used in zoos) but the read distance is extremely small (1-10cm) and reading is unreliable (Noted that they cannot change the tags)
    - Planning to incorporate the RFID tags to all birds in the bird park and planning to attach them as a leg ring (current birds with injected rfid tags are also being planned to be tagged at the legs)
    - Want to be able to identify and trap injured birds so veterinarians can attend to them quickly
- Feeding/ behaviour:

    - Different birds eat at different places and consume different types of food
    - Feeding trays as replenished 2 to 3 times a day and are washed at the end of every day
    - Feeding trays left on the ground are more likely to attract different types of insects and rodents
    - During breeding season, some birds tend to be more vocal and do different things/ have different calls
    - Birds shed feathers every few weeks and it is difficult to differentiate between individual birds within a species
    - When birds are sick or injured, they tend to eat less/ stop eating (which result in decrease in weight)
    - Feeding trays should be welcoming to the birds as some birds might not eat if it is scared (different birds could do with different designs to encourage/deter selected birds)
    - Birds tend to defecate anywhere (even places where they feed)
- Park activity:
    - Moving away from caged enclosure to walking/ large scale aviaries
    - Feeding trays should be hidden from the public
    - Activities are planned at different parts of the park, involving different birds, to help increase engagement and visibility of the birds

Glen Tan, Former Junior Nutritionist Assistant from Jurong Bird Park

- Mainly in charge of waterfall aviary
- Noted that there is not really much of a research team in the bird park

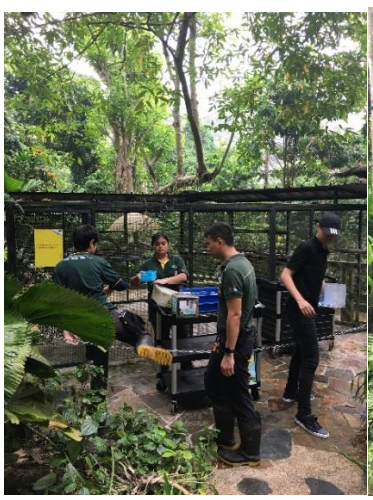
- Moving to new aviaries, will have little enclosures left (probably only those specific ones like penguins, etc)
- Current waterfall aviary would be similar good for comparison to how the new bird park aviaries are like
- There are no fixed feeding schedules/ methods, number of staffs assigned also varies (noted that probably can take one staff, but less staff means everything more rushed and might not have enough time and overflow to next day)
    - Logistics stuff are also included during feeding (like clean-up, maintenance, head counting)
    - Effort needed depends on type of aviaries (noted that penguins are easy to keep track of/ time the feeding, etc)
    - For bigger aviaries, like waterfall aviary, they just roughly head count (noted that we sometimes only see a bird once a week, sometimes don't even see for a month or more, so difficult to discern whether they dead or escaped)
- Different species of birds compete for space, so usually for some enclosures only one type of bird can be in it; if there are different species of birds in one enclosure, that compete for space, and given limited space they might fight (they fight for breeding/living space and also food present) (noted that some birds do not mind sharing same trees though)
- There are different locations in the aviary that they feed, and they can roughly track the some of the distinct clusters that specific species of birds live in but not all (some live throughout the aviary, some prefer to stay together in a small area)
- Birds can forage quite a distance from their nest, so it is hard to really pinpoint where they live (at most can guess and estimate)

- Different birds have different tastes (e.g for nuts or fruits) so he placed different food types during different timings at different areas (to reduce the need for birds to fight for space and can just occupy their own area)
- Keepers used to just feed in the morning buffet style but notice that it spoils very fast and once it spoils no birds will eat it, so now they dispense the food hourly which is a better system (keeps the birds eating well while minimising food wastage)
    - The hourly method is mainly for morning/ early afternoon as keepers have other things to do and also nearing knockoff time. [Placing food out takes around 10mins, then they will spend 50mins to do the other things in the aviary/preparing for next feeding]
- Sometimes difficult to keep track of birds because they breed and lay birds (sometimes hard to see and access the eggs, especially for small flying birds)
    - It is not good to let the eggs hatch unnecessarily because it will cause more fighting due to more birds in a small space (unless they need more of that specific bird)
    - Typically, the eggs can be given/traded to other zoos or used as feed for other animals (in the zoo)
- Smaller aviaries/ for aviaries where feeding clusters are closer or little tends to have more fighting between birds (when comparing waterfall aviary (larger) to wings of asia aviary (smaller))
    - This also results in less aggressive birds not getting near the feeding area to feed, even though it will be slightly easier to count the birds this way
- Normally cages are only used in aviaries to reintroduce/ quarantine specific birds

