A LOW COST AUTOMATED LIVESTOCK TRACKING SYSTEM

A Thesis

by

JASON GRUBB

Submitted to the Graduate School

Appalachian State University

In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2010

Department of Computer Science

A LOlW COST AUTOMATED LIVESTOCK TRACKING SYSTEM

A Thesis

By

JASON GRUBB

August 2010

APPROVED BY

_____

James B. Fenwick Jr.

Chairperson, Thesis Committee

_____

E. Frank Barry

Member, Thesis Committee

_____

James T. Wilkes

Member, Thesis Committee

Chairperson, Department of Computer Science

_____

Edelma D. Huntley

Dean, Research and Graduate Studies

ABSTRACT

A LOW COST AUTOMATED LIVESTOCK TRACKING SYSTEM (August 2010)

Jason Grubb, B.S., Appalachian State University

M.S., Appalachian State University

Chairperson: Dr. James B. Fenwick Jr.

Successful farming has always required intense manual labor and acute management skills. The technological advancements of two agricultural revolutions reduced the quantity of manual labor required but human direction is still necessary (Rasmussen, 1962). In the last recent years, the level of automation in farming processes has increased significantly. A main component of these new strategies is livestock monitoring information. Animal tracking provides valuable information including recent location, movement and feeding patterns, and land usage. The collection and storage of this information as well as actions based upon the information are becoming more automated. Technologies such as global positioning system (GPS), radio frequency identification (RFID), wireless networking, and mobile computing systems are being utilized to target specific needs of farmers (Barbari, Conti, & Simonini, 2010).

This research will develop and evaluate a prototype data acquisition system for tracking livestock. Open source, freely distributed technologies will be utilized whenever possible in an effort to reduce cost. This study will evaluate the performance and cost of this livestock management system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Livestock Tracking

The identification of livestock in the United States began in the late 1800s and was used as a way to show ownership and deter theft (USDA, 2010a). It was difficult for a thief to sell livestock that had been branded, since it could be traced back to its rightful owner. This identification has traditionally been achieved via hot iron branding, ear notches, paint marks, and even tattoos. The same marks were typically made to an entire herd, which makes tracking a single animal impossible. Tracking different herds of animals meant a manual identification of the symbols, which led to complications when markings from one herd were indistinguishable from another. The transfer of the ownership of animals presented complications since cattle had to be rebranded in order to be identified with the new herd. Some more modern techniques, such as numbered ear tags provide a quicker, easier, and more humane method of identification. However, they frequently have many of the same drawbacks as the older methods.

The demand for livestock tracking mechanisms has increased substantially in recent years. This demand is being fueled by a number of factors including disease concerns (control, eradication, surveillance, monitoring), regionalization, global trade, livestock production efficiency, consumer concerns over food safety, and emergency management programs (USDA, 2010a). The increasing public awareness of the advantages of livestock tracking has also pressured governments to get involved. In the

U.S., incidents such as domestic reports of mad cow disease have prompted the U.S. Department of Agriculture (USDA) to create a nationwide animal tracking system in 2004 (Johnston, 2003).

This tracking system, dubbed NAIS, was initially established as a voluntary system to help protect against the spread of animal diseases. The NAIS proposed that every farm register with the USDA and provide location and contact information. The next step in NAIS implementation is the tracking of animals and storage of these records in local and state databases (USDA, 2007). Through this national database structure, animals can be tracked throughout their lifecycle. In the event that a particular animal was found to carry a disease such as mad cow, the origin of the animal and every transfer along the way could be found. Any potentially contaminated animal could then be identified.

Irrespective of the advantages that the NAIS had to offer, it had strong opposition from a substantial population of livestock farmers. Larger farming facilities have been more receptive of the proposed regulations since many of them have portions of the requirements already in place. They are also able to absorb the higher prices more easily since implementation costs can be amortized over a larger herd (Jeffries, 2006). Many small farms oppose the NAIS as they foresee it moving from a voluntary to a required system. The added costs incurred by the small farmer would have to be passed on to the local meat markets and consumers, causing many small farms to go out of business (Jeffries, 2006). Many farmers were also concerned about confidentiality and privacy with this national database. Due to these concerns, the USDA announced on February 5, 2010 that it would cancel the NAIS and revise its policy on animal identification and

offer new methods for tracking livestock (APHIS, 2010). While the revised policy is not complete at the time of this research, the goals are to develop a broad set of criteria and leave it up to the states to determine methods of implementation. In addition, the USDA is only targeting animals destined for interstate commerce, and is encouraging the use of lower cost technology with the new policies and procedures (USDA, 2010a). This "open" policy allows a wide variety of possible implementations.

1.2 Automated Systems

As herd sizes grow and identify requirements increase, it becomes more difficult for farmers to remember and to record the large amount of data accumulated for each animal they oversee (Rossing, 1999). Through the use of automated systems, the collection and storage of tracking data can be completed with little or no human intervention. The automated identification and tracking of individual livestock enables herds to be managed more effectively and efficiently, giving even small farmers additional incentives for implementing a tracking system beyond becoming NAIS compliant.

The most significant identification advances have come via electronic tracking methods such as radio frequency identification (RFID). RFID systems use a tag with a unique serial number and an antenna. A reader device accesses the tag information via the antenna. RFID ear tags are just like standard livestock ear tags except they contain the proper circuitry to be read by an RFID antenna. Ear tags are the most common type for use with larger livestock. GPS systems have also been used as an electronic tracking method. This system places GPS receivers on every animal so that their exact location can be found. The GPS collars required for this system can cost up to $4500 each,

making this solution much less common (Farren, 2008). Barcodes placed on existing ear tags enable electronic identification without having to re-tag an animal. Reading these tags with a handheld reader requires close proximity and a direct line of sight to the barcode. Like GPS, this method has not gained broad acceptance of RFID tracking.

