A Multi-Modal Sensing and Communication Platform for Continental-Scale Migratory Bird Tracking

David J. Anthony
University of Nebraska-Lincoln, djanthony@gmail.com

Follow this and additional works at: http://digitalcommons.unl.edu/ceendiss

Part of the Computer Engineering Commons, OS and Networks Commons, Systems and Communications Commons, and the Systems Architecture Commons

http://digitalcommons.unl.edu/ceendiss/16

This Article is brought to you for free and open access by the Electrical & Computer Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in Theses, Dissertations, & Student Research in Computer Electronics & Engineering by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.
A MULTI-MODAL SENSING AND COMMUNICATION PLATFORM FOR CONTINENTAL-SCALE MIGRATORY BIRD TRACKING

by

David J. Anthony

A THESIS

Presented to the Faculty of
The Graduate College at the University of Nebraska
In Partial Fulfilment of Requirements
For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Mehmet C. Vuran and Matthew B. Dwyer

Lincoln, Nebraska
May, 2012
This thesis presents a novel platform for tracking migratory birds on a continental scale. Cellular technology is used to augment the short-range radios that have traditionally been used in wireless sensor networks. The platform utilizes multiple sensors, including a GPS and solid state compass. By using these sensors, the platform is capable of not only tracking a bird’s migration path, but also provides information on a bird’s behavior during its life-cycle. Testing methodology utilizing simulations and aspect-oriented programming is used to reveal faults in the platform prior to deployment on wild animals. In collaboration with the International Crane Foundation, and the Crane Trust, the platform is evaluated on multiple species of birds (Wild Turkeys, Siberian Cranes, Sandhill Cranes), over 6 months, to evaluate its effectiveness. These deployments reveal that the system is capable of not only tracking birds, but monitoring their behavior. By utilizing cellular technology, the system is capable of delivering information about a bird within 24 hours, which is much faster than current systems used to track Whooping Cranes.
COPYRIGHT

© 2012, David J. Anthony
DEDICATION

To Andrea.
ACKNOWLEDGMENTS

I am deeply thankful to my advisors, Dr. Mehmet C. Vuran and Dr. Matthew Dwyer, for their patience, guidance, and support of my academic career. I would also like to thank Dr. Sebastian Elbaum for all of his advice, lending his expertise to the crane tracking project, and serving on my committee. Thank you to Dr. Carrick Detweiler, who has always taken the time to answer my electronics questions, and has also served on my committee. Dr. Mark Bauer, of the Electrical Engineering department, has also given me substantial guidance on electronics and how to manufacture them.

Paul Bennett has been an invaluable partner and colleague, and I have enjoyed collaborating with him on the Whooping Crane project. Collin Lutz, Aldo Arizmendi, Derek Homan, and Caleb Huenefeld all helped out with the project as undergraduates, and performed many thankless tasks.

The International Crane Foundation and the Crane Trust have been a constant source of insight into wildlife behavior and new project ideas. In particular, Mike Engels, Anne Lacy, and Walter Wehtje have been very generous with their time, and have patiently answered many questions.

This work was completed utilizing the Holland Computing Center of the University of Nebraska.

Finally, I would like to thank my wife Andrea, who has continually supported and encouraged me.
GRANT INFORMATION

The university has generously supported me through a Mohr and Othmer fellowship. I have also been partially funded through NSF CAREER grant 0953900.
Contents

List of Figures xi

List of Tables xiv

1 Introduction 1
  1.1 Wireless Sensor Networks .............................. 1
  1.2 Contributions ........................................... 2

2 Related Work 4
  2.1 Wireless Sensor Network Deployments .................. 6
  2.2 Wildlife Tracking ....................................... 7
  2.3 Whooping Cranes ....................................... 8
  2.4 Testing .................................................. 10
  2.5 Energy Management ..................................... 11
  2.6 Conclusions ............................................ 12

3 Whooping Crane Monitoring Project 13
  3.1 Whooping Crane History ................................ 13
  3.2 Whooping Crane Migrations ............................. 14
3.3 Whooping Crane Behavior ........................................... 16
3.4 Mortality ................................................................. 17
3.5 Prior Tracking Efforts ................................................... 17
3.6 Project History .......................................................... 21
3.7 Whooping Crane Monitoring Goals ................................. 21
3.8 Whooping Crane Monitoring Requirements ....................... 23
3.9 Conclusions .............................................................. 24

4 Development of the CraneTracker100 Hardware .................. 25
  4.1 MTS-420 ................................................................. 25
    4.1.1 MTS-420 Description ........................................... 26
    4.1.2 MTS-420 Evaluation ............................................. 27
  4.2 CraneTracker100.v1 ................................................... 28
    4.2.1 CraneTracker100.v1 Description ............................... 28
    4.2.2 CraneTracker100.v1 Evaluation ................................. 32
  4.3 CraneTracker100.v2 ................................................... 33
    4.3.1 CraneTracker100.v2 Description ............................... 33
    4.3.2 CraneTracker100.v2 Evaluation ................................. 35
  4.4 CraneTracker100.v3 ................................................... 37
    4.4.1 CraneTracker100.v3 Description ............................... 38
    4.4.2 CraneTracker100.v3 Evaluation ................................. 40
  4.5 Enclosure and Attachment ........................................... 46
    4.5.1 Leg Attachment ................................................... 46
    4.5.2 Backpack Attachment ............................................ 48
  4.6 Platform Development Conclusions .................................. 50

5 Bringing the CraneTracker100 to Life: Software Development Process 53
5.1 Division of Labor ........................................... 53
5.2 TinyOS .................................................... 54
5.3 Lower Level Components .............................. 55
5.4 Upper Level Software .................................... 58
5.5 Backend Software ........................................ 61
5.6 Software Development Conclusion .................. 62

6 Pre-deployment Software Testing ...................... 64
6.1 Background ................................................ 65
6.2 CraneTracker100 Testing ............................... 68
6.3 Testing Platform .......................................... 69
   6.3.1 Application Under Test ............................... 71
   6.3.2 Log File Analysis .................................... 71
   6.3.3 AspeCt-Oriented C Compiler ....................... 73
6.4 Results ..................................................... 76
   6.4.1 Execution Times ..................................... 76
   6.4.2 Executable Size ...................................... 78
   6.4.3 Implementation Cost ................................. 78
   6.4.4 Violations ............................................. 79
6.5 Testing Conclusions ..................................... 80

7 Deployment Experiences with the CraneTracker100 .... 81
7.1 Preliminary Evaluations ................................. 82
7.2 Field Experiments with Wild Turkeys ................. 84
   7.2.1 Experimental Turkey Setup .......................... 85
   7.2.2 Experimental Turkey Results ....................... 86
   7.2.3 Ecological Observations ............................. 90
7.3 Captive Siberian Crane Experiments ................................ 91
  7.3.1 Captive Crane Experiment Setup ............................... 91
  7.3.2 Captive Crane Experiment Results ............................. 92
  7.3.3 Captive Crane Experiment Summary ........................... 94
7.4 Wild Sandhill Crane Deployments ................................. 95
  7.4.1 Wild Sandhill Experiment Setup ................................ 95
  7.4.2 Wild Sandhill Experiment Results ............................. 96
  7.4.3 Wild Sandhill Experiment Analysis ............................ 102
  7.4.4 Wild Sandhill Experiment Ecological Observations .......... 107
7.5 Experimental Deployment Conclusion ........................... 111

8 Conclusions .............................................................. 112
  8.1 Contributions .......................................................... 112
  8.2 Future Work ............................................................ 113

Bibliography .............................................................. 118
List of Figures

2.1 Captive Whooping Crane ........................................ 9

3.1 AWBP Migration route ........................................... 15
3.2 VHF Tracker .................................................. 18
3.3 Satellite tracker ................................................ 19

4.1 MTS-420 Tracker ................................................ 26
4.2 CraneTracker100.v1 Communication power requirements .... 30
4.3 CraneTracker100.v1 ............................................ 32
4.4 Flexible solar panel ............................................. 34
4.5 CraneTracker100.v2 ............................................ 35
4.6 Solar panel experimental setup ................................. 36
4.7 GE865 and GM862 size difference ............................ 38
4.8 Power supply design ........................................... 41
4.9 CraneTracker100.v3 ............................................ 41
4.10 Hardware component architecture ............................. 42
4.11 Tracker platforms .............................................. 42
4.12 Microcontroller power consumption .......................... 43
4.13 Sensor power .................................................. 44
4.14 CraneTracker100.v3 Communication power requirements .... 45
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.15</td>
<td>GSM current while sending SMS messages</td>
</tr>
<tr>
<td>4.16</td>
<td>CraneTracker100.v2 enclosure, top side</td>
</tr>
<tr>
<td>4.17</td>
<td>CraneTracker100.v2 enclosure, bottom side</td>
</tr>
<tr>
<td>4.18</td>
<td>Backpack mounting</td>
</tr>
<tr>
<td>4.19</td>
<td>Layout evolution</td>
</tr>
<tr>
<td>5.1</td>
<td>Compass driver</td>
</tr>
<tr>
<td>5.2</td>
<td>GPS driver architecture</td>
</tr>
<tr>
<td>5.3</td>
<td>GSM driver</td>
</tr>
<tr>
<td>5.4</td>
<td>Power monitoring component</td>
</tr>
<tr>
<td>5.5</td>
<td>GSM Manager</td>
</tr>
<tr>
<td>5.6</td>
<td>Radio manager</td>
</tr>
<tr>
<td>5.7</td>
<td>Crane storage component</td>
</tr>
<tr>
<td>5.8</td>
<td>Crane manager</td>
</tr>
<tr>
<td>6.1</td>
<td>System behavior for testing</td>
</tr>
<tr>
<td>6.2</td>
<td>Compilation process with aspects</td>
</tr>
<tr>
<td>7.1</td>
<td>Communication delay and speed over time</td>
</tr>
<tr>
<td>7.2</td>
<td>Received power vs speed</td>
</tr>
<tr>
<td>7.3</td>
<td>CDF of GSM localization error</td>
</tr>
<tr>
<td>7.4</td>
<td>CDF of turkey GPS error</td>
</tr>
<tr>
<td>7.5</td>
<td>Turkey positions</td>
</tr>
<tr>
<td>7.6</td>
<td>Battery voltage of control and turkey tracker</td>
</tr>
<tr>
<td>7.7</td>
<td>Siberian Crane with CraneTracker100.v3</td>
</tr>
<tr>
<td>7.8</td>
<td>CDF of GPS error</td>
</tr>
<tr>
<td>7.9</td>
<td>Three-axis acceleration readings from a Siberian Crane.</td>
</tr>
</tbody>
</table>
7.10 SH-Chick and BB-Female migration routes ........................................ 97
7.11 CDF of communications and queuing delay ........................................ 98
7.12 CDF of only communication delay ...................................................... 99
7.13 SH-Chick altitude and speed ............................................................... 100
7.14 Solar power and battery voltage for SH-Chick ..................................... 101
7.15 Solar power and battery voltage for JB-Female .................................... 102
7.16 Low voltage recovery .......................................................................... 103
7.17 Distance between SH-Chick and family and neighbors ....................... 104
7.18 CDF of solar current, 2 hour intervals, 1mA granularity ......................... 105
7.19 CDF of solar current, 4 hour intervals, 1mA granularity ......................... 106
7.20 CDF of solar current, 2 hour intervals, 5mA granularity ......................... 107
7.21 CDF of solar current, 4 hour intervals, 5mA granularity ......................... 108
7.22 Solar CDFs of individual birds ............................................................. 109
7.23 GPS locations of JB-Female over 24 days ........................................... 109
7.24 SH-Chick Acceleration during migration ............................................. 110
8.1 Sensor control using POMDPs ............................................................... 116
List of Tables

4.1 Quantitative solar panel results .............................................. 37
4.2 Tracker dimensions ................................................................. 43
4.3 Tracker summary ................................................................. 50

6.1 Overhead of aspect in terms of CPU time and executable size .......... 77

7.1 Number of solar data points collected ........................................ 107
7.2 Experiment summary ............................................................... 111
Chapter 1

Introduction

1.1 Wireless Sensor Networks

Wireless sensor networks (WSNs) are comprised of many small devices, called motes. These motes utilize a small set of sensors and short range communications to form ad-hoc networks. By collaboratively sensing their environment and communicating, the motes are able to relay information on their environment to distant locations. These capabilities have been used in prior efforts to monitor the movements and environments of animals [30, 50, 51]. Due to the success of these experiments, WSNs have been raised as a tool for monitoring the behavior and migrations of birds on a continental scale. However, existing approaches rely on short range communication, and simple sensing, and do not meet the needs of next generation research efforts. To this end, a new class of WSNs is developed, which utilizes multiple modes of communication to deliver sensor data in a timely, energy-efficient, and cost-effective manner. Furthermore, these devices are equipped with sensors that enable the analysis of an animal’s behavior. To support the long mission duration required by biological research, the devices utilize energy harvesting to extend mission lifetimes and reduce the weight of the energy reserves.
carried by a device.

An example of these new WSNs is the CraneTracker100. This mote contains both a GPS and solid state compass for monitoring a bird’s behaviors and movements. In order to supplement its meager energy reserves, a solar panel is used to harvest ambient solar energy from the environment. This new mote is used in a project to track Whooping Cranes throughout North America, and to analyze their behavior.

Developing and deploying the CraneTracker100 revealed several challenges in developing next generation WSNs. The extended mission duration of the Whooping Crane project (>5 years) necessitates environmental energy harvesting. The unpredictable, continent spanning, migration routes of the cranes force the CraneTracker100 to use multi-modal communications. The sensors of the platform have to effectively monitor complex behaviors, and consume significant amounts of power. The trade-offs between the sensing and communication must be weighed against the stochastic nature of the energy harvesting. All of these factors combine to create a much more complex platform than has traditionally been used in WSNs. This complexity also impacts the behavior of the mote, as the system must be capable of adapting to the behavior of the bird, energy harvesting, and sensors.

1.2 Contributions

The contributions of this work are as follows.

**Novel hardware platform:** A novel tracking platform, the CraneTracker100, is constructed. This platform includes multi-modal sensing and communication capabilities. This platform can be used for not only tracking wildlife, but also for conducting detailed research into wildlife behavior. This type of research will yield significant insights into crane behavior, that are not attainable with the current systems employed by ecologists.
**Testing methodology:** The CraneTracker100 is significantly more complex than the motes used in most research projects. This complexity, coupled with the extended mission durations common to wildlife research, demands rigorous testing techniques. To this end, an existing software tool for aspect-oriented programming is adapted for use with TinyOS [45] software and the TOSSIM network simulator [44]. This tool significantly increases the testing capabilities of the developers, and benefits from reusing existing tools.

**Field tests:** The combined hardware and software system is evaluated through a series of field trials. Given the extreme rareness of Whooping Cranes, this evaluation is conducted on “proxy” animals, including Wild Turkeys, Siberian Cranes, and Sandhill Cranes. These animals share behaviors and habitats with the Whooping Cranes. Using these animals for testing provides a safe method for evaluating the system, without endangering the already rare Whooping Cranes.

The remainder of this work discusses the development the CraneTracker100 and its results in tracking wild birds. Chapter 2 presents a high level overview of work related to this project, including previous WSN deployments, wildlife tracking, testing, and energy management. Following that, Chapter 3 presents a detailed background on Whooping Cranes and their conservation efforts to frame the rest of the work. Chapter 4 discusses the development and evolution of the CraneTracker100. Following a description of the hardware development process, the software for the system is discussed in Chapter 5. Following this description, Chapter 6 presents work that was conducted to test the software used on the platform. The results of this combined hardware and software platform are shown in Chapter 7, in which the platform is deployed in several scenarios. Finally, a conclusion is given in Chapter 8.
Chapter 2

Related Work

Wireless sensor networks (WSNs) were originally conceived as being made of many small, low cost, low complexity motes [11]. The low cost of these motes made it practical to deploy many redundant sensors in an area where some environmental phenomenon was to be monitored. Practical examples of deployments include monitoring forest fires and the nesting grounds of birds [50, 89]. Most real-world deployments of sensor networks have been characterized by relatively low mobility applications that operate in limited geographic regions. Additionally, the sensors on these devices are used to monitor simple environmental phenomenon. However, by expanding the capabilities of the motes through multiple communication channels and sensors, exciting new applications become possible. To this end, the CraneTracker100 has been developed. The CraneTracker100 demonstrates the possibilities of these next generation sensor networks by monitoring migratory birds on a continental scale. This platform outperforms previously used technology for wildlife monitoring by delivering detailed behavioral data in a timely manner, while the migrations are in progress.

Processing: Sensor motes utilize low cost microcontrollers. These small microcontrollers place a premium on power efficiency, which is a necessity for the oft-limited
energy reserves on a mote. The processing unit on the motes allows for limited in-network processing of data. Communications costs are typically assumed to dominate communication costs in WSNs, so in-network data processing to reduce network traffic is commonly used to reduce energy usage [12, 38, 65].

