

## ABSTRACT

Title of thesis: IMPROVEMENT AND ANALYSIS  
OF NETWORKED ANIMAL-BORNE  
SENSORS

Eli Lorenzi  
Master's of Science, 2018

Thesis directed by: Professor Nuno Martins  
Department of Electrical and Computer Engineering

Over the past few decades, advances in semiconductor technology have enabled the evolution of smaller and lighter embedded systems. Many researchers have utilized this technology to achieve new perspectives on animal behavior by developing animal-borne sensors and recording devices. Such devices have facilitated significant improvement in ecological research capabilities. However, there are many ways in which animal-borne sensor technology has yet to be harnessed.

The Networked Crittercam systems developed by the University of Maryland Institute for Systems Research in conjunction with the National Geographic Society offer a more sophisticated tool than previous animal-borne sensors. By engaging in real time analysis of sensor and telemetry data to determine when to trigger video recording, system designers can conserve precious battery life while maintaining the collection of pertinent data.

In order to best utilize the Networked Crittercam systems, it is necessary to understand the physical capabilities of the hardware and to develop software tools which augment the system. To achieve this goal, a total of 29 Crittercams were deployed onto two different species in Gorongosa National Park, Mozambique. Additionally, the systems underwent controlled tests to quantify performance metrics such as battery life and network connectivity. Lastly, a suite of software tools was developed in order to facilitate efficient and repeatable deployment efforts in the future.

IMPROVEMENT AND ANALYSIS OF NETWORKED  
ANIMAL-BORNE SENSORS

by

Eli Lorenzi

Thesis submitted to the Faculty of the Graduate School of the  
University of Maryland, College Park in partial fulfillment  
of the requirements for the degree of  
Master's of Science  
2018

Advisory Committee:  
Dr. Nuno Martins, Chair/Advisor  
Dr. Ankur Srivastava  
Dr. Anthony Ephremides

© Copyright by  
Eli Lorenzi  
2018

## Dedication

I would like to dedicate this thesis to my family, who has provided unlimited and unconditional love and support throughout my education.

## Acknowledgements

I would like to thank my advisor, Dr. Nuno Martins, my supervisor at National Geographic, Kyler Abernathy, and my predecessor on this project, Shinkyu Park. Each one of you has provided invaluable assistance to me throughout the course of this endeavor, and this document would not exist without your contributions.

I would also like to acknowledge the financial support of the National Science Foundation, who funded the Networked Crittercam project through award number 1135726.

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Objectives . . . . .	2
1.3	Outline of Thesis . . . . .	3
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	Similar Technologies . . . . .	5
2.2	System Design . . . . .	7
2.2.1	Hardware . . . . .	8
2.2.2	Software . . . . .	9
<b>3</b>	<b>Gorongosa 2016 Deployment</b>	<b>14</b>
3.1	Mission Preparation . . . . .	14
3.2	On-Site Deployment . . . . .	17
3.3	Data Collection and Analysis . . . . .	18
3.4	ArcGIS Development . . . . .	19
<b>4</b>	<b>Software Development</b>	<b>21</b>
4.1	Device Initialization Software . . . . .	21
4.2	Data Processing Interface . . . . .	24
<b>5</b>	<b>Device Performance Analysis</b>	<b>26</b>
5.1	Battery Life . . . . .	26
5.1.1	Empirical Model . . . . .	26
5.1.2	Deployment Results . . . . .	29
5.2	Radio Transmission Range . . . . .	31
5.2.1	Empirical Model . . . . .	31
5.2.2	Deployment Results . . . . .	34
<b>6</b>	<b>Conclusion</b>	<b>37</b>
6.1	Results . . . . .	37
6.2	Skills Acquired . . . . .	38
6.2.1	Python . . . . .	38
6.2.2	Java . . . . .	38
6.2.3	Software Development . . . . .	38
6.3	Future Work . . . . .	39

## Chapter 1: Introduction

### 1.1 Motivation

Until recently, continually observing most wild animals in their natural habitat has been an impractical task. Biologists and ecologists had to rely on secondhand data, such as fecal samples, to attempt to reconstruct the behavior of animals in unobserved environments [1]. Recording technologies have been used to create camera traps [2] and record encounters that occur within visual range of human observers, but these techniques are limited only to those locations that researchers can physically access. Additionally, the presence of a human observer in biological encounters can effect the subject's behavior limiting accurate insight into the animal's behavior [11]. To collect more useful data on animal behavior, researchers and conservationists require the subject's perspective on the world around it.

By having access to the animal's perspective, researchers gain numerous advantages. Most importantly, researchers gain access to areas that are either physically inaccessible or expensive to observe, such as under water or densely forested habitats. Furthermore, this new perspective enhances data collection by contextualizing animal behavior as a product of decisions. According to Wilmers et al, "answering the most interesting ecological questions will nearly always require fine-scale data on the ecological drivers of the behavioral and physiological measures that are being collected." [13] Having the first-person point of view not only provides information regarding what the animal did, but also what fine scale factors prompted the behavior. Whereas fecal samples can tell researches what plants an animal decided to consume, first-person video evidence allows the researchers to profile the vegetation suite from which the selection was made. Furthermore, this perspective can provide vital information regarding inter-animal interactions, unperturbed by human interference [3].

As advances in computing technology allow for the reduction in size of embedded devices, researchers can now achieve this more useful first-person perspective of animal behavior. There have been a variety of animal-borne recording systems created and deployed since the 1980's, including systems small enough to be deployed on domestic cats, as well as underwater systems [7][4]. However, the recording capacity of these devices is bottlenecked by battery consumption, as battery size and weight cannot be scaled in the same way microprocessors and cameras have been. In order to deploy these systems for longer periods, smart battery conservation techniques must be used, while also limiting loss of data collection capabilities.

The networked Crittercam systems attempt to solve the problem of battery capacity by using a microcontroller to determine the best times to record video. Since the battery cost of recording GPS data and communicating over radio is minute compared to the cost of recording video, the systems can afford to constantly be acquiring and analyzing sensor data. The motivation behind the Networked Crittercams is to create an animal-borne sensor that has the capability of capturing significant biological data over longer periods of time than systems that trigger recording off a timer. Additionally, these new, smarter systems can be used to deepen biological analysis, offering a side by side comparison of GPS, accelerometer, radio, and video data [12]. By offering features that no animal-borne monitoring systems have before, these Networked Crittercams mark a new step forward in biological research. The goal of this thesis is to quantitatively examine the viability of dynamically triggered animal-animal borne cameras.

## 1.2 Objectives

Having taken over research responsibilities on the Crittercams after the design and development of the systems, I was given two general objectives as a part of my thesis. Primarily, I was assigned to get the systems ready for deployment in Gorongosa National Park, Mozambique, in August 2016. Over the course of a month, I was tasked to gain an understanding of the systems' architecture, implement a small amount of code changes, initialize the devices with the appropriate deployment parameters, and test the devices for proper functionality. Once this was completed,



the devices were deployed onto twenty-five African Waterbuck and four African Water Buffalo by trained researchers and veterinarians.

Included also in the deployment process was the extraction of data from the devices. I was expected to format the raw data and distribute it to the appropriate recipients. Additionally, I was expected to perform some simple analysis in order to quantify the device performance for various National Science Foundation (NSF) reports.

I was also expected to make a variety of improvements to the Crittercam system, the associated software, and the deployment process in general. It was expected that I would design software solutions for programming the device parameters, as well as deployment data extraction. I was also asked to create a visualization of the deployment data collected in Gorongosa, to supplement NSF reports and to provide an intuitive starting point for any further ecological analysis. Lastly, I was asked to quantify some performance metrics for the devices such as radio transmission range and battery usage, as well as create a deployment guide to be used in future deployments.

### **1.3 Outline of Thesis**

The first portion of this thesis is meant to provide the background information necessary to contextualize the work completed on the Crittercam systems in the last two years. Motivations and objectives for the project have already been listed in this section, and previous iterations of animal-borne sensing technologies are listed in detail. Lastly, the contributions to the project completed before my involvement are listed to provide context for my own contributions. This includes a summary of the design and development methodology for the initial iteration of the Networked Crittercam systems.

Each of the next few chapters of the thesis correspond to the completion of explicit objectives for the Crittercam project. Chapter three outlines the process of deploying the devices onto animals in Gorongosa National Park, Mozambique in August 2016. This includes the process of learning the hardware and software structure of the devices, testing the devices for proper functionality, and extracting the pertinent data from the devices when the deployment was over. Chapter four

describes the software solutions created to increase usability of the Crittercam systems by non-programmers. Design methodologies for both the deployment programming interface and data extraction interface are discussed, as well as a small suite of additional python scripts used for data analysis. Chapter five outlines the attempt to quantify the performance of the Crittercam systems in two crucial areas, battery life and radio connectivity, in an effort to better inform future Crittercam deployments.

Lastly, chapter six summarizes the results of my time working with the Crittercams and attempts to analyze the viability of similar systems moving forward. This section also lists the skills that I acquired during my time working on the Crittercam project. This chapter also offers potential directions for future improvement to the systems and potential deployments.