Farm management systems using the new technologies allow livestock inventories to be electronically monitored at all times. After the initial setup of joining a particular animal with a unique ID, the time, date, and location are recorded every time the animal is within range of an RFID reader. These readers can be placed at key locations throughout the farm, such as feeding stations, barns, choke points, gates, and other custom locations as necessary for a particular farm's need. In the case of GPS solutions, even more accurate location information may be possible. Knowing the most common locations for animals, as well as their travel patterns, allows the use of fields and shelter to be optimized. The frequency of location changes for an animal, as well as their rate of recurrence at feeding stations, can also indicate health issues (RFID, 2010).

1.3 Thesis Research Scope

The lack of technology standardization by the federal government and the push for cost conscious animal identification solutions opens the door for solutions like this research provides. This thesis proposes that a low-cost, automated livestock tracking system built on open-source software is possible. Reducing automation costs makes such a system a viable alternative for smaller budget farms, minimizing one of the major issues that farmers had with the NAIS. The proposed system, LCTracker, will integrate the automated tracking virtues of an RFID based electronic identification system with a web based management system. Utilizing freely available software technologies and

providing the system software open-source allows for a cost effective, yet expandable solution. The future work section of this research will explore the expansion of this system beyond the scope of its initial implementation.

CHAPTER 2: RELATED WORK

2.1 Commercial Systems

There are many commercial livestock management systems available today. The features of these systems vary from manual-entry systems to those designed for complete business management. These packages were explored and evaluated for their features, cost of implementation, and ability to integrate with electronic identification solutions.

2.1.1 Herd-Pro

Herd-Pro offers livestock management software called StocKeeper (Herd-Pro, 2008). This product provides a Microsoft Windows main user interface and can be expanded to include a Palm OS mobile solution. The main interface has the ability to set warning alerts, define custom fields and reports, and has multi-language support. StocKeeper works with Destron RFID tags and readers by Digital Angel. This type of RFID equipment operates in the low frequency band, and therefore provides a maximum read distance of around one meter. This does not pose a problem for wand type readers, but to use automated panel readers the livestock must be placed in a chute or other type of confinement. A Microsoft Access database is used for persistent storage, and is accessible either directly or from within the application. The database is stored locally which restricts access to the data and requires a backup solution be provided by the farmer.

The Herd-Pro software does not have any setup fees and can be purchased outright or leased from the company. Two versions are offered, standard and professional. The standard product has a maximum head count of 300 and provides only limited customization. The professional version has an unlimited head count and offers custom fields and inventory analysis not available in the standard edition. The leasing program is a lease-to-own option, finalizing after the fifth year. If the contract is broken before the fifth year, licenses are voided. As shown in Table 2.1, support is also offered on a yearly basis. An active support contract offers the end user technical support and product patches and upgrades (Herd-Pro, 2008).

Table 2.1

StocKeeper purchase and lease price data

	Purchase	**Lease**	**Support**
Standard	$495	$225/yr.	$125/yr.
Professional	$995	$430/yr.	$250/yr.

Herd-Pro has an informative website that details the features of the system and the pricing options available. It gives a simple explanation of RFID identification, but does not elaborate on its use within StocKeeper. Email inquiries were responsive, answering all questions within 24 hours. Phone support is available, but the number is not toll free.

2.1.2 Cow Sense

Midwest MicroSystems offers management tools targeted to the beef cattle producer (Midwest, 2008). This software offers data entry and reporting tools similar to the other commercial systems that were evaluated. Cow Sense also offers analysis on

performance, breeding, and economic contributions down to the individual animal. It can also be customized by creating user defined fields and reports. This product is marketed as a base product for management, with add-on modules for specific tasks. These modules offer functionality for marketing, animal transfer, electronic data transfer, as well as an interface that works with the PocketPC OS of Microsoft Windows mobile computing systems. Cow Sense offers the Nomad handheld computer for $1354. The RFID vendors supported by Cow Sense are Allflex and Destron. This is the same type of RFID equipment supported by the Herd-Pro product. It retains the same features and limitations such as one meter maximum read distances and wider use of wand readers vs. automated panel readers. This Microsoft Windows based software uses Microsoft Access for its database, which is only accessible through the Cow Sense software.

The pricing for Cow Sense is the most complicated of the evaluated systems. The base product is available in five versions, with many add-ons available. The pricing scheme runs from a limit of 75 animals to unlimited animals with unlimited users. Table 2.2 shows pricing for the main product, which has a feature set similar to the other evaluated systems. Telephone support is included for 30 minutes over 30 days with purchase. An online version is also presented, which moves the interface and database off-site. This requires internet access on the computer being used. This version offers very basic functionality that is far below what is offered in the standard product (Midwest, 2008).

Table 2.2

Cow Sense purchase price data

	CS75	Unlimited	Commercial	Purebred	Multiuser	Online	Support
Price	$149	$265	$445	$535	$1995	$4.95/mo.	$1/min.

Midwest MicroSystems has a website defining all of the Cow Sense modules and pricing information. The support telephone number is not toll free, and is only available from 8-5 CST. There is voicemail with the option for a callback. Email response, however, was quick and answered technical details thoroughly.

2.1.3 TrackLivestock.net

The simplest commercial management software evaluated was the TrackLivestock.net system (Grow, 2006). This online only solution has fixed fields for predefined types of animals, including cattle, goats, hogs, and sheep. There are no custom fields or reports that can be created. TrackLivestock.net also doesn't have the ability to connect to an electronic ID technology. However, an Internet connection is all that is required and all data is hosted remotely and backed up nightly with no interaction from the end user. Data can be exported from the system in Microsoft Excel format for local use. While this program does not have the customization and automation features of the competing products, it has one main asset going for it. This software is free to set up and free to use, regardless of the number of animals tracked. Support is only available via email, with no telephone number provided (Grow, 2006).