    - For wings of asia aviary, they have cages at the sides so that visitors can see specific birds (if not some birds are hard to find)
- Different aviaries have different places visitors can walk
    - For example, lory loft aviary has a specific path that visitors can walk by, which keepers also mainly use to feed the birds, so birds far away from that boardwalk are not examined and the close proximity of feeding also can cause increased fighting
- There are trap cages that keepers currently use to attempt to trap injured birds, but it is hard and may take multiple attempts to catch the injured bird, as other birds will also try to steal the bait

Appendix B: Park Visit

Waterfall Aviary:

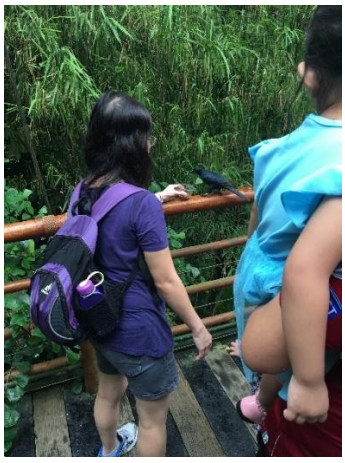


Picture 1: Bird introduction cage 1 *Picture 2: Trapping/feeding cage* *Picture 3: Free roaming bird with leg tag*

Picture 4: Current feeding system 1

Picture 5: Multiple birds on one tray

Picture 6: Meshed ceiling



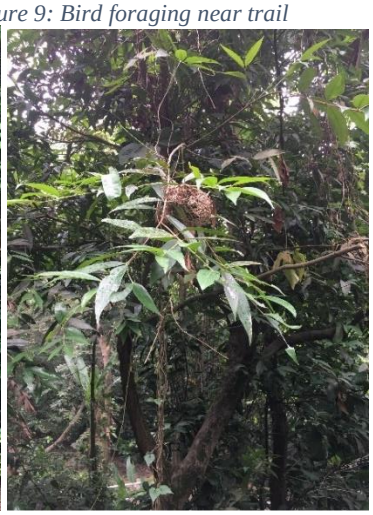
Picture 7: Feeding activity

Picture 8: Bird introduction cage 2

Picture 9: Bird foraging near trail

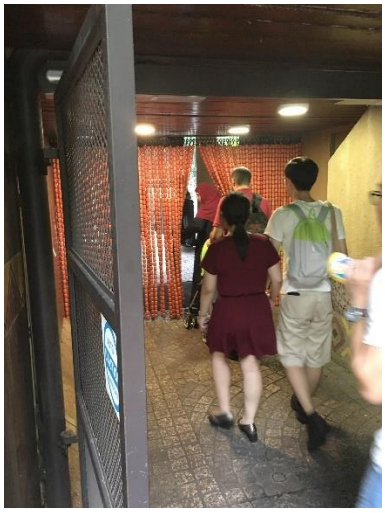
Picture 10: Different feeding trays near trail

Picture 11: Manmade nest box



Picture 12: Bird's nest

Picture 13: All bird species in the aviary



Picture 14: Double door at exit

Wings of Asia Aviary:

Picture 15: Types of bird species

Picture 16: Double door

Picture 17: Cages lined at sides

Picture 18: Feeding trays on ground

Picture 19: Feeding system

Picture 20: Feeding tray on ground

Picture 21: Feeding system

Picture 22: Trap/feeding cage

Picture 23: Bird's Nest

Picture 24: Suspended bird feeder

Picture 25: Bird group

Picture 26: New bird cage at corner

Picture 27: Camouflaged pillar

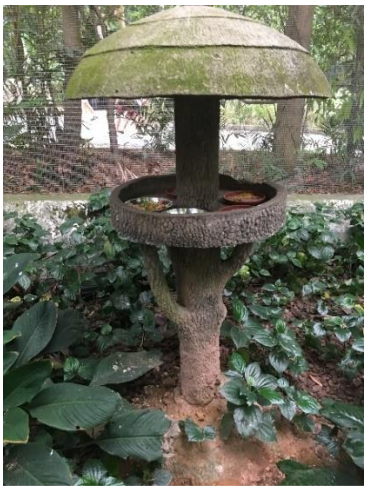
Picture 28: Feeder tossing feed on trail

Picture 29: Birds following feeder

Flight Aviary:

Picture 30: Weight curtain at door

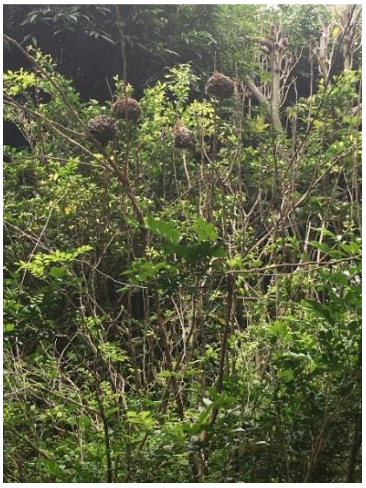
Picture 31: Feeding system



Picture 32: Feeding System

Picture 33: Small trap cage at corner

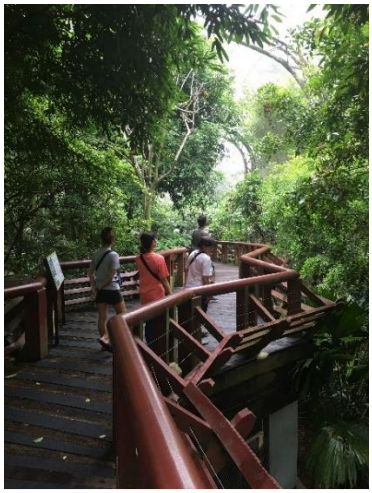
Picture 34: Feeding system



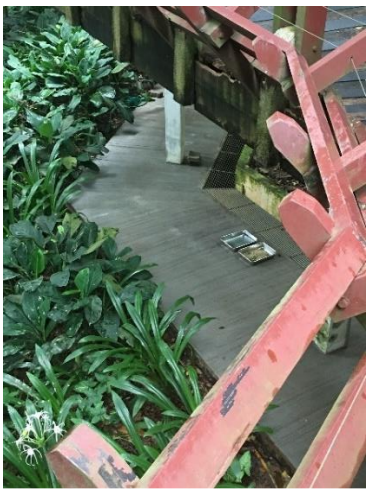
Picture 35: Bird's Nests

Picture 36: Birds group feeding

Picture 37: Trap cage



Picture 38: Boardwalk trail



Picture 39: Feeding trays

Picture 40: Birds in pond

Picture 41: Birds fighting on cage

Lory loft Aviary:

Picture 42: View from outside birds

Picture 43: Small trap cage

Picture 44: Tourist feeding

Picture 45: Feeding system

Picture 46: Boardwalk

Picture 47: Bird group feeding

Picture 48: Feeding trays under tourists

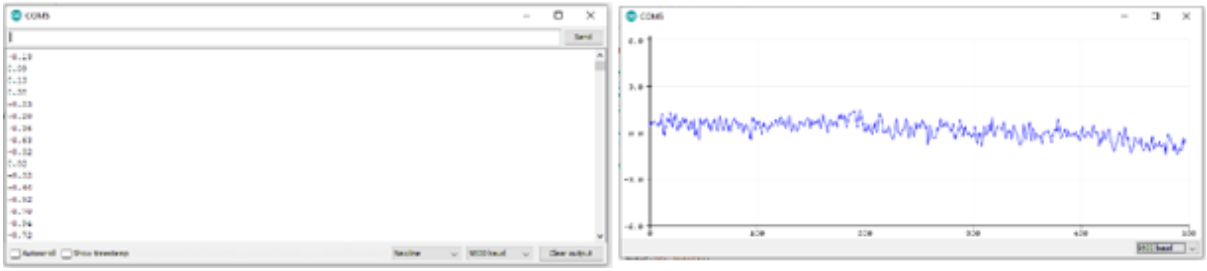
Picture 49: Birdbath

Picture 50: Trays near boardwalk

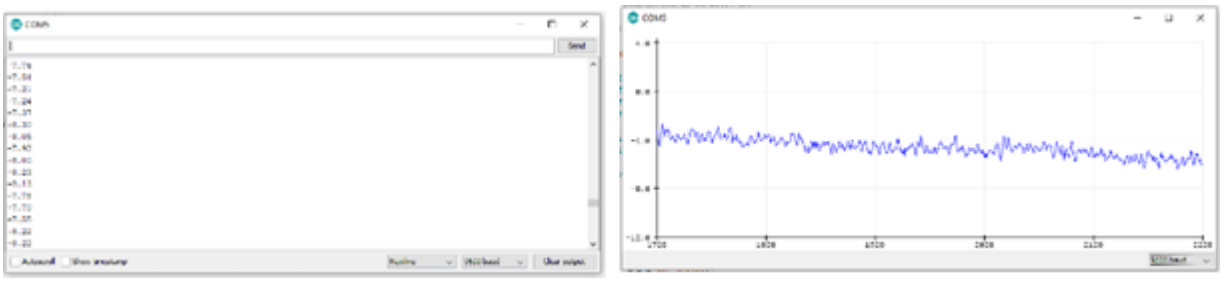
Appendix C: Load Cell Tests

```
#include "HX711.h"

#define calibration_factor 200.0 //This value is obtained using the SparkFun_HX711_Calibration sketch

#define DOUT  5
#define CLK  6

HX711 scale(5,6);

void setup() {
  Serial.begin(9600);
  //Serial.println("HX711 scale demo");

  scale.set_scale(calibration_factor); //This value is obtained by using the SparkFun_HX711_Calibration sketch
  scale.tare(); //Assuming there is no weight on the scale at start up, reset the scale to 0
  while(Serial.available()){
    Serial.read();
   }

  //Serial.println("Readings:");
}

void loop() {
  //Serial.print("Reading: ");
  Serial.println(scale.get_units(), 1); //scale.get_units() returns a float
  //Serial.print(" grams"); //You can change this to kg but you'll need to refactor the calibration_factor
  //Serial.println();
  delay(1000);
}
```

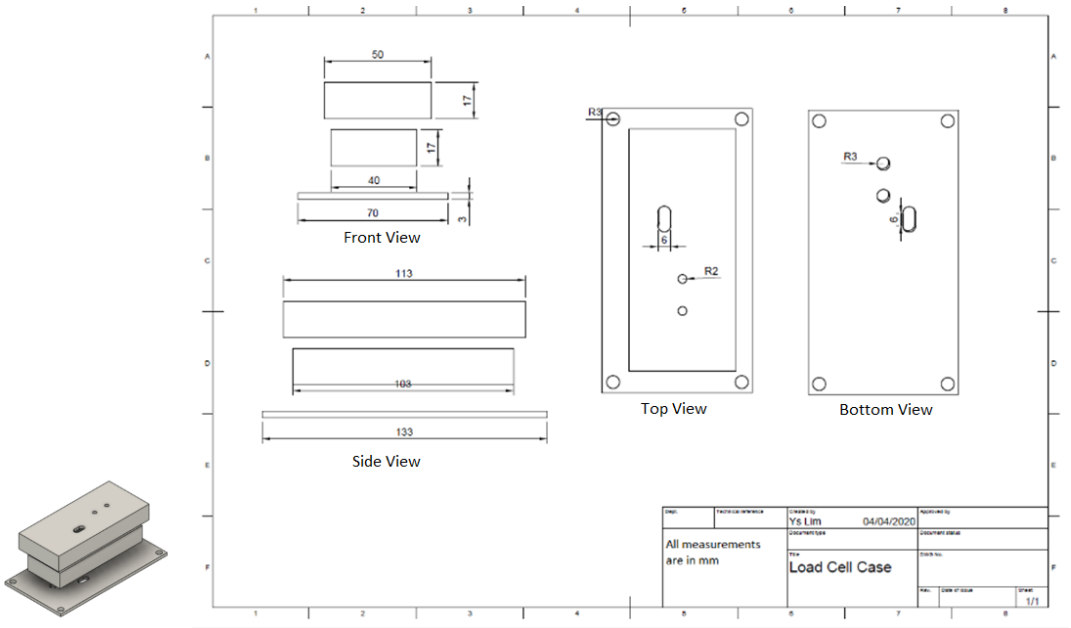
Picture 51: Arduino code with calibration factor tweaked to get accurate weight readings (in grams)