## Chapter 2: Background

### 2.1 Similar Technologies

A number of conservation efforts have decided to utilize animal-borne technology to glean behavioral information from species. The Crittercam program at National Geographic has spearheaded many of these efforts, augmenting their systems with new technology in preparation for unique deployment environments. Standard on board Crittercam systems is a static temporally triggered recording system. However, to gather data in denied environments, it is often necessary to supplement the system with protective features or additional sensors. Since the inception of the program in 1987, Crittercam systems have been deployed on 80 unique, non-captive species on all 6 continents, and have included augmentations such as waterproof housings and infrared image capturing [5]. National Geographic's various partnerships with conservation efforts have established their Crittercam program as an eminent source of advancement among animal-borne technology and have allowed them to inquire into previously impregnable biological and ecological questions [13].

In addition to the Crittercam program, there are many scholarly groups using animal-borne technology to address ecological research topics. By obtaining data from previously inaccessible underwater locations, animal-borne sensors have been used to bolster research regarding oceanographic climate change. [9] Video capturing sensors have additionally been used to answer pressing questions regarding the unconventional diet of leatherback turtles. [6] In 2009, devices similar to the Networked Crittercams were deployed onto white tail deer in what was documented as "the first terrestrial, store-onboard AVED (Animal-borne video and environmental data) collection sys-

tem developed for large mammals.” [10] Lastly, in 2014, cameras mounted on various species of falcons were used to analyze the relative position of prey during pursuit [8].

The increasing usage of animal-borne sensors demonstrates the wide range of applications for such technology. Many different hardware systems have been developed to address the ever-growing list of ecological questions. However, these systems have largely taken a ”brute force” approach to collecting video data. Prior to the development of these devices, all CritterCam systems were programmed to trigger recording on regular intervals, independent of ecological context. This strategy is the standard operating procedure for most existing animal-borne recording systems.

Camera-traps, stationary devices that are commonly dynamically triggered using infrared or motion-based monitoring, demonstrate the utility of dynamically triggered recordings [2]. A variety of pertinent ecological data can be acquired through these devices, often at a much lower cost than using animal-borne sensors. However, the placement of camera-traps is predicated on the assumption that observers know where significant behavior is going to occur. This places a limit on the versatility of camera-traps as a data collection tool.

Post-deployment integration of video and sensor data has been demonstrated to be a useful data collection and verification technique. Cross-referencing sensor data with video recording not only allows for noise reduction, but also enables the collection of novel biological metrics. For instance, comparing accelerometer data and video recordings of penguins has allowed researchers to chart the number of krill captured versus the speed of thrusts of the penguins’ beak [12].

It is evident that advances in technology have enabled novel approaches to biological surveillance. However, there is still much room for improvement. None of the above systems operate on the collected data in real time in order to facilitate the more effective collection of significant data. This is where the Networked Crittercams represent a new paradigm in animal-animal borne technology. The Networked CritterCams use data collected by auxiliary sensors to make informed decisions about when to trigger recordings. Details regarding the implementation of this functionality are presented in the next section.

## 2.2 System Design

The inception of the Networked Crittercam project within the Institute of Systems Research at UMD was in 2012, roughly four years before my involvement in the project. To properly contextualize my contributions to the project, it is necessary to review the contributions of the previous investigators on the project, and the design of the systems at the time of my involvement. This is not meant to be a comprehensive description of the system components, but rather the background information necessary to understand the context of this thesis.

As stated at the outset of the NSF grant, the following are the design objectives for the networked Crittercam systems:

- Must acquire location and motion data as needed for post-deployment study as well as local real-time computations.
- Must share location and motion information via wireless communication for the operation of distributed algorithms.
- Must have sufficient processing power to complete computations required for autonomous decisions, including wireless sharing of data and recording of video.
- Must be compact and lightweight so as to not interfere with the animals' well-being or normal behavior (less than 3 percent body weight of animal).
- Must be durable for use on wild animals and suitable for quantity production using commercial contract manufacturers at reasonable cost.

In addition to meeting the requisite design requirements for the systems, the Crittercams were expected to optimize deployment length (or battery efficiency) while minimizing the amount of data loss, particularly the loss of useful video recordings. To meet these requirements, the system designers were required to implement customized hardware and software solutions, which are outlined below.

## 2.2.1 Hardware

The realization of this schematic in hardware requires multiple independent components, all communicating through an Arduino-based microcontroller board custom designed for these systems. The most up to date hardware design, as deployed in Gorongosa National Park in 2016, is shown in figure 2.1. A brief description of each hardware component included in the system is provided.

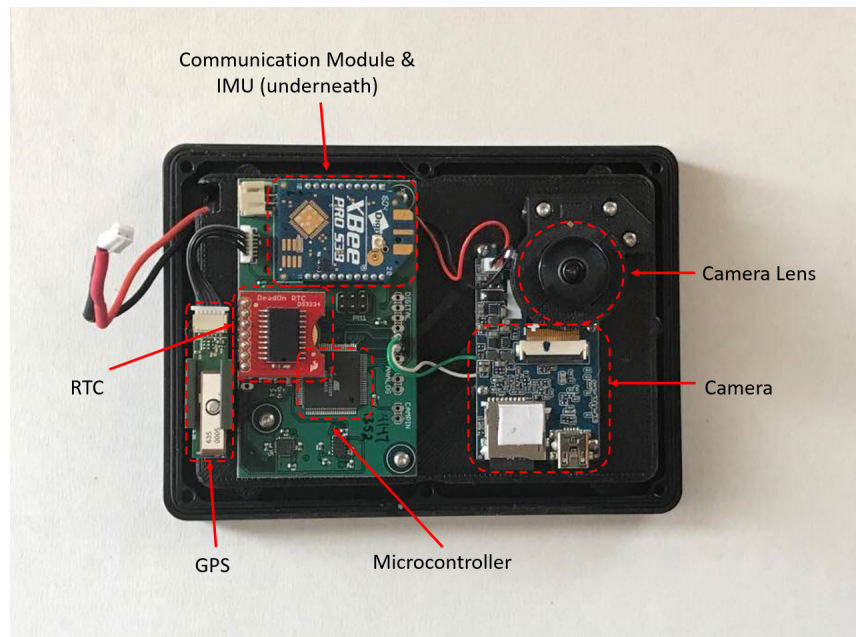


Figure 2.1: The hardware modules of the Networked CritterCam device

### Microcontroller Unit

The MCU is comprised of a custom fabricated circuit board housing an Arduino Mega Pro 2560V 3.3v processor, integrated with an inertial measurement unit (IMU) and augmented with the proper interfaces to the peripheral devices. The IMU includes an accelerometer, gyroscope, and compass, providing a comprehensive suite of orientation and control metrics to the system in real time.

## **Camera**

Video recording is done by the “Mobius Action Camera,” designed by the manufacturer to be used as a portable or dashboard camera. The housing has been removed to allow for direct access to the control mechanisms. The cameras can record video to auxiliary SD cards in up to 1080p quality.

## **Real Time Clock**

Timing capabilities are provided by Sparkfun breakout board DS3234, maintaining an on-board clock down to the second. The clock has an auxiliary battery module to keep the date and time up to date while the system is powered off.

## **Global Positioning System**

The GPS receiver module is also provided by Sparkfun GP-735 and is connected to the MCU via a serial port.

## **Radio**

Radio functionalities are facilitated by a network of XBee Pro XSC RF modules. The Xbees provide a “black box” implementation of the radio capabilities that can be controlled serially. The physical parameters of the Xbee modules can be adjusted separately from the rest of the system using free XCTU software. This design makes it simple to adjust RF parameters such as power and frequency without modifying the Crittercam firmware or writing any code.

### **2.2.2 Software**

The software structure for the Crittercam includes a large network of interconnected libraries, allowing each individual hardware module to contribute their data to the control algorithms. The big picture software structure and key algorithms are outlined below.

The backbone of the Crittercam firmware is coded in Arduino C. This code takes on the

familiar Arduino structure setup and loop blocks, the latter of which runs indefinitely while the system is on. The Arduino code references a suite of C++ libraries, which house the bulk of the firmware code. Each of the separate systems in the Crittercam is operated through a library of C++ functions that are called in the Arduino loop. A table of summary of the C++ libraries is provided in Figure 2.2.

Library Name	Description
CamController.cpp	Contains Routines for operating camera.
DataFilter.cpp	Contains routines for managing distributed location algorithm.
FSM.cpp	Contains routines for dealing with the device's activity based
RadioController.cpp	Assists with the construction of data packets.
Time.cpp	Handles time and date calculations.
TinyGPS.cpp	Contains routines for operating the GPS module.
Xbee.cpp	Interfaces the Arduino with the Xbee module.

Figure 2.2: Summary of software libraries present in Crittercam firmware

In the setup block, deployment parameters are uploaded from the SD card into the main memory of the system, and the initialization routines are run for each hardware unit. The loop block subsequently queries each subsystem, and if the designated time has elapsed since the subsystem last updated, an update routine is called. Each subsystem is updated at its own designated frequency, some of which are statically determined, while others may vary between deployments. A summary of the significant software routines utilized by the primary Arduino code is provided in figure 2.3.

### **Animal Activity Finite State Machine**

One of the primary means of battery conservation implemented in the Crittercams is the finite state machine (FSM) that limits certain functionalities when the animal occupies a “less active” state. This allows the camera to devote the most energy to data collection only at times when useful information can be cleaned. Transitions between states are triggered by increased



### Setup:

- Initialize system parameters from SD card.
- Run setup routines for each hardware module.