2.1.4 CattleMax CS

CattleMax CS livestock management software has many of the same features as the other products evaluated, with custom reports and advanced reporting and analysis (Cattlesoft, 2010). What separates CattleMax CS from its peers is its intuitive graphical user interface. This slick icon-driven interface is the most modern interface reviewed and is similar in functionality to the familiar Microsoft Office suite of programs. The RFID options for CattleMax are Allflex and Destron, like many of the other systems evaluated. Allflex and Destron use the low frequency readers and tags that restrict read range to about one meter. CattleMax CS requires Microsoft Windows XP, Vista or System 7 and at least 1024 x 768 resolution. Software acquisition is a one time purchase and is based on the number of animals tracked and commercial or registered use. This is detailed in Table 2.3. Registered use adds genetic performance tracking and reporting. Support is free and is available via toll free number or online (Cattlesoft, 2010).

Table 2.3

CattleMax CS purchase price data

	< 50 Animals	**Unlimited Animals**	**Online Backup**
Commercial	$125	$245	$60/yr.
Registered	$295	$495	$60/yr.

CattleMax offers a well designed and professional looking website offering all information given above. When attempting to call the toll free number, however, it always went to voicemail requesting a callback number. This occurred even during listed operating hours. There was also no email response as of this writing. As the most

complete product evaluated, it was surprising that there was no response to email inquiries.

2.1.5 RFID Hardware

The common RFID equipment manufacturers among the evaluated software vendors were Destron and Allflex. These vendors offer RFID readers and tags in the low frequency radio band that is common for the electronic identification (EID) of cattle. Both companies produce similar products at similar pricing. Cost data for these systems was accumulated from vendors such as QC Supply (QC Supply, 2009), EarTagCentral (DHIA, 2008), and CattleStore (CattleStore.com, 2008). Table 2.4 shows average pricing for Destron and Allflex RFID equipment. Regular tags are standard size, extended range tags. These tags can be read at distances up to four feet. The combo tags house an RFID tag as well as display a visual lot number. This type of tag is useful for quick human identification. Wand type readers require a human to move the wand over the tag on an animal. The readings can be stored in the reader or to a connected computer via an RS-232 interface. The Panel readers consist of a combined reader and antenna combo, such as the Allflex, or separate items, like the Destron.

Table 2.4

RFID Pricing

	Regular Tag	**Combo Tag**	**Wand Reader**	**Panel Reader**
Allflex	$2.45	$3.35	$400	$2,700
Destron	$2.00	$3.20	$600	$3,800

2.2 Academic Research on Systems

Academic research on RFID based animal identification systems has expanded in recent years. Studies have been done on management systems and implementations such as this one, as well as on the economic viability of such processes. The findings from these works demonstrate the gains, both economic and social, for such systems. They also present the economic hurdles that slow the transition to RFID identification systems as a replacement for current methods. The findings from the research outlined in this section substantiate the need for a low cost RFID animal management solution.

2.2.1 FARMA

FARMA is a research project and implementation of an RFID based animal identification and farm management system. It explores the use of low frequency RFID tags for animal identification, databases, and networking technologies to develop an integrated framework for animal identification and monitoring (Voulodimos, Patrikakis, Sideridis, Ntafis, & Xylouri, 2010). This research is one of the first that not only explores the use of RFID for identification and tracking of animals, but is a complete system built around this information. The FARMA platform consists of a central database that contains information about all farms in the system, including their owners, location, and RFID numbers used. Each farm then has its own local database that only stores information about the animals on that farm. The RFID implementation incorporates an animal's tag, a portable RFID reader, and a mobile electronic device that can interface with the reader and store the gathered information.

FARMA includes, in its user interface, menus for both the farmer and veterinarian. Appropriate options are presented to each type of user. For instance,

veterinarians can manage examinations, vaccinations, diseases, and animal death, while farmers can control new births, tagging, weighing, and sale categories (Voulodimos et al., 2010). This menu system updates the local database, which has an interface with the central database. Periodic updates can then be made to keep the two storage systems in sync.

The FARMA proposal and implementation includes a network system to integrate the disparate components of its structure. Both wired and wireless options are available. Portable devices can connect to the local database wirelessly through either 802.11 wireless networking or through mobile broadband connections (Voulodimos et al., 2010). When neither of these wireless options are available, the collected data are stored within the portable device and then transferred to the local database when a wired connection can be made.

The FARMA implementation was field tested on a small sheep and goat farm (Voulodimos et al., 2010). Data were collected, stored, and managed by both the farms owner and veterinarian. The system presented no serious problems and was generally well received by animal and operator alike. Multiple tagging options were explored, with ear tagging determined to be the most viable option. The wireless networking technologies that were employed provided good results, but were determined to be potentially cost prohibitive or too complex for average farmer implementation. Direct connection of the mobile device used to temporarily store the gathered data was determined to be the best default solution for rural farms. The limitations of the international standards for RFID of animals were also realized during the field trial. The number of devices compliant to these regulations is small and they are limited when

compared to emerging trends in the market. Future enhancement possibilities that were explored include the use of Global Positioning System (GPS) modules in the system to provide precise location information.

The FARMA system incorporates many of the same features and ideas as the LCTracker system. The fundamental designs of having remote locations networked to a central database system are equivalent, but the means are different. Where FARMA uses the 802.11 wireless protocol, LCTracker uses 802.15 radio standard for communication. LCTracker also increases automation opportunities by utilizing ultra-high frequency RFID panels that can read animal tags at a distance. This mitigates the labor required in identifying the animals. The LCTracker system reduces costs by not only providing the software free of charge, but by the use of open source platforms for development. Where FARMA uses costly Microsoft development, hosting, and storage subsystems, LCTracker uses freely available products from Apache, MySQL, and PHP.

2.2.2 RFID Animal Identification Economics

Gary Halverson studies actual cost vs. gain for producers utilizing RFID animal identification methods in his dissertation completed at Utah State University (Halverson, 2008). Halverson first set out to determine the costs incurred by producers utilizing RFID identification techniques within their herd management styles. To do this, he obtained costs from three sources: a survey of producers using RFID, databases and animal ID software providers, and third-party facilitators (Halverson, 2008). The surveys requested the type of RFID equipment in use, size of operation, other forms of animal ID, and additional time activities required with the use of RFID.

