Building an Energy-Efficient Ad-Hoc Network for Wildlife Observation

Luis Camal and Baris Aksanli *

Electrical and Computer Engineering Department, San Diego State University, San Diego, CA 92182, USA; lcamal@sdsu.edu
* Correspondence: baksanli@sdsu.edu

Received: 1 May 2020; Accepted: 8 June 2020; Published: 12 June 2020

Abstract: This study evaluated the design of an energy-efficient ad-hoc network used for wildlife observations, particularly in order to understand the social relationships in an animal group, where the distance between individuals, i.e., proximity, can be used to measure a relationship. Our proposed network consists of a full mesh topology and contains nodes that communicate via Bluetooth Low Energy (BLE) in advertisement mode. The initial hardware configuration and software algorithm duty cycles the BLE communication to on and off states using a parameter called the BLE active triggering interval. The algorithm is improved by placing the BLE subsystem and CPU in deep sleep when there are no BLE or CPU tasks to process. This improves the power performance by up to 94.48%. To scale up power optimization and track the trade-off between power and throughput, we created a simulator that modeled our network with dynamic wireless sensor nodes. The simulator verified the base case hardware results. It also showed a median power performance increase of 97.79% in comparison to the base case, yet throughput decreased by 66.65%. The highest power performance increased by 98.89% when a wireless sensor node was configured with a BLE active triggering interval of 50 s and its CPU was set to 14 MHz; however, the simulator showed a throughput drop of 79.97%. Depending on the application, a design may tolerate the decline in throughput to achieve higher power performance.

Keywords: ad-hoc mesh network; energy-efficient; network throughput; wildlife observation

1. Introduction

Tracking animals in the wild has gained momentum, and the advancements in technology and the Internet of Things (IoT) have challenged scientists and engineers to design, build, and create devices that outperform predecessors. There are cases where animals are tracked through stationary nodes, but in such cases, the technology heavily depends on the animal being present at a particular location. Others use wireless sensors and technologies that heavily depend on a large power source. These designs include technology that is bulky and power inefficient. The main reason such technology is bulky is due to the batteries used. In some cases, the battery is the bulkiest component of the design [1]. Other devices rely on wireless technology that is power inefficient, and battery degradation happens within hours and the harvested energy is consumed quickly. For example, in Ghazali’s et al. [2] tracking system the battery voltage dropped by more than 20% within an hour. Additionally, several of these technologies and applications lack a network with data collection capability between wireless nodes. The devices are capable of tracking the physical location of an animal but cannot collect data that pertains to the behavior of a species. Many times the behaviors examined are solely based on the physical migration of a particular species. The behavior tracked pertains to a group of species as a whole but never the behavior that corresponds to interactions between individuals in those groups.
Similarly, a few companies, such as Lotek [3], Sigfox [4], and Telemetry [5], are producing wearable or implantable devices for animal monitoring and tracking.

In this study, we specifically targeted chimpanzees, one of humans’ closest living relatives [6], to understand their social behavior. They live in large social groups, up to more than a hundred individuals, with various group dynamics that result in different daily behavior [7]—splitting during the day, foraging, etc. These relationships can be cooperative, exploitative, or competitive, so can be seen as resembling to human cultures [8]. The behavior of each individual is deeply affected by their relationship with others in the group, and relationships are usually measured in terms of the proximity of different individuals over time [9]. Thus, automated behavioral monitoring of chimpanzees has very high potential for understanding the social dynamics of a chimpanzee group. These dynamics are traditionally studied manually by the scientists, which is labor-intensive and time-consuming. This is because in this method, the scientists manually identify the chimpanzees, follow them, and record their behaviors over time. Because of this manual, labor-intensive workload, it becomes very difficult to capture all interactions among the individuals. Thus, an automated solution can significantly improve the efficiency of this process.

Although other technological tracking studies or products are available in the literature or the market, they are not best-suited for our target of automatically understanding the relationships in a chimpanzee group for multiple reasons: (1) These solutions are mostly GPS-based (where a GPS unit is known to be power-hungry and we cannot use it for behavior tracking that requires frequent data collection); (2) These solutions are usually designed for less intelligent or capable animals (such as fish, cows, etc.), thus the designs are quite simplistic, not considering behavior dynamics or the possibility that the animals can take them off; (3) The devices require communication with or through a central device (e.g., a local gateway, a server connected to the Internet or a satellite in the case of GPS), and thus the individual devices would not necessarily have the capability to communicate among each other (i.e., peer-to-peer).

Instead, we propose to develop a device that uses Bluetooth Low Energy (BLE) to facilitate an ad-hoc mesh network among the individuals. We believe that our proposed method to track wildlife can introduce a new and efficient way to track chimpanzees in their natural habitat to obtain information about their social interactions. Since we are interested in group dynamics, which requires proximity, our focus is not to track the actual physical location, but rather track social behavior within a group of a species, which can be measured by physical proximity. We want to know how often they are together and participate in activities together. To that end, we are taking an animal-in-the-loop approach, introduced in [10,11], where the technology considers the movements and physical characteristics of the animal that is being monitored. Next, we explore different technological approaches for wildlife tracking and address their trade-offs with respect to our application. At the end of this section, we discuss the limitations of the existing methods, and include comparisons of their operational efficiency and effectiveness, if applicable, to our application.

2. Related Work

2.1. Image Processing, Machine Learning, and UAV Wildlife Tracking

Patman et al. [12] developed a technology called Biosense to track animal movement. Their main sensor is a camera, and they use video image processing to track animal movement and behavior. They can track spatial location, velocity, region preference, and other features in certain species. Norouzzadeh et al. [13], Guobin et al. [14], and Aodha et al. [15] make use of relatively modern algorithms in each of the respective fields. They specifically make use of deep neural networks (DNN), convolutional neural networks, or a variation to expand performance [16]. The main difference is that in [13,14] their sensor of choice is the camera, which they use to collect frames or images of wildlife. In Ref. [15], the authors use microphones to collect acoustic data for species in the wild. However, in these algorithms, they all require what is known as a training set, with labeled data that require a
high manual work overhead. Other studies include using video-tracking for fish [17], kangaroos [18], domestic cats [19], etc. Due to the success in machine learning, others such as Gemert et al. [20], Kellenberger et al. [21], and Lopez-Tello et al. [22] have extended their research and studies to wildlife tracking and identification using UAVs (unmanned aerial vehicles). These applications make use of sensors such as cameras and microphones, but the main advantage is that they can move along with the detected species and obtain more data.

Image processing and machine learning systems have significant limitations with respect to our application. Accurate machine learning applications usually depend on high processing power that requires power-hungry devices. Systems like these need one or more large power supplies to keep them running. It is not feasible to deploy such technology for our application purposes, where power will be limited; thus, deploying these systems in remote areas is not ideal. These systems also hamper budget constraints, as high-power processors are expensive. A potential solution would be to decrease the image or video quality so that the required data and processing power can be reduced. A limitation of such a solution would be that the system may not be able to distinguish individuals from one another. Images and sound may be able to distinguish wildlife from one species from another, but may not uniquely identify one individual from another in the same species group, which is what our application requires. For example, Marcot et al. [23] state that their video-based solution can work efficiently under some conditions for some species, but ultimately requires more validation from the researchers, not eliminating the manual work significantly. In another study [24], the efficiencies of camera trap solutions were analyzed, and the study found that the system failed to capture more than 40% of the events.

2.2. VHF and UHF-Based Wildlife Tracking

Unlike image processing, machine learning, and UAVs, which have been integrated and implemented into technology the past few decades, the use of radio frequencies in communications has been studied for over a century. RF has been integrated into many scientific and commercial applications, including wildlife tracking. Yoo et al. [25] and Taylor et al. [26] made use of very high frequencies (VHF) to track certain species of interest. To track wildlife, a combination of VHF tags and base stations are used. The VHF tags radiate frequency burst of either 150.10 MHz (Western Hemisphere) or 166.38 MHz (Europe). These bursts are broadcast and are detected by base stations that can interpret the signals and identify the species being tracked. The base stations can capture this data and transmit or route the data through other means of communication. A base station’s location can also be used to determine the approximate location of the tracked species of interest. Another very relevant study is Encounter tags [27], which uses proximity tags on birds. The study uses individual tags to record proximity information among birds; however, it relies heavily on local data collection gateways. In other words, the system assumes that the animals will get closer to data collection points frequently, as an individual node can keep only a certain number of events (due to using internal device RAM rather than local storage). Additionally, the study also reports that power consumption is the biggest limiting factor: when in the proximity mode, the device’s lifetime is 10 h. This is because to make the tags small, the designers sacrificed battery storage. Although these tags can be useful for our proximity tracking, the implicit assumption of a separate data collection gateway and low lifetime due to power requirements make it unsuitable for us.

2.3. GPS Based Wildlife Tracking

Kays et al. [28] claimed that the most common and popular technologies for wildlife tracking are GPS based trackers which can be designed for both small and large mammals. In Kenya, there was a problem between human encounters with wild elephants, and there were incidents of both human and elephant deaths. Wall et al. [29], designed a system that notified locals of elephants in the area. The design consists of trackers that are placed on the elephants. The sensor contained a GPS
module and used cellular towers. Whenever an elephant’s location was found near a human boundary, it notified the locals through a short message service (SMS).

EcoNet is a technology created by Huang et al. [30], and they claim that their technology can be used as an inter-species tracker. Their implementation consists of base station nodes and wireless transceivers that are harnessed around the animals. Their technology uses GPS and 3G communication between the tracker and the base stations. For node to node communication, the trackers use Zigbee communication. Wireless nodes that come in contact with each other share their unique identifiers with the neighboring nodes. The neighboring nodes record information until they are out of reach of each other. When the wireless nodes come close to a base station, the nodes uplink their stored data to the base station.

Liu et al. [31,32] developed the G-tracker, which is a wireless technology that was specifically designed to track birds. This technology makes use of GSM and GPS alike. The tracker records the birds’ locations using GPS, and stores the data in device memory. The GSM is used as a base station and the tracker on the birds uplink data to it. However, due to power constraints, the GPS is turned on every twenty minutes by using a wake-up timer. The GSM technology suffers constraints as well. Wild birds tend to fly high above for long intervals of time when migrating. When the birds are soaring at high altitudes, the GSM may not detect the trackers, which makes it difficult to uplink the data stored in the MCU. If there is no uplink when the data in the MCU is full, the data in between will be lost as newer data replaces older data. Similar GPS-based wildlife tracking studies include Fischer et al. [33] for terrestrial wildlife tracking, Foley et al. [34] and Allan et al. [35] for low-cost collars, Hertel et al. [36] for tracking individual bear behavior (using costly commercial collars [37] to equip bears), and Weiser et al. [38] for wild animal tracking. Taylor et al. [26] use commercial trackers from Lotek Inc. [3]. Trackers from this company that are suited for terrestrial wildlife include power-hungry GPS modules with very bulky battery installments.