### Loop:

- IMU Routine (Every 20ms):
  - fetch IMU data
- FSM Update (Every 20ms):
  - update activity FSM based on IMU and radio data
- GPS Update (Frequency determined at deployment time):
  - acquire a non-noisy GPS fix
- Data filtering for Distributed Algorithm (If neighbors detected):
  - formulate and transmit data packet for distributed algorithm
- Radio Control Routine (Frequency varies depending on state):
  - send or listen for radio transmission as appropriate, based on timer
- Camera Update Routine (Once per minute):
  - determine if camera needs to be turned on or off, as determined by camera triggering algorithms
- Data Recording:
  - write the acquired data back to the database

Figure 2.3: Software structure of the Crittercam Arduino sketch

accelerometer and radio activity. A detailed depiction of the finite state machine mechanism is provided in Figure 2.4.

### Camera Triggering

The Crittercam firmware triggers the camera recording functionality when one of the following conditions are met:

- The system recognizes that it is a designated time of day that calls for camera triggering. These designated times are programmable and vary among deployments.
- The system recognizes that it is within a designated distance of another deployed device, which has been assigned a different species value from that system. The triggering distance is programmable and can vary among deployments, and species designated as “predators” can be assigned their own triggering distance.

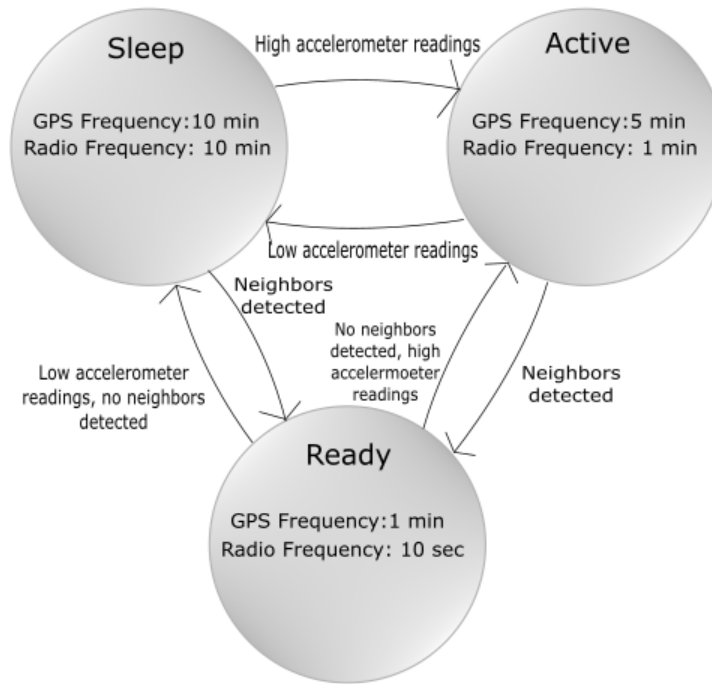


Figure 2.4: Finite State Machine representation of animal activity

- The system recognizes that it is within a designated distance of a geographic longitude/latitude pair. This distance and these points are hard coded into the Crittercam firmware.

These conditions are evaluated once per minute in the current firmware. There is an upper bound on the amount of time each of these conditions can trigger the camera daily, which can be programmed before deployment. Recordings are saved to an auxiliary SD card located on board the camera.

### Data Output

In addition to the video recordings, a significant amount of numerical data is collected by the array of sensors on board the Crittercam. Accelerometer data, GPS readings, radio transmission and reception records, battery voltage readings, and camera triggering records are recorded in binary format. After recovering the devices, these files can be translated into databases which can be queried in whatever manner is useful to research.

	datetime	node	lat	lon	hdop	speed	course	numsats
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	2016-08-15 2...	1	-18.97216033...	34.465454101...	4294967295	0.0514444410...	0	255
2	2016-08-15 2...	1	-18.97216033...	34.465454101...	367	0.0514444410...	0	4
3	2016-08-15 2...	1	-18.97217559...	34.465454101...	367	0.2057777643...	0	4
4	2016-08-15 2...	1	-18.97219276...	34.465454101...	367	0.0823111012...	0	4
5	2016-08-15 2...	1	-18.97220420...	34.465454101...	367	0.4012666344...	0	4
6	2016-08-15 2...	1	-18.97222709...	34.465465545...	366	0.2469333112...	0	4
7	2016-08-15 2...	1	-18.97222709...	34.465465545...	366	0.2469333112...	0	4
8	2016-08-15 2...	1	-18.97224044...	34.465473175...	366	0.3446777760...	0	4
9	2016-08-15 2...	1	-18.97224044...	34.465473175...	366	0.3446777760...	0	4
10	2016-08-15 2...	1	-18.97224807...	34.465473175...	366	0.2520777583...	0	4
11	2016-08-15 2...	1	-18.97224807...	34.465473175...	366	0.2520777583...	0	4
12	2016-08-15 2...	1	-18.97224998...	34.465473175...	366	0.3035221993...	0	4
13	2016-08-15 2...	1	-18.97223281...	34.465446472...	366	0.3909777402...	0	4

Figure 2.5: Format of the data collected by Crittercam systems

## Chapter 3: Gorongosa 2016 Deployment

After the development requirements were met for the Networked Crittercams, it was necessary to demonstrate their functionality in large-scale deployments onto multiple species. Deployments of this kind occurred in 2015 and 2016, in Gorongosa National Park, Mozambique, using African Waterbuck and Buffalo as subjects. I was directly involved as the lead engineer on the second and larger of the two deployments. Below I will outline my contributions to the deployment before, during, and after the actual deployment period.

### 3.1 Mission Preparation

My involvement in the Crittercam program began just one month before the scheduled deployment date. Shinkyu Park, the previous lead engineer left me with documentation detailing the system design and the deployment process, which were my primary resources for mission preparation. I was made aware of the expectations for the deployment in a meeting with Principal Investigator Kyler Abernathy. I was expected to add a “Geographic Location Triggering” functionality to the device firmware, test this new functionality along with all the previous ones, and prepare thirty updated devices for use in Gorongosa.

The “Geographic Location Triggering” functionality was designed to accompany PhD candidate Jen Guyton with her research on the African Waterbuck grazing habits. “Steady plots” of specific vegetation were planted throughout the park, and what the Waterbuck chose to eat from within these plots was of interest. It was therefore necessary to trigger video recording at times when the animals were likely grazing in the “steady plots.” It was decided that if the device detected that the animal had spent five consecutive minutes within fifty meters of the geographic

center of the plot then the camera should turn on. Additionally, the device should not trigger again unless the device detects that the animal has left the triggering radius and re-entered it.

This functionality was implemented in the CamController C++ library via a routine that is called at a frequency of once per minute under the current device configurations. The following pseudo code outlines the implementation of the “Geographic Location Triggering.”

---

```

lats ← [triggering latitude 1, triggering latitude 2...triggering latitude n]
lons ← [triggering longitude 1, triggering longitude 2...triggering longitude n]
recents = (0,0,0,0,0,0)
geodata ← most recent gps reading
found ← 0
for i = 0 ← n do
  distance ← pythagorean distance between (geodata,(lats[i],lons[i]))
  if distance < maximum trigger distance then
    found = 1
    for j = 5 ← 1 do
      recents[j] ← recents[j - 1]
      recents[0] ← 1
    end for
    break;
  end if
end for
if found = 0 then
  for j = 5 ← 1 do
    recents[j] ← recents[j - 1]
    recents[0] ← 0
  end for
end if
if recents = (1,1,1,1,1,0) then
  turn on camera
  return()
end if

```

---

Figure 3.1: Pseudo code for camera triggering algorithm

The result of this implementation is a finite state machine, consisting of incremental states, leading to the eventual triggering of the camera or reset to the initial state. It is trivial to extend or reduce this model to include any number of incremental states. The finite state diagram is pictured in Figure 3.2

Although the “Geographic Location Triggering” was designed to meet a very specific need, the principle can extend to a wide range of scenarios, and can provide useful biological data in a variety of deployments. These potential scenarios are discussed in later sections.

Once the changes to the firmware had been made, I began the sequence of steps necessary

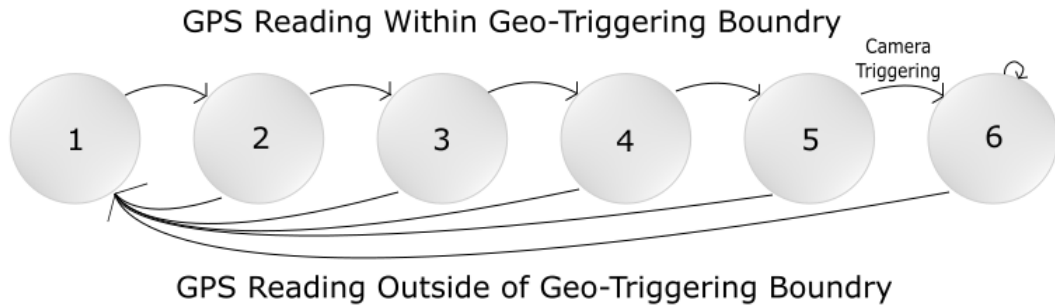


Figure 3.2: State diagram for camera triggering algorithm

to prepare each system for testing and eventual deployment. Each device underwent the following process:

- **Arduino Bootloader Upload:** After receiving the boards from the fabrication company, it was necessary to flash them with Arduino bootloader software, so the boards would be able to run Arduino sketches.
- **Calibration Parameters:** An accelerometer calibration sketch was uploaded to the boards that provided calibration parameters for each device. These parameters are then uploaded to the device in the “system parameters” binary.
- **Radio Initialization:** The Xbee modules were initialized with the correct parameters and settings for proper operation and communication with the Crittercam systems.
- **Clock Setting:** An Arduino sketch was used to initialize the clock for each device to GMT.
- **Deployment Parameters:** Deployment specific parameters were decided upon by the stakeholders, and were uploaded to the device via the “system parameters” binary file.
- **Firmware Upload:** The deployment firmware sketch was uploaded to the systems, finalizing the software based preparation for the deployment.