Halverson determined that a one size fits all cost-per-animal calculation is impossible since animal identification is an ongoing activity along with all other farm activities (2008). Management techniques caused the largest variance in cost-per-animal calculations, with production size and producer involvement causing smaller, but potentially significant, variances. If producers were already restraining animals for management tagging, branding, or vaccinations, then there is no significant labor cost added by RFID tagging the animal. If, however, the previous management style of the producer did not involve restraining the animal before the time necessary to apply an RFID tag, then the labor for this operation was found to be the most significant factor in RFID implementation costs (Halverson, 2008).

Additional implementation costs beyond tagging labor include the physical tags, database and software subscriptions, labor for entering data, and RFID readers. It was found that the visual management tags that were commonly in use averaged $1.00 each and RFID cost an average of $2.00 each (Halverson, 2008). After RFID tags are purchased, they have to be entered into a database and associated with a particular animal. The labor for this, along with possible charges by the database provider, were found to be $0.40 - $-.50 per animal. The database providers surveyed had annual subscription costs ranging from $0 to $1750, however those with no annual costs had per-entry fees and some had both (Halverson). All totaled, the cost-per-animal calculations based on the gathered data ranged from $2.91 to $10.51 (Halverson, 2008).

The cost for RFID readers was not included in the calculations performed by Halverson (2008). Reader cost was seen as a significant barrier by many providers, so many chose not to implement them. This leaves the reading of the tags up to the unit

where the animals are being transitioned, such as another producer, feed lot, or slaughterhouse. The price of an RFID reader varies based on type and technology. Wand type readers are cheaper than fixed panel readers, but require additional labor since the wand has to be placed in close proximity to the tag on each animal. Over time, the additional labor cost will outweigh the price differential between wand and panel readers (NAIS, 2009). Panel readers are placed at fixed locations and read tags when they are within range. These readers vary in reading range from a few inches to many feet. For close proximity panel readers the animals must be placed in a head catch or run through a chute in order to read the tags. Readers that have increased range do not require this and therefore reduce the risk of injury and animal stress, which can lead to weight loss (Halverson, 2008).

To determine price premiums for RFID tagged cattle, Halverson used a sale dataset that included over 29,000 lots totally over 3.25 million head of cattle. RFID tagging was evaluated as a characteristic of the lot, similar to vaccination status, natural vs. hormone, etc. To ascertain if RFID was a value-added characteristic, the marginal physical product (MPP) multiplied by the unit price of the output (P_o) is compared to the cost of the input. A producer may pay for individual characteristics that increase MPP or P_o if the value of marginal product is greater than the cost of input (Halverson, 2008) The price premium deduced from the dataset for RFID tagged cattle was $1.50 per 100 pounds. At an average sale weight of 500 lbs., the producer would see a net gain as long as the total cost for RFID implementation was less than $7.50 per head. Halverson's research showed that, depending on current management style, many producers were able to see net profit gains from implementing RFID animal identification.

Halverson's dissertation also addressed the lack of standardization in RFID based animal identification. In the US, the USDA is recommending the individual tracking of livestock, but is remaining technology neutral (USDA, 2010b). This has resulted in many different companies trying to make a profit by becoming a technology provider for producers with no consistencies between them (Halverson, 2008). With standardization, the use of RFID in animal identification will increase, and economies of scale will act to lower equipment costs. The increased use of RFID will also begin to replace current methods, thus lowering costs further (Halverson, 2008).

Halverson's calculations show that farms can see positive economic gains for using RFID tracking systems, provided costs can be kept down. Thus, a need for a low-cost system is truly present. LCTracker uses freely available and open source software with generic hardware systems to reduce implementation costs. Software licensing costs are eliminated completely, as LCTracker is offered to farms at no charge. By providing RFID tracking, as well as increasing automation, LCTracker offers farms the opportunity for financial gain.

CHAPTER 3: TECHNOLOGY OVERVIEW

3.1 RFID Technology

The fundamental technology behind this research is radio frequency identification, or RFID. RFID is a wireless technology that uses radio frequency induced electromagnetic signals to transfer a serial number from a tag to a reader (Domdouzis, Kumar, & Anumba, 2007). The basic RFID system includes an antenna, a transceiver, and a transponder. This system is typically connected to a computer or integrated device that logs or takes action based on the reading.

The antenna portion of an RFID implementation consists of a coil that transmits radio signals in order to establish communication between the transceiver and transponder (Domdouzis et al., 2007). The transceiver can then read and write data to the transponder, also known as the tag. Antennas and transceivers are often packaged together and are simply referred to as readers. RFID antennas operate at different frequencies and must have appropriate tags and transceivers that operate in the same frequency range. The most common commercially available frequency ranges are 125 – 135 kHz (low frequency), 13.56 MHz (high frequency), and 433 – 956 MHz (ultra-high frequency) (Domdouzis et al., 2007).

RFID tags consist of a radio antenna and a microchip. Passive tags receive their power through the electromagnetic radio frequencies generated by the reader and antenna. Since there is no other power source for passive tags, they can only operate

when in range of an RFID reader. Once powered, the tag adds its serial number to the signal that is sent to the reader. This process is referred to as passive backscatter (U.S., 2005). The reader can then extract the serial number from the original signal. Active tags have a built-in power source so that they do not have to be powered from the reader. Instead of being idle until within range of a reader, active tags send out a beacon type of signal. This allows the tag to be read from greater distances, as well as send additional data from attached sensor devices (U.S. Department of Homeland Security Smart Border Alliance [USDHS], 2005). While these are definite advantages, the price of this type of tag makes them cost-prohibitive for most applications.

RFID tags for animal tracking are manufactured in three main types. Boluses are capsules that are ingested by an animal with multiple stomachs and remain in one of the first two (Voulodimos et al., 2010). Injectable tags are small glass or plastic tubes containing an RFID tag that are placed just under the skin of an animal. RFID ear tags are just like standard livestock ear tags except they contain an RFID chip and antenna.