The main limitation of GPS-based methods is the high power requirements. The GPS limits the lifetime of the system but also the granularity at which the system can record data. To compensate for a longer system life span, the GPS is duty-cycled to be active every few minutes or hours. Alternatively, energy harvesting-based methods can be used to increase the lifetimes of similar systems [39,40]. For example, Foley et al. [34] sets GPS data collection frequency at 1 or 10 reads per hour; Hertel et al. [36] uses 1 h data collection intervals. Farine et al. [41] use a GPS module to monitor bonobos that collects data at 1 Hz, but the device’s lifetime is limited to 14 days, and to account for high power usage the device includes a bulky battery (up to 300 g). Multiple minute or hour-long granularity will degrade the quality of data being collected. Our application requires frequent data collection, as we want to measure proximity, since social interactions can happen in the order of a couple of minutes or seconds, if not less. However, using GPS for this purpose would require a lot energy to feed into the GPS module, draining the batteries of the wearable device very quickly.

2.4. LPWAN Wildlife Tracking

The low power wireless area network is wireless technology that uses sensors which consume a small amount of current, and thus power. The main restriction these devices have is the distance of communication. Most technology that uses LPWAN sensors consists of nodes or devices that are limited to tens of meters in range.

Joshi et al. [42] created a system that consists of anchor nodes or fixed nodes that form small granular grids into an overall larger grid. These can be considered as base station nodes, and these nodes use GPS to determine the exact location. However, their mobile node is a dynamic node implemented with Xbee wireless technology. The sensor is placed on small turtles, and the device sends a beacon to the base stations. Each grid of four sensors listens for this beacon, and an algorithm estimates the location of the turtle in the bounded area formed by the anchor nodes. A drawback from this implementation is the base stations. The base station depends on an unlimited power source; thus, this system was deployed on a campus with many accessible power outlets. This implementation
would not be ideal in remote areas where power is limited. This implementation also fails if at least one anchor node is down since the algorithm to track turtles depends on each anchor node. Duda et al. [43] developed their own proprietary radio method to track bats. The authors say that the reason for this was to be able to develop a smaller module that is suitable for bats.

Bluetooth Low Energy (BLE) is another low power technology with several operational modes. In most commercial applications, BLE forms a connection to establish a constant data stream. However, it can also be set as a beacon that sends signals or data at certain intervals of time. Ayele et al. [44] proposed a system wherein BLE can be used to obtain data from distinct individuals in a species. Several nodes transmit their data to a source node. The source node concatenates all the different data to a packet and forwards it to a sink node that consists of a long range module (LoRa), which is also a low power device than can transmit up to 10 km. Trogh et al. [45] also used BLE for their sink or anchor nodes. They placed BLE devices on four cows and estimated their distances using the receiver signal strength indicator (RSSI) from nine different anchor nodes. They claimed that their tracking devices had an average of 4.2 m of accuracy and 2.7 m in standard deviation. Hartmann et al. conducted a similar study for bats [46]. LPWAN devices and systems are promising due to their low power consumption. However, the limitations may include the distance from which a sensor can be located; i.e., if the distances are far, tracking may not be possible. Additionally, some solutions, such as LoRa, require gateways to be deployed for data collection.

2.5. Limitations of Current Wildlife Tracking Approaches with Respect to Our Application

Table 1 summarizes the different technologies available for wildlife tracking, their accuracy and effectiveness highlights, and their limitations for our wildlife tracking application. We see that video-based tracking has high potential for capturing more events, as long as the video feed is active continuously. Energy-saving approaches for video tracking, such as using video-traps or reducing image quality, can lead to a significant number of events being missed, or individual animals may not be distinguished. Additionally, video-based solutions need a lot of power to process the images and videos, making them power-hungry, and not very suitable for our application’s purposes.

| Technology | Accuracy or Effectiveness Comments | Limitations for Our Application |
|------------|-----------------------------------|---------------------------------|
| Video [16–19,23,24] | (1) More than 40% of events can be missed with camera traps [24]  
(2) 60% more events detected than manual observing [23]  
(3) Additional processing required, such as background subtraction, Kalman filtering, etc. [17] | (1) High data volume  
(2) High processing power  
(3) High battery requirement  
(4) Difficult to distinguish individual members of the same species |
| UHF/VHF [25–27] | (1) Battery life can be low (less than a day) [27]  
(2) More than 60% of interactions are recorded one way [27]  
(3) More than 10% missing or corrupted data [27] | (1) Need for additional ground locations (e.g. base stations)  
(2) Inaccurate or frequently missing data points  
(3) Need for high power and battery when frequent readings are required |
| GPS [26,33–36,38] | (1) Infrequent data collection, i.e., low fix rate (e.g. 1 reading per hour) [36]  
(2) 2–15 m inaccuracy in readings [33,34] | (1) Power-hungry modules  
(2) High battery requirement  
(3) Unnecessary to obtain absolute location  
(4) Commercial GPS-based collars can be costly |
| LPWAN [42–44] | (1) High spatial resolution is possible [43]  
(2) Long lifetime possible [43]  
(3) More feasible to track multiple animals at the same time [43] | (1) Dependency to signal power, hence careful power settings are required  
(2) Need for additional ground locations (e.g., gateways) |
Some of the limitations with VHF trackers include the external use of large batteries, inaccurate location readings, and the need for base stations. Large batteries may be too heavy or bothersome for certain species. Inaccurate location readings are a product of inconsistent spatial coverage and low-resolution position data. Tracking wildlife with VHF technology can give inaccurate readings in the range of hundreds to thousands of meters. Their limitations are also set by their need of stationary base stations. According to Taylor et al. [26], base stations track VHF best for wildlife in flight than for wildlife on the ground. Thus, VHF may not be ideal to track wildlife that is constantly on the ground. Tracking through a base station is also prone to environmental factors and technical factors, such as poor weather, power outages, or other unexpected difficulties.

GPS devices consume a significant amount of power; thus, many systems use a duty cycle which turns the GPS on and off for particular periodic intervals of time. The duration of time for which the GPS is turned off is much larger than the duration in which the GPS is on. For example in Ref. [30], to compensate for a longer system life span, the GPS is duty-cycled to be active every ten minutes. In another study [36], the GPS reading rate is set to be 1 h. This type of granularity will degrade the quality of data being collected. Applications that depend on higher granularity, such as our application, cannot work well with this system. To understand the dynamics between individuals of a group, we need to collect frequent data, in the order of a few seconds, as behavioral activities can happen fast. Additionally, the cost of GPS trackers can range from hundreds to thousands of US dollars. We observed that custom-made devices such as [33,34] provide GPS-based wearables that cost a few hundreds of dollars, whereas devices that are bulky with significant battery capacity included may cost more than thousands of dollars. The main reason GPS-based devices are not suitable for our application is twofold: (1) a GPS unit would consume a lot of power with the data collection frequency we need; (2) the location inaccuracy can range up to 15 m [33,34] which is not suitable when meaningful interaction range between individual animals can be less than 10 m. LPWAN devices and systems can be promising due to their low power consumption. The greatest limitation for these systems is the limited tracking distance. If the distances are too far from one LPWAN sensor to another, tracking may not be possible. Additionally, some technologies require additional ground stations, such as a gateway, to facilitate the network and data transmission.

Our goal was to design a cost-effective and power-efficient tracking device due to budget constraints. Thus, we did not want to rely on expensive equipment such as high-grade processors, or items that require costly manufacturing for specific wireless communication, as the number of individual animals we want to track is high. We also could not use technology with power consumption, as the system was to be deployed in a remote area with limited access to batteries and power sources. As such, we could not include GPS technology or technology with base stations that are constantly consuming power. As a result, we chose to create a LPWAN-like network that can have peer-to-peer communication to detect meaningful interactions between individuals. For the LPWAN technology, we used BLE, which is cost-effective and known for low-power applications. The average distance of BLE communication would place a constraint on many tracking applications. Thus, we have to do careful selection of the device settings and parameters to understand the relationship between collected data quality and device lifetime (battery capacity). Additionally, our design focuses more on proximity rather than exact distance, since our focus is to track social interaction amongst individuals in a species. This is because we are not interested in where exactly each individual is in the monitored area, but whether they get close enough to each other; hence, measuring proximity is the number one goal of our application.

3. Our Wildlife Monitoring Approach and Requirements

The main trend for wildlife tracking has been in recording spatial and temporal data of a particular species. But wildlife tracking is missing the opportunity to track behavior among individuals in a species. While we are interested in spatial-temporal data, we are also interested in capturing data that track relationships individuals have over time. Not only do we care about where a species is and
at what time, or identifying who is around, but we also care about how individuals come together
and we want to identify the individuals that impact a single individual’s behaviors and actions. Our
wildlife tracking target is understanding the behavior and social lives of chimpanzees in the wild.

3.1. System Hardware Selection

Our system uses the following components to make our wildlife tracker:

- CY8C4248LQI-BL583 module by Cypress: This comes with a 32-Bit ARM0 processor and BLE wireless communication. We used this module because of its ease of use and low cost. This module provides us with the BLE functionality and the capability to interface with the inertial measurement unit and the SD card module (please see below).
- LSM6DS module: This is an inertial measurement unit that has an accelerometer and a gyroscope. This unit is placed for future use in the device, and its usage is out of scope of this paper.
- An SD card module with 32 GB memory: This provides additional storage for the data captured both by the BLE communication and the LSM6DS module.
- Battery: A 3.6 V lithium thionyl chloride battery with a charge capacity of 2600 mAh to power the components.

We have a prototype device that puts together all these individual units, shown in Figure 1.

3.2. System Network

Each tracker in the system is a node, and the protocol we are using for the nodes is 802.15.1, or Bluetooth Low Energy. This is typically used for applications consisting of a WBAN (wireless body area network); however, because of its low power consumption, we are using it to create a wireless sensor node for tracking chimpanzee dyadic levels. Cognitive scientist researchers use dyadic levels to determine the relationship between two individuals in the species of great apes [47]. The dyadic level depends on measured distances between two individuals. In the next section, we address a solution to obtain this measurement.