Picture 52: Hx711 reading at the start of experiment (with no weight): Range is from -1g to 1g



Picture 53: Hx711 reading after 30min from start (with no weight): Weight drift to -8g to -7g



Picture 54: CAD design for case to store load cell

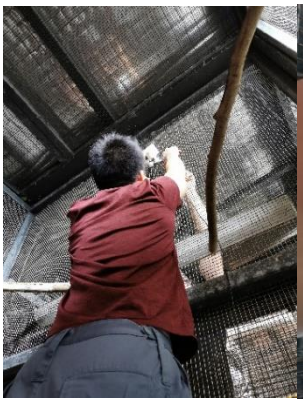
Appendix D: Deployment Pictures

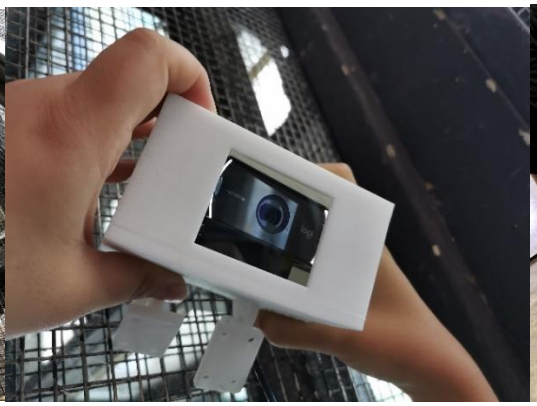
Picture 55: Deployment team picture

Picture 56: Software setup

Picture 57: Setup discussions

Picture 58: RFID setup





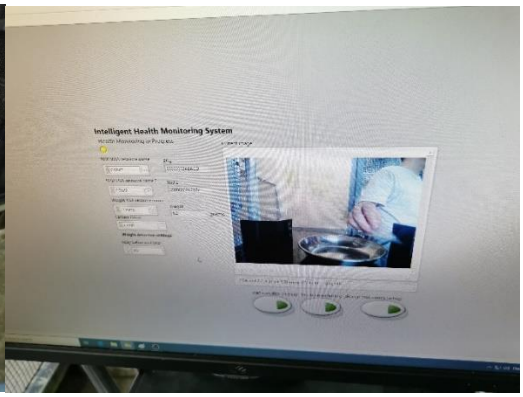
Picture 59: Setting up overview camera

Picture 60: Wiring

Picture 61: Cleaning components

Picture 62: Checking components

Picture 63: Initial system testing



Picture 64: Initial testing results