RFID transceivers, or readers, generate the radio signals sent by the attached antenna. They also receive responses from tags within range. The receiver is able to decode the serial number and additional data sent by the tag and return it to a digital format to be processed by the attached device. Receivers also have the functionality to handle the anti-collision algorithms that allow them to read more than one tag at a time. The attached device, be it embedded or a host computer, typically controls operation of RFID readers.

Each of the frequency ranges commonly used in RFID systems have different attributes and uses. The low frequency (LF) band was used in early RFID

implementations and is still in use today for animal tagging, access cards, and other close proximity uses. The read range for these frequencies ranges from two inches to two feet. Readers and tags that operate at the high frequency (HF) range are typically used for smart cards and access control systems and operate within a range of three feet. Ultra-high frequency (UHF) RFID equipment has the advantage of longer read distances, with U.S. approved frequencies achieving distances up to 49 feet. UHF tags are commonly used for entry/exit systems such as parking garages (USDHS, 2005).

Our research explores the use of UHF readers and tags for use in animal identification. Traditionally, LF radio frequency band RFID equipment has been used for tracking and identifying livestock. The very short read range of LF tags inhibit the options available for tracking and automation when managing animal lots. Substantial labor is necessary when using wand readers to get within range of an LF tagged animal. It also causes undue stress on the animal to place it in a head catch or force them single file through a chute. By exploring the use of UHF RFID equipment, this research seeks to take advantage of the increased read distances to reduce labor and increase automation opportunities.

3.2 RFID Hardware

We have selected the DL910 reader from Daily RFID co., Limited (2009). This generation two UHF reader contains an integrated antenna in a sealed enclosure that offers weather protection. The specifications for this reader list maximum read range at 26 – 49 feet, which allows for fewer readers required to cover a given area. This reader can read multiple tags at a time and contains an RS-232 serial interface for communication with another device. This unit was also chosen because it comes with a

software development kit and sample code. This allows full control of the reader by an external device.

There are two main types of RFID readers. The DL910 reader is a panel type reader. This type of reader is mounted at a fixed location and reads any tags within range. Panel readers can typically be connected to multiple antennas for additional coverage. Wand readers are portable units with integrated antennas that are placed in close proximity to the single tag to be read. These readers require human control and can connect to mobile computing devices or store the values for later retrieval.

The RFID tags that were selected operate at the same UHF frequency as the DL910 reader. Three type of tags were used for different attachment options. Wristband tags can be attached and removed from necks or limbs of an animal for temporary identification. Label tags can be affixed to existing tags, allowing quicker attachment. Hard plastic tags are like standard ear tags and are made of sturdier construction.

Many other types of RFID tags exist. The retail sector attempts to thwart shoplifting by attaching RFID tags to their merchandise. These tags are deactivated at the time of purchase. RFID panels are placed at store exits and sound an alarm when a tag passes through them that is still active. Hospitals are also making use of RFID technology by attaching ankle bands to newborn babies (Hospital, 2005). The unique serial number of the tag is linked to the mother's wristband to prevent a mix up and readers at hospital exits detect the unauthorized removal of the infant.

3.3 Sun SPOTs

A Sun Microsystems research project developed the Small Programmable Object Technology (SPOT) device as a platform for diverse wireless device development (Sun,

2010). Sun sells these in development kits with 2 free-range SPOTs and a base station SPOT. The free-range SPOTs are mobile devices, while the base station connects to a computer and establishes a wireless connection with other SPOTs. These small devices contain ARM processors with on-board RAM and ROM storage. This gives them the potential to not only collect and store data, but to process that data as well. The wireless functionality is provided by an 802.15.4 radio and integrated antenna, which provides approximately 328 feet of range. Because SPOT devices contain a microprocessor, they provide for transmission hopping. Thus, data from a distant SPOT device can be transferred along a series of in-range SPOTs to the base station. Each free-range SPOT device includes a rechargeable 3.7V lithium-ion battery. Recharging is accomplished by connecting the mini USB port to a powered host. Also stacked within the same enclosure is a sensor board that is shipped in each free-range SPOT. This sensor board demonstrates sensor integration possibilities and includes a 3-axis accelerometer, temperature and light sensors, eight tri-color LED lights, six analog inputs, two momentary switches, five general purpose I/O pins, and four high current output pins. The high current output pins can drive external servos, motors, and control units. The I/O pins can provide a serial interface between the SPOT and another device (Sun, 2010).

The entire Sun Labs research project is available open source. The hardware schematics and everything necessary to recreate the units are provided at no charge. The virtual machine, as well as all demo and API software is also available under the GNU General Public License. This gives the option of buying prebuilt units for rapid prototyping and development, or using the open source hardware information to build

production units specific to an application. Significant cost savings can be achieved by using only the portions of the prebuilt units that are necessary for the particular use.

Sun SPOTs employ the Java programming language exclusively. The software loaded to the device completely controls the device, thus providing the developer with a high degree of flexibility. SPOTs run the Squawk virtual machine (VM), a Java VM written in Java that was developed to support micro-embedded development (Oracle, 2010b). Java programs written for the devices are compiled on a computer and then deployed to the device via the Ant (Apache HTTP Server Project, 2010) build tool. This can be done at the command line or through the NetBeans Java development environment. NetBeans Plugins are available to assist in SPOT development, and provide programming templates and automatic deployment functionality.

We will employ Sun SPOTs to control and gather data from the RFID readers in the field. The SPOTs will then transmit these data, possibly via transmission hops across other SPOTs, to the base station unit attached to a PC. This remote data collection allows for an increased number of automated reading stations without an increase in computers or handheld reading devices, since the reader SPOT handles the data acquisition from the reader as well. The frequency of RFID reader tag detection, as well as the frequency of data transmission, is controlled by each individual SPOT and can be set independently to allow for different use case scenarios.