BLE usually consists of two wireless sensors forming a connection, where one is assigned a central role and the other the peripheral role. The central role is the sensor that requests data, and the peripheral is the sensor that provides data when queried by the central node. To do this, the protocol establishes that the central role and the peripheral role should form a connection. Barsocchi et al. [48] mentions that a BLE device has latency when forming these connections, and if we place a wireless node to sleep constantly, it could take a toll on both power and the quality of service (QoS) for a given WBAN. For our implementation this is not optimal since establishing a connection can lead to a less efficient system in terms of power and throughput. Instead, we decided to leave the BLE devices in
the advertisement state without establishing a connection. Instead of connecting multiple devices one after another, each node can directly receive advertising data from other nodes within the vicinity of its receiver’s capabilities. In short, the network is designed to be a full mesh network, but the connection is established only when two nodes are close enough to each other, as depicted by Figure 2. This also shows again that our main goal is to measure proximity, i.e., when two nodes come close enough to each other, rather than knowing the exact locations of the nodes.

3.3. System Tracking Implementation

The 802.15.1 protocol used in BLE has access to the RSSI (receiver signal strength indicator) value, and with the RSSI we can obtain the distance. This is shown in [49], where the RSSI is calculated by the Equation (1). $d$ is the distance in meters and $P_{Tx}$ is the power for the BLE transmission in dBm.

\[
\text{RSSI} = -20 \times \log_{10}(d) + P_{Tx}
\] (1)

Solving for the distance, $d$, we can then derive Equation (2) from (1).

\[
d = 10^{\frac{P_{Tx} - \text{RSSI}}{20}}
\] (2)

Therefore, our implementation does not rely on satellites or transmission of RF signals at large distances. With BLE, we can approximate the vicinity of each chimpanzee with respect to another. Although Ref. [50] states that localization—to find absolute or exact location—solely based on BLE has low accuracy. However, rather than determining an exact location for animal tracking, we use BLE to track social behavior; specifically, the dyadic levels. Rather than measuring absolute locations of each individual, we measure proximity as the distance between individuals. For the wireless node, we use the RSSI value to make estimates in distance rather exact location measurements. Traditionally, the didactic relationships are usually calculated with distances measured and estimated by human eye. These often have errors. Substituting them with distances measured with RSSI is ideal. Measuring with RSSI can cut manual labor costs and time, and it could eventually achieve more accurate results. For our purpose, we use the RSSI for proximity, and the precision in distance calculations is beyond the scope of this paper.

Challenges: One of the main concerns when designing a wildlife tracker is the life span of the device. Many articles [27,43] address this challenge but do not provide a solution to power-hungry tracking devices. Optimizing the battery life of a tracker leads to less granular data or less throughput, as the system duty cycles the communication to on and off states. In our wildlife tracking approach, we are interested in a wireless sensor node that can capture valuable data and be a power-efficient device. Thus, the main focus of this paper is to optimize the power consumption in a wireless sensor node that is used to track wildlife behavior, without compromising the throughput of the network. To simplify the solution to our problem, this paper focuses on optimizing power and throughput specifically for...
the BLE wireless communication subsystem, which consists of the CY8C4248LQI-BL583 module by Cypress. This module contains the main controller which is powered by a 32-bit ARM0 processor and contains a BLE component that uses the 802.15.1 protocol. Optimizing the total power of the system will be left for future work. In the next chapter, we will begin to address a test bench setup that will explore the initial power performance of our design.

4. Our Solution for Proximity-Based Wildlife Monitoring

Our design consist of a wireless sensor node that is mobile; therefore, we require a battery as the main power source. Batteries have finite lifetimes; thus, our wireless sensor node also has a limited lifetime. Though this design works, it may have not been the most power optimal design. Thus, we are interested in improving the lifetime of our wireless sensor node. We must first understand our initial performance and take current measurements. Obtaining current measurements will evaluate the system’s initial power performance and the initial lifespan of our wireless sensor. The initial design and the initial software algorithm will be referred to as the base case design and the base case algorithm respectively. As we later improve the power performance of our design we can compare the current consumption and lifespan to the base case design.

4.1. BLE System Specifications

The BLE antenna, in the CY8C4248LQI-BL583 BLE module, consumes 15.6 mA at 0 dBm when transmitting and 18.7 mA when receiving. The processor consumes 1.7 mA at 3 MHz when it is inactive and 1.5 μA when in deep sleep mode [51]. However, some facts are obscured here. Running the processor at 3 MHz is not valid for our wildlife tracker since some of the external components added to the wildlife tracker depend on much higher frequency clocks; thus, testing at frequencies below 10 MHz is unrealistic for our implementation. The data sheet lists the current consumption at varying CPU clock frequencies; however, it does not include the additional current consumption when BLE communication is active. Thus we had to set a test bench to measure the average current drawn at different clock frequencies while transmitting and receiving advertisement data from the BLE system. This test setup is described and illustrated in the next section.

4.2. Base Case Software Algorithm

The base case algorithm, shown in Algorithm 1, is the initial software algorithm that the wireless sensor nodes ran. The CPU clock is kept at a high frequency of 46 MHz and the transmission power of the BLE device is set to positive 3 dBm. Duty cycling a system to on and off states is proven to increase the power efficiency of that system [52]. The BLE subsystem in the CY8C4248LQI-BL583 module is set to broadcast and receive every ten seconds for a second. When the BLE subsystem is broadcasting, it transmits an advertisement packet of 31 bytes which holds its name and node ID (Address), and it can also be any custom data. Our current implementation uses the node name and node ID only. During the time in which a node is transmitting it also receives the node names and node IDs of other nodes in the network. It is also important to note that the BLE subsystem, along with the CPU, remained idle yet active, or on, during the period wherein there was no BLE advertising communication.

The flow diagram illustrated in Figure 3 is a visual of the base case system. First, the system and subsystems, such as the BLE subsystem, are initialized. Then, the program consistently checks if ten seconds have elapsed. If ten seconds have elapsed, then we set the BLE subsystem to discover other nodes and simultaneously advertise. It is important to note that in the algorithm shown here, a state machine is hidden for simplicity. It is also worth mentioning that the ground team in the wild shared that the chimpanzees in a 24 h day roam the sanctuary for 12 h from 6 a.m. until 6 p.m. As 6 p.m. approaches, the chimpanzees move towards the sleeping area and they remain in that area from 6 p.m. until 6 a.m. of the next day. Therefore, our algorithm includes a state machine that places the BLE
subsystem to sleep after a 12 h day period. From 6 p.m. until 6 a.m. the system no longer duty cycles the BLE subsystem from non-discoverable and non-advertising to discoverable and advertising.

Figure 3. Base case flow diagram.

4.3. Test Bench Hardware Setup

To measure the current consumed by the CY8C4248LQI-BL583 BLE module from the 3.6 volt lithium thionyl chloride battery, we used a 10 Ohm resistor and a measuring device called LabJack T7 Pro. The LabJack T7 Pro can measure voltages through the various analog to digital converters (ADC) at a 12-bit resolution [53]. Figure 4 shows the configuration of our test setup. To power the BLE module, a 3.6 volt battery with a charge capacity of 2600 mAH is connected in series with it. To measure the current drawn from the device, the power line between the positive terminal of the power supply and the BLE module is broken and the 10 Ohm resistor is placed in between and in series. Two ADCs, A0 and A1 were used to measure the differential voltage of the resistor. This same setup was also used to test our optimized algorithms. Knowing the differential voltage and the resistor value we can calculate the current using Ohm’s Law \((V = I \times R)\). We then derive the current drawn from the BLE system as:

\[
I_{BLE} = \frac{(V_{A1} - V_{A0})}{R}
\]  

where \(I_{BLE}\) is the current drawn from the BLE system, \((V_{A1} - V_{A0})\) is the differential voltage probed by the LabJack T7 Pro instrument, and \(R\) is the 10 \(\Omega\) resistance. Sampling from the LabJack’s ADCs and applying Equation (3) in software allows us to collect and plot current consumption data. The visuals will later be illustrated. From the derivation of Equation (3), we can also calculate the instantaneous power consumed by the BLE system by multiplying the voltage of the battery, which is 3.6 V, with the current values obtained in our data.
4.4. Base Case Results

Configuring the base case with the power measurement setup resulted in the data portrayed in Figure 5. The base case average current when the BLE communication is on is 31 mA. However, when the BLE communication is off and the CPU is left idle, the average current consumed is 17 mA.

Referring to Figure 6, we can see 1 s pulses with higher current consumption and a lower, 9 s constant current. The high pulses represent a second in which the BLE device is both transmitting and receiving, while the flat current represents 9 s in which the BLE device is idle, neither transmitting nor receiving. From the previous statement, we can conclude that there are two main states for the BLE system; that is, an active BLE state and idle BLE state respectively.

We are also interested in calculating the average current used by the BLE communication system in a 24 h day. It is important to recall that the BLE duty cycling only happens for a 12 h duration in 24 h. In the remaining 12 h period, the BLE communication system is turned off and the CPU is left
idle and consumes the average current of 17 mA. However, Equation (4) is used to calculate the actual average current of the system throughout 24 h.

$$I_{avg} = \frac{\sum_{i=0}^{n} I_i \times T_i}{\sum_{i=0}^{n} T_i}$$

(4)

where $I_{avg}$ is the average current, $n$ is the number of different device states, $I_i$ is the average current in state $i$, and $T_i$ is the time spent in state $i$. The average current for 24 h was calculated to be 17.7 mA. Therefore, multiplying the voltage of the battery, 3.6 V, with the average current, amounts to the average power consumed by the BLE system of 63.7 mW. Given the battery capacity of 2600 mAh and the average daily current, we can also calculate the life span of the system by using Equation (5).

$$B_{Life} = \frac{B_{Capacity}}{I_{avg}}$$

(5)

where $B_{Life}$ is the battery life in hours, $B_{Capacity}$ is the battery capacity in mAh, and $I_{avg}$ is the average current. Using Equation (5), we calculate that the system will operate for 146.9 h, which is 6.12 days.

In the next sections, we explore different methods to optimize our limited battery capacity and analyze these methods and the improvements they have on power.