**Communication:** The motes are capable of establishing ad-hoc networks, without the need for centralized architecture. This allows for flexible deployments that place the motes where they are needed, in a timely manner [11]. The communication mechanisms are typically short-ranged devices, due to the size and power constraints of the device. This necessitates multi-hop communications amongst the nodes, since most motes will not be in communication range of their destination device.

**Sensors:** The motes in a wireless sensor network include sensors for monitoring their surrounding environment. These sensors used in wireless sensor networks are incredibly diverse, ranging from GPS [56], to simple accelerometers [55], and even moisture sensors for measuring soil moisture [28].

**Power:** Size, weight, and cost requirements put severe restrictions on the power supply of motes. Methods to save power are the focus of much of the research in WSNs. At the hardware level, several approaches have been taken to address the limited energy reserves of the motes. While some type of battery typically provides the majority of the energy reserves of the device, recent approaches incorporate supercapacitors or energy harvesting to improve the lifetime of the device [39, 72, 83].

The system developed in this work has significant differences from these prior approaches. Rather than relying solely on short range communication systems, the devices include cellular technology to communicate at long distance. The sensors on the device are capable of monitoring complex animal behavior. Finally, it includes the ability to harvest solar energy. Unlike prior solar energy harvesting works, the energy harvesting process is influenced by the particular behaviors of the animal it is attached to.
2.1 Wireless Sensor Network Deployments

The combination of low cost and ad-hoc networking make WSNs ideal for many environmental monitoring applications. Their ad-hoc networking capabilities mean they can be deployed in areas where it is impossible or cost prohibitive to add fixed infrastructure. Examples of these deployments are WSNs for volcano monitoring [85] and underground networks for soil moisture monitoring [28]. Deployments have even been conducted underwater, where acoustic communication channels are used instead of traditional radio frequency communications [26].

Wireless sensor network deployment models typically fall into one of three categories. In the first type of deployment, the nodes are deployed at fixed, pre-determined locations. Underground sensor networks are a prime example of this deployment [28]. The motes that comprise the network are buried in soil. Once this is accomplished, it is very costly and time consuming to recover the motes. These types of deployments have the advantage of having a fixed physical topology. However, if the physical environment changes, or different area needs to be monitored, it is very difficult to redeploy the network. Initial deployment costs may also be high, as the motes need to be intelligently placed in a well defined pattern, which is time consuming.

The second type of deployment randomly places immobile motes at random locations in an area. Such deployments deployments may be accomplished via air drops, or even artillery shells [11]. In comparison to the pre-planned placement deployments, these types of deployments can be accomplished much faster, and are therefore cheaper. However, they suffer the same drawbacks of being difficult or impossible to redeploy. Furthermore, the random deployment often results in a poor network topology, which wastes energy and increases the chances of network failure.

The final category of deployments places the motes on mobile devices. These types of
deployments are common for wildlife tracking [51] and social behavior monitoring [68]. These deployments are often seen in vehicle area networks (VANETs) [73], in which the motes are placed on vehicles that probabilistically travel through streets. These types of networks are often the most challenging. The dynamic topology makes it difficult to route data between nodes. Moreover, the changing topology can lead to routing delays as nodes establish and lose connections. These topology changes also waste energy, as the motes must resend data to motes that have lost connectivity.

Tracking migratory animals on a continental scale is an extreme case of a mobile deployment. The mobility of birds during these migrations eclipses the parameters considered in most WSN deployments [34]. The unpredictable and complex flight patterns of the birds makes it difficult to employ short range communication devices to monitor the birds during migratory periods. Thus, new approaches are needed to compensate for the high mobility, that are able to deliver information in a timely manner.

### 2.2 Wildlife Tracking

Several prior projects have utilized WSNs in interesting manner for tracking wildlife and monitoring their habitat. In an early example of these efforts, collars were attached to zebras that included GPS receivers and VHF radios for communications [50]. These collars are capable of forming ad-hoc networks for delivering information on the movements of zebras to distance researchers. However, the collars are quite bulky and heavy, which makes them unsuitable for the monitoring of birds.

Lighter weight approaches have been used to track badgers while underground [51, 53]. These approaches rely on a fixed set of motes located aboveground to track the badgers using RFID tags or magneto-induction. Since the motes must be deployed in a relatively dense pattern above the badgers’ habitat, these approaches are ill-suited for
tracking highly mobile birds.

Specific approaches have been tailored to bird monitoring. One early example used WSNs to monitor the micro-climates of bird nests [50]. In this work, devices were placed near bird nests to track the small scale environmental conditions of the nests. Thus, the devices did not directly monitor the birds, but rather their environment.

Several other efforts have tracked birds and other animals by places equipment directly an animal. The approaches that most closely resemble our work utilize either short range radios, cellular communications, or satellite transmitters to send information to researchers [9, 30, 48, 84]. The approach taken in our work differs from these prior efforts in two critical areas. First, our approach utilizes multiple means of communications. The use of multiple communication channels allows the device to utilize the strengths of multiple approaches, while mitigating the shortcoming of using a dedicated communication method. This allows our system to function effectively during the periods of high mobility during migration, as well as periods of low mobility in nesting grounds. Second, our device allows researchers to infer detailed behavioral patterns from the sensor readings, which exceed the capabilities of previous devices.

### 2.3 Whooping Cranes

Whooping Cranes are a species of crane that is indigenous to North America. As seen in Figure 2.1, the Whooping Cranes are a large, predominantly white, bird. Whooping Cranes stand approximately 1.5m in height, and adult males typically weigh 7kg. In the spring of 2011, approximately 575 of these birds remained in the world. The remaining wild population is split between two main groups. These groups are differentiated by their migratory paths and nesting grounds. The Aransas-Wood Buffalo National Park (AWBP) population migrates annually between Aransas National Wildlife Refuge
in southern Texas, and Wood Buffalo National Park in Alberta, Canada. The second population migrates between Florida and Wisconsin. This population has been created from chicks taken from the AWBP population.

![Figure 2.1: Captive Whooping Crane](image)

Whooping Cranes have been the focus of extensive conservation efforts over the past years [42, 43, 84]. The migration periods of these birds is of particular interest for conservation efforts, as this is when most of the crane mortalities occur. Over the past decades, many approaches have been used to track these birds during their migrations. These approaches include leg-banding, VHF transmitters, and satellite tracking. All of these attempts have fallen short in terms of quality of collected data, latency, and cost. A more detailed analysis of these approaches will be given in Chapter 3, but these shortcomings motivate the need for better platforms to track the Whooping Cranes on a continental scale.
2.4 Testing

Testing WSNs is a notoriously difficult proposition [14, 24]. Several factors contribute to the difficulties in testing WSNs.

**Environment:** A large part of the appeal of WSNs is the ability to deploy them to remote or difficult to access regions. This capability is a liability when it comes to testing and maintaining the network. It may be difficult to gain access to the motes to check their status. This difficulty can come from either a difficult physical environment, or because of unreliable communications. In the case of network failures, it may be necessary to gain physical access to remote systems. In one case, a network failure resulted in 4 days of downtime while researchers redeployed a network on a volcano [86].

In the case of tracking Whooping Cranes, environmental factors are especially hard to recreate. The deployed motes will face an extremely large set of environmental conditions, from potentially freezing temperatures, hot weather, low energy, and submersion in water. Recreating these environmental conditions in a laboratory setting is incredibly difficult and time consuming.

**Connectivity:** The communications channel in a WSN can form a bottleneck to testing. Transmitting debugging information over wireless channels affects both the communication links and energy reserves of a WSN [22, 40, 90]. The very act of debugging a sensor network can change its behavior, and hide a bug from a developer.

**Interaction Faults:** WSNs are highly distributed systems, and many software faults may be caused by the interaction between motes [40]. These faults are especially difficult to locate, capturing and recreating a network’s state at a particular instant in time is very difficult.

**Hardware Constraints:** The need to reduce the physical size of the motes can result in debugging or testing interfaces being removed. Moreover, the limited peripheral inter-
faces on the embedded microcontrollers may make it difficult to interface to the system. For example, UART lines are a convenient method of interfacing to a microcontroller. However, the microcontroller may only include one set of these lines. If the UART is used to interface with a sensor, and there is a defect involving this interaction, it may be complicated to retrieve information related to this defect.

Developing systems to track highly mobile animals means all of these challenges must be addressed. The high mobility, and long term deployments, of animal tracking applications makes it extremely expensive, if not impractical, to test the device in the exact conditions that a device will encounter. Moreover, a high degree of reliability is needed in these systems, so that animals are not needlessly encumbered by failed devices. This motivates the need for extensive testing with effective methodologies.

2.5 Energy Management

The efficient use of energy reserves dominates almost every aspect of WSN development [11]. From communications to sensing, every action a mote takes will diminish its limited energy reserves. The three main users of energy in a mote are the communications, processing, and sensing. Traditionally, communication has been assumed to dominate the other two. Accordingly, many MAC and routing protocols have been developed to optimize the communications protocol stack to reduce this cost [21, 81, 82]. However, it will be shown in Chapter 4, that the sensors can utilize a significant portion of the energy budget.
2.6 Conclusions

WSNs can be an effective means of tracking wildlife. However, when tracking highly mobile wildlife, such as Whooping Cranes, significant challenges are imposed. The high mobility of the birds makes it difficult to maintain network connectivity to the birds. The extended mission deployments place a heavy burden on the energy reserves of the system. Given the inaccessibility of the motes, and the environmental conditions they will face, testing the motes prior to deployment can be difficult. In the following chapters, these challenges will be addressed.
Chapter 3

Whooping Crane Monitoring Project

In this chapter, background information on Whooping Cranes, their conservations efforts, and behaviors are given. This background information motivates the need for the CraneTracker100. Furthermore, it helps motivate the design choices and work described in Chapters 4, 6, and 7. The goals and requirements for the final platform are presented.

Most of the background information on the Whooping Cranes and their behavior has been gained through personal communication with the research personnel at the International Crane Foundation (ICF) [4] and The Crane Trust [8]. Unless otherwise noted, the information presented is assumed to have come from these personal contacts.

3.1 Whooping Crane History

Whooping Cranes are indigenous to North America. While once common, hunting pressure eventually forced this birds to the brink of extinction. In the 1940s, the Whooping Crane population was estimated to be approximately 15 breeding pairs. As a result, many conservation efforts have been made to help the population recover and return to its original habitat. Some of these conservation efforts include preserving the birds’
habitat, reintroducing birds to areas they once inhabited, and rearing chicks in captivity for reintroduction to the wild.

These efforts have had some impact in helping the population recover. There are now approximately 600 birds remaining in the wild. Whereas the wild population was once contained in the AWBP population, a second flock has been established. This second flock migrates between Wisconsin and Florida. Despite these efforts, the crane population remains extremely fragile, and its recovery has been slow.

3.2 Whooping Crane Migrations

Whooping Cranes annually migrate between their southern and northern breeding and nesting grounds. Figure 3.1 shows the migration path for AWBP cranes. In the breeding and nesting grounds the cranes are loosely spread out over a sizable region, but are not highly mobile. In the northern grounds the birds hatch and rear their chicks. After the chicks have matured into young juveniles, the families migrate to the south. During the migration, the families travel as a unit, but rarely associate with other families. Typically, the birds begin their southern migration in October or November. Completing the migration takes one to two months. The birds stay in their southern nesting grounds until March or April, and then return to the north. The northward journey typically takes around one month. It is important to note that the birds do not spend this entire migratory period flying. The cranes instead fly in bursts, where they travel 950km/day at speeds approaching 105km/h. Doing so consumes a significant amount of energy, so they must then rest for several days and forage for food to replenish their energy reserves.

The cranes have a highly distinctive method of flying [43]. In order to conserve energy, the birds avoid flapping their wings. Instead, the birds spend most of their flight
time searching for thermal updrafts. Once these updrafts are located, the birds hold out their wings rigidly. This allows them to ascend the thermal updrafts without exerting significant energy. Once the birds have attained sufficient altitude, they exit the updraft and glide for kilometers at a time, again without flapping their wings.

Even with these energy saving flight mechanisms, the Whooping Cranes still expend a tremendous amount of energy during flight. To compensate for the activity, the birds gain up to 25% of their body weight prior for departing on their migration. The cranes also spend time during the migration foraging to replenish their energy reserves. However, the factors influencing the choice of roosting spots are not well understood. Tracking where the birds fly to is thus of high importance for conservation efforts. Moreover, understanding the motivation for choosing a roosting spot motivates not just tracking the birds, but also monitoring their environment and behavior.
3.3 Whooping Crane Behavior

Migrations are not the only time when cranes have interesting behavior that need to be monitored. Attempts have been made to model the energy expenditures and intake of animals [17, 32]. Now, researchers have turned their attention to Whooping Cranes. By developing an accurate energy model of a bird, researchers can estimate how much food will need to be in the environment to sustain the bird. This knowledge can then be used to predict the roosting locations of the bird, and how many birds an area will support.

Current efforts to accomplish this goal are extremely labor intensive. The birds are typically kept under close surveillance, and their actions are recorded by a researcher. Each action is categorized by how much energy it takes, and from this the energy expenditures of the bird can be estimated. Recording an animal’s energy intake can be a complex process. First, the amount and type of food that the bird consumes must be measured. The nutritional content of these sources must be estimated. Finally, the bird may expend varying amounts of energy to acquire the food, as some sources are more easily obtained than others. For instance, grain lying on the ground may be easily obtained. Small animals such as toads and crabs may offer a crane more nutritional value and calories, but can be more difficult to obtain.

Given the amount of labor it takes to perform this type of analysis, it is highly desirable to streamline the process through the use of automated monitoring techniques. Prior studies have shown that simple measurements with accelerometers can classify the behavior of a crane with an accuracy rate of 92% [74]. This impressive accuracy was achieved through a simple Linear Discriminant Analysis (LDA). However, such monitoring efforts have been conducted on short time scales, and the energy consumption at the high sensing rate used is not sustainable for long periods of time. Thus, novel methods of inferring the birds’ behavior must be constructed in order to develop higher precision
models of a bird’s energy budget over its entire lifetime.

### 3.4 Mortality

Determining the location and causes of crane mortality is of great importance to conservation efforts. The process of collecting this information is complicated by several causes. First, the cranes experience a much higher mortality rate during migration, compared to when they are in their long term nesting and breeding habitats. It is much more difficult to monitor and track the cranes during their high mobility periods, thus determining causes of death is harder. Second, when a crane dies there is a limited window of opportunity to locate the crane and study it. Both the climate and scavenging predators can quickly destroy evidence of why a bird died.

Detecting these events is not easy. The readings from sensors on the crane tend to be very noisy because of how they are attached, and how the environment affects the readings. Moreover, the sensors used cannot interfere with the bird or its health, which in practice means a bird’s health must be inferred from indirect measurements. There is also a high penalty for falsely reporting that a crane has died. When a crane is suspected to have died, researchers may have to travel to remote locations in an attempt to retrieve the corpse. The field researchers’ limited resources are squandered when this happens, and the tracker may waste its own precious energy reserves in reporting that a crane has died.

### 3.5 Prior Tracking Efforts

Whooping Cranes have been the focus of several prior tracking attempts [42, 43, 84]. The first attempts used colored bands that were attached to a bird’s leg [42]. These colored
bands uniquely identified the bird and made it possible to easily distinguish one bird from another. When migrations started, ground based spotters were alerted to watch for the banded birds along the migration path. Additional spotters actively followed the birds in vehicles. Each sighting of a bird was recorded for later analysis.

This visual tracking system has many obvious drawbacks. First, maintaining visual contact with the birds during migratory periods is extremely difficult, and prone to errors. This leads to missing and incomplete observations sets, which lowers the quality of the collected data. Second, the labor intensive nature of this approach makes it very expensive to conduct extensive studies.

The visual tracking system was eventually supplemented with Very High Frequency (VHF) transmitters [43]. These transmitters are placed on the legs of birds, and use a simple timing circuit to intermittently broadcast a signal. This location of this signal is triangulated using radio direction finding equipment. This method is a great improvement over visual tracking, since the VHF signals can be heard tens of kilometers from the bird.

(a) Length of VHF transmitter

(b) Height of VHF transmitter

Figure 3.2: VHF tracker. 117mm x 30mm x 15mm. Not pictured: 29 cm antenna.
A VHF transmitter that was recovered from a prior wildlife study is shown in Figure 3.2 [1]. This tracking device was attached to the back of a bird. To do this, cloth loops were passed through the holes seen at the left and right ends of the device. The 29 cm long antenna (not fully shown) extended down the bird’s back. This device is activated by removing the magnet clearly shown in Figure 3.2(b).

![Magnet](image1.png)

(a) Length of satellite transmitter

![Magnet](image2.png)

(b) Height of satellite transmitter

Figure 3.3: Satellite tracker. 64mm x 27mm x 30mm. Not pictured: 17 cm antenna.