3.4 MySQL

MySQL is a relational database management system (RDBMS) that is a popular storage solution for web application development. It provides the power and dependability of many commercial RDBMS solutions, but is free to use under the GPL

open source license agreement. MySQL is a key component as part of the AMP (Apache, MySQL, PHP) software development stack and is available for many operating system environments such as Microsoft Windows, Linux, and Mac. (Oracle, 2010a)

The low cost automated livestock tracking system outlined in this research will utilize the MySQL software for data storage. The location, timestamp, and any other collected data about an individual animal is saved in this database. MySQL will also be used to store all user-configurable, interface, and system settings. All of this data will be delivered to the user interface that the farmer will use.

3.5 Apache

The Apache http server is an open source web server supported by the Apache Software Foundation (Apache HTTP Server Project, 2009). This freely available software is responsible for serving almost 55% of the worlds web pages (Netcraft, 2010.). Apache is available for multiple operating systems, including Microsoft Windows, Mac OS, and Linux. Combining the Apache http server with the Linux operating system creates a web hosting solution free of software licensing fees. The low cost automated livestock tracking system developed during is written in PHP and utilizes the Apache http server to host the system's user interface.

3.6 PHP

PHP is a web development scripting language that is embedded in HyperText Markup Language (HTML) for creating dynamic web content (The PHP Group, 2010). Utilizing the fundamental Internet language of HTML for displaying static content, developers can leverage PHP to create interactive web pages. Content can be pulled from other web sources or from databases. This connection to a database and its persistent

storage is what makes the PHP language useful for creating user interfaces. The display, interaction, and customization of the low cost automated livestock tracking system occurs in a user interface built with PHP.

CHAPTER 4: DESIGN

4.1 System Description

LCTracker is the prototype low cost automated livestock tracking system developed for this research as an alternative to current herd management options on the market. Programs such as NAIS and its forthcoming replacement are pushing for farmers to track their livestock. These programs are seen as an extra cost for the producer, with little benefit for the average farmer. LCTracker is designed to meet the needs of a national livestock tracking program, while also providing useful animal management information as a benefit to farmers. All movement, feeding, and farmer-entered information can be stored and linked to the unique ID for the animal. These data provide movement patterns, which farmers can use to identify over and under-used areas. The data also show feeding patterns that farmers can use to tailor their animal weight gain strategies.

The high level depiction of LCTracker in Figure 4.1 shows the main components and how they interact. This diagram illustrates one sample configuration for tracking cattle at a remote location. Field number one signifies a remote location where cattle are monitored. A good example of this is a feed station. Each cow is tagged with an RFID ear tag and is allowed to roam as they would normally. A UHF RFID reader is placed at one end of the feed station, facing toward the food supply. This allows the reader to pick

up all RFID tags within 49 feet of the reader. In doing so, cattle coming to the feed bin are monitored.

The RFID reader is connected to SPOT number one via an RS-232 serial cable. The SPOT acts as a control unit. Determining the rate at which the reader detects tags and stores all tag IDs that are read in the memory of the SPOT device. This data is then broadcast wirelessly to all SPOTs that are in range. In the scenario of Figure 4.1, SPOT 2 is within range of the first spot and receives the data being sent. SPOT 2 and SPOT 3 in the adjacent field are not connected to readers or computers and act as repeaters for the data transmission to and from field one. Within range of one another, SPOT 2 receives data from SPOT 1 and repeats this data to SPOT 3. This continues until the data is received by the base station, SPOT 4, which is connected to a computer inside the barn or home.

The computer in the barn contains the system database as well as the graphical user interface (GUI). The tag ID data received by SPOT 4 is stored in the database and available for view in the GUI. The result of one read in the example shown in Figure 4.1 would be the ID storage of the three cattle at field one's feed bin in the database on the computer in the barn. This all happens automatically, without any human intervention.