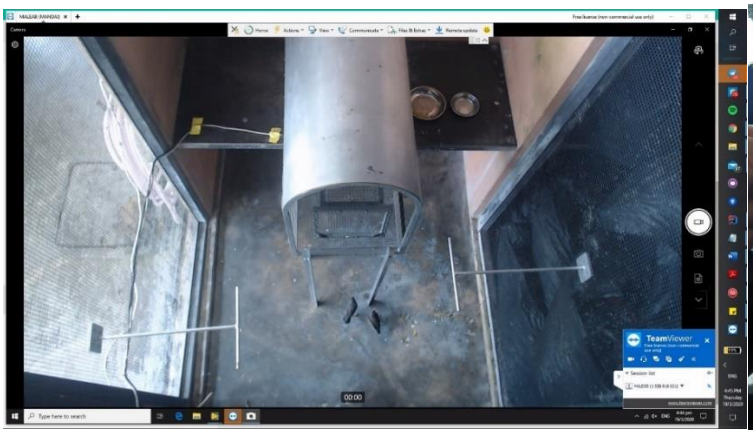
Picture 65: Bird tagging

Picture 66: Tagged bird 1

Picture 67: Bird tagging team

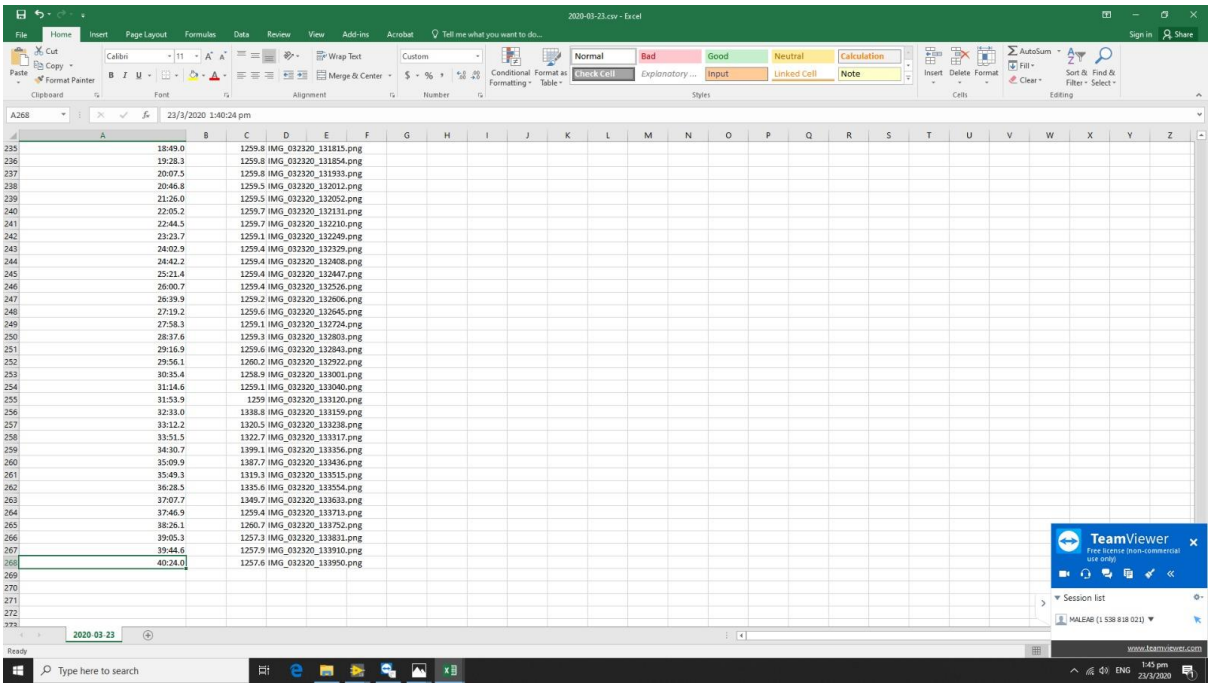
Picture 68: Platform with bait

Picture 69: Birds on platform



Picture 70: Remote access to check on system (with TeamViewer)

Picture 71: RFID checks



Picture 72: Issues where RFID was not read, despite datetime, weight and image being stored

Appendix E: Code

All codes used in the project can be found at:

https://github.com/limyongsong/fyp_software

The different systems (shown in the github link) allow for different configurations of the system, which were done with minimal code changes. Prototype versions of system were removed from the code for clarity.