However, VHF transmitters are far from an ideal tracking solution. The most fundamental flaw of these designs is the VHF transmitters double as both the localization and communication mechanism. Therefore, when communications are lost with a device, information on the bird’s location is also lost. The transmitters suffer from range limitations, and it is possible to lose contact with the birds for extended periods of time when they are in rugged, rural areas. Furthermore, tracking the bird’s with these devices is still labor intensive, as it requires researchers to be in constant communication range.
with the cranes.

The current state of the art tracking devices used for Whooping Crane research combine a GPS receiver with a satellite transmitter [84]. This combination addresses several of the defects of the VHF transmitters. First, the satellite transmitter removes the need for researchers to follow the birds into the field. Instead, the data can be retrieved from through the Internet at any convenient location. Second, the data from the GPS receiver can be cached on board the device. If there is a short term communication failure, these recorded positions are still transmitted. Thus, communication difficulties result in a larger latency in determining the bird’s location, rather than losing this information entirely.

An example of a satellite tracker is shown in Figure 3.3 [7]. This particular transmitter has been mounted to a leg band. Half of this leg band can be seen in Figure 3.3(b). The antenna for the device extends to the right for 17 cm.

The satellite transmitters still leave room to improve upon. The largest drawback to the satellite transmitters is their high cost. Each device used in monitoring Whooping Cranes costs approximately $4,300. Additionally, satellite communication bandwidth must be purchased to operate the trackers, which adds another $700 to the annual cost of operating each transmitter. These high upfront and long term costs make satellite tracking unattractive for large studies. The communication energy costs is also quite high in these devices, which leads to depleted batteries and periods of unavailability. Moreover, the communications tend to have high latency. Field experiences have shown 48 hour delays between recording a GPS location and sending the data to researchers. Finally, the long antennas the VHF and satellite transmitters need are a potential point of failure. The cranes tend to break pieces of the antenna off over time, which degrades their performance [84].

While the location information collected by the GPS is better than that of VHF trans-
mitters, the satellite transmitters still fail to deliver many types of information that are desired by researchers. In fact, the only other sensor on the device is an accelerometer. The data from this device is heavily filtered and coarsely quantized. It is impossible to use this data to analyze crane behavior. These sensor limitations also make it difficult to determine whether or not the bird is alive if it is staying in one location. These limitations motivate the need for better platforms to track and monitor the cranes’ behavior with better communication and sensor capabilities.

### 3.6 Project History

The work presented in this thesis is best understood with knowledge of the project history. Rahul Parundare originally began working on wildlife tracking as a project for the Wireless Sensor Networks (CSCE896). Rahul collaborated with Dr. Felipe Chavez-Ramirez of the The Crane Trust, in Wood River, NE. Paul Bennett and I continued developing the project after the class ended. After Dr. Chavez moved on to another position, we continued working with Walter Wehtje at The Crane Trust. Eventually, we were put into contact with the International Crane Foundation at Baraboo, WI. This partnership proved extremely fruitful. Anne Lacy and Mike Engels helped expand the scope of the project past simple tracking, and into behavior monitoring. This type of research is proving valuable to ecologists working to conserve the Whooping Crane population.

### 3.7 Whooping Crane Monitoring Goals

The preceding sections of this chapter have described the attributes of cranes, WSNs, and the prior attempts at tracking them. To improve on this prior efforts, the following
goals are set for a new hardware platform to track and monitor the cranes.

**Migration tracking:** Information needs to be collected on where the birds fly and nest during the migration periods. Collecting this information will create insights on what habitats are desirable, and why. This information can be used to preserve these locations for future migrations, and the impact of human change on the birds’ habitat. If the information is collected with low enough delay, then it may be possible to recover the bird’s when they perish, and ascertain a cause of death.

**Reduced latency:** As stated in the previous goal, low-latency is a high priority goal of this project. This low communications delay is needed not only in the case where a bird dies, but also when it may be in trouble, or behaving unexpectedly. If the delay between collecting information and reporting it to researchers is low enough, then researchers can be dispatched to observe the cranes in the field.

**Behavior and movement characterization:** Scientists lack quantitative information on bird’s flight movements and daily routines. The existing data has been collected using opportunistic visual observations. By quantifying this data, it becomes possible to estimate a bird’s energy budget. This information can in turn be used to determine the needed food reserves in a bird’s habitat. Movement data can also be used to assess the impact of climate and habitat change on the cranes. Finally this data also indicates the health of a bird.

**Long-term operation:** Forming statistically valid conclusions requires multiple years of data from individual cranes. Once deployed, it is extremely undesirable to recapture a bird. Doing so may impact the crane’s behavior, and thus taint the results. Therefore, any tracking device must operate unattended for years. This necessitates energy efficient operation and careful power supply design.

**Flexible operation:** Currently, there is insufficient baseline data to highly customize the software, sensing, and communication routines to a crane’s behavior. However, once
this data is collected, the project’s focus will shift to optimizing the software to maximize the data quality. The platform should be flexible enough to support these future goals, so that the project can progress without any major interruptions.

**Alternative mounting:** Prior crane tracking experiments have attached the devices to the legs of the birds. Ecologists have conjectured that this adversely affects the birds by hindering copulation and unbalancing the specimens. Consequently, future tracking devices should support attachment to the back of a bird.

**Low cost:** Several commercial satellite tracking systems come close to meeting the sensing, communication, and lifetime needs. However, they are prohibitively expensive. The complete system cost must be minimized as much as possible.

### 3.8 Whooping Crane Monitoring Requirements

While the preceding goals must be achieved for the new platform to be a success, there are restrictions on the device, and strict sensing requirements to be met.

**Weight Restrictions:** Using guidelines developed for other birds, the tracking device and harness equipment combined must weigh less than 2% of a bird’s bodyweight [80]. The average Whooping Cranes weighs 6kg, which leads to a weight of 120 grams.

**Communication Latency:** While there is no hard deadline on when information must be delivered, 24 hours is the target of this project. If sensor readings are delivered in this time frame, then causes of death may be determined if the crane perishes. This low communications delay also allows field researchers to follow and observe the birds, if desired.

**Sensing:** Tracking the birds requires determining their position twice a day. Ideally, the bird’s location would be determined once during the day, and once during the night. This pattern reveals the nesting habits of the birds, and information on how the bird’s
The requirements for monitoring a crane’s behavior are less well defined, because of the lack of prior research in this area. For the initial research conducted by this project, a target sampling rate of 0.5Hz was used. This sampling rate is assumed to be fast enough to analyze crane behavior.

### 3.9 Conclusions

With this knowledge of crane behavior, information on prior tracking efforts, and goals for new tracking experiments, it is possible to develop a new platform. This new platform is capable of collecting more detailed experimental data, maintain a higher degree of connectivity, and reduce the deployment costs.
Chapter 4

Development of the CraneTracker100 Hardware

To achieve the goals of Chapter 3.7 and meeting the requirements of Chapter 3.8, several hardware platforms were constructed and evaluated. A commercial product was first evaluated, but was found to be deficient. This motivates the need for a custom tracking platform, the CraneTracker100. This platform underwent several iterations before reaching its final form. In this chapter, the evolution of this platform is described.

4.1 MTS-420

The first platform that was evaluated for use in the crane tracking project was a commercial offering from Memsic [5]. Memsic offers a variety of sensor platforms. Most of Memsic’s products utilize a design where the communication, processing, and data storage is implemented on one circuit board. For clarity, this will be referred to as the mote. These motes are can be attached to another circuit board containing the sensing devices. This component will be referred to as the sensorboard in the following.
4.1.1 MTS-420 Description

Initially, an MTS-420 sensorboard [56] and Iris mote [54] were evaluated for use in the crane project. This combination was chosen for several reasons. The Iris and MTS-420 are well known and widely used inside the WSN research community. Since the devices are so widely used, there is a great deal of documentation and resources available. The widespread use of this device includes the Cyber-Physical Networking (CPN) laboratory [2], of which the author is a member. The CPN laboratory has many of these devices available, which made it a convenient prototyping device.

![MTS-420 and Iris](image)

Figure 4.1: MTS-420 Tracker

More importantly, the MTS-420 and Iris offered sensing and communication abilities that appeared to match the needs of the project. At the beginning of the project, only the tracking of the Whooping Cranes was considered a necessity, and any other data collected was considered as side benefit, and not part of the primary mission. The Iris contains an Atmel Atmega1281 processor, RF230 802.15.4 compatible radio, and AT-45 flash memory. The Iris is powered via AA batteries, distributes this power to any
attached sensorboard.

The *MTS-420* is equipped with a LEA-4M GPS receiver [79] that requires an external antenna to be attached. The *MTS-420* is equipped with several other sensors, namely a 2-axis accelerometer, magnetometer, barometer, thermometer, humidity sensor, and light sensor. This package has the same width and length as the *Iris* mote, so the combined package is compact. The *MTS-420* and *Iris* are shown in Figure 4.1.

The *Iris* and *MTS-420* were augmented prior to deployment in two ways. First, the entire package was attached to a leg mount. Additionally, they were potted in epoxy. This epoxy not only waterproofed the devices, but also securely fastened the two together.

### 4.1.2 *MTS-420* Evaluation

The *MTS-420* and *Iris* combination rapidly proved to have severe shortcomings. First and foremost, it relied on a pair of AA batteries for power. Preliminary tests and power modeling showed that AA batteries lacked the capacity to support the desired sampling rates for the mission duration. Assuming the device only acquired one GPS fix a day, used the radio for 5 minutes, and sampled the other sensors 10 times a day, a simple power model projected that the device would only last 205 days. Moreover, this model did not account for such factors as temperature on the battery, which made the lifetime estimates optimistic.

This lack of power was exacerbated by the outdated GPS receiver used by the *MTS-420*. The datasheet of the receiver states that the peak current draw of the device is 70mA [79]. The GPS proved to be slow when acquiring satellites, and often failed to achieve a fix when tested in the urban setting of the campus. In fact, testing revealed that approximately half of the fixes acquired by the GPS took over 1.5 minutes to obtain. The external antenna used by the GPS is bulky and heavy, and proved to be difficult to
waterproof and mount to the bird.

In addition to the limitations of its GPS, the MTS-420 is crippled by its 2D accelerometer. When mounted on the bird, the orientation of the device is variable and difficult to determine. This is especially true of the legs, which are tucked behind the bird during flight, and extend down when the bird is on the ground. Since the device is free to move in three dimensions, a 2D accelerometer is incapable of characterizing the movements. The magnetometer is similarly limited.

The Iris mote possesses insufficient communication capabilities for the mission. Testing with the mote revealed that a realistic range of 100 meters between motes could be expected. With a high gain Yagi antenna, this range is extended to almost 500 meters. Even with this long range, it is insufficient for tracking cranes. The cranes’ flight paths and roosting spots are too unpredictable to set up base stations at during the migration. Even at the summer and winter roosting and breeding grounds, the cranes cover a large enough area that it would be costly and labor intensive to cover with 802.15.4 coverage.

4.2 CraneTracker100.v1

After evaluating the MTS-420, it was clear that it did not meet the project requirements. Therefore, a custom hardware platform, the CraneTracker100.v1, was developed. This platform sought to rectify the deficiencies of the MTS-420 by including enhanced communications, sensing, and power capabilities.

4.2.1 CraneTracker100.v1 Description

Upgraded communication capabilities were the first priority when developing the CraneTracker100.v1. The unpredictability of the continental scale migratory paths made choosing a communications technology difficult. Short range solutions, such as 802.15.4, are
impractical because of the need for pre-deployed infrastructure along the migratory paths. However, longer range solutions, such as satellite communications, are expensive in terms of price and energy. A Telit GM862 cellular module was chosen to solve this problem [18, 77].

Cellular communications are an attractive option in this project. First, cellular units using GSM (Global System for Mobile Technology) can communicate with base stations at a range of 30km. The power requirements of the cellular devices are high, as shown in Figure 4.2. In this figure, the power consumption of the different communication devices, and their modes, is shown. The power required by the Iris while active, is shown first as a reference. The RF230 radio used by the Iris consumes significantly more power than the microcontroller. Also, the power consumption of the RF230 is approximately the same, whether it is transmitting or receiving information. The final three bars in the figure show the power requirements of the GSM. These requirements are much higher than that of the RF230. Significantly, the GSM consumes far more energy while actively transmitting, than when it is idle on the cellular network.

GSM cellular devices require a Subscriber Identity Module (SIM) card to fully function, which must be accounted for in the design process. The GM862 simplifies this design by including the SIM card holder in the cellular module. However, the GM862 requires an external antenna to function properly. The GM862 connects to the Crane-Tracker100.v1 through a 50 pin connector [63].

The cellular modem offers two practical methods of transmitting data for this project. The first is to use the General Packet Radio Service (GPRS). GPRS is a packet switched data transmission protocol that is part of the GSM standard [10]. The second is to encode the sensor data into a format compatible with the Short Message Service (SMS) format. While the GPRS protocol offers much higher bandwidth than SMS, it also consumes more energy and requires a higher quality of service (QoS) when compared to SMS.
Additionally, GPRS is much more complex to implement and requires more software testing to be reliable. These issues, and the drivers for supporting both modes, were investigated by Paul Bennett [18]. For the initial steps of the crane tracking project, it was decided to use SMS messages, and investigate using GPRS for future work.

The sensing capabilities of the CraneTracker100.v1 were also enhanced, when compared to the MTS-420. The first enhancement was adding a more modern, compact, and better performing MN5010 GPS receiver and antenna [6]. This upgraded GPS solution reduced the peak current to 46mA. Testing also revealed it was much quicker to acquire fixes, and more consistent in acquiring its position in an urban environment. Using this receiver also benefited the project from a software aspect. The MN5010 uses the same
NMEA0183 communication protocol as the receiver on the MTS-420, so much of the driver code could be shared between the platforms.

The GPS receiver was not the only sensor enhancement. A HMC-6343 solid state compass was included with the CraneTracker100.vi [37]. This solid state compass provides heading, pitch, and roll information. Moreover, it includes a high precision 3D accelerometer and temperature sensor. The combination of accelerometer and compass data allows for an accurate characterization of a bird’s movements, and behavioral information. The temperature sensor collects information about the ambient environment, which is also valuable. All of this information is delivered via I²C lines, using a driver Paul Bennett developed.

The CraneTracker100.vi still uses the Iris for power, processing, storage, and the 802.15.4 radio. The CraneTracker100.vi interfaces to the Iris using the same type of connector used in the MTS-420 [3]. The combination of 802.15.4 radio and GSM is particularly powerful. The cellular modem enables the CraneTracker100 to exploit predeployed cellular infrastructure during its migration to maintain connectivity. Once the cranes reach their establish roosting and breeding grounds, then the 802.15.4 radio can be used for more power efficient communications.

The power supply of the system was upgraded. The GM862 cannot reliably operate at the voltage levels supplied via the AA batteries connected to Iris. This problem is compounded as the batteries discharge, and provide a lower supply voltage. To correct this problem, a single cell Lithium-Polymer (LiPo) battery was used instead. The LiPo batteries used in this project have a nominal supply voltage of 3.7V, and a peak of 4.2V, which is what the GSM is designed to operate at. The power supply was also augmented through BJT transistors that control the power to individual components. These transistors are controlled by the general purpose input-output (GPIO) lines of the Iris. The CraneTracker100.vi sensorboard is shown in Figure 4.3.
4.2.2 CraneTracker100.v1 Evaluation

The CraneTracker100.v1 presents a significant advancement over the MTS-420. The compass allows the bird’s flight mechanics to be accurately characterized. In addition, the GPS is faster and more reliable in achieving satellite fixes ($\approx 80$ s faster on average), as well as weighing significantly less. The GSM also proved to be a reliable mechanism for communicating data.

While an important milestone in the project, the CraneTracker100.v1 suffered from many shortcomings. First, while the compass and Iris tolerate the voltage supplied by the LiPo battery, the voltages are at the limits of the stated maximums for the devices. Second, as can be seen by the connector in Figure 4.3(a), the Iris was oriented at a right angle to the rest of the sensorboard. This would have made the assembled package very awkward when placed on cranes. Finally, a 500mA-hr LiPo battery was to be used with this system, due to weight requirements. This small capacity, coupled with the high energy needs of the GSM, made it unlikely the device would last for more than a few months without depleting its energy.

The GSM also posed significant challenges. First, the size of the device was quite large compared to the other components, which limited how small the system could be
made. The GSM also requires an external antenna, which once again adds significant weight and makes it much more difficult to package the device. Finally, the GSM is extremely power hungry. While the average instantaneous current draw is reasonable (19mA), the module requires that the power supply be capable of handling 2A spikes. Finally, the GM862 uses an MMCX (Micro-Miniature Coaxial) connection to connect to its external antenna. This external antenna has the same drawbacks as the external GPS antenna used by the MTS-420.

4.3 **CraneTracker100.v2**

While the CraneTracker100.v1 was an important step in this project, it still did not address all of the challenges that had to be overcome. Most importantly, a high capacity LiPo battery is incapable of meeting the mission duration requirements on its own. After assessing the CraneTracker100.v1, a second revision was made, called the CraneTracker100.v2, as described next.