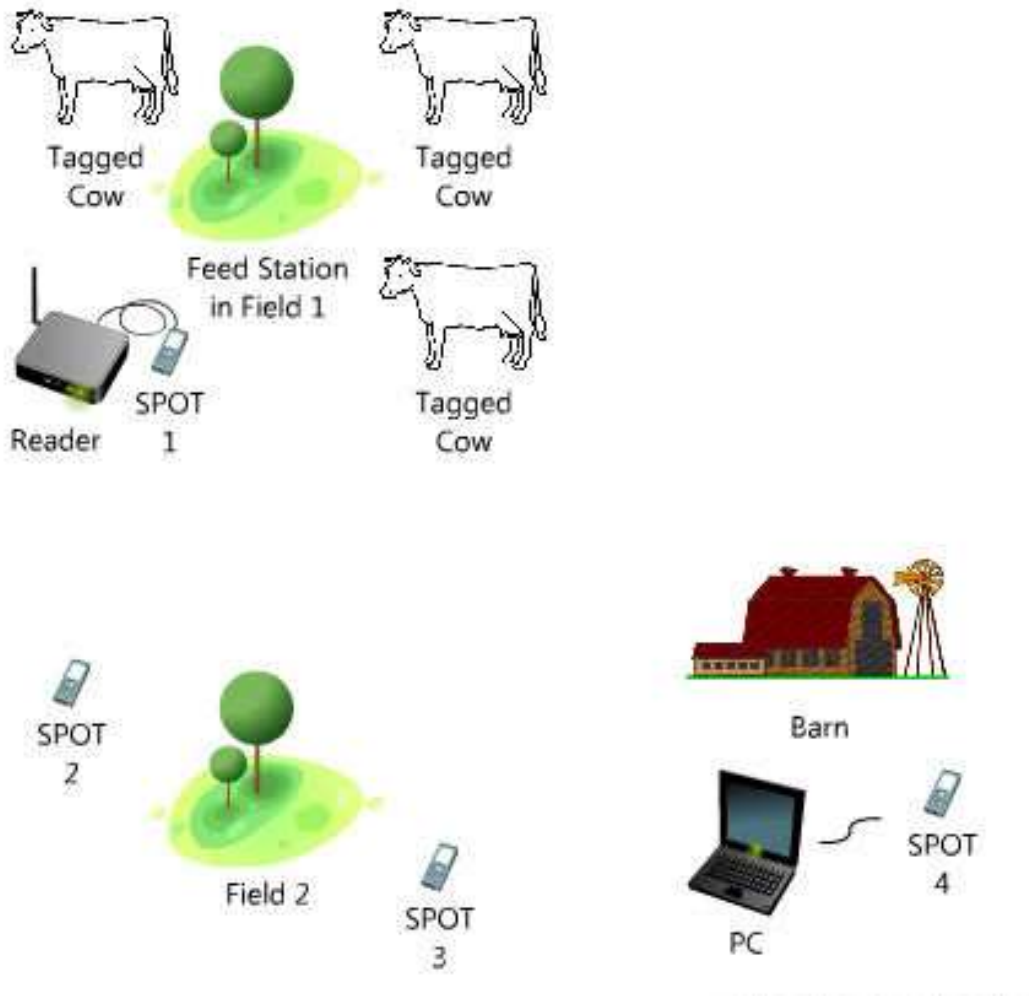

Figure 4.1. LCTracker example scenario.

4.2 LCTracker Setup

Initial system setup at a new location requires a minimum set of components to have a working product. There must be a computer at a location that has power and is shielded from the weather. This workstation must have an AMP software stack installed with running servers for Apache, MySQL, and PHP. Java must also be installed for the operation of the connected SPOT, and a USB port must be available to plug it into. All of the required software is available at no cost.

At least two SPOT devices must be also be present. One SPOT can act as the base station and is connected to the workstation. The other SPOT is a self-powered free-range SPOT that is connected to an RFID reader. In order for the system to operate with just two SPOTs, they must be within wireless range (approximately 328 feet) of each other. Care must also be taken for the SPOT connected to the reader so that it is protected from the weather. Minimally, one RFID reader is required, as is one RFID tag per animal. This reader must operate in the UHF frequency band and have an RS-232 connection for data transfer and control. A serial cable is necessary to connect the SPOT to the reader. Power for the reader must be provided at this location, either from an A/C outlet or via battery. As a part of this prototype LCTracker system, we have fashioned an inexpensive acrylic panel enclosure to provide weather protection.

In order to set up an LCTracker system, the installation steps in Figure 4.2 must be followed. These steps show the process for initial installation, as many of the components and steps are only required once. The system can be expanded from the base installation by repeating steps 6, 7, and 8 in Figure 4.2. Figure 4.3 gives the steps for installing a new reader.

Figure 4.2. Initial installation and setup.

1. Tag livestock with RFID tags.
2. Install RFID reader at desired location. This install includes mounting the reader and connecting it to a power source.
3. Install LCTracker software on reader SPOT.
4. Mount SPOT near reader and connect to it with a serial cable.
5. Place protective cover over SPOT if necessary.
6. Install LCTracker repeater software on free-range SPOTs.
7. Add repeater SPOTs every 100 yards or less, adding protective cover if necessary.
8. Install AMP stack, database, and interface to workstation computer.
9. Connect base station SPOT to computer.
10. Install LCTracker base station SPOT software.
11. Locate computer in shelter with A/C power available such that it is within 100 yards of the closest SPOT.
12. Start web and database servers, reader, and SPOT software.
13. Use GUI to set up run-time options and load areas, reader locations, and animal information.
14. Once these steps are complete, the system functions automatically. Whenever livestock come within range of the reader, the time and date of that location is stored. Running histories for areas and animals are available in real time via the system's interface.

Figure 4.3. New RFID reader installation and setup.

1. Install RFID reader at desired location. This install includes mounting the reader and connecting it to a power source.
2. Install software on reader SPOT.
3. Mount SPOT near reader and connect to it with a serial cable.
4. Place protective cover over SPOT if necessary.
5. Install repeater software on free-range SPOTs.
6. Add repeater SPOTs approximately every 100 yards to complete a connection to an existing SPOT.

4.3 Hardware Features

The hardware components that make up the LCTracker system were all selected with cost, automation, and features in mind. Current commercial systems utilize RFID hardware with capabilities that are now far below the capabilities provided by recent advancements in equipment. The RFID reader and tags utilized in LCTracker operate at UHF frequencies. This expands the maximum reading range from around three feet to 26

– 49 feet. The short read range of LF panel readers require animals to be restrained or run through a chute in order to read tags. While power can be attenuated in the UHF reader to reduce reading range if necessary, the extended range allows for the use of unattended panel readers in an open area. This increases convenience and automation opportunities, as well as reducing labor costs.

The evaluated commercial systems also use LF frequency animal tags. These tags are similar to the tags used in LCTracker, with the main difference being their operating frequency. To comply with NAIS, specific tags must be used. These tags have dedicated ranges of serial numbers and are only sold by select providers. LCTracker makes use of generic RFID tags available at a wide selection of outlets. NAIS compliance is provided by the LCTracker system software, which maps tag serial numbers to NAIS issued numbers. RFID tags can be ordered with specific serial numbers, so tags ordered must be unique to an LCTracker implementation. With increased competition in the sale of generic tags, cost savings can be seen by using them. Using generic tags also increases the diversity of sizes and styles available, increasing the chance that the farm manager can get the style they prefer.

The Sun SPOTs utilized in LCTracker allow remote RFID readers to communicate readings wirelessly to the base station computer. They operate autonomously via a rechargeable lithium-ion battery and can be custom constructed for specific needs. Reader SPOTs require full free-range SPOT capabilities, but repeater nodes simply relay wireless data to in-range devices. Repeater SPOTs can either serve other automation tasks or be built with only the components necessary for wireless communication to save cost. Sun SPOTs are multipurpose devices, supporting the

control of other systems beyond RFID readers. Thus, extensions could have SPOTs opening/closing gates, turning on/off water supplies, etc.