4.5. Understanding System Modes

Referring to the CY8C4248LQI-BL583 datasheet, the BLE communication system is a subsystem of the entire BLE module [54]. The BLE has its own 24 MHz external crystal oscillator (ECO). This ECO is used to generate the correct BLE communication frequency, which is 2.4 GHz. Thus, the BLE is the only peripheral that uses the ECO. When the ECO is in use, more power is consumed; however, as mentioned before, the BLE communication is only established every ten seconds for one second, and this is for both transmission and reception. This means that at all other times, that is, for a 9 s interval, we can place the BLE in one of two sleep modes: sleep mode or deep sleep mode. When the BLE is not being used, in an idle state, the BLE subsystem can be set to sleep mode; however, the ECO is still left on. The ECO can be left on for applications that use the ECO as an external oscillator, or as an input to another subsystem. The system does not make use of the ECO except for BLE communication, and the system’s power can be optimized by placing the BLE subsystem in deep sleep mode. To reiterate, in the deep sleep mode, the ECO is turned off and less power is consumed.

The BLE subsystem also has an internal main oscillator (IMO) for the CPU, which can be set from 3 MHz to 46 MHz with the use of an internal divider. The IMO is used for code instructions, for CPU I/Os, and peripherals that the CPU manages. Similar to placing the BLE subsystem to sleep and stopping the ECO, the CPU oscillator can be put into different modes, including sleep mode and deep sleep mode. Using CPU power mode information from the device data sheet [55], we can derive:

$$I_{BLE} = 0.85 + (F_{CPU} \times 0.26) + 17.2 mA$$

(6)

where $I_{BLE}$ is the current drawn by the BLE module, $F_{CPU}$ is the BLE module CPU frequency, and 17.2 mA is the BLE transceiver average current estimate. Using Equation (6) we can calculate the theoretical current drawn from the BLE device and can apply this equation to three different frequencies: 14 MHz, 24 MHz, and 46 MHz. The purpose of this calculation is to compare the actual recorded results from the BLE module and validate the small scale test bench. Additionally, the three different frequencies were selected to compare the current and power savings between them. Table 2 shows the different theoretical currents and actual currents drawn from the BLE device at the different frequencies we are transmitting and receiving.
Table 2. CPU current drawn at different frequencies.

| CPU Frequency | Theoretical Current | Actual Current | Percent Difference |
|---------------|---------------------|----------------|--------------------|
| 14 MHz        | 21.8 mA             | 21.3 mA        | 1.8%               |
| 24 MHz        | 24.3 mA             | 23.3 mA        | 4.1%               |
| 46 MHz        | 30.0 mA             | 27.7 mA        | 7.7%               |

The percent difference or error to each of these is a product of estimating our average BLE current to be 17.2 mA from averaging the BLE transmission current, 15.6 mA, and the BLE receiver current, 18.7 mA. The averaging assumes that the transmission and the reception use the same amount of time, which is not the case, since the BLE device can be scanning or receiving for a different time interval than the transmission. Explaining the workings of this scheme is not the focus here; however, the rough estimate allows us to determine the validity of our hardware measurement. We obtained less than 10% difference on our hardware validation and we can confidently say that our hardware test bench was measuring accurately.

4.6. Optimizing System Power

Gathering all the information learned, we modified the base case firmware. The flow diagram illustrated by Figure 7 shows the flow of the updated system. The system components, which are the CPU, the real-time clock (RTC), and the BLE subsystem, are initialized. We then enter a loop and check if 10 s have elapsed. If ten seconds have elapsed, we turn on the BLE subsystem by setting it to discovery mode, which enables reception, and advertisement mode, which enables transmissions. The BLE subsystem can then receive and transmit for a one-second interval. Next, a BLE process event is checked. The system searches if there is any reception or transmission packet that needs to be fulfilled from the BLE stack. If there is, the BLE subsystem architecture takes care of it in the background. However, if the BLE subsystem has no other pending events, such as advertising, attempting to form a BLE connection, or receiving, then it enters the deep sleep mode. The only events that take place in the BLE stack for the system are advertising and receiving during the one-second interval.

![Figure 7. Improved system flow diagram.](image)

The final step to the flow of the software algorithm is to check whether the CPU can be placed in a deep sleep mode as well. The CPU can only enter deep sleep when the BLE subsystem is in deep sleep mode. This criterion is set to ensure that no BLE events are missed. If this condition is met, the CPU
system enters the deep sleep mode by turning the CPU IMO off. Additionally, a built-in real-time clock (RTC) is managed via an interrupt that is triggered every second by the watch crystal oscillator (WCO), a low-frequency clock of 32 kHz. Using the WCO is promising in contrast to the IMO or the ECO. The WCO is still accessible when both the BLE subsystem and the CPU are in deep sleep low power modes.

4.7. Optimized Algorithm Hardware Results

We implemented the algorithm outlined in the flow by Figure 7 on three different frequencies: 14 MHz, 24 MHz, and 46 MHz. Using the same test bench setup with the LabJack T7 pro, we measured both the current drawn when the BLE device is transmitting and receiving and the current drawn when the BLE device is set to deep sleep. Inclusively, we took measurements of the BLE transmission at both 3 dBm and 0 dBm. We noticed that there was no major difference with the amount of current consumed at different BLE transmission powers. Table 3 shows the currents measured for all frequencies at different transmission power levels and for three different tests.

| CPU Frequency | Test 1 Current | Test 2 Current | Test 3 Current | Total Average Current |
|---------------|----------------|----------------|----------------|-----------------------|
| 14 MHz, 0 dBm | 19.6 mA        | 19.3 mA        | 19.6 mA        | 19.5 mA               |
| 14 MHz, 3 dBm | 19.3 mA        | 19.6 mA        | 19.5 mA        | 19.5 mA               |
| 24 MHz, 0 dBm | 23.3 mA        | 23.4 mA        | 23.3 mA        | 23.3 mA               |
| 24 MHz, 3 dBm | 23.0 mA        | 23.5 mA        | 23.3 mA        | 23.3 mA               |
| 46 MHz, 0 dBm | 27.4 mA        | 27.5 mA        | 27.8 mA        | 27.7 mA               |
| 46 MHz, 3 dBm | 27.8 mA        | 27.3 mA        | 27.9 mA        | 27.7 mA               |

There is no significant difference between the 0 dBm and 3 dBm settings. Testing for actual average current values for deep sleep resulted in 2 μA for all modules set at different frequencies and different transmission powers. Running the system at 2 μA when the BLE is not in use leads to a significant improvement in power consumption. Figure 8a–c shows the current consumption during the active portion of the BLE system. The displayed current is specifically taken from our hardware test bench with CPU frequency set at 14 MHz, 24 MHz, and 46 MHz respectively. The transmission power for these configurations was set to 0 dBm. We do not intend to show the current for the 3 dBm configuration as the results are similar. These images are significant, and, in comparison to the base case, when the BLE is idle, we can see that the average idle current dropped to a very small current. The next section will demonstrate that placing the BLE subsystem and CPU into deep sleep has a great and positive impact on power consumption.

Using Equation (4), we calculated the average current and power for our updated algorithm and the different CPU frequencies in 24 h. The results reflect 12 h of duty cycling the BLE communication to on and off, and a 12 h period where the system is idle. With the updated software algorithm, when the system is in an idle state, the BLE subsystem and the CPU are in deep sleep mode. Placing the idle states in deep sleep allows us to significantly increase the power performance of the system, since
the idle current is then 2 $\mu$A instead of 17 mA. Table 4 shows the power performance improvement between the different CPU clock frequencies and the base case configuration. The percentages of improvement at the different CPU frequency configurations are comparisons of improvement in power performance against the base case algorithm at a frequency of 46 MHz.

| CPU Frequency     | Current Drawn | Power Consumed | Percent Improvement |
|-------------------|---------------|----------------|---------------------|
| Base Case (46 MHz)| 17.70 mA      | 63.72 mW       | 0.00%               |
| 14 MHz            | 0.98 mA       | 3.52 mW        | 94.48%              |
| 24 MHz            | 1.17 mA       | 4.20 mW        | 93.41%              |
| 46 MHz            | 1.39 mA       | 4.99 mW        | 92.16%              |

Table 5 shows the comparison and improvement in battery life against the base case algorithm. The comparison is shown in both hours and days. At 14 MHz the battery lives well over six times longer, at 24 MHz the battery lives well over five times longer, and at 46 MHz, which is the same frequency as the base case algorithm, the battery lives over five times longer as well.

| CPU Frequency     | Lifetime Hours | Lifetime Days |
|-------------------|----------------|---------------|
| Base Case (46 MHz)| 146.89         | 6.12          |
| 14 MHz            | 2661.48        | 110.90        |
| 24 MHz            | 2228.13        | 92.84         |
| 46 MHz            | 1874.68        | 78.11         |

5. Scaling Up Power Optimization Using a Dynamic Wireless Node Simulator

In this chapter we further explore the power saving values of our method by varying the active triggering time interval of the BLE communication. The wireless sensor node was deployed to advertise and receive for one second for every ten seconds; however, we want to see greater improvements in power consumption. We want to analyze the power savings for active triggering time intervals greater than ten seconds. This suggests that the BLE communication will be less active which would lead to less power consumption. To verify this on a large scale, we developed a simulator that uses our actual current hardware results listed in the previous chapter. The simulator uses Equation (4) to calculate the current and derive the average power and the lifetime of a wireless sensor node with varying BLE active triggering intervals. We know that increasing the BLE active triggering interval will affect the throughput of the network. For our purposes, the throughput is the total number of nodes that can be read within the vicinity of a particular node that is set by a threshold. In a later section, we mention that the threshold is a 10 m radius. The simulator will also verify the impact power optimization, through larger BLE active triggering intervals, has on the throughput of the network. We are interested in continuing to optimize the power performance of the wireless sensor nodes; however, we do not want to compromise the QoS of the network. In summary, the simulator will verify the increased power performance for varying BLE active triggering intervals and it will verify the impact it has on the network’s throughput performance. It is important to address that without the simulator, verifying the trade-off between power and the network throughput performance would require extensive testing, observations, and analysis. Our goals with simulations are as follows:

- Make feasible experiments for various unique network settings, scaling the network size up.
- Analyze the optimization of power by setting different lengths of active triggering time intervals.
- Ensure that further power optimization does not degrade network throughput.
5.1. Simulator Overview

Though the simulator allows us to explore improvements to our algorithm’s power performance, it is of significant importance that it can mimic realistic random movements of wireless nodes. Specifically, the wireless nodes represent the movement of chimpanzees in a sanctuary. For our purposes and simplicity, we condense the sanctuary to be represented by 1 hectare, an area of 100 m by 100 m or 10,000 square meters. We also know the following information:

- The chimpanzees have a designated area for feeding.
- The chimpanzees are free to move in this area throughout 12 h in a day.
- The chimpanzees have a designated time to sleep and have a designated area to sleep.