4.3.1 **CraneTracker100.v2 Description**

The CraneTracker100.v2 primarily focused on addressing the power challenges that the CraneTracker100.v1 failed to solve. The GM862 cellular mode, HMC-6343 compass, and Iris mote were retained from the CraneTracker100.v1. The power supply was upgraded by adding a solar panel, improved switches for sensors, and voltage regulation. The orientation of the 51 pin connector to the Iris was rotated 90°, so that the device was narrower and easier to attach to a bird. The new system can be seen in Figure 4.5.

The addition of a solar panel caused the most challenges with this design. LiPo batteries must be charged in a specific manner, or else they will be damaged. The simplest method of meeting these requirements is through a dedicated charging chip.
To this end, a Microchip MCP73831 charging integrated circuit (IC) was added to the design [60]. This IC is designed to charge single cell LiPo batteries, such as the ones used in the crane tracking project. In particular, it prevents the battery from damage from overcharging. It also will safely recharge the battery in case it becomes overdischarged.

The MCP73831 is capable of charging the battery from a wide variety of energy sources. A flexible solar panel, MPT4.8 – 75, from PowerFilm Solar was chosen for this project [69]. The solar panel is shown in Figure 4.4. This solar panel has many desirable characteristics related to this project. First, the solar panels weights only 1.9g, so it works well with the weight requirements of the project. Second, this panel is rated to produce 4.8V and 50mA of current. This high voltage level makes it simple to charge the LiPo batteries used in the CraneTracker100.v2. Finally, it is flexible. This is a highly desirable property. When the CraneTracker100.v2 was being developed, it was still thought that the device would go on the cranes leg. A flexible solar panel can be unobtrusively wrapped around a crane’s leg.

A 3.3V voltage regulator was added to the system [49]. This voltage regulator supplied power to the Iris, compass, and GPS. The GM862 still connects directly to the
batteries to meet its voltage requirements. Using this voltage regulator moved the supply voltages of the sensors and Iris back to the designed operating points, which reduces the chance of damage to them. This particular voltage regulator is capable of providing 300mA of current, can accept input voltage from 1.8V to 20V, and has a dropout voltage of less than 270mV.

The next important improvement was the replacement of the BJT transistors with purpose-built power switches \[59\]. These power switches are very compact, and do not require the biasing resistors that the BJTs require. The switches are designed to be operated with the logic levels the system uses, and can source over 2A of current. The switches are based on MOSFET technology, which is more power efficient than the BJT transistors in this application.

![Top Side](image1.png) ![Bottom Side](image2.png)

Figure 4.5: CraneTracker100.v2

### 4.3.2 CraneTracker100.v2 Evaluation

The CraneTracker100.v2 was an incremental step in achieving the goals of the system. The sensors remained the same as those in CraneTracker100.v1. The voltage regulator and switches improve the robustness of the system and efficiency. These leaves the solar panel as the major novelty, and it is challenging to evaluate. When deployed in on an actual bird, the solar panel can be obscured by feathers, wings, and foliage. The solar
panel will not be facing the sun at all times, so it can be difficult to evaluate how much power it will actually provide.

As shown in Figure 4.4, the solar panel consists of individual cells arranged in rows and columns. The panel was evaluated in Lincoln, NE on a sunny day in August 2011. The panel was evaluated by measuring the open circuit voltage across the sides of the panel. After the voltage was measured, a $1\Omega$ resistor was placed across the terminals of the device, and the current passing through the resistor was measured. Various combinations of cells were covered with opaque electrical tape to gauge the impact of the solar panel being covered in the wild. Prior to the experiment it was known that the width of the solar panel contributed to the voltage, while the height of the solar panel increased the current.

Figure 4.6: Solar panel experimental setup, uncovered cells are green, covered cells are red
The results from the August experiment with the solar panel are shown in Figure 4.6. As expected, covering a column of cells decreased the voltage, while covering the rows decreased the voltage. The results show the impact of covering the cells can be quite dramatic. Table 4.1 contains quantitative results of the experiment. The experiment number in this table corresponds to the order in which it appears in Figure 4.6. The results highlight the potential difficulties in using solar panel to monitor wildlife. The energy provided by a solar panel is already a stochastic process. When an animal is capable of covering the device, or orienting it in a non-ideal orientation, it will further reduce the energy in an unpredictable manner. Testing on ‘proxy’ animals in later experiments would reveal the full impact of an animal’s behavior on the energy harvesting process.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Voltage(V)</th>
<th>Current(mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>5.95</td>
<td>65</td>
</tr>
<tr>
<td>b</td>
<td>5.78</td>
<td>53</td>
</tr>
<tr>
<td>c</td>
<td>4.56</td>
<td>64</td>
</tr>
<tr>
<td>d</td>
<td>5.81</td>
<td>65</td>
</tr>
<tr>
<td>e</td>
<td>5.89</td>
<td>66</td>
</tr>
<tr>
<td>f</td>
<td>5.65</td>
<td>42</td>
</tr>
<tr>
<td>g</td>
<td>3.53</td>
<td>47</td>
</tr>
</tbody>
</table>

Table 4.1: Quantitative solar panel results

### 4.4 CraneTracker100.v3

Using the accumulated experience from CraneTracker100.v1 and CraneTracker100.v2, as well as feedback from the ICF researchers, the CraneTracker100.v3 was created. Rather than adding new features, this revision focused on refining the sensors and power supply, as well as optimizing the form factor for deployment. This is the final version of the system that will be presented in this work, and thus will receive the most detailed description.
4.4.1 **CraneTracker100.v3** Description

The largest change in the **CraneTracker100.v3**, compared to the **CraneTracker100.v2**, was the replacement of the **GM862** cellular module with the Telit **GE865** module. As shown in Figure 4.7, this module is half the size (22mm x 22mm) of the **GM862** (44mm x 44). However, the **GE865** does not have an internal SIM card holder, and thus an external holder had to be added to the board [31]. The smaller GSM module, coupled with the flexibility of placing the SIM card holder, makes for a much more compact form factor. Additionally, the **GE865** and SIM card holder weigh 11 grams less than the **GM862**.

The **GE865** uses a different antenna than the **GM862**. Whereas the **GM862** connected to the **CraneTracker100.v1** and **CraneTracker100.v2** through a SMD connector, the **GE865** is a Ball Grid Array (BGA) package. The antenna of the device does not connect to the **GE865** through an MMCX connector, but rather a trace is ran from a solder point on the module to an SMA connector. A whip antenna is connected to the SMA connector with a right angle connector [71]. In this configuration, the antenna is parallel to the main board, which makes the **CraneTracker100.v3** very compact.

![Figure 4.7: Size difference between GE865 (on left) and GM862](image)

The power supply circuitry was once again enhanced in the **CraneTracker100.v3** com-
pared to the CraneTracker100.v2. Initial testing with the CraneTracker100.v2 revealed that the solar panel could not be expected to always meet the system’s energy needs. More concerning were cases of the battery completely discharging during testing, which can damage LiPo batteries. These results will be more extensively discussed in Chapter 7.

To prevent the LiPo battery from being damaged, a voltage detector was added to the power supply of the CraneTracker100.v3 [61]. This voltage detector controls the shutdown line to the system’s voltage regulator. When the battery voltage drops below 2.9V, an output line turns off the voltage regulator, in turn removing power to the Iris and sensors. The GSM is incapable of operating at this voltage level, so the entire system stops consuming power. This allows the solar panel to safely recharge the battery.

The voltage regulator of the CraneTracker100.v2 was replaced with a different model [58]. Two reasons motivated this change. First, the new voltage regulator was a lower dropout voltage, which wastes less power. Second, the package is simpler (8 pins vs. 6 pins), and much easier to hand assemble.

The voltage regulator on the CraneTracker100.v3 is not as capable as the regulator on the CraneTracker100.v2, but these differences do not impact the final system. Most significantly, the CraneTracker100.v3’s voltage regulator can only supply 150mA of current, compared to 300mA on the previous revision. This does not pose a problem, since all of the regulated components combined consume less than 80mA of current. Even with the lower maximum current, the new regulator provides surplus power that can be used for future expansion.

The regulator on the CraneTracker100.v3 does not accept as large of a range of input voltages as the CraneTracker100.v2 does, but the 5.5V maximum input voltage is still far above the 4.2V maximum of the LiPo battery. It also has slightly higher output noise (30µV), but the power line filtering on the other components is capable of dealing with this noise.
Significantly, the power supply was upgraded to include monitoring circuitry capable of measuring both the voltage and current of the solar panel, and the voltage and discharge current of the battery. This was partly done out of necessity, as previously the battery voltage was measured through the internal voltage reference of the microcontroller. Since the CraneTracker100.v3 regulates the voltage to the microcontroller, the value will always be 3.3V. Therefore, a voltage divider is connected to an ADC line so that the microcontroller can measure the true battery voltage.

Monitoring the voltage of the solar panel is accomplished by using a voltage divider to reduce the input voltage to a range the Iris can measure. The input current of the solar panel, and the discharge current of the battery is measured via op-amps [52]. While these components are not necessary to carry out the biological research aspects of the project, they will prove useful in developing solar energy models for wildlife tracking. The total power supply design is shown in Figure 4.8.

The final improvement of the CraneTracker100.v3 was a change to the GPS receiver. A new receiver and antenna was added to the system [35]. This new solution uses a ceramic patch antenna placed on top of the receiver. This is a more compact solution (16mm x 16mm) than the receiver found on the CraneTracker100.v2. The ceramic patch antenna should perform better in real world situations than the small chip antennas found on the CraneTracker100.v2. The final CraneTracker100.v3 design is shown in Figure 4.9. A more conceptual view, presenting the different interfaces and communication buses, is shown in Figure 4.10.

### 4.4.2 CraneTracker100.v3 Evaluation

A comparison of the physical dimensions of the CraneTracker100.v3 to the prior platforms is shown in Table 4.4.2. The weight savings of the CraneTracker100.v3, compared to the
earlier versions, are due to the smaller form factor and compact antennas. These differences are visually shown in Figure 4.11, which shows all of the devices. The weights
Figure 4.10: Hardware component architecture

Figure 4.11: Tracker platforms. Clockwise from top: Satellite tracker, VHF transmitter, CraneTracker100.v3 CraneTracker100.v2 CraneTracker100.v1 MTS-420.

of Table 4.4.2 include all of the antennas needed for the devices to function, but do not include the solar panel weight. While the CraneTracker100.v3 is slightly longer than
the MTS-420, its upgraded communication and sensing capabilities are well worth the trade-offs.

<table>
<thead>
<tr>
<th>Version</th>
<th>Weight (g)</th>
<th>X-Dimension (mm)</th>
<th>Y-Dimension (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTS-420</td>
<td>29</td>
<td>32</td>
<td>57</td>
</tr>
<tr>
<td>CraneTracker100.v1</td>
<td>48</td>
<td>45</td>
<td>63</td>
</tr>
<tr>
<td>CraneTracker100.v2</td>
<td>53</td>
<td>62</td>
<td>94</td>
</tr>
<tr>
<td>CraneTracker100.v3</td>
<td>33</td>
<td>32</td>
<td>84</td>
</tr>
</tbody>
</table>

Table 4.2: Tracker dimensions

![Graph showing current (mA) for different operating modes](image)

Figure 4.12: Microcontroller power under different operating modes [15]

The multitude of sensors, communication channels, and operating modes of the CraneTracker100.v3 creates a complex power consumption profile. The Iris uses an AT-mega1281 processor from Atmel [15]. This processor has a variety of operating modes, which can have a dramatic affect on the energy consumption of the device. In Figure 4.12, the three most common operating modes are shown, for a system operating at 3V. The difference between the lowest power sleep state, and the normal operating mode is 3 orders of magnitude. This huge difference motivates the need for the software to be power aware, and force the microcontroller into a low power sleep state whenever possible.
Unlike many WSN designs, where the power consumed by the sensors is assumed to be minimal, the CraneTracker100.v3 sensors consume a significant amount of power when operational. As seen in Figure 4.13, the HMC-6343 compass consumes slightly more current than the microcontroller used by the Iris. An even more dramatic difference is seen by examining the GPS power profile. When the GPS is attempting to locate its position, it consumes 48 mA. After the location has been determined, the operating mode changes, and the current consumption drops to 37 mA. Since the GPS can take minutes to acquire its location, collecting as much information as possible once it has acquired a fix is desirable. This energy profile may even motivate the application designer to leave the GPS on when multiple samples are needed, rather than turning the GPS on and off.

![Sensor power, with microcontroller for reference.](image)

The communication devices consume a significant amount of power as well. The values shown in Figure 4.14 were taken from datasheets [16, 76], with the exception
of the SMS power for the GSM. This value was empirically determined by measuring the average current consumed by the module while in a laboratory setting. This figure highlights the high power consumption of the GSM relative to the 802.15.4 radio on the *Iris*. Moreover, the SMS protocol may operate on a channel at a rate of only 765 kbit/s [10], while the *Iris* radio utilizes a channel with a 250 kbit/s rate [16]. The high energy and slow data rate of the SMS protocol relative to the *Iris* radio motivates the use of the 802.15.4 radio while the cranes are in their nesting grounds, so that energy may be conserved. The trade-offs between SMS communication and the other protocols on the GSM have not yet been investigated.

Figure 4.14: Communication power, with microcontroller for reference. 802.15.4 is in green, GSM is in red

In addition to the high cost of sending a message with the GSM, the cost of associating with the cellular network must be accounted for. Figure 4.15 shows the current consumption of the GSM module as it associated with a cellular network and sent mes-
sages. As can be seen by this figure, the association costs are not trivial.

![Figure 4.15: GSM current while sending SMS messages](image)

**Figure 4.15:** GSM current while sending SMS messages

### 4.5 Enclosure and Attachment

The electronics construction is only part of the hardware challenge of this platform. Once the electronics are designed and built, they must be packaged for deployment. This packaging must simultaneously protect the devices from the environment, avoid interfering with the operation of the device, and not injure or impair a crane.

#### 4.5.1 Leg Attachment

Initially, the trackers were to be attached to the cranes' legs, as previous studies have done. To accomplish this, the motes were to be potted in epoxy and attached to a
prior leg attachment device. Several attempts were made to pot the MTS-420 and Crane-Tracker100.v1 in epoxy, but these attempts failed. These failures can largely be attributed to the ungainly antennas used in these designs, and the irregular shaped caused by the combination of an Iris and sensorboard. It was extremely difficult to apply the epoxy in a thin layer in this shape, protect the antennas, and keep within the weight restrictions.

As a result of these experiences, a custom enclosure was fabricated that mated to the existing leg bands. This design is shown in Figures 4.16 and 4.17. This enclosure was fabricated using a selective laser sintering (SLS) process. The enclosure was tested on Wild Turkeys, and will be further discussed in Chapter 7.
4.5.2 Backpack Attachment

The leg attachment design was abandoned, partly due to problems discussed in Chapter 7, and partly due to general concerns of leg band mounting. Recently, concerns have been growing in the conservation community that leg band designs are negatively impacting the cranes, by unbalancing them, and making it awkward to mate.

These concerns forced the project to use an alternative design that has not been used on Sandhill or Whooping Cranes before. Rather than attach the device to the leg, it is
placed in a backpack mount. The backpack is attached to the bird using cloth straps. These cloth straps are looped over a crane’s wings. One loop is secured using threat that will slowly degrade over time, and cause the device to fall off. This loop is made prior to deployment, so that it can quickly be fitted to a captured crane. The second loop is left free until deployment, when the loop is closed using a permanent epoxy. The backpack mount is shown being mounted on a Sandhill Crane in Figure 4.18.

Figure 4.18: Backpack mounting

To protect the sensitive electronics from the environment, they are inserted into a tube made from fluorinated ethylene propylene (FEP) plastic. This plastic is transparent and does not degrade from sunlight, which ensures that the solar panel will operate at high efficiency for extended periods of time. It is also well suited for harsh environment, and resists harsh chemicals. These properties shelter the electronics for the duration of the deployment. A package of silicone desiccant is placed inside the tube along with the
electronics, so that condensation is absorbed and does not affect the electronics.

### 4.6 Platform Development Conclusions

The CraneTracker100 underwent a substantial development process, from using an off the shelf solution, to a custom sensorboard that possessed much greater communications and sensing capabilities. These abilities, coupled with the solar energy harvesting, make it possible to conduct multi-year wildlife monitoring missions. During these missions, the animals are able to maintain a high degree of connectivity over large geographic regions, using a variety of communication channels. These capabilities will be demonstrated in Chapter 7. A summary of the different designs, with their notable features, is given in Table 4.3.