Once the devices were prepared, it was necessary to test the crucial functionalities of the devices to ensure proper behavior during the deployment. The following functionalities were tested

before deployment:

- Do the devices properly transition between activity states upon device acceleration?
- Do the devices trigger camera recordings at the preset times?
- Do the devices communicate over the radio network?
- Do the devices trigger camera recording when brought within proximity of a device assigned to a different species?
- Do the devices trigger camera recording exactly once when brought within proximity of a geographic triggering location for an extended duration?
- Do the devices write accurate output to the binary file?

The results of these tests were verified for each of the 29 devices before they were deemed ready for deployment.

## 3.2 On-Site Deployment

After arriving in Gorongosa, it was necessary to verify that the functionalities were still operational. During the 2015 deployment, the geographical change to the southern hemisphere had an unanticipated effect on the GPS readings, rendering the communication between devices ineffective. To avoid similar issues in 2016, select devices again completed the suite of tests listed above to verify that functionalities had not changed. Additionally, after consultation with Jen Guyton, the list of geographic triggering points for the “steady plots” were finalized and the firmware was edited appropriately. During this process, one battery became unusable, and the number of deployed devices was then reduced to 29.

Once the final changes were made and the devices were tested, the cases were sealed and attached to National Geographic’s custom designed animal collars. Each collar was uniquely labeled, and augmented with an RF beacon and a timer-based release mechanism. The collars

were set to release after the deployment period had passed, and then were located with their RF signals.

With the assistance of multiple parties, the devices were then deployed onto twenty-nine animals across different parts of the park. Four were applied to buffalo, eight were applied to waterbuck in the floodplain region, eight were applied in the salt-pan region, and nine were applied in the forest. The animals were tranquilized from a helicopter by trained veterinarians, and were quickly attended to by a team of researchers on the ground. In addition to taking many biological readings, the team would apply the collar to the animal before administering the de-tranquilizer, and allowing the animal to return to daily activity.



Figure 3.3: An African Waterbuck sporting a CritterCam collar

### 3.3 Data Collection and Analysis

The 2016 deployment in Gorongosa generated over 29 gigabytes of total data along with 170 hours of video. After regaining possession of the deployed devices, it was my job to translate the output data into a useful form and distribute it to the proper investigators. Additionally, it was my job to process the deployment data and provide a deeper level of understanding of the



deployment metrics. The details of the data processing methods are left for a later section, but most notably, GPS data was interpolated and sampled, and radio reception patterns were analyzed. This information was sent to Princeton as a part of their research into conservation efforts, and to National Geographic to be included in their NSF reports.

### 3.4 ArcGIS Development

Because numerical GPS data is not useful as a demonstration tool when presenting the deployment results to observers, it was necessary to create a visualization to provide context as to the scale of the deployment. ArcMap provides a robust framework for displaying geographic data including a multitude of features and documentation. Many types of ArcMap templates were considered for the visualization, but without a server with which to publish the more complex templates, it was decided that a simple interactive ArcMap file would be created, and that it could be embedded in photos and videos for the purposes of sharing results.

The output data was formatted such that ArcMap could create a time-labelled line features between each GPS datum. Data was visualized using the “time slider” tool, to create a small-scale time lapse representation of the deployment. The GPS data was sampled at an hourly rate to provide a detailed but not frenetic visualization of the animal activity. Buffalo and waterbuck were labelled with separate color schemes as well as separate patterns. Radio transmission data was also included on the map, again sampled so that radio communication between two devices would appear once per hour.

This map was augmented with video files from the 2016 deployment and included in the final NSF report submitted in November 2017. Software for formatting future deployment data for a similar ArcMap template was developed and saved, details on the code is included in the software development section.

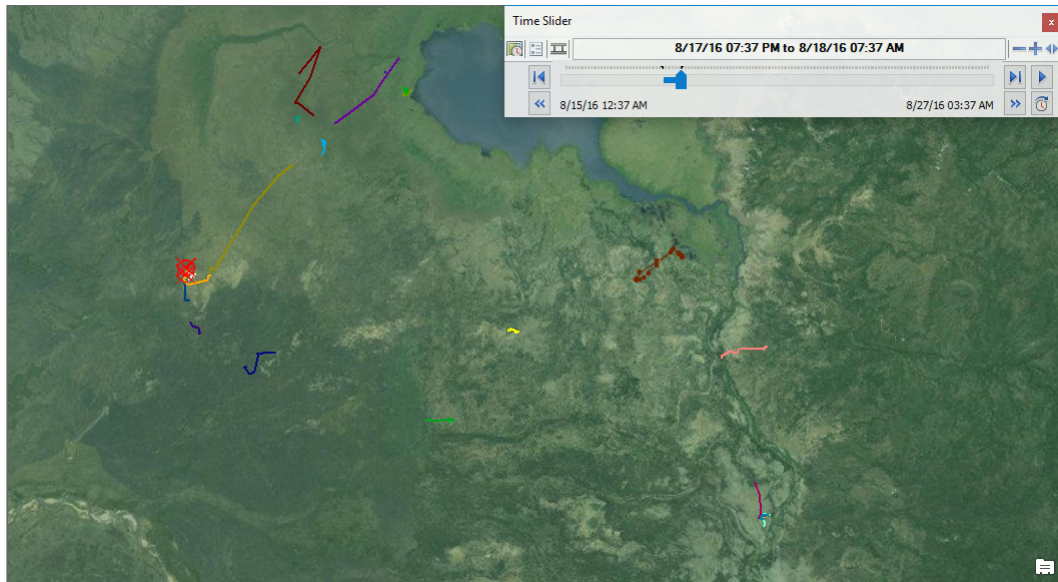


Figure 3.4: A snapshot of the ArcMap interface for visualizing deployment data

## Chapter 4: Software Development

### 4.1 Device Initialization Software

As previously mentioned, the Crittercam devices offer deployment initialization functionality so that it is not necessary to rewrite the firmware for each mission. The Crittercams read these deployment specific parameters from an SD card that is physically inserted into the device. While this is a convenient mechanism, it requires the user to be able to translate their deployment specific parameters into a binary file of the expected format. When I joined to the project, this file was generated using a templated Excel workbook as the user interface, with a visual basic script working in the background to convert the cell contents to a binary file.

The Excel interface offered a number of advantages, such as saving capabilities and a familiar user experience, but was lacking in many ways as well. Most notably, the Excel workbook required reformatting if the user wanted to change the number of devices to deploy or if the number of time based triggers was changed. If the user wanted to add a device, he or she would have to create an entire new worksheet from scratch. Furthermore, many of the parameter values are constant across all of the devices for a given deployment. In this case, it was desired that the user would only have to input this data once. Lastly, the Excel interface has no means of ensuring that the values entered by the user will not cause any unforeseen errors in the deployment parameters. It was decided that a new interface should be created, and I selected Java as the tool to create it with, because of its easy to use Swing interface. The development process for the interface mimicked closely the standard cycle of design, implementation, testing, and feedback. Intermediate versions of the interface were brought to my supervisor at National Geographic for testing and feedback,

each visit generating a new list of desired features for the interface. The final iteration of the device initialization software is described below.

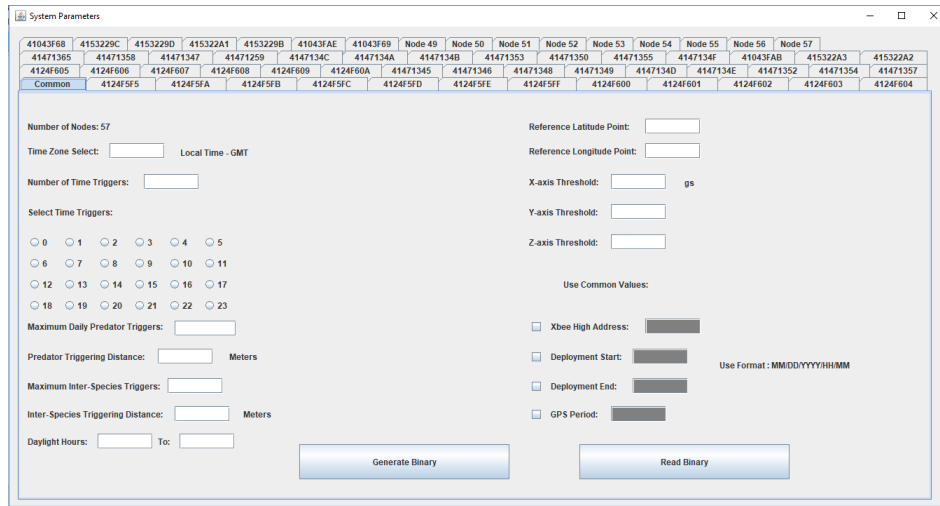


Figure 4.1: Parameter entry fields common to all devices in a deployment

The user interface consists of one tab for each deployable device, and one tab that contains inputs for parameters common to all devices. In its current iteration, there are nine blank tabs, left in case of the addition of more Crittercam devices to the current set. The first tab (the one for common values) includes inputs for parameters such as number of nodes, deployment time zone, and reference latitude and longitude points, among others. The fields that require units are appropriately labeled, and fields with a relatively small number of options are represented using radio buttons. In the bottom right corner of the first tab are the fields that are technically unique to each deployed device, but are often common, such as deployment start time and GPS frequency. In the case that the user wants to use a common value for all the devices, checking the box next to the corresponding value on the first tab will write the common value to all devices, and black out the corresponding entry field on all other tabs. This functionality greatly reduces the amount of time required to deploy a set of devices.