We divided the hectare area into four quadrants. From left to right and top to bottom, the first quadrant represents the feeding area, the second and third quadrants represent areas in which the chimpanzees are free to roam around, and the fourth quadrant represents the designated sleep area. Figure 9 gives a visual of the quadrant assignments. The visual shows the wireless sensor nodes, which would be placed on the chimpanzees as green dots; however, each chimpanzee is represented by a specific and unique coordinate.

![Figure 9. Simulator visual.](image)

The simulation runs for a specified time in seconds. The chimpanzees will move to different quadrants at given times throughout the day. For example, at around noon, the chimpanzees are fed in a designated area and at a specified time. The chimpanzees know this and will begin to move to that area once the time approaches. Analyzing Figure 9, we can tell that the feeding time was near, as most of the chimpanzees (nodes in green) were in quadrant I, the eating area. Similarly to this example, other events take place based on the chimpanzees’ behavior. Recall that the wireless sensor nodes capture the interactions an individual animal has with another animal based on their proximity. We want the simulator to capture data as the wireless sensor nodes would in a real-world scenario.

5.2. Simulator Model

For the simulator to behave as realistically as the chimpanzees moving through the sanctuary, we used Markov chains. A Markov chain is a stochastic process where the next state of a system only depends on the present state \[56\]. The Markov chain state diagram we implemented is shown in Figure 10, where each circle represents a state of a chimpanzee in a given quadrant. The first quadrant is the eating area, the second and third quadrants are free space area, and the fourth quadrant is the sleeping area. The diagram also includes directional arrows from one state to another, including a directional arrow from a state to itself. Each directional arrow contains a weight which is the probability from transitioning from the present state to the next state. Mathematically, the probability from one state to another can be represented and shown in Equation \(7\) \[57\]. This equation represents the
conditional probability a chimpanzee transitions from one region $i$ at time $t$ to another region $j$ at time $t+1$. The conditional probability is then represented as $P_{i,j}$, and is shown in Figure 10 near each state transition or directional arrow. We use half an hour or 1800 simulated seconds as the granularity of time $t$.

$$Pr((X(t+1)) = j \mid X(t) = i) = P_{i,j} \tag{7}$$

We mathematically translate the Markov chain and use a matrix called the transition matrix, or probability matrix, to model the transition from one state to another. A transition matrix contains the probabilities for each state transition shown in Figure 10. The transition matrix contains rows and columns that represent each state in the Markov chain. Specifically, the rows represent the current state at a given time $t$. The columns represent the next state of a transition that will take place at time $(t+1)$. Thus, each element in the transition matrix contains the conditional probability of transitioning to the next state $j$ given the current state $i$, where $i = 1, 2, 3, 4$ and $j = 1, 2, 3, 4$.

$$Transition\ Matrix = \begin{bmatrix} P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \\ P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} \end{bmatrix} \tag{8a}$$

It is important to address that a single transition matrix represents a single event in the simulation. This happens because different events occur throughout a chimpanzee’s day. An event can be classified as a process that drives a chimpanzee to a specific response. For example, when a chimpanzee knows that the feeding time approaches, the chimpanzee will move towards the eating area with a higher probability. Thus, different time boundaries trigger different events, or Markov chains, in which different probabilities are used to determine the transition of a chimpanzee from one area to another. To account for important events, we created a transition matrix for different activity movements observed from the chimpanzees. We have a total of five probability matrixes which represent the following activities: uniform movement throughout the four quadrants, movement towards the designated eating area, active movement after the chimpanzees are fed, movement towards the designated sleep area, and still movement, which represents grooming. The probability matrices are in Equation (9).
Each row and column pair represents a state transition. The left-most column and the rightmost column represent quadrant 1 and quadrant 4 respectively. Similarly, the topmost and the bottom-most rows represent quadrant 1 and quadrant 4 respectively. The rows represent the current quadrant and the columns represent the quadrant a chimpanzee is transitioning into. Thus, the index pair (row = 1, column = 2) represents the state transition from quadrant 1 to quadrant 2. If the column were to be column equal to 3, then that index pair would represent state transition from quadrant 1 to quadrant 3, and so on for the remaining index pairs. Therefore, a transition matrix is used to represent the transition of a chimpanzee when it is found in a quadrant at a particular time. For example, the uniform matrix has an evenly distributed probability of 25% for each of the quadrants. This suggests that a chimpanzee has equal probability to transition to any of the four quadrants or areas, including its current quadrant position. The active matrix suggests that a chimpanzee is most likely to move to other quadrants, with equal probability, but with a lesser probability to stay in the quadrant that the chimpanzee is currently in. Thus, for the current quadrant, there is only 10% probability and the remaining probability is evenly spread amongst the other quadrants—that is, 30% each. The sleep matrix is used to represent the times of day where the chimpanzees are most likely to be in or near the sleeping area. Thus, the matrix holds a much higher probability in column 4, which represents that there is a higher probability for a chimpanzee to transition towards the sleeping area. The eat matrix is used to model the movement of the chimpanzees during their time to feed. Thus, a higher probability is assigned in column 1. The other quadrants were assigned much smaller probabilities but evenly distributed. The still matrix models the times that a chimpanzee is most likely to stay in its current area. Thus, the diagonals hold the highest probability and the other entries hold much smaller, but evenly distributed, probabilities. The simulator is not limited to these transition matrices, but versatile enough to work with additional transition matrices if needed.

The transition from one state to another state also depends on time boundaries that signal a different Markov chain. It is at the time boundaries a simulated chimpanzee, with a wireless sensor node, is able to transition from one area to another. These time boundaries are simulated time events that occur in a real world scenario. The time boundaries are as follows:

- **init_start_time**: The start of the day, which is second 0 of the day.
- **awake_time**: The time the chimpanzees begin to move away from the sleeping area.
- **feed_time_start**: The time at which the chimpanzees begin to be feed and go to the feed area.
- **feed_time_end**: The time at which the chimpanzees are no longer fed.
- **time_to_sleep_start**: The time at which the chimpanzees begin to move to the sleep area.
- **use_probability_matrix_interval**: The time interval at which the currently selected probability matrix should set the new position of the nodes in the wireless full mesh network.
5.3. Simulator Algorithms

The simulator needs an algorithm that can feasibly perform our goals. The flow diagram of the algorithm is depicted in Figure 11. The simulator’s algorithm is outlined in Algorithm 1, and it implements the Markov chain model. It also uses the hardware calculations obtained from Section 4 and contains several parameters that must be predefined.

![Simulator algorithm flow diagram.](image)

The simulation framework is implemented with MatLab [58]. The algorithm has four main loops. The first loop iterates through the different number of nodes in the node list, \( n \), the current number of nodes, defines the density of the wireless full mesh network for the current iteration. The second loop iterates through the different active period intervals and sets the active period interval, \( active\_period\_interval \), for each of the nodes, \( n \), in the current iteration. The third loop iterates through 50 random seeds for the current wireless full mesh network that consists of the current number of nodes, \( n \), and the current active interval period, \( active\_interval\_period \). Once we are in the inner loop, we take the current seed, \( s \), and set the random seed generator on that seed. We then initialize the position of all the nodes into the defined grid of 100 m by 100 m. The fourth loop represents the time, in seconds, the BLE system is in use. It particularly represents the 12 h period, from 6 a.m. to 6 p.m., that the node is switching between its active cycle and sleep cycle. During the latter 12 h of a day, from 6 p.m. to 6 a.m., the wireless sensor node is placed in a deep sleep state; therefore, there are no throughput data, and the average current consumption of a wireless sensor node during this period is constantly at 2 \( \mu A \). Inside the timing loop, we set the new position for each network node. The new position is calculated using Algorithm 2. Let us analyze Algorithm 2 in depth.
Algorithm 1 BLE dynamic node simulator for a full mesh network

Require: Deployed area in meters, Area Divided into Quadrants, Probability Matrices, Time Boundaries, Define WSN Distance of Rx Sensitivity, List for Different Number of Nodes, Set Distance of Rx Sensitivity, List for Different BLE active period intervals, Number of Random Seeds

1: for n = node in [10, 15, 20, 30, 40, 50] do
2:   for ble_active_triggering_interval = ble active triggering interval in [10, 12, 15, 20, 25, 30, 35, 40, 45, 50] do
3:     for s = seed in [1...50] do
4:       setRandomSeed(s)
5:       positionOfNodes = createNetworkMatrix(n)
6:       for t = time in [1...(12 × 60 × 60)] do
7:         positionOfNodes = getNewPositionofNodes()
8:         if t ≥ init_start_time && t ≤ awake_time then
9:           probability_matrix = uniform_probability_matrix
10:          else if t ≥ feed_time_start && t ≤ feed_time_end then
11:             probability_matrix = eat_probability_matrix
12:            else if t ≥ time_to_sleep_start && t ≤ time_end then
13:                probability_matrix = active_probability_matrix
14:               else if t ≥ time_to_sleep_end && t ≤ time_end then
15:                   probability_matrix = sleep_probability_matrix
16:               else
17:                 probability_matrix = still_probability_matrix
18:               end if
19:             end if
20:           if mod(t, ble_active_period_interval) == 0 then
21:             positionOfNodes = moveQuadrant(probability_matrix)
22:           end if
23:         end for
24:       end for
25:       throughput_matrix = recordThroughput(positionOfNodes)
26:     end for
27:   end for
28: end for
29: recordAverageThroughputPerNodePerActivePeriodInterval()
30: recordStandardDeviationOfThroughputPerNodePerActivePeriodInterval()
31: calculate24HourPowerConsumptionPerActivePeriodInterval()
32: recordNetworkAverageTotalThroughput()
33: end for

Algorithm 2 Calculate the new position of a node in the full mesh network

Require: Current X and Y coordinates, Step Size Gain

new_position_x_max = current_position_x + step_size
new_position_x_min = current_position_x - step_size
new_position_y_max = current_position_y + step_size
new_position_y_min = current_position_y - step_size
new_position_x = randChoose(new_position_x_max, new_position_x_min)
new_position_y = randChoose(new_position_y_max, new_position_y_min)