<table>
<thead>
<tr>
<th>Version</th>
<th>Sensors</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CraneTracker100.v1</td>
<td>GPS, Compass</td>
<td>First custom sensorboard. Includes GSM. Changed to Lithium-Polymer battery.</td>
</tr>
<tr>
<td>CraneTracker100.v2</td>
<td>GPS, Compass</td>
<td>Incorporates solar recharging. Regulated power supply.</td>
</tr>
<tr>
<td>CraneTracker100.v3</td>
<td>GPS, Compass</td>
<td>Changed GSM module. Improved energy harvesting and monitoring. Changed GPS solution. Smaller and lighter weight.</td>
</tr>
</tbody>
</table>

Table 4.3: Tracker summary

These increased capabilities increased the complexity of the system. The original CraneTracker100.v1 designed used 19 components, and was easily assembled by hand. The CraneTracker100.v3 required 60 components. The boards were much more difficult
to assemble, as ball-grid array components are used, which are difficult to reliably place by hand. This complexity is reflected in the layouts of the devices, which is shown in Figure 4.19. Reliably assembling the quantity of boards needed for the experiments described in Chapter 7 proved difficult to do by hand, so an outside company constructed the boards using automated equipment.
Despite this increase in complexity, the cost of the CraneTracker100 was quite reasonable. Using a commercial PCB assembly business, 10 of the CraneTracker100.v3 can be constructed for ≈ $450 dollars. The Iris costs another 125. The total cost is an order of magnitude less than commercially available solutions. This reduced cost enables more trackers to be deployed for a given experiment, which increases the amount of data collected, and the quality of the results.

The reliability of the CraneTracker100.v3 was improved compared to the MTS-420. The redundant communication channels, energy harvesting, and power control circuitry combined to make the device robust, and capable of operating in a wide variety of areas. However, the hardware platform is only half of the solution in wildlife tracking. The platform needs software that is capable of taking advantage of all of its features. This software is described in the next chapter.
Chapter 5

Bringing the CraneTracker100 to Life: Software Development Process

The hardware platform developed in Chapter 4 requires software to fulfill its objectives. In this system, the software is conceptually divided into two levels. The lower level is responsible for interfacing to the peripherals on the sensorboard. The upper level manages the higher level functionality of the system, such as choosing when to activate components and communicate.

5.1 Division of Labor

The software of the CraneTracker100 was developed as a collaboration between Paul Bennett and myself [18]. Paul wrote the low level sensor drivers, the backend software, timer utilities. The upper level system manager was originally written by myself. Paul has taken over maintaining and upgrading this component as well. I was responsible for writing the flash memory component, the 802.15.4 radio software, upgrading the GPS driver, and the initial versions of the system manager. In addition, I re-wrote and
re-factored the upper level GSM manager component.

5.2 TinyOS

The system’s software is based up TinyOS [45]. TinyOS is lightweight operating system designed for sensor networks. TinyOS is designed to operate with limited resources and minimize the energy usage of the system. While the core of TinyOS is written in C, TinyOS applications are typically written in nesC [33]. nesC embodies an event driven design. The goal of this event driven design is to mitigate concurrency issues. WSNs are also assumed by the creators of nesC to be event driven systems, so nesC reflects the nature of WSNs.

The core of the CraneTracker100 software is based on a variant of TinyOS 1.x called MoteWorks. MoteWorks is produced by Memsic [5], to support their line of commercial WSN motes. The largest difference between standard TinyOS and MoteWorks is the radio protocol stack. MoteWorks implements a custom radio protocol that is designed to interface to Memsic’s WSN control and monitoring software. For the remainder of this work, unless specifically noted, all references to TinyOS should be understood as references to the MoteWorks software.

The decision to use MoteWorks was driven by its support for the MTS-420. MoteWorks has fully functioning and supported drivers for all of the MTS-420’s sensors and peripherals. Thus, it was a natural choice at the beginning of the project to quickly prototype and test software designs for the system. Even after transitioning to the CraneTracker100 platform, the MoteWorks software was retained. The software was retained because of its driver support for the Iris mote, radio communications stack, and GPS drivers. All of the GPSes used in this project communicate with the NMEA0183 protocol, so the drivers for these devices are easily portable.
5.3 Lower Level Components

Several significant upgrades to the TinyOS core have been made for this project. The default $I^2C$ TinyOS driver proved to be too slow to meet the timing requirements of the HMC-6343 compass. To fix this problem, it was replaced with an implementation that utilized hardware interrupts to control the communication timing. The compass driver software is represented as an abstract device to the upper layers of the system. This component then interfaces to the lower level component that contains the knowledge of the specific $I^2C$ commands needed to retrieve data from the HMC-6343. The upper level of this software is shown in Figure 5.1(a), while the lower level details are shown in Figure 5.1(b).

![Figure 5.1: Compass driver](image)

(a) Upper level compass abstraction

(b) Lower level, device specific, components

The MTS-420 GPS driver was upgraded at the beginning of the project so that the GGA sentence was understood by the driver. This sentence includes the altitude of the receiving unit, which is useful for ecological studies. Since the GGA sentence is a standard sentence that is transmitted by most GPS receivers, the modification was used
in all subsequent versions of the CraneTracker100. The GPS driver is shown in Figure 5.2. Derek Homan later modified the driver so that it only reported information to the upper layer of the system when a valid fix was achieved. Prior to this change, the driver triggered events in the upper layers of the software every time it received data from the GPS, even if the data was invalid.

![Figure 5.2: GPS driver architecture](image)

The GSM is controlled by sending serial character strings over the UART lines to the modem. These commands are defined as part of the standard AT command set [10]. The GSM driver is shown in Figure 5.3. This driver provides a high level interface to the cellular modem. For example, the driver can be commanded to associate with the cellular network. It converts this command into the set of serial commands that the cellular modem can understand. It also understands the responses sent by the GE865, and can report errors to the upper layers.

A lower level component then translates these commands into a format the GE865 understands, and sends them over the UART lines. This architecture is shown in Figure 5.3.

To support the additional power monitoring capabilities of the CraneTracker100.v3, a power monitoring module was constructed that is capable of interfacing to the different power monitoring devices. With this module, the system is able to query the different ADC lines to check the current voltage and current levels of the system. Additionally, the voltage levels are converted from an ADC count to a true voltage level. This conversion
Figure 5.3: GSM driver

Figure 5.4: Power monitoring component

simplifies the process of checking for low voltage conditions at the higher levels of the system software. This new component is shown in Figure 5.4.

The final low level component that was added to TinyOS was a change to the timer software. TinyOS’s timer software module was modified so that it kept track of how long the system had been running. This change was made to support debugging applications. Initial testing with the device was complicated by the system resetting. When this happened, it was difficult to detect because the system had no memory of how long it had been running. The default TinyOS distribution allows the application to set timers. These timers trigger an upper level event periodically or after a certain amount of time has elapsed. These software timers are supported by hardware based timers that periodically fire interrupts. This event was enhanced so that it continually increments a global 64 bit system timer, with millisecond precision, so that the system uptime can be tracked.
This interface also supports synchronization to a received time from the GPS, so that the system has knowledge of the global time frame.

5.4 Upper Level Software

The upper level of the software consists of four main components. These components control the high level system operation, communications, and data storage. The first of these components is the GSM manager. The GSM manager acts as an intermediary between the high level components and the GSM driver. This component masks details from the upper level component about how the GSM associates with the cellular network, encodes data, and handling errors generated by the cellular module. When commanded to send data, the manager retrieves the unsent data from the flash storage and sends it to the GSM driver. The component for the GSM manager is shown in Figure 5.5.

The GSM manager has a counterpart for managing the 802.15.4 radio. The crane radio component listens for 802.15.4 base stations. When it establishes contact with the base station, it fetches data from the flash storage. It then sends this data over the 802.15.4 radio, and resends the data in the case of packet loss. If too many packets are lost, the crane radio ceases communications with the base station. This component utilizes the
default MoteWorks radio protocols to send the data. The radio manager is shown in Figure 5.6.

Flash storage is handled in a very primitive manner by MoteWorks. There is little to no abstraction in the flash storage. Any application that interfaces to the flash is capable of writing arbitrary sized blocks of data anywhere in the flash device. This lack of abstraction and proper filesystem makes it difficult to handle errors with the flash. In particular, early testing with the device had trouble coping with a complete loss of system power. When this happened, it was impossible to determine what data had already been sent, and the device would resend everything in its memory.

To address these shortcomings, a new storage component was created that is shown in Figure 5.7. This component changes the representation of the flash storage as a circular FIFO queue. Two pointers are always written to the first 8 bytes of memory. These
pointers keep track of where new data is to be written, and what data has been sent over a communications device. In the case where the circular queue wraps, and the entire memory is filled before data can be written, the oldest data in the system is overwritten. These behavior was chosen for a specific reason. The most likely time for the cranes to face extended communication outages is in their nesting and breeding grounds. This behavior is very likely if a base station cannot be located near to the bird. In this case, the flash storage will most likely be filled with data from the nesting grounds. This data is the least interesting data from the biological research perspective, so when the crane begins migrating again, this uninteresting data should be overwritten.

Each entry in the queue occupies 31 bytes. Each entry contains a specific type of data, such as GPS readings, acceleration in a particular axis, heading, pitch, roll, or solar data. These different types of data are placed in a union. A common set of data, including the system voltage and time, is appended to each record. Thus, each entry is the same size. Given there are 512 kB of data, and 8 bytes is used for storing pointers, 16,912 records can be stored in the flash at any time. The final application studied in Chapter 7.4 writes 50 records to the flash per day. At this rate, the system is capable of caching over 338 days worth of data in the flash. Since the system is able to cache this amount of data, under normal operating conditions, the system will almost certainly be in communications range at some point, and be able to offload its data.

The low level flash storage was also changed to better support the mission. The AT-45 flash chip on the Iris utilizes a write cache, so that the flash write latency is masked from the upper levels of the system. Unfortunately, the sensor readings generated by the CraneTracker100 would not always fill up these buffers during a sampling period. In this case, the buffer would not be permanently committed to the actual flash storage. If the device lost power before the next sampling period, all of the data in the flash cache would be lost. This is highly undesirable. To rectify this problem, a control was added
to the flash storage so that the upper levels could force the caches to be committed to the main storage. The system uses this control to force the data to be written, and thus minimize the chances of data loss in the case of power loss.

The final component is the most important component of the system. This is the crane manager, which controls the flow of sensing, communication, and sleeping. The heart of this component is a finite state machine, which controls what sequence the actions are taken in. After taking an action, the system checks the current time, and sleeps for a pre-determined period. Before taking any action, such as sampling a sensor or using a communication channel, the battery voltage is checked. If the battery voltage is too low to successfully operate the component, its action is skipped, and the system sleeps to recover energy. This process will be described in more depth as part of Chapter 6. The interaction between this high level control flow and the rest of the system is shown in Figure 5.8.

### 5.5 Backend Software

Backend software was create that handles the reception of the data, storage, and visualization [18]. The CraneTracker100 sends its SMS messages to a Google Voice account. From here, the messages are automatically transferred into a database. This database is accessed through a web based front end. This front end is capable of displaying the sensor readings from the bird, as well as overlaying their positions on a Google Maps display.
5.6 Software Development Conclusion

The software for the CraneTracker100 was developed using the well known research platform, TinyOS. Using TinyOS allowed for the re-use of software designed for the Iris and MTS-420. At the same time, the capabilities of TinyOS were expanded through drivers to support the CraneTracker100 additional support for NMEA0183, and enhanced timer capabilities. At the highest levels, an application for efficiently managing the system’s resources was developed. This system is able to adapt its behavior based on the available energy. To make the collected data easily accessible, a backend component was developed that stores and visualizes the collected information from the trackers.
In many ways, the application developed for the CraneTracker100 is much more complex than those seen in many WSN deployments. Traditionally, the communication protocols have been considered the most complex software used in WSNs. However, in the CraneTracker100 both the high level behavior and sensor drivers are more complex than traditional networks. To ensure reliable system operation, the CraneTracker100’s software must be thoroughly tested prior to deployment. This is challenging, given the wide variety of locations and environmental conditions the motes will experience. These challenges, and a method for resolving them, are given in the next chapter.
Chapter 6

Pre-deployment Software Testing

The system’s hardware and software is designed to operate for years without intervention. Given this extended mission duration, it is critical that the systems be mechanically and electrically robust. One factor that is often overlooked in WSN deployments is the reliability of the software. Traditionally, reliability in WSNs has been derived from overlapping sensor coverage [11]. A large number of nodes are deployed to an area. Losing a certain number of nodes is anticipated, with a high enough percentage of redundant nodes, adequate sensor coverage and network connectivity can be maintained [41].

The CraneTracker100 differs from traditional WSNs in several key aspects. First, it is only possible to deploy one tracker per bird. Unless an entire family of birds is captured and tagged, there will be no redundancy in tracking the birds. Second, multiple communication mechanisms make it much more complex than many WSNs. This additional communications channel accounts for a large amount of the complexity in the CraneTracker100 code. The GE865 has a variety of failure modes, that must all be accounted for in the code. Third, the compass and GPS are more complex than most of the sensors that are deployed on WSNs. The GPS is especially complicated, since it communicates using text strings that must be parsed correctly, and may have errors. Finally,
the behavior of the motes is highly coupled to their environment. The more complex the environment, the more complex the motes’ operation will be. In the case of the CraneTracker100, the environment is extremely complex. Not only is the device reacting to the behavior of a wild animal, it is also facing a variety of environmental conditions on a continental scale on a multi-year time scale.

These factors make the CraneTracker100 a challenging project to debug and test prior to deployment. To address problems in testing the mote, without attempting to recreate all of the possible environmental conditions, aspect oriented programming was integrated into the crane tracking project. The work in this chapter is a subset of the work Paul and I carried out [14]. The work discussed in this chapter is part of a larger effort, that includes a GPS and crane migration simulator, that is wholly the work of Paul Bennett, and are thus not discussed here [18]. The work that is presented was either developed entirely, or substantially, by myself.

6.1 Background

The increasingly widespread adoption of WSNs has lead to the development of new testing techniques. In particular, two tools have been widely used in several works. The first tool is TOSSIM [44]. TOSSIM is a discrete event simulator. TOSSIM replaces the toolchain for compiling TinyOS code for microcontrollers with an alternative implementation that allows the TinyOS code to run on PCs. Low level components, such as the UART ports, are replaced with components that simulate their behavior. By moving the application execution in a simulated environment on a PC, TOSSIM seeks to be highly scalable, and capable of simulating motes’ behavior at a high degree of fidelity.

Avrora [78] takes another approach to testing WSN applications. Avrora is a cycle accurate simulator for the AVR architecture. Avrora seeks to be a more accurate predictor
of system behavior than TOSSIM. In this respect, it does achieve a great deal of success. The primary drawback to Avrora is that it is only capable of simulating AVR based architectures.

In addition to these general purpose simulators, several other testing approaches have been taken to emulate the impact of environmental phenomena on WSNs [87]. Static code analysis has been utilized to guarantee program correctness in WSNs [66]. Run-time monitoring of motes has been attempted by running mote software inside of a simulator or on a mote [46, 47, 67, 75]. These approaches suffered from long execution times from exhaustively checking the program execution space, or missed potential faults by simulating an ideal mote environment.

During the initial development of the CraneTracker100, using the MTS-420, two testing approaches were taken to verify the behavior of the program. These two approaches were: (1) Log file analysis and (2) Aspect-oriented programming. Log file analysis separates program execution processes from fault detection methods. In this testing method, the program is modified to output data to some storage mechanism so that it can be analyzed by a separate process without interrupting the execution of the original program [13]. The analyzer program is a state machine that uses the outputs of the original application as inputs. The analyzer program decides that the application was correct if its state machine accepts the input, otherwise it rejects it.

Log file analysis can be a powerful tool to test software against formal specifications, but it does have some drawbacks [13]. Developing the analyzer and guaranteeing its correctness can be difficult. The size of the generated log files can also pose problems [88]. Relatively simple Java programs have been shown to produce logs containing several million lines.

Aspect oriented programming is designed to aid in the development and testing of programs by making it easier to address the cross-cutting concerns of a program. Cross-
cutting concerns are properties of a system that are developed separately from each other, but must work together correctly in order for a program to function correctly [57]. For example in our system, the voltage is often checked prior to activating a peripheral, to ensure that the system has sufficient power to correctly function. These voltage checks are cross-cutting concerns, since each voltage check can impact many parts of the program.

To deal with cross-cutting concerns, functions called “aspects” are written separately from the target application. These aspects can monitor or even change the program behavior where cross-cutting concerns arise. Areas of code involved with the cross-cutting concerns are specified by a developer and then the aspects are “weaved” into the program at compile or run-time at these locations.

The AspeCt-oriented C Compiler (ACC) [36] brings aspect oriented programming support to the C language. ACC allows for compile-time weaving of aspects to monitor properties of the system at run-time. These aspects can cleanly capture many complex system properties. Unfortunately, run-time monitoring has been shown to inflict a severe performance penalty as demonstrated by [20]. To overcome this, several approaches have been proposed that reduce the number of states that must be analyzed by the run-time monitors [29, 47].