The rest of the tabs are labelled with the unique Xbee low address of their corresponding device, which is hard-coded into the binary generation. There is an entry field for each parameter unique to the individual devices, which are blacked out if a common tab is being used. The

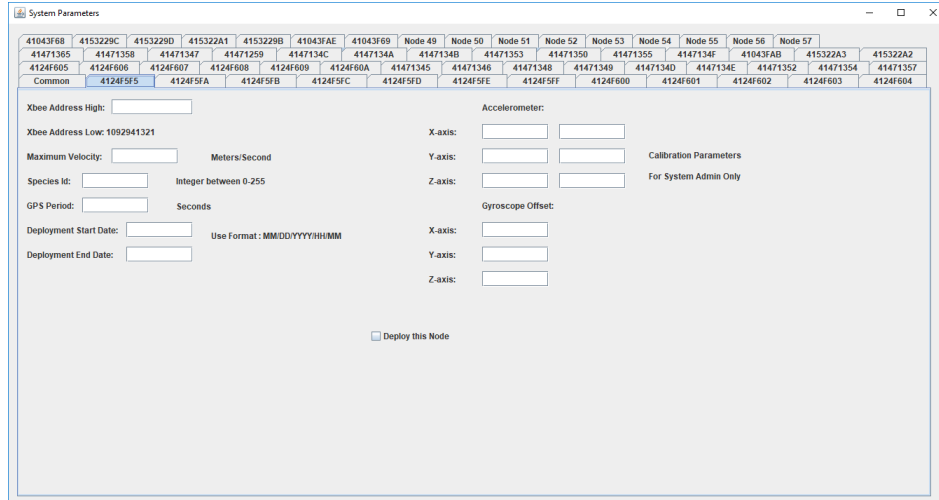


Figure 4.2: The deployment parameter fields for a single device

accelerometer calibration fields, however, do not require entry, and the calibration values for each device are hard coded into the file writing routine. The user can overwrite these values by entering values into the corresponding field, in which case the software will ask for a password upon making the binary file. Password protection is necessary because the calibration parameters for each device have already been determined, and only need to be changed upon the replacement of a device. There is a check box that should be selected for each node that the user plans on deploying. The new interface makes a number of checks to ensure that the values entered by the user will not cause any large errors in the deployment. Common pitfalls such as input values being out of range, or the deployment start time being after the deployment end time are checked when the file is created, and the user will be prompted to fix the error before the binary is written. I believe this to be the greatest benefit of the new interface over the excel workbook.

One apparent disadvantage of using a Java executable for the interface is its lack of memory capability. When the program is closed, the entries that the user made are lost. For this reason, it was necessary to include a “restore” functionality that could read in the values from a previous deployment, so the user does not need to start from scratch. This was included in the “Read Binary” button. Clicking this button prompts the user to select a binary file from their device that is parsed and rewritten into the proper text fields. Using this button allows the user to make

slight alterations to previous deployment parameters without rewriting all the values.

## 4.2 Data Processing Interface

Severely lacking from the capabilities of the Crittercam infrastructure when I joined the project was an easy way to extract the recorded data from the devices after deploying them. Although it may not always be of crucial importance to parse the output of the devices as quickly as possible, during testing, when the device may be “deployed” multiple times each day, reading the device output quickly can save the user a lot of time. Furthermore, the recipients of the Crittercam data want the data in a particular format or sampled at a particular rate, so having the ability to easily convert and sample data without writing a processing script from scratch is often useful. The output data from the deployment is written to the on board SD card in a binary format to conserve processing resources and space. Again, however, this requires the user to translate the data in order to access it. I was provided with a Python script that parses the binary file and writes the data to a database file, which was necessary in order to access any readable data from the device. From this starting point, a script for sampling the data and converting it to the more commonly used .csv format was created. In the end, a number of scripts were produced, corresponding to the varying needs of the data users. Most basically, one script provided strict conversion of any table from the database file to a .csv file, without any loss of data. Second, a script allowing the user to sample the GPS recordings at any desired rate and print the results to a .csv file was developed. Furthermore, a script that interpolated GPS data using a cubic spline was created, but to provide useful output to the user, the interpolated track required sampling, and the sampled data is written to a .csv file. The utility of the interpolation function will be demonstrated in the following section. Lastly, to facilitate the creation of further ArcMap displays, a script was written to generate the .csv files needed to create a “line” shapefile in the ArcMap software. Since running Python from the command line is not familiar to all of the Crittercam users, all of these Python scripts were packaged together into the backend of another Java Swing interface. This interface prompts the user to select the input files and enter the sampling rates,

constructs the proper command line arguments, and then runs the script behind the scenes. All that is required of the user is to have a working version of Python installed on their machine.

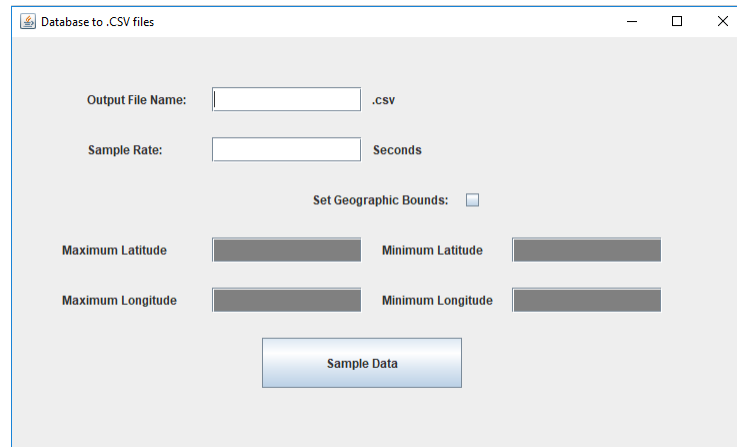


Figure 4.3: Screen shot of the database sampling mechanism

## Chapter 5: Device Performance Analysis

In order to properly assess the fitness of the Crittercam systems for a variety of potential future deployments, it is crucial to understand the physical limitations of the system in both constrained and unconstrained scenarios. The two primary system components in need of testing are the battery life and the radio network connectivity. In addition to the inherent testing the systems underwent in their deployments in Gorongosa, it is necessary to isolate and identify the performance of these components in a controlled experiment. Both components underwent such testing, and results were compared to the data collected during deployments.

### 5.1 Battery Life

#### 5.1.1 Empirical Model

##### Methodology

To accurately identify the expected battery life of the systems in future deployments with varying demands, it is necessary to understand the basic battery usage associated with the basic action of each device component. Measuring these relatively small effects on battery usage proved to be somewhat difficult, and multiple methods were attempted before accurate readings could be recorded. Fluctuations in battery usage were too small to be detected by a standard watt meter, so eventually, at the suggestion of Eric Berkenpas at National Geographic, a Hall Effect meter was used to detect the current draw of the devices during different modes. The Hall Effect signal was input into a standard laboratory voltmeter, and corresponding voltage levels for each functional mode were easily identifiable.



## Results

The static voltage signal output the constant running the microprocessor and the real-time clock is 2.2 volts, corresponding to a static current draw of 22mA (the Hall Effect device was set to output 1mV per 10mA of current measured). The single most impactful component on the overall battery life of the systems was the camera. When the camera was running, a fairly constant current draw of 300 mA was measured, which matches the nominal current draw detailed in the specifications for the Mobius cameras. Figure 5.1 displays the current draw from the Crittercam as it completes one cycle of processing data.

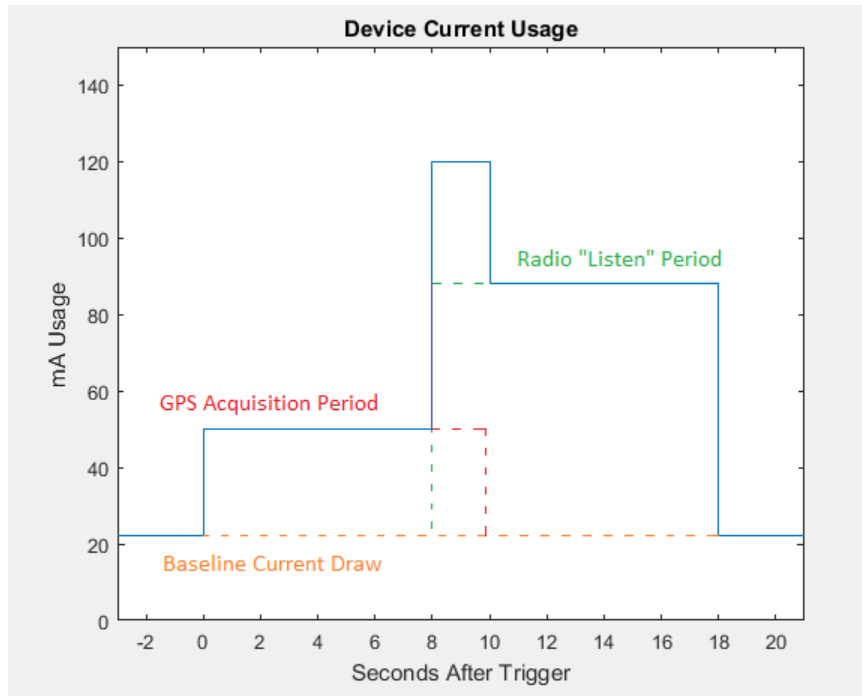


Figure 5.1: The current draw of a CritterCam as it completes one cycle of gathering, transmitting, and receiving data

The current iteration of the Networked Crittercam firmware triggers the periodic radio actions off of the devices' acquisition of a new GPS fix. At periodic intervals which may be varied among deployments, the device attempts to retrieve a GPS reading for ten seconds resulting in an additional 28 mA of current being drawn during this period. Eight seconds later, the radio

functions begin, resulting in an additional 66mA of current draw for another ten seconds. In the two second period where the functionalities overlap, the overall current draw was measured to be 120mA.