To calculate the new position of a node, we need the current node position in both coordinates and the step size at which the node can change for both coordinates. We set the nodes to move by a step of 0.1 m, which is set as a constant during the simulation. With these parameters, the algorithm finds the new maximum position and minimum position for both the X and Y coordinates by adding the step size and subtracting the step size respectively from both the X and Y coordinates, obtaining the minimum and maximum coordinate values after a movement in one time step. The last step is to randomly choose a position within the movement range for both coordinates. Back to Algorithm 1, the next step is to check the time boundaries to set the appropriate transition matrix. Setting the correct transition matrix is crucial so that the simulation can closely track a real-world scenario. If we are within the time interval use_probability_matrix_interval, we set the new positions of the wireless nodes based on the transition matrix. As mentioned before, the transition matrix determines the new quadrant and position of each node stochastically. Algorithm 3 shows the algorithmic process to determine a node’s new quadrant and position when a transition matrix is activated.
Algorithm 3 Move node position utilizing the probability matrix

Require: Current $4 \times 4$ Probability Matrix, X and Y Coordinate of each node

\begin{verbatim}
current_quadrant = determineCurrentQuadrant(X, Y)
cumulative_probability_array = computeCummulativeProbabilityArray(current_quadrant, current_probability_matrix)
random_probability = getRandomProbability()
for single_cumulative_probability = cumulative probabilities in cumulative_probability_array do
    if single_cumulative_probability_index == 1 then
        if random_probability <= cumulative_probability_array(single_cumulative_probability_index) then
            next_quadrant = quadrant
            break
        end if
    else
        if random_probability >= cumulative_probability_array(single_cumulative_probability_index - 1) \&
        random_probability <= cumulative_probability_array(single_cumulative_probability_index) then
            next_quadrant = single_cumulative_probability_index
            break
        end if
    end if
end for
\end{verbatim}

Algorithm 3 begins with the given X and Y coordinates of the node to determine the current quadrant in which the node is located. Next, we calculate the cumulative probability array using individual probability values in respective transition matrix. These probabilities each represent a probability at which a node in the current quadrant can transition to any of the four quadrants. Returning to Algorithm 3, a random probability is generated and assigned to a variable called random_probability. The algorithm iterates through the different cumulative probabilities in the array cumulative_probability_array. If the random probability, random_probability, lies within the boundaries of two cumulative probabilities in the cumulative probability array, then the next quadrant is equivalent to the index of the current iteration. The index is given the variable name single_cumulative_probability_index.

Back to Algorithm 1, the last step in the inner loop is to verify that the current "time" corresponds to a time at which the BLE should be actively transmitting and receiving—this simulates the wireless sensor nodes’ behavior, which is determined by the software algorithm. If it does, then we record the throughput based on the current positions of the nodes and store that information in a matrix called throughput_matrix. The next lines to the algorithm simply record all the valuable information that we need to analyze power and throughput performance for the different combinations of network densities, different BLE active triggering intervals, and different initial node locations.

5.4. Simulator Parameters

Our wireless sensor node is meant to detect proximity with respect to other nodes. Field researchers placed the constraint of ten meters and below on meaningful proximity data. In the simulator, there is node-to-node throughput or communication when the Rx (reception) sensitivity is an equivalent distance of 10 m or less. As mentioned in Section 4, we use the actual hardware results pertaining to the current drawn from our experiments in Section 3 to find the correct hardware parameters. We took different current measurements of the BLE system at different CPU clocks or frequencies. The simulator takes a list of the hardware’s actual average currents with respect to the BLE transmission at different CPU clocks. Therefore, the simulator will loop for each CPU frequency and record data for the BLE system set at that particular clock. The list consists of the following currents for 14 MHz, 24 MHz, and 46 MHz respectively: 19.5 mA, 23.3 mA, and 27.7 mA.
A major goal for the simulations is to scale up and test several network densities; therefore, we specify a list of the different numbers of nodes for a full mesh network. The node list consists of: 10 nodes, 15 nodes, 20 nodes, 30 nodes, 40 nodes, and 50 nodes. Node counts of 10 nodes, 15 nodes, and 20 nodes are more realistic and align more with the number of nodes that cognitive science researchers would want to initially deploy. However, we wanted to show that the system could work for a larger wireless full mesh network. Additionally, we would like to see the power and throughput results with the wireless sensor nodes with different BLE active triggering intervals. To analyze the impact of the distinct BLE active triggering intervals, we use the following times: 10 s, 12 s, 15 s, 20 s, 25 s, 30 s, 35 s, 40 s, 45 s, and 50 s. Note that the first three intervals differ by 2 s and 3 s respectively. This was done to analyze the effect between smaller increments in BLE active triggering time intervals. We gathered data that showed an improvement in the power performance, with drop in throughput, however. With power and throughput data we can discover the optimal solution for the wireless sensor node.

To model animal movement, we use Markov chains that consist of transitions from one area to another. For example, we specifically know that the chimpanzees start and end their day in a designated area for sleeping. We also understand that they can move differently each day or start and end their day differently. At the beginning of a day, an individual may start at a different coordinate or location. Thus, we gather network throughput data for different days, by running experiments with fifty random seeds for each node number and BLE active triggering interval pair. A summary of the important simulator parameters is listed in Table 6.

| Name                        | Description                                                                 | Value(S)                                                                 | Explanation                                                                                       |
|-----------------------------|-----------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| Node Maximum Distance Read  | The maximum distance at which a node will record another node within its vicinity. | 10 m                                                                                                                     | The ground team does not consider a distance greater than 10 m a meaningful relationship to record. |
| Average CPU Currents with active BLE session | A list of average currents when the BLE module is active. | 19.5 mA, 23.3 mA, 27.7 mA                                                                                           | These currents are actual currents measured from the hardware.                                  |
| Sleep Current               | The current when the BLE communication is not active (when the BLE subsystem and the CPU are placed to sleep). | 2 µA                                                                                                                      | This current is an actual hardware average current measured from the BLE subsystem and CPU when placed to a sleep state. |
| BLE Active Triggering Intervals | Time intervals that wake up the CPU and trigger the BLE subsystem to be active for a 1 s interval. | [10 s, 12 s, 15 s, 20 s, 25 s, 30 s, 35 s, 40 s, 45 s, 50 s] | The base case uses 10 s and using larger intervals leads to lower power as the system gets less active. |
| Network Densities           | A list of different network densities. (each value is the number of nodes in the network) | [10, 15, 20, 30, 40, 50]                                                                                          | The initial network has 10 nodes, and we are interested in up-scaling and showing that our system works for larger network sizes. |
| Initial Node Position       | The initial position of each node in the network. These are set randomly and initialized in the sleep area. | 24 h                                                                                                                     | The chimpanzees wake up each morning in a designated sleep area.                                |
| Simulated Run-Time          | The total simulation time for an experiment. (the nodes are actively duty cycling in 12 h based on the BLE active triggering interval and the other 12 h the nodes are strictly sleeping.) | -                                                                                                                        | -                                                                                                 |
| Output Power                | The average power of a single node. The value depends on the BLE active triggering interval (in mW). | All nodes will have the same power consumption if they have the same BLE active triggering interval. This does not depend on throughput. | -                                                                                                 |
| Output Throughput           | The number of times a node reads another node in its vicinity. The value depends on the number of nodes within the vicinity of a node. | This depends on the random movement of the nodes as simulated by the simulator. | -                                                                                                 |

5.5. Results

After running our algorithm for various scenarios, we gathered information regarding power consumption and corresponding throughput values. Many scenarios will not be covered, but the most crucial data will be analyzed and discussed.
5.5.1. Power Results

The power results depend on the BLE active triggering interval and the CPU frequency of the wireless sensor node. The wireless sensor node is duty cycling between an active BLE communication and no BLE communication for 12 h. Activating the BLE communication depends on the BLE active triggering interval, \textit{ble\_active\_triggering\_interval}. This means that every \textit{ble\_active\_triggering\_interval} seconds the BLE communication is on for a one-second interval. In the remaining 12 h, the wireless sensor nodes are in sleep mode and only consume 2 µA. For different CPU frequencies, the simulator sets the nodes to simulate current consumption based on one of the following three currents: 19.5 mA, 23.3 mA, and 27.7 mA. The currents correspond to the CPU set at 14 MHz, 24 MHz, and 46 MHz respectively. We obtain information for the nodes with different combinations of BLE active triggering intervals and CPU frequency currents. For each combination, we obtain different power results. Since we have three different current consumption values, based on the three CPU frequencies, and 10 different BLE active triggering intervals, we obtain 30 power results. All power results (average current, average power, node lifetime) apply to a single node.

As we increase the BLE active triggering interval, the average power consumption drops. This is true at all frequency settings, however, recalling from the hardware experiments, the current drawn from the BLE device at a higher frequency is higher. Thus, at higher frequencies, we experience higher power consumption in comparison to lower frequencies. To compute the average power, we first use Equation (4) to obtain the average current consumption in a simulated 24 h day. This current is then multiplied with the voltage of the battery, which is 3.6 V. The results are shown in Table 7 and represent the average power consumed for a single wireless sensor node in 24 h.

| CPU Frequency | 10 s | 12 s | 15 s | 20 s | 25 s | 30 s | 35 s | 40 s | 45 s | 50 s |
|---------------|------|------|------|------|------|------|------|------|------|------|
| 14 MHz        | 3.52 mW | 2.93 mW | 2.35 mW | 1.76 mW | 1.41 mW | 1.18 mW | 1.01 mW | 0.88 mW | 0.79 mW | 0.71 mW |
| 24 MHz        | 4.20 mW | 3.50 mW | 2.80 mW | 2.10 mW | 1.68 mW | 1.41 mW | 1.21 mW | 1.06 mW | 0.94 mW | 0.85 mW |
| 46 MHz        | 4.99 mW | 4.16 mW | 3.33 mW | 2.50 mW | 2.00 mW | 1.67 mW | 1.43 mW | 1.25 mW | 1.12 mW | 1.00 mW |

Next, we calculate the percentage of average power saved between adjacent values in the list of BLE active triggering intervals. That is, the average percentage of power saved between 10 s and 12 s, 12 s and 15 s, 20 s and 25 s, etc. The percent of power saved transitioning from one frequency to the other shares a similar pattern. Table 8 shows the percentages calculated. By thoroughly examining the numbers, we can see that the power saving increases when transitioning from 10 s to 12 s, 12 s to 15 s, and 15 s to 20 s. As we transition from a large BLE active triggering interval to a slightly larger active triggering interval, our power saving begins to decrease significantly. As the BLE active triggering interval grows larger, the system as a whole consumes much less power since the system duty cycle is kept longer in deep sleep. Thus, approaching larger BLE active triggering intervals we approach closer to a more constant deep sleep average power. Figure 12a is a visualization of the power saving pattern across all three frequencies. As seen in the plot, the peak saving is made between the transition of 15 s to the choice of a 20 s BLE active period interval.