ACC has been used to build our mote application for a simulator to conduct run-time monitoring and property checking. The simulator uses data derived from prior crane tracking experiments to provide a realistic model and focus the debugging efforts on likely scenarios. By utilizing ACC, we have leveraged existing, already tested technology. Furthermore, by weaving the aspects into the C code generated by the nesC compilation process, it is possible to reuse this process for TinyOS 1.x and 2.x, without major changes.
6.2 *CraneTracker100 Testing*

The behavior of the *CraneTracker100* is highly coupled to its surrounding environment. The most important impact the environment has on the *CraneTracker100* is the solar recharging performance. The amount of energy recovered from the environment is dependent on several processes. The first is the weather, as has been shown in several studies [62, 64]. The second major impact on the solar recharging process is the position of the solar panel. Solar panels for industrial power production can move during the day to achieve maximum efficiency [70], but in this project, the solar panel position is tied to the bird’s behavior. In general, there is no reasonable expectation that the bird will orient itself in a manner which maximizes the solar recharging performance. Finally, according to our partner ecologists, the birds will probably preen themselves so that the device is located closer to their skin, and be partially obscured by their feathers.

The processes that consume energy in the *CraneTracker100* are also stochastic. Two prime examples of these processes are the GSM communications and the GPS sensing. The GSM module takes a variable amount of time to connect to the cellular network and send data, depending on the channel conditions. During this time, it can adjust its transmit power, and may have to resend data in the case of packet loss. Thus, the energy the GSM consumes is variable. The GPS also takes a variable amount of time to acquire its position. The time between turning the GPS on, and it determining its position, is called the time to first fix (TTFF).

These types of processes have a major impact on the system’s energy reserves. For the device to correctly operate, it must then constantly check the voltage level of the battery to ensure it has enough energy to carry out an action. The peripheral sensors and communication modules must be turned off after use, so that they do not waste energy. This energy management process is complicated by potential error conditions.
raised by the devices, which may cause their operation to terminate in an unexpected manner. Enumerating, let alone recreating, all of these error conditions is extremely difficult, and may not be practical in a laboratory setting. Furthermore, these errors may be caused by physical damage to the device, which is difficult to model. Whereas much of the prior work in testing WSNs has focused on testing interaction faults between the motes, and the general communication software protocols [40, 44], the CraneTracker100 requires extensive testing of its power management and error handling.

Further complicating the testing of the CraneTracker100 is the long mission duration. During these extended missions, the mote will encounter many different environmental conditions, and experience a wide variety of different energy states. These long missions give slowly manifesting faults a generous time window in which to manifest themselves. This motivates the use of simulators for testing purposes, so that different environmental conditions and long mission durations can be tested in a reasonable time frame.

The crane tracking application was tested using several different approaches. The first method tried was the use of log file analysis. After log file analysis was tried, a system was developed for run-time monitoring of the application in TOSSIM, using ACC. In particular, we are interested in ensuring that the system sampled its sensors and communicated in a particular sequence. Another behavior of particular interest was the system’s capability of recovering from an error, which caused a particular component to run for too long without finishing.

### 6.3 Testing Platform

In this chapter, the log file and aspect oriented programming platforms are explained in detail. The TOSSIM simulator is heavily used in both approaches. Using TOSSIM, it is possible to simulate years of a mote’s operation in a matter of minutes. Also, multiple
TOSSIM simulations can be run in parallel on the computing grid at UNL. By using the computing grid, it is possible to simulate the operation of hundreds of networks at once, rather than being limited to the few dozen physical motes that are available in the lab.

The goal of these tests was to test two main features of the application software.

**Power monitoring:** The system must continually check the battery voltage before attempting to use a communication device or sensor. For example, the GSM requires 3.3V to operate correctly. If the battery level is not at this level, then the GSM may fail to associate with a cellular tower. Even though it may not be capable of joining the cellular network, it will still consume energy in a futile attempt. Therefore, it is imperative that it not be activated unless there is sufficient power to operate it.

**Peripheral control:** After a sensor or communication device is used, it is very important to turn it off. After using a device, the microcontroller typically enters a low power operational mode to conserve energy, and give the battery a chance to recharge through the solar panel. If a sensor or radio is left on when a sleep cycle begins, it will not reduce its power usage. Thus, the battery will continue to drain during the energy recovery process.

These features were tested using two different techniques. The first used log file analysis to analyze the behavior of a mote after a simulation has completed. The second technique used aspect oriented programming to monitor the system behavior during run time. Both of these techniques utilized TOSSIM. The low level component that simulates the voltage level of the system was modified to generate random voltage levels, so that the system would be forced to use all of its power management features. This randomized input checks the system’s low voltage behavior more rapidly than allowing a real world system to slowly deplete its energy reserves. In some ways, it also offers a more strenuous test than a real world battery probably would provide. This additional stress is caused by the system’s energy state switching much more rapidly, and in a
much more unexpected manner than a real world battery.

### 6.3.1 Application Under Test

This software testing was performed early in the crane tracking project. The *MTS-420* was the only platform that had been developed when the testing was being performed. As a result, the application differs in some aspects from the final application. Most importantly, the system did not have energy harvesting, so the system attempted to conserve energy during most of the year, and only perform high frequency sampling during the migratory periods. The *MTS-420* lacked cellular communications, so the 802.15.4 radio was infrequently used to send data back. Finally, the wide variety of non-GPS sensors on the *MTS-420* are collectively thought of as “ADC” sensors.

The application behavior is shown in Figure 6.1. The transitions show the multiple duty cycles employed by the system. One duty cycle persisted over the course of the year, and determined the periods of sleeping and high sampling rate and communications. Another duty cycle was employed during the cranes’ northerly migration, in which the sensors were sampled at a much higher rate. The final duty cycle controlled the communication period, during which base stations were assumed to be present.

### 6.3.2 Log File Analysis

Log file analysis is used as a basis for measuring the effectiveness and complexity of using aspects for run-time monitoring. The log file analysis method uses debugging statements that are manually inserted to the program. These debugging statements output symbols that correspond to the state of the program, i.e. sleeping, sampling GPS, communicating, etc. as shown in Figure 6.1. The program is run in TOSSIM for several years of simulated time. The output of the simulation is recorded in log files. These log
files are then post-processed using the following procedure. If the system is capable of observing and logging an erroneous behavior, then the log file should reveal the error.

Regular expressions are developed from the finite state machines that represent the ideal program execution behavior. A program then parses the output of the simulation and compares the state changes in the simulation to the allowed states that are represented by the regular expressions. If the output of the simulation matches the patterns
in the regular expressions, the program is assumed to have performed only legal operations and passes the test. If the output of the simulation does not match the regular expressions, it fails and a fault is reported. Using this testing, we were able to reveal instances where the system did not change its state properly in response to a component becoming unresponsive.

6.3.3 AspeCt-Oriented C Compiler

ACC is incorporated into the build process of the application to provide run-time monitoring. Aspects describing different properties of the system are written and woven into a simulator executable. This process allows the system properties to be monitored at run-time, as opposed to log file analysis which is conducted after the program execution. When erroneous behavior is detected it is possible to not only report the violation, but to give information of the state of the system when the violation happens. This contrasts with the log file analysis, where only the violation is reported.

To weave aspects, the compilation process for the motes has to be altered as shown in Figure 6.2. The original application is written in nesC. The nesC application is compiled using ncc which is an extension of gcc that compiles nesC code for TinyOS. As part of its compilation process, ncc produces a C file containing all of the application code. This application source code is then processed using gcc to make it conform to ACC specifications. This transformed source code is then passed into ACC, where aspects written in C are woven into the source file. This final source file is then passed back into ncc to produce the final simulator executable.

To test the system, seven aspects are defined to monitor different properties. The properties are focused around preserving energy and checking the order of system events. Others are written to illustrate the power of the aspects and their capabilities.
These aspects are listed below.

**timers\_fired**: This tracks when a system timer is fired. In TinyOS, it is possible set many timers with distinct names. When a timer is fired, a specific callback function is executed. This aspect is capable of logging which timer was fired, and when it happened.
Often times in WSNs, complex processes are started based on these timer events. When debugging these applications, it is very common to put debugging statements on the callback functions, which can be a tedious process. Therefore, this aspect is useful for automatically capturing these events with a single aspect, that captures all timer events.

**voltage_sensor:** This aspect checks that the voltage level of the system is checked before sampling any of the non-GPS sensors. As previously discussed, this ensures that a sensor can correctly operate, given the state of the system’s energy reserves.

**no_stalls:** This aspect monitored for watchdog timer events. The watchdog timers acted as a failsafe in case a component failed, and caused the system to wait for some event that would never occur. For example, normally the GPS portion of the system checks for a valid GPS fix for a fixed number of NMEA messages. If the system has not achieved a fix in this amount of time, then it is turned off to conserve power. If the GPS became damaged during this process, the entire application would essentially be halted, waiting for a NMEA message that would never arrive. The watchdog timers were intended to catch these exceptional events, and force the system to continue its execution. Therefore, monitoring for a watchdog timer serves to notify the developer when the system is stuck waiting for an event.

**gps_off:** This monitors the GPS to make sure it is always turned off after it is turned on. Conceivably, the GPS would be turned off after acquiring a fix, failing to acquire a fix in a specified time period, or completely failing, and triggering a watchdog timer. This aspect was capable that each of these events would result in the GPS being turned off, so that power is conserved.

**state_change:** This monitors the internal program state, to check if it is following the control flow shown in Figure 6.1. This control flow is somewhat complex, and it is important the long term behavior of the application match its expected behavior.

**voltage_gps:** This is similar to **voltage_sensor.** This checks to make sure the system
always checks the battery voltage level before turning on the GPS receiver.

**trace**: This trace generates a complete trace of the application’s execution. This is used to generate the power of aspects, and for debugging purposes.

### 6.4 Results

To test the impact of aspects on application behavior, each aspect is woven into the application at compile time into the TOSSIM Framework that can be run on a PC [44]. A TOSSIM application simulates TinyOS and its execution at its lowest level. This provides high fidelity simulation of TinyOS applications. Many of these simulations were then conducted in parallel, using the PrairieFire supercomputer.

The simulations are performed for a longer period of execution time (3 years) than the motes were expected to last in the field. This provided extensive coverage of different execution scenarios very quickly, resulting in the unmodified application taking only 15 minutes to run and using only 0.03s of CPU time.

The results of each simulation provide a comparison of performance between the original application and each individual aspect. The performance metrics of the simulations include CPU time of the simulation, size of the executable, the output generated from each simulation, and faults that existed in the application. These results are shown in Table 6.1. A more detailed analysis of the testing results follows.

#### 6.4.1 Execution Times

The impact of the run time monitoring is of the highest concern. As shown in Table 6.1, all of the aspects had a substantial impact on the execution time of the simulation. If just the total run time is considered, then the change in execution time seems to be reasonable. However, this is a deceiving metric. For one, the simulation that was ran is...
Table 6.1: Overhead of aspect in terms of CPU time and executable size

<table>
<thead>
<tr>
<th>Aspect</th>
<th>CPU Time (s)</th>
<th>CPU % Increase</th>
<th>Executable Size (bytes)</th>
<th>% Size Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unmodified</td>
<td>0.0327</td>
<td>N/A</td>
<td>258607</td>
<td>N/A</td>
</tr>
<tr>
<td>timers_fired</td>
<td>0.57</td>
<td>1642</td>
<td>283665</td>
<td>9.7</td>
</tr>
<tr>
<td>voltage_sensor</td>
<td>0.06</td>
<td>83.3</td>
<td>284231</td>
<td>9.9</td>
</tr>
<tr>
<td>no_stalls</td>
<td>0.0382</td>
<td>16.7</td>
<td>279134</td>
<td>7.9</td>
</tr>
<tr>
<td>gps_off</td>
<td>0.1009</td>
<td>208.3</td>
<td>281077</td>
<td>8.7</td>
</tr>
<tr>
<td>state_change</td>
<td>0.0463</td>
<td>41.7</td>
<td>286067</td>
<td>10.6</td>
</tr>
<tr>
<td>voltage_gps</td>
<td>0.0518</td>
<td>58.3</td>
<td>283048</td>
<td>9.5</td>
</tr>
<tr>
<td>trace</td>
<td>31069</td>
<td>94933258</td>
<td>721783</td>
<td>179.1</td>
</tr>
</tbody>
</table>

extremely simple. In general, most WSN simulations will include components like an RF channel model, which will greatly slow down execution. In these cases, many of the aspects may create unacceptable delays.

These results show that the performance suffers less when the aspects are focused on select areas of the code. For example, the state_change aspect revealed several bugs, and was one of the least expensive aspects to run. Aspects such as these can run without the computationally expensive RF simulation components, so they may be practical to run. The less focused aspects, such as the trace aspect, are prohibitively expensive to run in all cases.

Even with the dramatic increase in execution times, some aspects may still be capable of running on a real mote. This is an intriguing option, and one that would add another tool to the system developer’s toolbox. For this to happen, a method of communicating the aspect results needs to be developed, which may pose challenges. Another difficulty in moving aspects to a real hardware platform is the effect of the slow execution speed on time critical portions of the system. In general, TOSSIM does a poor job at exposing time related flaws in code [78]. Therefore, this experiment does not imply aspects will be easily portable to real hardware.
6.4.2 Executable Size

Another important performance metric is the impact of the aspects on the size of the produced executable. While this does not impact the functionality of the simulator process, if this process is to be used on actual mote hardware there will be much more severe constraints on the size of the program that can be used. Single aspects were woven into the application and the size of the resulting executable was recorded as shown in Table 6.1.

The results reveal that the aspects have a very small impact on the executable size. With the exception of the trace aspect, all aspects add less than 11% to the size of the executable. Therefore, if a corresponding change is made to the hardware build process, it is very likely the program will still fit in the memory on the mote. It is interesting to note that even though the aspects may have a small impact on the size of the system image, this does not directly correlate to the effect on the execution time. Therefore, caution must be taken in assuming that infrequently woven aspects will have a non-trivial effect on performance.

6.4.3 Implementation Cost

One major benefit the aspect oriented debugging had over the log file analysis was the speed in which the aspects were developed and implemented. Developing the aspect that checked and monitored all of the state changes took less than 4 hours. Developing the same functionality using log file analysis took approximately twice as long. The aspect is also much easier to modify. The aspect also automatically recorded any additional state changes, where the log file approach required constantly modifying the application code to continue supporting the debugging attempts.
6.4.4 Violations

The testing revealed several bugs in the application. The log file analysis and state checking aspect were able to catch several instances where the mote did not transition to a correct state or got caught in a state and never transitioned to a new one. This behavior was typically caused by a component taking too long to run, and being abnormally terminated through a watchdog timer. Another common cause of these failures was the system not having enough voltage to run a peripheral, and incorrectly choosing the next action to run. Had these violations been included in the actual deployment, it could have resulted in the motes running out of power in days or weeks and failing to complete their mission.

The aspect oriented approach revealed additional faults that did not affect the correctness of the program that the log file analysis did not detect. For instance, the GPS receiver was being commanded to turn on when it was already on. The log file analysis did not catch this error because the GPS was turned on twice within the same state. The log file analysis only checked when the system changed state, so it did not catch this error, but the aspect that checked the GPS power on sequence caught it immediately.

Finally, the use of the simulator to accelerate the mote testing process proved invaluable. The violations resulting from incorrect state transitions had a dependency on the power level of the mote. In general, finding these types of violations on real hardware can be a time consuming process, since many tests would be needed to try all possible power level combinations. It was also possible for the violations to manifest themselves only after weeks or months of run time, which is also difficult to test in a controlled fashion on real hardware.
6.5 Testing Conclusions

Testing for WSNs is a hard problem, and is particularly so in the CraneTracker100. The tight coupling of the mote’s behavior to its environment produces challenges when the software is designed to run for years unattended, and cover a large portion of North America. Using simulations and run-time monitoring enables the system developers to efficiently test their software. Moreover, the testing methodology presented in this chapter is adaptable to a variety of software platforms. Since ACC integrates with the C code generated by the NCC compilation process, it is possible to easily integrate this tool into other projects. For example, ACC was integrated into a project using TinyOS 2.x, using largely the same manner as was described in this chapter. This flexibility makes it possible to use the system in a variety of projects developed in the CPN laboratory.

One downside to the testing done in this chapter is it is not particularly effective at simulating hardware faults. These faults can have highly unpredictable affects on the system. For example, if a component becomes partially detached from the board, it may intermittently communicate with the microcontroller. The exact nature of these hardware faults is difficult to replicate. However, if the fault recovery action can be triggered, then the system described in this chapter can test that portion of the application.

With a reliable hardware and software platform developed, the motes are ready to be deployed in real-life scenarios and testing. These experiments, and their results, are presented in the following chapter.
Chapter 7

Deployment Experiences with the CraneTracker100

The CraneTracker100 has been extensively evaluated via component evaluations, controlled system experiments, and on-going field experiments. Component evaluations are used to evaluate the individual elements of the hardware platform. The software testing work that began in Chapter 6, is continued with an evaluation of the entire system’s software in real deployments. Field tests are used to not only test the system’s hardware and software, but to conduct valuable research for our partner ecologists. The results of these tests can be used for future experiments, by adjusting the system parameters to make the system more effective and efficient.