Because the frequency at which GPS fixes will be attempted is known in each functional mode, a model for the current draw in each function mode can be created. Furthermore, since there is an upper bound on the number of video recordings taken daily, given a rough estimate of the amount of time per day the device will spend in each mode, and the rate at which GPS readings are taken, we can make a fairly accurate estimate of the overall battery life of the devices in any given deployment. Figure 5.2 demonstrates the procedure used to create a model for estimating battery life, where  $\beta$  represents the battery capacity in mAh,  $h_{vid}$  and  $h_{act}$  are the number of estimated daily hours spent recording video and in the active state, respectively,  $mA_{vid}$ ,  $mA_{act}^+$  represent the additional current drawn from the recording and active states, respectively,  $mA_{sp}$  is the baseline current draw, and  $\rho$  is the period of GPS acquisition.

$$\begin{aligned}
 duration &= \frac{\beta}{mAh \text{ per day}} \\
 mAh \text{ used per day} &= h_{vid} * mA_{vid} + h_{act} * mA_{act}^+ + mA_{sp} \\
 mA_{act}^+ &= \text{number of events per hour} * mA \text{ per event} \\
 \text{number of events per hour} &= \frac{3600}{p}
 \end{aligned}$$

Figure 5.2: Formulation of the deployment duration model

Substituting the measured current values into  $mA_{vid}$ ,  $MAh_{act}^+$ , and  $mA_{sp}$ , we are left with the following equation as our model for deployment duration:

$$duration = \frac{\beta}{300 * h_{vid} + 946.8 * \frac{h_{act}}{p} + mA_{sp}}$$

Figure 5.3: Crittercam deployment duration model

It is the hope that this information will better inform future deployment efforts, and better allow researchers to decide how to best allocate battery usage resources. The model can be retroactively applied to the 2016 deployment, and its accuracy in this on case can be tested against the collected data. This is done in the following section.

## 5.1.2 Deployment Results

### Methodology

During the 2016 deployment in Gorongosa, the devices were deployed with uniform timing parameters, camera settings, and batteries. The specifics of these parameters can be referenced in the deployment section of this thesis. Without a substantial bank of experience to draw from, it was the estimation of the investigators that the systems would run for two weeks before dying. While the parameters of the deployment were held constant, there was no way to guarantee that the devices recorded the same amount of video or spent the same amount of time in each active state. As outlined above, the amount of time spent recording video is the dominant factor in the battery life, and is also easily quantifiable. Below, I compare the battery life of each device with the amount of video it recorded during the deployment, in an effort to create a more practical, albeit less exact model of battery usage.

### Results

A plot displaying the deployment duration and total recording time of each device is provided in [Figure 5.4](#).

It is important to note that these variables are co-dependent on each other. A device that takes fewer videos will likely sustain a longer deployment duration, while at the same time, a device that is deployed longer will likely have taken more videos. For these reasons, I do not think it is useful to express one metric as dependent on the other. Instead, since an increase in either metric indicates a superior performance by a device, in the future, it would be more useful to perform a Pareto analysis to compare the success of any given deployments.

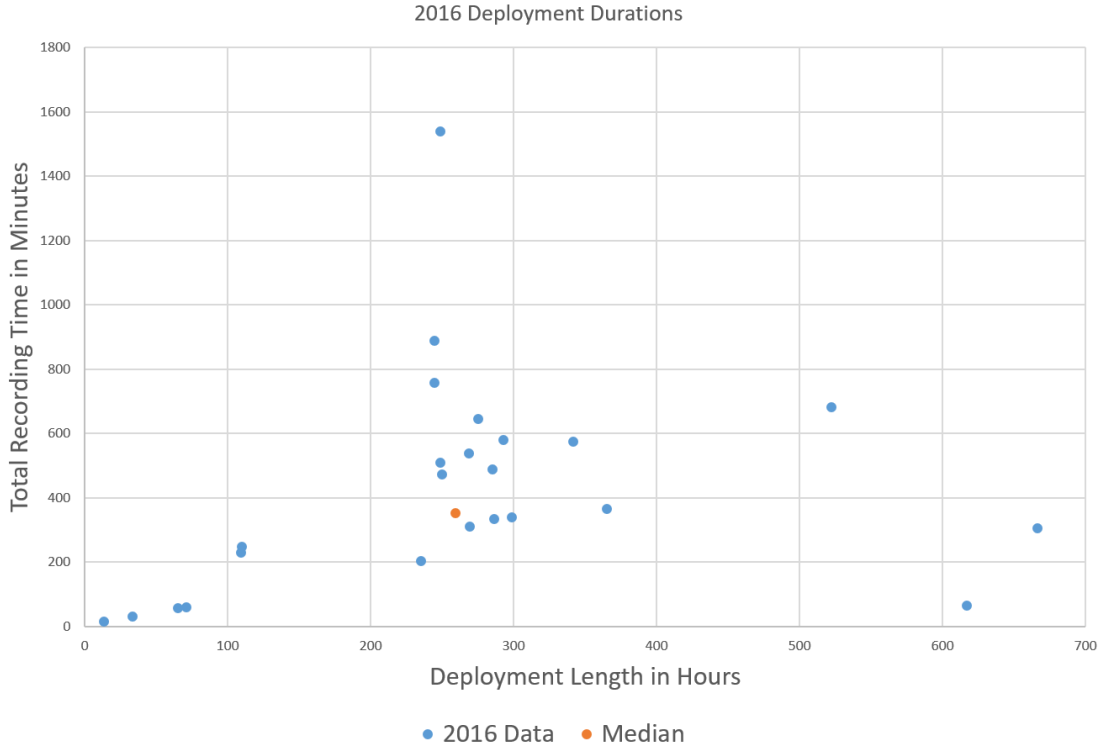


Figure 5.4: Deployment length as measured by battery duration and video capture time

This deployment data can also be used to retroactively test the accuracy of the deployment length estimation model created in the previous section. For the estimated input parameters to the model, we will use  $H_{vid} = 35$  minutes,  $H_{act} = 14$  hours,  $\rho = 60$  seconds, and  $\beta = 15600$  mAh. The result is an estimated deployment length of 16.88 days. When compared to the median deployment duration of the 2016 deployment, 259.265 hours, or 10.80 days, we see that the estimated deployment length overshoot the measured values by 56.3 percent. This overestimation is likely due to the discharge curve of the batteries dictating that at a certain mA usage, the requisite 3.6 volts cannot be supplied to power the camera. For this reason, it is necessary to re-evaluate the B value to represent the estimated point on the discharge curve where 3.3 volts can no longer be supplied.

Since the actual battery cells used in the deployment were prototypes, an exact discharge curve could not be provided. However, [discharge curve for a similar battery](#) with a 22000mAh capacity could be located. For these batteries, operating at room temperature, the discharge voltage drops below 3.6 at roughly 80 percent of capacity. We can subsequently modify our

estimated  $\beta$  value to 12480mAh, and our estimated deployment duration becomes 13.504 days, a more reasonable estimation when considered alongside the measured deployment duration.

## 5.2 Radio Transmission Range

### 5.2.1 Empirical Model

#### Methodology

Because the radio data collected from the deployments relies on interpolation of the GPS data, and the categorization of the habitat is not exact, it is useful to run some controlled experiments on the radio capabilities of the Crittercam. The goal of these tests originally was to identify the maximum communication range of the Xbee radios at different power levels, and to determine the effects of heavy forestation on this maximum distance. However, during the tests, it was determined that the boards are not suited to provide the Xbee modules with the current required to operate on power levels higher than the deployment level. Because of these limitations, tests were only performed at the deployment power level.

Four Crittercams were modified to transmit radio packets every ten seconds, and then split into pairs. One pair of devices remained stationary throughout the test, while the other pair was gradually moved 10m at a time away from the stationary devices. This movement occurred every twenty minutes. Results of preliminary tests, however, indicated a “radio reception saturation” where the devices could only receive a maximum number of transmissions in a given time period, regardless of how many were sent. With two devices sitting next to each other in this configuration, the test results were obscured by the fact that most received packets were sent from the neighboring device. Because of this effect, the number of devices used in the tests was reduced to two. The process of gradually moving the devices apart in increments of 10m was maintained.

Tests were performed in two distinct environments. To simulate an open, flat environment devoid of obstacles, tests were performed along the northeast branch of the Anacostia River. To simulate a heavily forested region, tests were performed along the perimeter trail of Greenbelt

Park.



Figure 5.5: Images from the "Forest" and "Plains" test environments

Precisely measuring distance between the devices presented a bit of a challenge, especially in the forest environment, where moving in a straight line over long distances is impossible. For this reason, distance was calculated using a GPS positioning phone app. Distance measurement was not always precise to the meter, but GPS coordinates were recorded so the actual distance could be calculated and recorded afterward.