| CPU Frequency | (10 s, 12 s) | (12 s, 15 s) | (15 s, 20 s) | (20 s, 25 s) | (25 s, 30 s) | (30 s, 35 s) | (35 s, 40 s) | (40 s, 45 s) | (45 s, 50 s) |
|---------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 14 MHz        | 20.00%      | 25.00%      | 33.33%      | 24.14%      | 20.83%      | 14.29%      | 16.67%      | 12.50%      | 14.29%      |
| 24 MHz        | 19.44%      | 24.13%      | 34.88%      | 26.47%      | 17.24%      | 16.00%      | 13.63%      | 15.78%      | 11.76%      |
| 46 MHz        | 20.68%      | 24.28%      | 34.61%      | 23.80%      | 20.00%      | 16.67%      | 15.38%      | 13.04%      | 9.52%       |
We explore further power savings by increasing the BLE transceiver active triggering interval. The results are shown in Figure 12b,c. A larger BLE active triggering interval is inversely proportional to the current consumed by the system. When the BLE communication is less frequently active there are current savings that will correlate to power savings. Thus, the lifetime to the wireless sensor node is directly proportional to the BLE active triggering interval and we see the lifetime increase. The percent power improvement from the base case ranges from 92.16%, which sets the CPU at 46 MHz and the BLE active triggering interval to 10 s, to 98.89%, which sets the CPU at 14 MHz and the BLE active triggering interval to 50 s. The lifetime of a node ranges from 78.11 days to 549.97 days for the respective settings. This is a significant improvement from the base case lifetime of 6.12 days.

5.5.2. Throughput Results

As shown in Table 6, throughput is the number of times a node reads another node in its vicinity, and it is not limited to reading one node but can read multiple nodes. The average throughput for a single node is the average throughput for fifty distinct n-dense network events, where n is the number of nodes in a network event. An event is composed of multiple unique interactions that each node has with other nodes during the time that the network is active. The total average throughput of the network is the sum of all the nodes’ average throughputs. The throughput results depend on the network density. The more nodes there are, the more probable that a single node can record more neighbors. The throughput results also depend on the number of events, which in our case is set by fifty random seeds and fed to a random generator. The last parameter is the BLE active triggering interval because this parameter turns on the BLE communication for one second every ble_active_triggering_interval number of seconds. Therefore, the longer the BLE active triggering interval is, the fewer times the BLE communication broadcasts and receives. This affects two aspects of the network. First, a longer BLE active triggering interval suggests that a node makes itself known less to other nodes. Second, a longer BLE active triggering interval suggests that a node can read other nodes in its vicinity less frequently.

Throughput changes are significant between the different combinations of node count and BLE active triggering intervals. The throughput results are important, as they allow us to understand the impact our power-saving algorithm has on the performance of our full mesh network. Figure 13a shows the average total throughput data with different network densities and different active period intervals. Looking at this figure, we see the average throughput decreases as active triggering interval gets bigger. We show that increasing the BLE active triggering interval decreases the power consumption at the expense of the network’s performance.
Next, we show the throughput percent drop as the ratio of the throughput of any of the BLE active triggering interval that is greater than ten seconds over the throughput for the base case (10 s). Figure 13b depicts the throughput decrease with increasing BLE active triggering interval. We see that the throughput decrease is larger for longer BLE active triggering intervals. Note that the throughput decrease is similar for different network sizes; thus, we show only one line in this figure. For our particular application, we do not want to lose more than 50% throughput, in order to obtain consistent and meaningful tracking data. If we desired to further optimize the power consumption of the system, BLE active triggering intervals below 20 s are the better performers for our purposes.

5.5.3. Scaling Up the Network Size

This section analyzes the effect of scaling up the network size. To do this, we analyze each node’s average total throughput and the standard deviation between the total average throughput for fifty random events. The average throughput for each node along with its respective standard deviation can tell us the variability that would be expected if the experiment would be repeated. Figure 14a shows the average total throughput for ten nodes for a simulation event in the thick blue bars. It also contains an error line which represents the standard deviation per average bar. This percentage in the error bar tells us the uncertainty in our reported throughput measurement. Let us examine a network with twice as many wireless nodes. Referring to Figure 14b, we see that the maximum variation happens at node number 11. The percent difference the node experiences from the average throughput is 17.72%, which is less than the previous example. Figure 14c shows the error bar for our densest network of 50 nodes. Node 32 experiences the highest variation in its average throughput for 50 random events, and the percent difference from the average throughput is 16.53%.

We took a further step and expanded our tests to include a network with size 75 and 100. The average throughput per node in 50 random events was generated for these network densities. Since
we are interested in the variation and uncertainty, we plotted the maximum coefficient of variation per n-dense network. The coefficient of variation consists of the throughput standard deviation between a node’s recorded throughput in 50 random events, divided by the average throughput for that same node in those same events. We take the maximum coefficient of variation since this indicates the largest variation we see per n-dense network. The coefficient of variation in the simulator, for network densities ranging from 10 nodes to 100 nodes, is illustrated in Figure 15. The results show more throughput variability for less dense networks relative to the average throughput per node. Hence, there is more confidence in denser networks. As the network size scales up, the simulator can handle it with diminishing variation.

![Figure 15. Maximum coefficient of variation per network for varying BLE active triggering intervals.](image)

6. Conclusions

This paper evaluated the design of an energy-efficient network used for wildlife observation. The network consists of a full mesh topology and contains nodes that communicate via Bluetooth Low Energy (BLE) in advertisement mode. The initial hardware configuration and software algorithm duty cycles the BLE communication to on and off states using a parameter called the BLE active triggering interval. This parameter triggers the BLE communication to be on for 1 s. Turning the BLE communication off results in lower power consumption. The base case wireless sensor node has a BLE active triggering interval of 10 s and a CPU clock set to 46 MHz.

The algorithm was improved by placing the BLE subsystem and CPU in deep sleep when there are no BLE or CPU tasks to process. This alone improves the power performance by 92.16%. We improve power performance further by 93.41% and 94.48% by changing the CPU clock of the wireless sensor node from 46 MHz to 24 MHz and 14 MHz, respectively. To further optimize the power of our device, we used longer lengths of BLE active triggering intervals. This implies a drop in the network’s throughput performance. We tracked this effect for varying network densities by scaling up power optimization and analyzing the trade-off between power and throughput. We created a simulator that models our network with dynamic wireless sensor nodes. The simulator verified the base case hardware results. It also showed a median power performance increase of 97.79% in comparison to the base case, yet a throughput decrease of 66.65%. The highest power performance increased by 98.89% when a wireless sensor node was configured with a BLE active triggering interval of 50 s and its CPU was set at 14 MHz; however, the simulator showed the throughput drop by 79.97%. Depending on the application, a design may tolerate the decline in throughput to achieve higher power performance.

**Author Contributions:** Conceptualization, L.C.; Funding acquisition, B.A.; Investigation, B.A.; Methodology, L.C. and B.A.; Project administration, B.A.; Resources, B.A.; Software, L.C.; Supervision, B.A.; Visualization, L.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.

References

1. Roberts, E.; Last, D.; Roberts, G. Accurate current measurement using drogued buoys employing Decca Navigator and Argos. IEEE J. of Ocean. Eng. 1991, 16, 307–312.
2. Ghazali, M.A.; Othman, K.A.; Isa, M.A.H.M.; Baharuddin, M.A. Wireless sensor development for Malaysian forest monitoring and tracking system. In Proceedings of the 2017 21st International Computer Science and Engineering Conference (ICSEC), Bangkok, Thailand, 15–18 November 2017.
3. Lotek Inc. Lotek Wireless—Fish and Wildlife Monitoring. 2020. Available online: http://www.lotek.com/ (accessed on 1 May 2020).
4. Thomson Reuters. Tiny Trackers Connect Endangered Rhinos to the Internet of Things to Fight Poaching. 2020. Available online: https://reuters.screenocean.com/record/1369808 (accessed on 1 May 2020).
5. Telemetry Solutions. Premium Gps Telemetry Solutions for Wildlife and Livestock. 2020. Available online: https://www.telemetrysolutions.com/ (accessed on 1 May 2020).
6. Krupenye, C.; Kano, F.; Hirata, S.; Call, J.; Tomasello, M. Great apes anticipate that other individuals will act according to false beliefs. Science 2016, 354, 110–114.
7. Aureli, F.; Schaffner, C.M.; Boesch, C.; Bearder, S.K.; Call, J.; Chapman, C.A.; Connor, R.; Fiore, A.D.; Dunbar, R.I.; Henzi, S.P.; et al. Fission-fusion dynamics: New research frameworks. Curr. Anthro Pology 2008, 49, 627–654.
8. Whiten, A.; Goodall, J.; McGrew, W.C.; Nishida, T.; Reynolds, V.; Sugiyama, Y.; Tutin, C.E.; Wrangham, R.W.; Boesch, C. Cultures in chimpanzees. Nature 1999, 399, 682.
9. Silk, J.; Cheney, D.; Seyfarth, R. A practical guide to the study of social relationships. Evol. Anthropol. Issues News, Rev. 2013, 22, 213–225.
10. Pons, P.; Jaen, J.; Catala, A. Towards future interactive intelligent systems for animals: Study and recognition of embodied interactions. In Proceedings of the 22nd International Conference on Intelligent User Interfaces, Limassol, Cyprus, 13–16 March 2017; pp. 389–400.
11. Mancini, C. Animal-computer interaction (ACI): Changing perspective on HCI, participation and sustainability. In Proceedings of the CHI 2013, Paris, France, 27 April–2 May 2013; pp. 2227–2236.
12. Patman, J.; Michael, S.C.J.; Lutnesky, M.M.F.; Palaniappan, K. Biosense: Real-time object tracking for animal movement and behavior research. In Proceedings of the 2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), Washington, DC, USA, 9–11 October 2018; pp. 1–8.
13. Norouzzadeh, M.S.; Nguyen, A.; Kosmala, M.; Swanson, A.; Palmer, M.S.; Packer, C.; Clune, J. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. Proc. Nat. Acad. Sci. USA 2018, 115, 25.
14. Chen, G.X.; Han, T.X.; He, Z.X.; Kays, R.X.; Forrester, T.X. Deep convolutional neural network based species recognition for wild animal monitoring. In Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP), Paris, France, 27–30 October 2014; pp. 858–862.
15. Aodha, O.M.; Gibb, R.; Barlow, K.E.; Browning, E.; Firman, M.; Freeman, R.; Harder, B.; Kinsey, L.; Mead, G.R.; Newson, S.E.; P; et al. Bat detective—Deep learning tools for bat acoustic signal detection. PLOS Comput. Biol. 2018, 14, 3.
16. Bowley, C.; Andes, A.; Ellis-Felege, S.; Desell, T. Detecting wildlife in uncontrolled outdoor video using convolutional neural networks. In Proceedings of the 2016 IEEE 12th International Conference on e-Science (e-Science), Baltimore, MD, USA, 23–27 October 2016; pp. 251–259.
17. Lantsova, E.; Voitiiuk, T.; Zudilova, T.; Kaarna, A. Using low-quality video sequences for fish detection and tracking. In Proceedings of the 2016 SAI Computing Conference (SAC), London, UK, 13–15 July 2016; pp. 426–433.
18. Herbert, C.A.; Dassis, M.; Pye, M.; Jones, P.W.; Leong, P.H.W.; Thomas, G.; Cope, H.; Jarman, A.; Hobbs, R.; Murray, P.E.; et al. Development of light-weight video-tracking technology for use in wildlife research: A case study on kangaroos. Aust. Zool. 2020, 40, 364–378.
19. Huck, M.; Watson, S. The use of animal-borne cameras to video-track the behaviour of domestic cats. Appl. Anim. Behav. Sci. 2019, 217, 63–72.
20. Van Gemert, J.C.; Verschoor, C.R.; Mettes, P.; Epema, K.; Koh, L.P.; Wich, S. Nature conservation drones for automatic localization and counting of animals. *Eur. Conf. Comput. Vision 2014*, 2014, 255–270.