The Whooping Cranes’ small population creates special challenges for testing. New tracking technology must first be tested on “proxy” species before being used on Whooping Cranes. Testing on these alternative species ensures that the CraneTracker100 will not harm or otherwise adversely affect a wild crane. These proxy species are chosen because their share characteristics or habitat with the Whooping Crane. Results from these “proxy” species can then be used to predict the performance of the CraneTracker100 on
the Whooping Cranes.

The first field test used the CraneTracker100.v2 to monitor Wild Turkeys (*Meleagris gallopavo*) at the Crane Trust in Wood River, NE. These turkeys live in habitat along the Platte River that Whooping Cranes use during their migrations. Moreover, the turkeys are abundant, which reduces the barriers to testing with them. Our colleagues at the Crane Trust have experience working with turkeys, which is also convenient.

The second set of field tests were conducted at the ICF. In these tests, Siberian Cranes (*Grus leucogeranu*s), were used to test the short range communications, compass, and mounting design. In the third set of tests, several devices were deployed on Sandhill Cranes (*Grus canadensis*). These birds are closely related to Whooping Cranes, and have many of the same behaviors. In particular, the Sandhill Cranes migrate to the same regions as the Whooping Cranes, and in similar time frames. Unlike Whooping Cranes, Sandhill Cranes are plentiful, which makes them an ideal testing platform.

### 7.1 Preliminary Evaluations

While traveling to Wood River to deploy the CraneTracker100.v2 on Wild Turkeys, several experiments were ran to evaluate the performance of the GSM. The trip to the Crane Trust used I-80. Since the interstate closely follows the Platte River, which the Whooping Cranes frequent during their migrations, the performance of the GSM is of great interest in these areas. This trip took over 2 hours to complete, and 160km of road were traveled.

In Figure 7.1, the received power of the cellular signal is compared to the speed at which the car was traveling. The x-axis is the time since the experiment started. The first y-axis shows the time between collecting a data sample, and the backend server storing the data. This latency includes the time it took for the CraneTracker100.v2 to send the data to the cellular network, the time the cellular network took to deliver the data, and
the time it took for our backend to process the received data. The second component of the y-axis shows the speed at which the car was traveling, according to the GPS. From this data, the average communication delay was determined to be 6.07 minutes, with a variance of 1.53 minutes. This data shows that during the experiment, a higher velocity lead to a higher latency. This correlation could have been caused by several factors. First, prior work has confirmed that communication channels are negatively impacted by velocity [25]. Second, at a higher level, the cellular base stations must hand off the modem between themselves, which can also lead to delays. Third, it is possible that the delay was caused by factors specific to the particular cellular towers being used, and the towers at the beginning of the trip were performing better than the final towers. This hypothesis is not possible to test, since we do not have access to such detailed information on the cellular network.

Figure 7.2 suggests a further link between speed and link quality. In this figure, the received power of the cellular signal is plotted against the speed of the vehicle, as determined by the GPS. While the vehicle was traveling at a slower speed, the signal strength was consistently better. Given that the cranes are capable of traveling in excess of 80km/h, and migrate over a large area, these experiments show that they will encounter a wide variety of communication conditions throughout their migration.

The cellular base stations can also act as a source of location information. The Crane-Tracker100.v2 sends data on which cellular base stations it associates with to the backend software. This information is then used to derive an estimate of the mote’s location. In Figure 7.3, the CDF of the error between the true position of the mote (as determined by the GPS), and the position estimated through the base stations is shown. This data reveals that the GSM is a viable method for tracking the general migration paths of the cranes. While the error is too great for individual roosting sites to be located, the position estimate can identify the general areas where the cranes travel.
This is a significant result. This demonstrates the value of multi-modal sensor platforms, that are capable of redundantly sensing information, in wildlife tracking. The very act of the device communicating with researchers will reveal information on an animal’s behavior in future applications. Also, redundancy is provided on the device itself, so it is more fault tolerant and resilient to physical damage.

7.2 Field Experiments with Wild Turkeys

The next set of field tests were conducted using Wild Turkeys at the Crane Trust in April and May of 2011. These experiments were performed using the *CraneTracker100.v2* platform, and served to motivate many of the changes between the *CraneTracker100.v2* and *CraneTracker100.v3*. 
Field observations have shown that 2/3 of the Whooping Cranes’ migration time is spent foraging and roosting in habitat very similar to that of Wild Turkeys’. Thus, these birds are used to evaluate the sensing, communication, and power aspects of the Crane-Tracker100. These early experiments yielded promising data, but also demonstrated some large faults in the CraneTracker100.v2.

The turkey experiments were carried out using two CraneTracker100.v2 platforms. The first device, henceforth referred to as the turkey tracker, was attached to the back of a captured turkey hen. The device was located inside one of the leg mount devices discussed in Chapter 4.5.1. The turkey was then released into the wild for a week. The second device acted as a control. The same hardware and software was placed in an
open area that was located within 1km of the hen’s habitat.

The deployed software followed a simple duty cycle. Every 4 hours, the system would wake up, attempt to acquire a GPS fix, collect 10 readings from the compass, and then attempt to send this data using the 802.15.4 and cellular communication links. This simple duty cycle was designed to collect a large amount of sensor data, as well as provide insight into the real world performance of the solar panel and GSM.

### 7.2.2 Experimental Turkey Results

Over the course of the 10 day trial, the turkey tracker delivered 480 SMS messages. 105 samples of GPS data were collected, and 9,170 sets of compass data were received. While there was no method included to determine if a message had been lost, the amount of
data received matched how much data we expected to generate in the given time period. With the exception of 9 messages, all of the SMS messages were sent with 4 hours of the data being recorded. This high reliability demonstrates that even in the rural areas where the cranes travel, cellular communications will still be available in some areas.

By using the data from the mote located in a fixed location, an empirical CDF of the GPS accuracy is determined. This data is shown in Figure 7.4. This data showed that the GPS component of the system could be improved upon. In general, the errors are much higher than expected, and are much higher than the predicted accuracy of the GPS’s datasheet. These large errors are caused by the software component that controls the GPS. After the GPS is turned on, it begins sending messages to the microcontroller,
indicating whether or not it has acquired its position. Unfortunately, these messages do not indicate the error in the GPS’s position estimate. When the GPS first reports it has acquired its location, it does not know the position with a high degree of accuracy. However, the system software accepted this location, and immediately turned the device off to save power. This lead to inaccurate GPS readings being recorded.

The GPS data collected from the turkey tracker revealed the hen moving in a 2km x 1km area. During this time, all of the data was relayed to the backend using the cellular connection. By dividing the data, based on when it was collected, it is possible to analyze the movement patterns of the turkey. This movement is shown in Figure 7.5. This data reveals that the turkey is more mobile in the early morning and evening. In these time periods, the turkey is traveling as far as 1km to reach its feeding grounds.
In Figure 7.6, the recharge performance of both trackers is shown. The vertical line on the figure indicates the start of the deployment. As expected, the control tracker harvested more energy from the environment. This difference is due to the control tracker being deployed in an open area, where the bird’s movements placed it in wooded areas. This sub-optimal recharging was exacerbated by the turkey spending its time in the woods during the middle of the day, when solar power is at its peak.

Even though the turkey’s behavior was not ideal for the solar recharging device, the control device had promising results. The Whooping Cranes tend to fly throughout the daylight hours. This means at times of peak solar power, the birds will probably be airborne, and giving the solar panel a clear view of the sky. This behavior should lead to better solar recharging performance than the turkey demonstrated.
The turkey experiment did reveal one major flaw in the system. The rigid plastic enclosure proved to be difficult to waterproof. Additionally, the material is extremely porous, which lead the enclosures actually absorbing water. These factors lead to the control unit failing from water intrusion. The unit on turkey eventually stopped recharging, and was not recovered. While it is impossible to know for certain, it is likely that the solar panel was either damaged by the bird, or water intrusion damaged the recharging mechanism.

7.2.3 Ecological Observations

The collected turkey data agreed with prior turkey research collected by field personnel. In particular, turkeys are known to spend up to 95% of their day feeding. The turkeys are more mobile during these periods, and can move over 200m over the course of an hour while feeding [27]. These field observations strongly correspond to the data collected from the turkey experiment.

The GPS data revealed several aspects of the turkey’s life. The night and daytime locations of the turkey are most likely its brooding and nesting areas. Other clusters of GPS locations are probably the bird’s feeding areas. The distance between these areas is in line with the turkey traveling at 200m/hr, as noted in other studies.

Even though it was not our primary goal, the turkey tracker data also provides important insight into turkey behavior. Turkeys spend up to 95% of the day feeding and are able to move at a rate of 200 m/hr during feeding periods [27]. The collected data supports these conjectures. The locations during the night and afternoon correspond to the hen’s brooding and nesting area. The other GPS positions are probable feeding areas, in which the turkey is found in the rest of the day. GPS positions from one day are also connected in Figure 7.5, where observed movements of up to 1 km over a 4 hour period
fall closely into the 200 m/hr observations [27].

7.3 Captive Siberian Crane Experiments

After testing the CraneTracker100.v2 on the turkeys, the CraneTracker100.v3 was developed to address the shortcomings in the platform. In July and September of 2011, this new platform was tested on three captive Siberian Cranes. These cranes were chosen as a result of our collaboration with the ICF, who are interested in not only tracking the cranes, but analyzing their behavior.

7.3.1 Captive Crane Experiment Setup

The three cranes used in the experiment are named A. Wright, Bazov, and Hagrid. These cranes are permanent residents at the ICF. At the ICF, these cranes are housed in separate pens that are approximately 10m x 10m. The packaging method described in Chapter 4.5.2 had not yet been finalized, so a waterproof cloth bag was used to hold a tracker to a bird’s back. This arrangement is shown in Figure 7.7. Note that in this arrangement, the solar panel is not used.

Two different sets of experiments were performed with these captive cranes. In July, the trackers attached to A. Wright and Bazov were programmed with a duty cycle operation that mimicked the deployment on the turkeys. When further experiments were conducted in September, the GPS behavior was changed so that 10 fixes were collected before the GPS was turned off, and the last GPS fix was transmitted to the base station. This was done to improve the accuracy of the position estimate.

A different experiment was used with Hagrid. The tracker on Hagrid was programmed to sample the compass at 10Hz for 30 seconds, and then sleep for 2.5 minutes. The recorded data was transmitted using the 802.15.4 radio. During this experiment,
Hagrid was recorded using a closed circuit camera. Her behavior was analyzed by a trained researcher who recorded what the crane was doing. This experiment illustrates the power of the accelerometer in analyzing a crane’s behavior.

### 7.3.2 Captive Crane Experiment Results

In July 2011, A. Wright immediately incapacitated his CraneTracker100 by slamming his the device into a fence. However, Bazov was much less bothered by the device, and generated useful data for 27 hours. The CDF of the GPS error was once again determined, and is shown in Figure 7.8. These results were calculated assuming that the crane stayed in the center of his pen, and did not move. This assumption introduces additional errors.
when calculating the CDF. However, even with that assumption, the results show that 58% of the GPS errors are less than 25m.

![Figure 7.8: CDF of GPS error](image)

With Bazov, over 27 hours, a GPS fix ratio of 97% was achieved out of 135 attempts. Figure 7.8 shows the CDF of the GPS error for Bazov, assuming that the bird was located in the center of the pen. Later, when the experiment was re-ran with the additional GPS fixes collected, the accuracy improved to the point where 58% of the GPS errors are less than 25m. This improvement in accuracy comes at a negligible energy cost, as these gaining these additional fixes only requires 10 seconds of additional operational time. Our partner ecologists are very satisfied with these results, and believe they can be used to reliably locate a bird in the wild.

The results of Hagrid’s experiment are extremely exciting. During the experiment in July, 21,882 records, each containing the head, pitch, roll, and 3D acceleration were collected. During the experiment, Paul was easily able to deduce the bird’s movements
from the accelerometer readings, without having visual contact with the crane. This strongly suggests that in the future, the accelerometer can be used to remotely analyze the behavior of a crane with a high degree of reliability [19].

A sample of these results is shown in Figure 7.9. These results include a time period where it was raining heavily. During this time, the crane stopped moving, as can clearly be seen in the marked area of the figure. Samples such as these indicate that the accelerometer can be used for developing and validating energetics models [74, 17].

![Figure 7.9: Three-axis acceleration readings from a Siberian Crane.](image)

### 7.3.3 Captive Crane Experiment Summary

The experimental results with the captive cranes are extremely exciting. The GPS accuracy was improved substantially with a minor software change. The acceleration data
collected from Hagrid points to interesting new research directions, and reveals the utility of the system.

Another interesting result revealed a flaw that had not been seen in the compass. The ICF is located on a anomalous magnetic field that is caused by a moraine. This magnetic field disrupts the heading, pitch, and roll readings of the compass. This result had not been seen during extensive testing in the Nebraska and Kansas regions. This flaw highlights the difficulty in testing systems that will be deployed in such large geographical regions, and the need to design for unexpected error conditions.

### 7.4 Wild Sandhill Crane Deployments

Given the encouraging results of the testing on turkeys and the captive Sandhill Cranes, the project was allowed to test the CraneTracker100 in an uncontrolled environment, on wild Sandhill Cranes. In this experiment, five cranes were captured from three different families. A CraneTracker100.v3 tracking device was attached to the back of each bird.

#### 7.4.1 Wild Sandhill Experiment Setup

With the help of trained professionals, five wild cranes were captured. While the cranes are detained, biometric data is collected on them. This data will be used to assess the health of the individual cranes, and track any health problems in the wild population. After this process, a tracking device is attached to each crane. The cranes are named using the names of the property owners of the locations they were first captured at. In this experiment the five cranes captured are: JB-Male and JB-Female, SH-Female and SH-Chick, and BB-Female. The cranes designed by JB are a pair of adult crane mates. The SH birds are an adult female, and her chick. Finally, BB-Female is an adult female crane.
The tracking devices were programmed with a routine similar to the ones used previously on the turkeys. This application is expected to run, without intervention, for several years. The software follows a schedule such that the system wakes up, collects a GPS fix, gathers 10 compass samples over 10 seconds, and then attempts to communicate with both the 802.15.4 radio and the GSM, and sleeps for 4 hours 5 minutes. The GPS fix is recorded after acquiring 10 valid fixes, which has been found to greatly improve the accuracy. To prevent excessive energy consumption, the GPS is turned off if a fix was not acquired within 5 minutes. The 4-hour 5 minute sampling interval and 10-second compass sampling duration was recommended by our partner ecologists. The sampling interval was selected so that the sampling times shift in time and in long-term, information from each time of day is collected.

### 7.4.2 Wild Sandhill Experiment Results

During the fall of 2011, SH-Chick and SH-Female were tracked during their migrations. The migration paths taken by the birds are shown in Figure 7.10. While SH-Chick flew a long migration path to Florida, BB-Female halted much more quickly in Indiana. This data demonstrates the unpredictability of the cranes. It also demonstrates the distances the cranes are capable of traveling, even in a short migration.

This experiment was the first time the cellular communications had truly been tested on a large scale. The results were extremely encouraging. First, the device was capable of communicating with the GSM module, even while the cranes were in flight. This information was inferred from the GPS locations, and the timing of the cellular communications. SH-Chick obtained a total of 330 GPS fixes. Of these, the bird was moving faster than it could walk 12 times. During 10 of these high speed movement times, the GSM communicated within 10 minutes of the fix being logged.
Further evidence of the cellular network’s power is shown in Figure 7.11. This figure shows the CDF of the delay between logging the data, and it being received in the backend software. This figure shows the delay for all of the cranes, except for JB-Male. Especially impressive is the delay for JB-Female and SH-Chick, where 98% of the data is received within 24 hours of it being measured. The other two cranes manage to meet this 24 hour delay bound 55% and 72% of the time.
Further analysis reveals that most of the delay in the system is an artifact of the FIFO queue used for storing the data. When the system fails to send all of its recorded data during a communication window, the delay will rapidly accumulate. The impact of this queue build-up is shown in Figure 7.12. In this figure, the delay is presented as if a LIFO queue were used instead. This eliminates the time the data sits in the flash memory, waiting to be transferred. In all cases, 94% of the delay is less than 24 hours, which justifies the use of cellular technology for the low latency monitoring of wildlife.

The communication aspects of the system held up very well in their first wild deployment. The sensors performed equally well, and revealed many interesting facts about the cranes’ migrations. One of the more interesting examples of the collected data is Figure 7.13. In this figure, the speed and altitude of SH-Chick is shown over the duration
of its migration. In this figure, the altitudes are referenced to the mean sea level, and not the actual altitude above the ground. This figure shows there is a strong correlation between the altitude of the bird, and their speed. This is expected, as the bird’s are capable of flying much faster than they can walk.