## Results

Figure 5.6 displays the rate at which transmissions were completed at each measured distance in each habitat.

It is immediately apparent that the recordings from this test indicate the "expected" result of a longer connectivity range in the less obstructed habitat. Aside from the 10m (where there were not any direct obstacles between the devices in the "forest") and 40m distances, the reception rate in the "plains" was higher at every point of measurement. The maximum reception distance in the "plains" is 30m higher than in the "forest."

While these results do not provide a comprehensive suite of radio performance data in

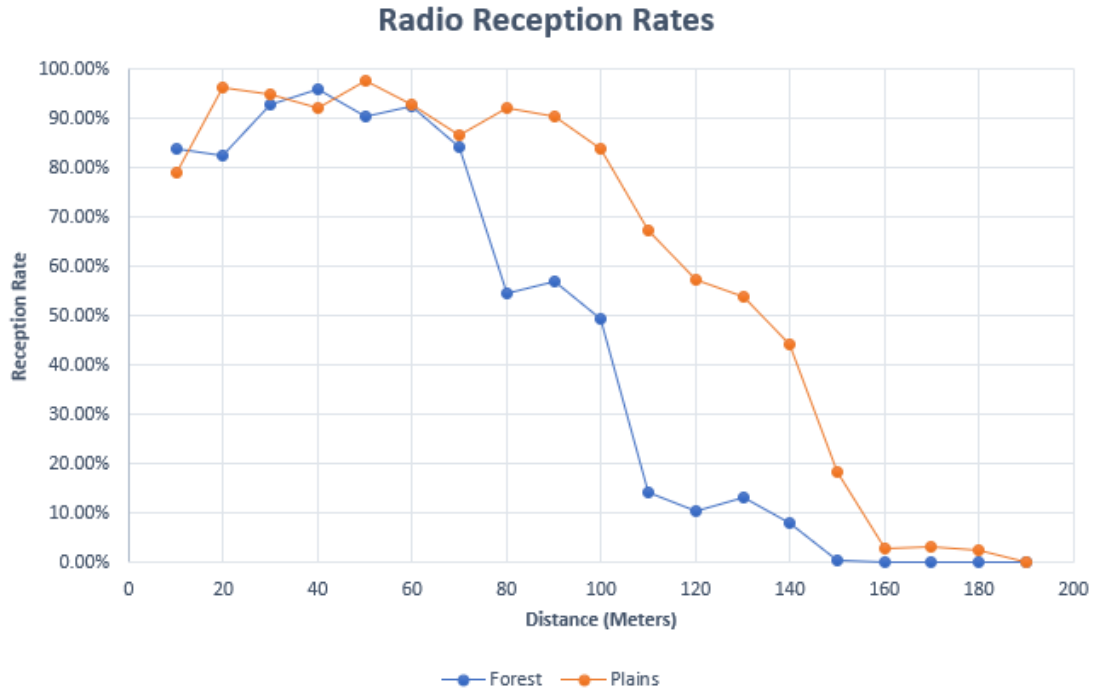


Figure 5.6: Radio reception rates in two distinct habitats

every habitat, they do provide an excellent starting point for approximating radio ranges in most potential deployment environments. The “plains” results provide a good baseline for the best case performance of the radio systems, and the “forest” results indicate the drop in performance caused by heavily obstructed environments. Armed with this data, researchers can have a good idea of how the Crittercam radios will perform and can adjust related parameters accordingly.

Another important feature of this data is the significant drop in reception rate at a particular distance in both habitats. The rate falls from the mid-forties to the teens between 100 and 110 meters in the “forest”, and between 140 and 150 meters in the “plains.” In deployments such as Gorongosa 2016, where the animals generally move around at a slow rate, it might be acceptable to consider the true maximum recorded transmission distance when planning for the deployment. However, in environments where animals will be moving quickly, it would be prudent to consider these drop off points as the maximum transmission distance. If the collared animals will only be interacting less than a minute, the odds of a successful radio transmission occurring after these

drop off points is very low. This is another instance of how this empirical data can better inform future deployment efforts.

## 5.2.2 Deployment Results

### Methodology

Utilizing radio transmission data from the Gorongosa deployments to quantify device performance presents a number of challenges. While understanding the radio connectivity of the devices during a large and sparsely populated deployment is of great importance, many assumptions and estimations had to be made to do so. Since GPS data is obtained periodically by the devices, interpolation of the latitude and longitudinal tracks is necessary to estimate the location of the devices at the time of radio communication. This interpolation was done using the cubic spline function in the Python SciPy library, independently applied to both the latitudinal and longitudinal coordinates as functions of time.

Since the GPS frequency during the 2016 deployment was a relatively high once per minute, this interpolation provides a reasonable estimate as to the animal's location at any time during the deployment (The GPS frequency drops when the device is in the sleep state, but that only occurs when the animal is not moving). Furthermore, in order to distinguish connectivity across different habitats, it is necessary to classify the entire deployment area into either "forest" or "floodplains" regions. Using satellite imagery, the different biomes were identified, and two rough boundary lines were drawn in between the regions. The distinction between the regions is depicted in Figure 5.7. It is important to note that within the region identified as "forest" there are many smaller "salt pan" regions that are devoid of trees and share more physical characteristics with the flood plains than the forests.

Once the GPS tracks had been interpolated and the habitats identified, distance for each transmission was calculated as points on a sphere, and the total number of transmissions completed at each distance was recorded.



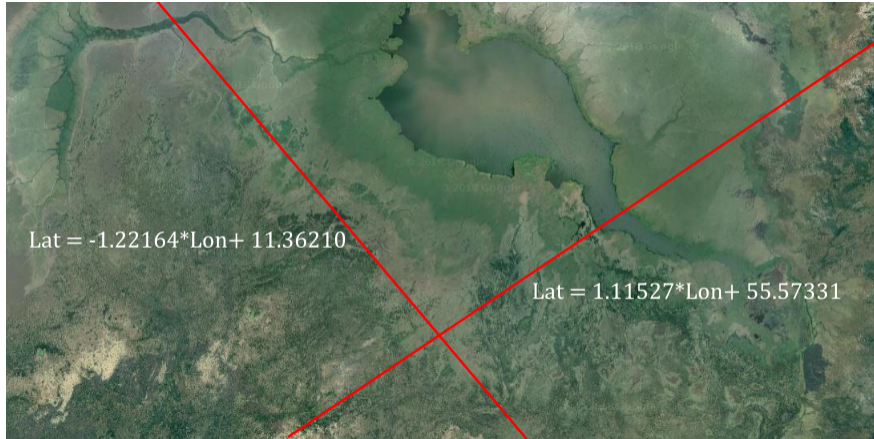


Figure 5.7: Delineation of “Forest” and “Floodplain” regions for radio analysis purposes

## Results

Figure 5.8 displays the number of radio transmissions that were recorded at each distance during the 2016 deployment. Transmissions were recorded at distances up to 500m, but the overwhelming majority of the transmissions occurred within 100m, and this range is where most of the meaningful data can be gleaned.

The most apparent result from the radio data is that the radio range in the floodplain appears to be smaller than the range in the forest, which is counter intuitive to what might be expected. However, this surprising result likely has more to do with the behavior of the Waterbuck than with the physical properties of the radio transmissions. The primary hypothesis to support this is that the Waterbuck tend to space themselves closer together when they are in the flood plain.

To try to verify this hypothesis, the GPS data for each node was sampled every five minutes, and the distance to each other node in the same habitat was estimated. However, this data did not reveal a significant tendency for the animals to space themselves differently across habitats. A more fine tuned examination of Waterbuck behavior is necessary in order to properly identify the ecological effect on the radio transmission distances.

Another important observation, gleaned from the ArcMap visualization of the deployment,

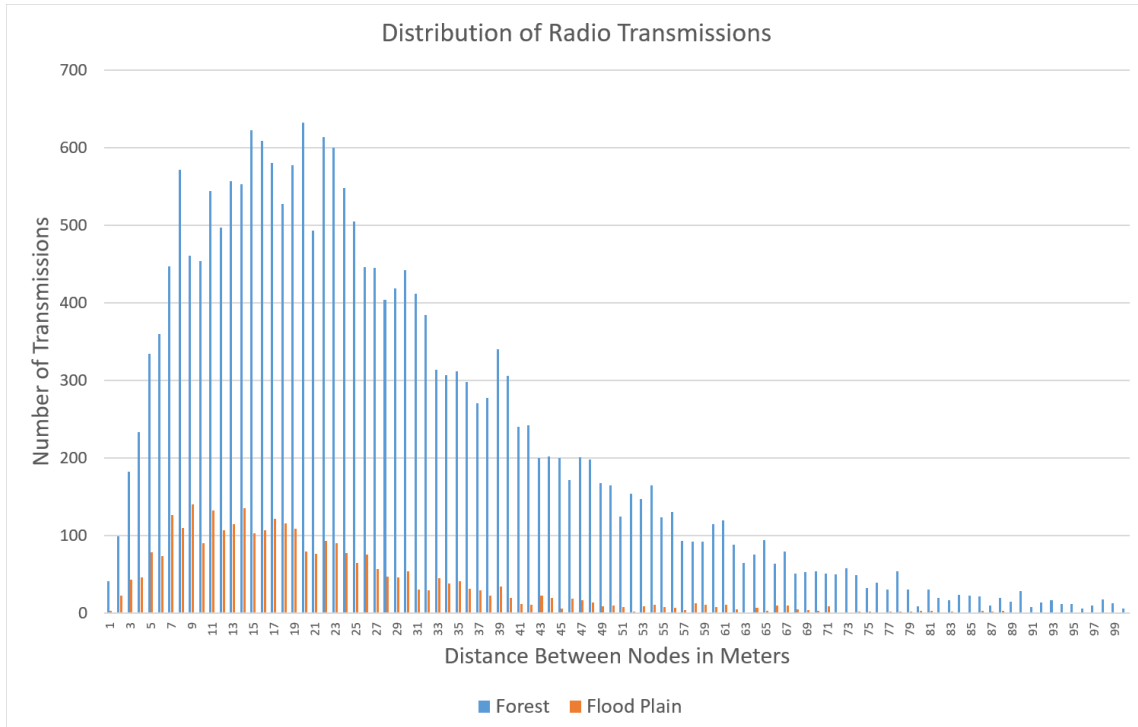


Figure 5.8: Distribution of estimated radio communication distances from Gorongosa 2016 deployment

is that high proportion of the time spent by the subjects in the "forest" region occurred near watering holes, an opening in the forest that is closer to the floodplain in features. The high amount of radio transmissions that occurred in this flood plain-like habitat further skew the data collected away from the physical capabilities of the systems and towards the realm of ecological analysis. A closer look at the Waterbuck behavior in this region would be a crucial first step in clarifying this distinction.