Snippets that are important for explanations are revealed below.

```
#This script aims to create folders at the target location every 1st day of the month (or any other day if there are no readings on 1st day)
```

```
#Files are created and updated everyday based on the readings (default values are
blank), Data stored are time,rfid,weight,imageID
#At the start of the next month (during the 1st reading), all data from previous
months will be collated into a single csv file
#End result would have monthly folders that contain daily files
import os
import time
from datetime import date
from datetime import datetime
from argparse import ArgumentParser
import pandas as pd

#every 1st day of the month, we want to create a new directory
def create_dir(today,dirpath):
        newdir = dirpath+"\\"+str(today)
        if not (os.path.isdir("./"+str(today))): # if dir does not exist
                os.mkdir(newdir) #create new folder to store data for new month

#compile the previous month data into a graph
def compile_prev(today,dirpath):
        if (str(today)[-5:-3]=="01"): #check the current month, cause want collate
previous month data
                str_month = str(12)
        else:
                month = int(str(today)[-5:-3])-1 #change month to correct format
                if month<10:
                        str_month = "0"+str(month)
                else:
                        str_month = str(month)
        lastmonth = str(today)[0:5] + str_month + str(today)[-3:]
        if (os.path.isdir("./"+lastmonth)):
                flag = 0
                for filename in os.listdir(dirpath +"\\"+lastmonth): #compile all the
csv file of the prev month together
                        if filename.endswith(".csv") and flag == 0:
                                my_data = pd.read_csv(dirpath
+"\\"+lastmonth+"\\"+filename)
                                flag = 1
                        elif filename.endswith(".csv"):
                                my_data = my_data.append(pd.read_csv(dirpath
+"\\"+lastmonth+"\\"+filename),ignore_index=True,sort=False)
                my_data.to_csv(dirpath +"\\"+lastmonth+"\\"+ "compiled_"
+str_month+".csv",mode='w+')
        else:
                None #here means previous month folder does not exist

#creates or updates the files everyday when there is a new reading
def update_file(dirpath, today, currentmonth, now, rfid, weight, imageID):
        currentmonth_folder = str(today)[0:5] + currentmonth + "-01"
        if (os.path.isdir("./"+currentmonth_folder)): # if the folder exist then do stuff
```

```
                if not
(os.path.isfile("./"+currentmonth_folder+"\\"+str(today)+".csv")): #if file not yet
created for that day
                        f = open(currentmonth_folder+"\\"+str(today)+".csv", "w+")
#create new file
                        f.write("time,rfid,weight,imageID\n")
                else:
                        f = open(currentmonth_folder+"\\"+str(today)+".csv", "a+")
                f.write(str(now) + "," + rfid + "," + weight + "," + imageID + "\n")
        else:
                newdir = dirpath+"\\"+str(today)[0:5] + currentmonth + "-01"
                os.mkdir(newdir) #create new folder to store data for new month if
day 1 not created
                if not
(os.path.isfile("./"+currentmonth_folder+"\\"+str(today)+".csv")): #if file not yet
created for that day
                        f = open(currentmonth_folder+"\\"+str(today)+".csv", "w+")
#create new file
                        f.write("time,rfid,weight,imageID\n")
                else:
                        f = open(currentmonth_folder+"\\"+str(today)+".csv", "a+")
                f.write(str(now) + "," + rfid + "," + weight + "," + imageID + "\n")

def main(rfid, weight, imageID):
        dirpath = os.getcwd()
        today = date.today()
        now = datetime.now()#.strftime("%H:%M:%S")# add this is dont want date
        if (str(today)[-2:]=="01") and not (os.path.isdir("./"+str(today))): #first day
of month do stuff
                create_dir(today,dirpath)
                compile_prev(today,dirpath)
        currentmonth = str(today)[-5:-3]
        update_file(dirpath, today, currentmonth, now, rfid, weight, imageID)

if __name__ == "__main__":
        parser = ArgumentParser(description='Store data from labview')
        parser.add_argument('-rfid', dest='rfid', default='XXXX-XXXX-
XXXX',help='input RFID, e.g., 00AB-11CD-22EF')
        parser.add_argument('-weight', dest='weight', default='XX',help='input
weight in grams, e.g., 10')
        parser.add_argument('-imageID', dest='imageID',
default='XXXXXXXX',help='input imageID, i.e. where to get the image, e.g.,
abcd1234.jpg')
        args = parser.parse_args()
        main(args.rfid, args.weight, args.imageID)
```

Picture 73: Full python script for data storage function