21. Kellenberger, B.; Volpi, M.; Tuia, D. Fast animal detection in UAV images using convolutional neural networks. In Proceedings of the 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, USA, 23–28 July 2017; pp. 866–869.

22. Lopez-Tello, C.; Muthukumar, V. Classifying acoustic signals for wildlife monitoring and poacher detection on UAVs. In Proceedings of the 2018 21st Euromicro Conference on Digital System Design (DSD), Prague, Czech Republic, 29–31 August 2018; pp. 685–690.

23. Marcot, B.G.; Lorenz, T.J.; Fischer, P.; Weinstein, B.G.; Cowell, S. Efficacy of automated detection of motion in wildlife monitoring videos. *Wildl. Soc. Bull. 2019*, doi:10.1002/wsb.1016.

24. Jumeau, J.; Petrod, L.; Rich, Y. (). A comparison of camera trap and permanent recording video camera efficiency in wildlife underpasses. *Ecol. Evol. 2017*, 7, 7399–7407.

25. Yoo, S.; Melde, K.L. VHF collar integrated antenna for ground link of GPS based location system. *IEEE Trans. Antennas Propag. 2013*, 61, 26–32.

26. Taylor, P.; Crewe, T.; Mackenzie, S.; Lepage, D.; Aubry, Y.; Crysler, Z.; Finney, G.; Francis, C.; Guglielmo, C.; Hamilton, D.; et al. The Motus wildlife tracking system: A collaborative research network to enhance the understanding of wildlife movement. *Avian Conserv. Ecol. 2017*, 12, 1.

27. Levin, I.I.; Zonana, D.M.; Burt, J.M.; Safran, R.J. Performance of Encounternet Tags: Field Tests of Miniaturized Proximity Loggers for Use on Small Birds. *PLoS ONE 2015*, 10, 9.

28. Kays, R.; Crofoot, M.C.; Jetz, W.; Wikelski, M. Terrestrial animal tracking as an eye on life and planet. *Science 2015*, 348, 6240.

29. Wall, J.; Wittemyer, G.; Klinkenberg, B.; Douglas-Hamilton, I. Novel opportunities for wildlife conservation and research with real-time monitoring. *Ecolgl. Appl. 2014*, 24, 593–601.

30. Huang, J.H.; Chen, Y.Y.; Huang, Y.T.; Lin, P.Y.; Chen, Y.C.; Lin, Y.F.; Yen, S.C.; Huang, P.; Chen, L.J. Rapid prototyping for wildlife and ecological monitoring. *IEEE Syst. J. 2010*, 4, 198–209.

31. Allan, B.M.; Arnould, J.P.; Martin, J.K.; Ritchie, E.G. A cost-effective and informative method of GPS tracking wildlife. *Wildl. Res. 2013*, 40, 345–348.

32. Hertel, A.G.; Leclerc, M.; Warren, D.; Pelletier, F.; Zedrosser, A.; Mueller, T. Don’t poke the bear: Using tracking data to quantify behavioural syndromes in elusive wildlife. *Anim. Behav. 2019*, 147, 91–104.

33. Vectronic Aerospace GmbH. Quality GPS Collars for Demanding Wildlife Studies. 2020. Available online: https://www.vectronic-aerospace.com/wildlife-monitoring/gps-collars/ (accessed on 1 May 2020).

34. Weiser, A.W.; Orchan, Y.; Nathan, R.; Charter, M.; Weiss, A.J.; Toledo, S. Characterizing the Accuracy of a Self-Synchronized Reverse-GPS Wildlife Localization System. In Proceedings of the 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Vienna, Austria, 12–14 April 2016; pp. 1–12.

35. Lu, C.; Tsui, C.; Ki, W. Vibration Energy Scavenging System With Maximum Power Tracking for Micropower Applications. *IEEE Trans. Very Large Scale Integr. Syst. 2011*, 19, 2109–2119.

36. Huang, Q.; Lu, C.; Shaurette, M.; Cox, R. Environmental thermal energy scavenging powered wireless sensor network for building monitoring. In Proceedings of the 28th International Symposium on Automation and Robotics in Construction, Seoul, Korea, 29 June–2 July 2011; pp. 1376–1380.

37. Farine, D.R.; Peshkin, A.; Couzin, I.D.; Berger-Wolf, T.Y.; Crofoot, M.C. Individual variation in local interaction rules can explain emergent patterns of spatial organization in wild baboons. *Proc. R. Soc. B 2017*, 284, 20162243.
42. Joshi, A.; VishnuKanth, I.N.; Samdaria, N.; Bagla, S.; Ranjan, P. GPS-less animal tracking system. In Proceedings of the 2008 Fourth International Conference on Wireless Communication and Sensor Networks, Allahabad, India, 27–29 December 2008; pp. 120–125.

43. Duda, N.; Nowak, T.; Hartmann, M.; Schadhauser, M.; Cassens, B.; Wägemann, P.; Mayer, F. BATS: Adaptive ultra low power sensor network for animal tracking. Sensors 2018, 18, 3343.

44. Ayele, E.D.; Meratnia, N.; Havinga, P.J.M. Towards a new opportunistic IoT network architecture for wildlife monitoring system. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018; pp. 1–5.

45. Troph; Jens; Plets, D.; Martens, L.; Joseph, W. Bluetooth low energy based location tracking for livestock monitoring. In Proceedings of the 8th European Conference on Precision Livestock Farming, Nantes, France, 12–14 September 2017; pp. 469–475.

46. Hartmann, M.; Nowak, T.; Patino-Studencki, L.; Robert, J.; Heuberger, A.; Thielecke, J. A low-cost rssi-based localization system for wildlife tracking. In Proceedings of the 2015 Radio and Antenna Days of the Indian Ocean (RADIO), Belle Mare, Mauritius, 21–24 September 2015.

47. Strandburg-Peshkin, A.; Farine, D.R.; Couzin, I.D.; Crofoot, M.C. Shared decision-making drives collective movement in wild baboons. Science 2015, 348, 1358–1361.

48. Barsocchi, P.; Lenzì, S.; Chessa, S.; Giunta, G. A novel approach to indoor RSSI localization by automatic calibration of the Wireless propagation model. In Proceedings of the VTC Spring 2009—IEEE 69th Vehicular Technology Conference, Barcelona, Spain, 26–29 April 2009.

49. Dong, Q.; Dargie, W. Evaluation of the reliability of RSSI for indoor localization. In Proceedings of the 2012 International Conference on Wireless Communications in Underground and Confined Areas, Clermont Ferrand, France, 28–30 August 2012.

50. Schmalenstroeer, J.; Haeb-Umbach, R. Investigations into Bluetooth Low Energy localization precision limits. In Proceedings of the 2016 24th European Signal Processing Conference (EUSIPCO), Budapest, Hungary, 29 August–2 September 2016; pp. 652–656.

51. Cypress Semicond. Corp., San Jose, CA, USA. PSoC 4: 4200 BLE Family Datasheet. 2018. Available online: https://www.cypress.com/file/416486/download (accessed on 3 November 2018).

52. Vera-Amaro, R.; Angeles, M.E.R.; Luviano-Juarez, A. Design and analysis of wireless sensor networks for animal tracking in large monitoring polar regions using phase-type distributions and single sensor model. IEEE Access 2019, 7, 45911–45929.

53. LabJack Corp., Lakewood, CO, USA. T-Series Datasheet. 2015. Available online: https://labjack.com/support/datasheets/t-series (accessed on 12 December 2019).

54. Kingsbury, M.; Sadasivam, K. PSoC 4 Low-Power Modes and Power Reduction Techniques. 2018. Available online: https://www.cypress.com/file/121271/download (accessed on 23 May 2019).

55. Agarwal, U.; Patel, K.; Vikram; Reddy, P.; Santosh, S. Designing for Low Power and Estimating Battery Life for BLE Applications. 2016. Available online: https://www.cypress.com/file/140991/download (accessed on 10 January 2019).

56. Kemeny, J.G.; Snell, J.L.; Knapp, A.W. Properties of Markov Chains. In Denumerable Markov Chains Graduate Texts in Math; Springer: New York, NY, USA, 1976; pp. 79–105.

57. Hastings, W.K. Monte Carlo sampling methods using Markov chains and their applications. Biometrika 1970, 57, 97–109.

58. MATLAB; Version 9.6.0.1072779 (R2019a); The MathWorks Inc.: 2019. Available online: https://www.mathworks.com/products/matlab.html (accessed on 1 May 2020).