The extreme speed of the chick illustrates the challenging nature of tracking the birds, and the communication challenges that must overcome in order to do so. In at least one point of the migration, SH-Chick was traveling at over 100km/h. These high speeds make it practically impossible to deploy enough short range radio links to maintain connectivity during the migration. Also, the high speed will create many communication errors, and introduce latency, as was seen in Chapter 7.1.

After the communications, the power control and energy harvesting were the major
Figure 7.13: SH-Chick altitude and speed

unknown factors in a wild deployment. Unfortunately, this aspect of the system had decidedly mixed results. In Figure 7.14 and Figure 7.15, the battery voltage and the solar power collected by the monitoring circuit is shown for JB-Female and SH-Chick. These figures show that there is a close correlation between the monitored solar power and the battery recharge rate. This shows the system requires a minimum amount of solar power, and that this level is often not met. Below this power level, the recharge circuitry is unable to utilize the power to recharge the battery. This stochastic energy harvesting process shows the need for adaptive system behavior, that can change its behavior in response to changing energy levels.

The differences in recharging behavior between different birds is partially due to their location. In the breeding grounds, where the cranes spend most of their time, the birds
are often in vegetation. This vegetation can obscure the solar panel. The data shows that JB-Female has a lower recharge rate, but the battery levels are above the critical threshold of 3.65V. As shown in Figure 7.14, during migration (16/11-28/11), the device is capable of recharging the battery in a matter of days.

Unfortunately, the system was not always able to keep functioning on the available solar panel. This lead to a series of unfortunate shutdowns and intermittent operation. Fortunately, the power control circuitry was able to remove power to the system components, and prevent the battery from being damaged. This process of discharging, turning off, and recovering energy to function is shown in Figure 7.16.

The GPS revealed interesting insights into crane family behavior. The distance between the SH-Chick and SH-Female, and SH-Chick and JB-Male, is shown in Figure 7.17.
While this data is useful for biological studies, the close proximity of the chick and the mother raise interesting possibilities for future WSNs. By utilizing their short range radio links, the CraneTracker100.v3’s could have split the sensing and communication duties. Such a division of labor could have decreased the individual motes energy use, thus improving reliability and network lifetime.

7.4.3 Wild Sandhill Experiment Analysis

In two occasions, JB-Male and SH-Female, the system failed. The first tracker was harnessed to JB-Male in July-August 2011 and transmitted information for 13 days, during which all GSM packets were received within 8.3 hours. However, the GPS was unable to acquire a valid fix, and the battery voltage steadily decreased without showing any signs of recharging. It was not possible to recover the device from the bird and determine an
exact cause of failure. However, from the symptoms, it is highly likely that the solar panel became disconnected from the circuit board. Once disconnected, it slid over the GPS antenna, thus blocking reception. This hypothesis lead improvements in how the device operates for the other four experiments.

JB-Male was not the only crane to have a major fault occur with its tracker. SH-Female had a fault that caused the battery to deplete itself, as shown in Figure 7.16. This figure shows the voltage dropped from a full battery level, at time A, until the system reset. Later, the system recovered enough energy to establish communications at time B. The system then proceeded to recharge its battery in a short time period. While the cause of this fault has not been determined, the root cause is suspected to be software.
Figure 7.17: Distance between SH-Chick and family and neighbors

Software is suspected because had the fault been with hardware, it is unlikely that the system would have recovered, and harvested more energy. In previous testing, when the hardware was physically damaged, the system never fully recovered, and continued to exhibit defective behavior.

However, after the system recharged the battery, another failure struck the device. The GPS never obtained valid GPS fixes. The coarse-grained localization data provided by the GSM enabled the experiment to continue. Unfortunately, after several days, a field observer saw the crane without the backpack. This, coupled with the GPS behavior, indicates that the system probably fell off the bird and sustained damage. This damage eventually resulted in the battery discharging, and the device was not found.

A further analysis of the collected solar data reveals more insights into the solar panel performance. Using data from all 4 cranes that were able to harvest solar energy,
the CDFs of the solar charging performance were calculated. These results are shown in Figures 7.18 and 7.19. This data is broken into two different analyses. In both analyses, the current was split into 1mA bins. In the first analysis, the time was split into 2 hour intervals. In the second analysis, time was split into 4 hour intervals.

![CDF of solar current, 2 hour intervals, 1mA granularity](image)

**Figure 7.18:** CDF of solar current, 2 hour intervals, 1mA granularity

These data shows the extreme variability of the energy harvesting device. Surprisingly, the data strongly resembles an exponential distribution. Prior to the deployment, it was conjectured that the solar data would follow a normal distribution. However, many times during the daylight hours the solar panel failed to recover energy. This motivates the need for energy conservation, and a healthy energy reserve in the system, as it will often go for periods of time with no solar energy.

This same analysis was conducted, but with the current divided into 5mA bins, and is shown in Figures 7.20 and 7.21. This data highlights the complexity of modeling solar
data. Dividing the CDF into large bins makes it easier to analyze and model, but the model loses its accuracy.

The prior figures give a qualitative description of the solar harvesting. Unfortunately, when attempting to fit the data to a curve, difficulties arose. First, the data is extremely noisy. This is compounded the relatively small number of samples that are used to form these distributions. Furthermore, the small population that was used to create this data leads to individuals heavily biasing the solar current. Finally, the data is not evenly distributed throughout the day, which introduces further difficulties in modeling the data.

The difficulties in modeling the data can be seen by comparing Figure 7.18 to the CDFs of the solar current for the individual birds. The individual CDFs are shown in Figures 7.22(a), 7.22(b), 7.22(c), and 7.22(d). These figures show that the collected data
differs significantly between the birds, which makes it difficult to model the combined solar data. The amount of data collected by each bird also has a significant impact on its quality. Table 7.1 contains the number of solar data measurements that were taken from each bird. Clearly, the more data that was collected by bird, the smoother and more well defined the CDF is.

<table>
<thead>
<tr>
<th></th>
<th>SH-Female</th>
<th>SH-Chick</th>
<th>JB-Female</th>
<th>BB-Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Points</td>
<td>1,180</td>
<td>10,258</td>
<td>5,919</td>
<td>1,840</td>
</tr>
</tbody>
</table>

Table 7.1: Number of solar data points collected

### 7.4.4 Wild Sandhill Experiment Ecological Observations

The 4-hour sampling interval and the staggered operation provides insight into the daily movements of cranes. In Figures 7.23(a) and 7.23(b), the locations of JB-Female are...
shown for different times of day over 24 days. In this figures, all the night and most of afternoon data correspond to two distinct roosting locations, indicated by circles. The general roosting locations in this area have been known by ecologists, but the collected information revealed individual information about where each specific crane roosted and the dynamics of roosting locations. The breeding territory of the couple is indicated by a rectangle, within which most of the morning and early afternoon data resides. This territory is used for daily activities such as feeding, and preening.

In Figure 7.24, the acceleration in the z- and x-axes for the SH-chick is shown, which reveals important insights on the small-scale movements of the birds. The absolute value of acceleration is heavily influenced by the gravitational force, especially in z-axis, and provides an information about the bird posture. Figure 7.24(a) shows that the bird stays still at night, and the sample variance is the lowest, with a mean close to 9.8m/s². The
Figure 7.22: Solar CDFs of individual birds

(a) BB-Female

(b) SH-Female

(c) JB-Female

(d) SH-Chick

Figure 7.23: GPS locations of JB-Female over 24 days

(a) Wide area view

(b) Zoomed in view
morning and afternoon data exhibit higher variance and the higher acceleration values, which suggests flying (\(\sim 0\text{m/s}^2\)) and takeoff (positive values).

![Acceleration plots](image)

(a) Along z-axis  
(b) Along x-axis

Figure 7.24: SH-Chick Acceleration during migration

The acceleration in the x-axis shown in Figure 7.24(b) has a similar trend in terms of the CDF shape. The steeper slope of the CDF during the night corresponds to limited movements compared to the morning and evening. The majority of the values during night are slightly higher than 0, which indicates the bird was standing still (see Figure 7.7 for bird posture, x-axis points towards the tail). As shown in Figure 7.23(a), during morning and evening, cranes generally do not fly and stay in their territory and roosting site, respectively. Consequently, x-axis acceleration can be used to reveal posture since the walking speed effects are negligible compared to gravity. The negative values suggest that the bird’s head was leaning to the ground, which is a typical posture during feeding. Similar trends have been observed for the other cranes.

A table summarizing the results of the experiments is given in Table 7.2. This table clearly shows that there is much room for improving the system reliability.
<table>
<thead>
<tr>
<th>Name</th>
<th>Exp. Duration (days)</th>
<th># SMS</th>
<th># Fixes</th>
<th>Distance Traveled (km)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>JB-Male</td>
<td>13</td>
<td>448</td>
<td>0</td>
<td>-</td>
<td>Failed to recharge</td>
</tr>
<tr>
<td>JB-Female</td>
<td>41</td>
<td>2,795</td>
<td>208</td>
<td>20</td>
<td>Unknown</td>
</tr>
<tr>
<td>SH-Female</td>
<td>29</td>
<td>525</td>
<td>468</td>
<td>4.3</td>
<td>Fell off</td>
</tr>
<tr>
<td>SH-Chick</td>
<td>66</td>
<td>1,890</td>
<td>330</td>
<td>1,725</td>
<td>Operational</td>
</tr>
<tr>
<td>BB-Female</td>
<td>121</td>
<td>2,213</td>
<td>468</td>
<td>603</td>
<td>Operational</td>
</tr>
</tbody>
</table>

Table 7.2: Experiment summary

### 7.5 Experimental Deployment Conclusion

The results of the experimental deployments are encouraging, but reveal the need for additional work. While the hardware and software performed very well, many motes were lost to problems in waterproofing and attaching them to birds. However, even with these failures, a great deal of interesting, high-quality, data was collected from these experiments. This data included biological observations, such as the birds’ travel and behavior patterns, and research for WSNs. This data, such as communication latency, energy harvesting performance, and localization accuracy, will be used to improve the system in future deployments. This data will enable the system parameters to be tuned, and novel energy management techniques to be implemented.
Chapter 8

Conclusions

In this work, the evolution and deployment of the CraneTracker100 platform was presented. From the deployment, the stochastic behavior of the energy harvesting and sensing process motivates the need to adapt the mote behavior to account for uncertainty in its environment.

8.1 Contributions

The contributions of this work are as follows.

**Novel platform:** The CraneTracker100 is a novel platform that contains multi-modal sensing and communication devices. This device not only tracks where the birds travel, but relays information about their behavior. This is a significant improvement on existing crane tracking efforts.

**Testing methodology:** An existing tool for aspect-oriented programming in C was adapted for use with TOSSIM and TinyOS. The testing process developed from this tool is usable in both TinyOS 1.x and 2.x. This process enables run-time monitoring of WSNs. This testing process makes extensive use of simulators to recreate environmental
conditions.

**Deployment experiences:** The *CraneTracker100* has been used in a real world deployments, that span multiple months. At the time of this writing, one device has been operating continuously for over 6 months. As a result of these experiments, a significant amount of data has been collected on crane movements and behaviors. Additionally, the real-world performance of a solar energy harvesting system and GSM communication latency has been evaluated through these deployments.

### 8.2 Future Work

The work presented is only a first step in monitoring wildlife, and energy harvesting WSNs. There are several directions in which this work will proceed.

**Reconfigurability:** There is a pressing need to make the system capable of being updated once deployed. Currently, a wildlife tracking deployment is designed to test a particular hypothesis or set of hypotheses about animal behavior before the system is deployed. If interesting or unexpected behaviors are discovered during the deployment, it is not possible to adapt the system to better monitor the behavior. Instead, another deployment must be made with an updated monitoring application. This is inefficient, and wastes many valuable research opportunities. While traditional WSNs have been capable of being reprogrammed, to the best of our knowledge, no existing wildlife tracking platform is reconfigurable, post-deployment.

By making the system reconfigurable, its value will be greatly enhanced. Multiple experiments can be conducted with one deployment. If the system is completely reprogrammable, its reliability may be increased. This increase in reliability is derived from the ability to correct software faults post-deployment.

**Future biological research:** Our research partnership with the ICF has been very
productive, and both parties are eager to engage in future research endeavors. One particular project that has already been proposed is to monitor crane nests using synthetic eggs. These eggs will be capable of monitoring bird behaviors with respect to their nesting habits. Such information can reveal why the cranes are abandoning their nests, before their eggs have hatched. This research will prove challenging for the WSNs, as the crane behavior must be indirectly monitored through the synthetic egg. These eggs also create the possibility of using ad-hoc networks to relay the information to researchers, which is a challenging research area.

**Ad-hoc networking:** The cellular and 802.15.4 are both reliable communication mechanisms. In the future, the powers of both will be combined to deliver information in an even more efficient manner. By placing trackers on several family members, the trackers can form ad-hoc networks. These networks can then divide the high cost of using the GSM modem between the family members. Additionally, since the birds may be behaving similarly, which creates redundant sensor readings. This redundant information can be filtered, so the amount of data sent is minimized.

**Platform improvements:** Many practical improvements can be made to the CraneTracker100. Many of the trackers are suspected to have failed because of poor waterproofing and mechanical reliability. The current system is difficult to deploy because the solar panel, Iris and CraneTracker100 are separate physical pieces that are not well integrated. This makes it possible for the individual components to become disconnected during the deployment. These separate pieces also make it difficult to waterproof the device without adding substantial weight to the package. To address these problems, a new revision of the platform will be made. This revision will integrate the functionality of the Iris into the sensorboard. Additionally, this new design will better integrate to the solar panel. These changes will make it possible to pot the entire device in a thin layer of epoxy, which will be much more robust.
**Better energy management:** Balancing the sensing and communication needs of the system against the energy harvesting capabilities is a challenging task. The fixed duty cycle and voltage monitoring of the current system is capable of preventing the system from overdischarging its battery, and operating in low voltage conditions. However, when the energy harvesting provides abundant resources, the system does not adapt its behavior to take advantage of the excess energy. In these cases, the system should take advantage of the excess energy to perform additional data collecting and communication.

The system is currently adaptive with respect to its energy reserves. In the future, the system should also adapt to the behavior of the crane it is attached to. For example, there are several “interesting” behaviors that the birds exhibit that our partner ecologists would like to monitor. In the current system, with the fixed duty cycle, these behaviors are only observed when the birds perform them during a sampling cycle. In the future the system should be able to adjust its behavior so that it can monitor these behaviors more frequently, and balance these sensing needs against the energy reserves and energy harvesting processes.

We have already conducted a great deal of work towards this goal. A mathematical framework utilizing partially observable Markov decision processes (POMDPs) has been developed. Using these POMDPs, a method of generating control policies for the sensors has been developed. By using the Markovian facets of the POMDP, the stochastic energy harvesting can be modeled. Furthermore, the imperfect knowledge generated by the sensors leads to partial observability of a bird. This effect can also be modeled within the POMDP. By incorporating all of these aspects into the POMDP, as well as the bird behavior, it is possible to develop control policies that determine how to use the sensors to monitor a bird’s behavior, account for the energy harvesting, and imperfect sensors.

Work has already begun on developing these control policies through POMDPs. One
scenario that has already been investigated examines the impact of varying the sensor accuracies of the device with respect to their abilities to detect when a bird has perished. The results of one such experiment are shown in Figure 8.1. In this graph, the different axes represent the accuracy of the sensors, which is a controlled variable through the experiment. The accelerometer is assumed to use less energy than the GPS. The size of the markers on the graph represent how frequently a sensor is used in a given policy. Interestingly, several times the accelerometer is used in a policy, even though it is not as accurate as the GPS. For example, one instance of this occurs when the GPS accuracy is 0.75, and the accelerometer accuracy is 0.7. This shows that the system will use a less
accurate sensor, so that energy is conserved.

This approach is one possible method of more efficiently controlling the sensors, that considers that energy harvesting capabilities of the system, and the inaccuracies of the sensors. Such mechanisms are needed to get the maximum performance from the system.
Bibliography


[10] 3rd Generation Partnership Project. 3GPP. http://www.3gpp.org/, April 2011. 4.2.1, 4.4.2, 5.3


Proceedings of the 11th ACM/IEEE Conference on Information Processing in Sensor Networks (IPSN’12), April 2012. 7.3.2


[23] F. Chavez-Ramirez. Personal correspondence, 2009. 3.1


[31] FCI. 7111S2015X02LF Drawing. www.fciconnect.com/, March 2012. 4.4.1


[34] Eugenio Giordano, Raphael Frank, Giovanni Pau, and Mario Gerla. CORNER: a step towards realistic simulations for VANET. In Proceedings of the ACM International Workshop on Vehicular InterNETworking (VANET’10), pages 41–50, Chicago, IL, 2010. 2.1


[69] PowerFilm Solar. MPT4.8-75. http://www.powerfilmsolar.com/, March 2012. 4.3.1