## Chapter 6: Conclusion

### 6.1 Results

When I took over engineering responsibilities on the Networked Crittercam project, the devices were in excellent functional condition, but as a whole, the system infrastructure was not suitable for repeated deployment. Along with getting the devices ready for the 2016 deployment, my primary responsibilities were to provide the systems with the requisite software and the users with the pertinent information so that the systems can be deployed quickly and successfully.

My contributions to the project have moved the Networked Crittercam from the development stage into the operational stage of the product life cycle. The systems have been tested in a large-scale deployment scenario on multiple species, and have demonstrated functional capabilities and durability. The systems now have a suite of intuitive deployment tools that can be used for deployments of different size and demands. Additionally, I have identified several important metrics regarding device performance. Lastly, I have identified and demonstrated means by which the data collected by the devices can be represented in a coherent interface.

As a result, National Geographic now possesses a tool that can be used by almost any user, without the need for a lengthy orientation process. Additionally, the metrics collected regarding device performance allow researchers to make more informed deployment decisions, and the software interfaces make it easy to make these changes in the field. With the additions made to the system infrastructure, I believe that the Networked Crittercams will offer a state of the art option for most large scale animal-borne sensor deployments.

## **6.2 Skills Acquired**

Over the course of my involvement with the Networked Crittercam project, I have acquired several useful skills that have bolstered my aptitude as a job candidate. This includes technology specific skills, such as programming languages, and also more general skills such as hardware troubleshooting. Outlined below are some of the more significant skills I either acquired or developed through my work on the Crittercam systems.

### **6.2.1 Python**

Before my work on the Crittercams, I had no exposure to the Python language. By the end of the project, I believe that Python is the programming language I am most comfortable with. Python offers many libraries that are useful to this project, including the SQL library that allows for interfacing with the database files, and the SciPy library that allows for interpolation of movement data. For this reason, Python was the default language for most of the Crittercam software, and I now feel confident in my use of Python in a professional environment.

### **6.2.2 Java**

The Swing library provided by Java offers a relatively simple tool to implement a Graphical User Interface and was therefore used as primarily for creating the Crittercam GUIs. Although I had some previous experience with Java, I had never worked with the Swing library or created a GUI before. Although my theoretical understanding of computer engineering was not expanded through my use of Java, it still provided practical experience in design and user interface development.

### **6.2.3 Software Development**

The Crittercam firmware code left to me when I joined the project was the largest segment of software I had worked with in my life, and at that time had been written entirely by other

coders. Because of this, I developed important software engineering skills such as version control and scalability and readability of code. Near the beginning of my time working with the code, I had a much harder time debugging my mistakes due to my own carelessness in how I was writing code. By the end of the project I believe I had developed coding practices closer to what is expected in a professional environment.

### 6.3 Future Work

With the development of a deployment software suite and the identification of significant device performance metrics, the Networked Crittercams are now situated as a useful tool for repeated biological research. Although the grant funding the development of the systems has ended, the Crittercam project is just now beginning to provide value to stakeholders through its deployment. In January 2018, four Crittercams were deployed to Buffalo enclosed within the American Prairie Reserve in Montana. While the results of this particular mission were not without importance, the true significance of this deployment was to demonstrate the Crittercams' functionalities and utility, with the intention of using them for larger scale investigations. In this deployment, the upper limit of the geographic triggering points the system could handle was tested. This was done as part of a planning effort for a potential deployment onto wild Bison in order to try to identify migration patterns. Geographic location triggering would be used to create "geofences" that could help allow the devices to indicate when anticipated migration checkpoints have been reached. This method could be used to isolate areas of interest along the migration trail of the bison.

The area that leaves the most room for improvement in regard to the Crittercams is the definition of effective conditions for camera triggering. This presents a complex problem that varies based on species, habitat, and research goals. However, the largest advantage that the CritterCams offer is their fine-tuned control over camera trigger procedures. The utility of this technology is limited by the accuracy at which moments of significance can be predicted. Given sufficient deployment data from the CritterCams, it is conceivable that a more accurate model for

triggering conditions can be developed by cross referencing video and sensor data.

CritterCam technology can be used to facilitate more efficient data collection in other ways as well. One such upgrade would entail enabling large scale data transfers between devices via radio communication. This would provide devices with a “history” of neighboring devices and facilitate more informed camera triggering decisions. It would also minimize data lost by devices that could not be recovered, and the transferred information would help recover dropped devices whose location is unknown.

Lastly, a comprehensive documentation suite for the maintenance and deployment of the Crittercams needs to be developed. With my departure from the project, there will no longer be any one person dedicated to these responsibilities. Knowledge on how to operate Crittercam systems is necessary in order to facilitate their sustained use and explore their utility. This thesis acts as a starting point for such documentation, but a more detailed and explicit instruction set is necessary to ensure that the Crittercam systems fulfill their potential utility to the ecological community.

## References

- [1] Ran Blekhman et al. “Common methods for fecal sample storage in field studies yield consistent signatures of individual identity in microbiome sequencing data”. In: *Scientific Reports* 3 (2016). DOI: [10.1038/srep31519](https://doi.org/10.1038/srep31519).
- [2] Anthony Caravaggi et al. “A review of camera trapping for conservation behaviour research”. In: *Remote Sensing in Ecology and Conservation* 3.3 (2017), pp. 109–122. DOI: <http://dx.doi.org/10.1002/rse2.48>.
- [3] Karen M. Carney and William J. Sydeman. “A Review of Human Disturbance Effects on Nesting Colonial Waterbirds”. In: *Waterbirds: The International Journal of Waterbird Biology* 22.1 (1999), pp. 68–79. DOI: [10.2307/1521995](https://doi.org/10.2307/1521995).
- [4] *Crittercam Education*. URL: <https://www.nationalgeographic.org/education/crittercam-education/>.
- [5] University of Georgia. *The National Geographic and University of Georgia Kitty Cams (Crittercam) Project*. URL: <http://www.kittycams.uga.edu/>.
- [6] Susan G. Heaslip et al. “Jellyfish Support High Energy Intake of Leatherback Sea Turtles (*Dermochelys coriacea*): Video Evidence from Animal-Borne Cameras”. In: *PLoS ONE* 7.3 (2012). DOI: <https://doi.org/10.1371/journal.pone.0033259>.
- [7] IOOS. *OOT: Animal Borne Systems*. URL: <https://ioos.noaa.gov/project/ott-animal-borne-sensors/>.

- [8] Suzanne Amador Kane and Marjon Zamani. “Falcons pursue prey using visual motion cues: new perspectives from animal-borne cameras”. In: *Journal of Experimental Biology* 217 (2014), pp. 225–234. DOI: [10.1242/jeb.092403](https://doi.org/10.1242/jeb.092403).
- [9] Clive R. McMahon et al. “Animal-borne sensors successfully capture the real-time thermal properties of ocean basins”. In: *Limnology and Oceanography Methods* (2005). DOI: [10.4319/lom.2005.3.392](https://doi.org/10.4319/lom.2005.3.392).
- [10] Remington J. Moll et al. “A terrestrial animal-borne video system for large mammals”. In: *Computers and Electronics in Agriculture* 66.2 (2009). DOI: <https://doi.org/10.1016/j.compag.2009.01.001>.
- [11] Darren Norris Fernanda Michalski Carlos A. Peres. “Habitat patch size modulates terrestrial mammal activity patterns in Amazonian forest fragments”. In: *Journal of Mammalogy* 91.3 (2010), pp. 551–560. DOI: <https://doi.org/10.1644/09-MAMM-A-199.1>.
- [12] Yuuki Y. Watanabe and Akinori Takahashi. “Linking animal-borne video to accelerometers reveals prey capture variability”. In: *Proceedings of the National Academy of Sciences of the United States of America* 110.6 (2012), pp. 2199–2204. DOI: <https://doi.org/10.1073/pnas.1216244110>.
- [13] Christopher C. Wilmers et al. “The golden age of bio-logging: how animal-borne sensors are advancing the frontiers of ecology”. In: *Ecological Society of America* 96.7 (2015). DOI: [10.1890/14-1401.1](https://doi.org/10.1890/14-1401.1).