Each hardware component used in LCTracker was designed for modularity. For instance, the RFID reader can be replaced with another type of identification method while the rest of the system remains intact. SPOTs were chosen for interfacing and networking, but could be replaced by ZigBee or Arduino mobile devices that are functionally equivalent. Modular components allow an LCTracker implementation to be custom built for specific purposes while using the same system design.

4.4 Graphical User Interface

The GUI for the LCTracker system provides the end user with the ability to control read rates, as well as search for particular animal IDs in the system. It was designed as a dynamic website so that it can run locally on the base station or be hosted and accessed from any machine with Internet access. This interface provides the most recent tracking data for livestock, as well as the ability to search for particular livestock and view their location history. Information specific to each animal can be entered and tied to the RFID tag used on it. Farm information, RFID locations, and areas can all be defined within the GUI.

The GUI also provides a setup section. This section has elements such as RFID locations that are required at initial installation, as well as other settings useful to the farmer. The farmer can set read timings for each particular reader so that they are appropriate for the intended location. A reader inside a barn may not need to take readings as often as one at a gate, so those adjustments can be made in the interface and disseminated to the readers. Notifications can also be set so that the farmer is alerted

about specific situations. Examples of these notifications are when an animal hasn't been picked up by any of the readers in a set amount of time or when an animal hasn't been to the feed bin in a certain time period.

Built as a dynamic website, the user interface can be run locally on the base station PC or delivered by a website hosting company. Running the interface from the base station allows the system to function without an Internet connection, and is free. Backing up the database requires external data storage, and must either be done manually or programmed into the GUI. Website hosts typically charge a small fee and a connection to the Internet is required, but they typically provide support and automatic backup options as incentives.

CHAPTER 5: IMPLEMENTATION

5.1 LCTracker

The current implementation of LCTracker is a proof of concept prototype, showing the feasibility of a low cost automated livestock tracking system. The design described in the previous chapter was followed as closely as possible, while some concessions were made with consideration to cost, availability, and functionality. This prototype shows that the LCTracker design can be the basis for a viable cost effective solution.

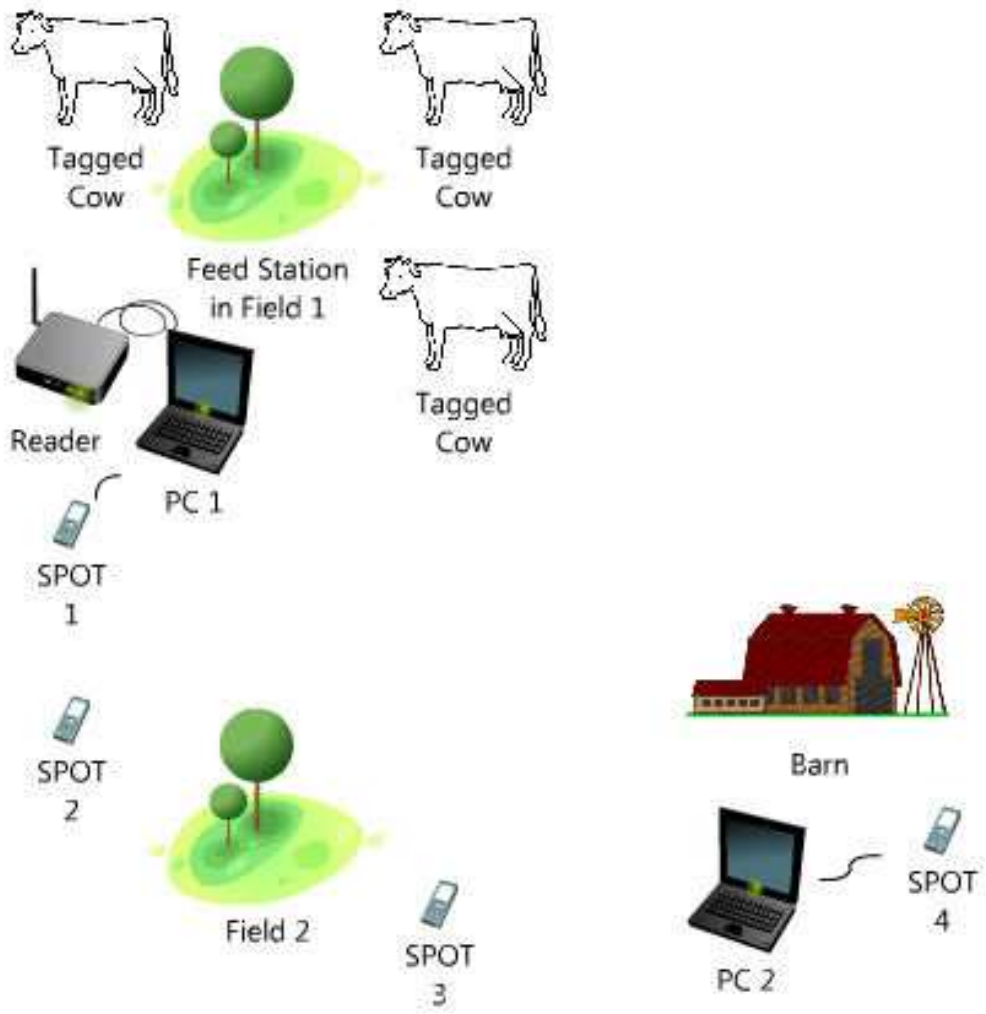

Figure 5.1. LCTracker Implementation Design.

Figure 5.1 shows a slight change made in the prototype. Functionally, this system still operates in the same way as the design of Figure 4.1. Tagged animals in field 1 are detected by the reader and stored there until their information is broadcast via SPOT 1. This data is then relayed to SPOT 2 and three and eventually to SPOT 4 that is connected to PC 2 in the barn. The only hardware difference in this design is the addition of PC 1 at the location of the RFID reader. This PC controls the reader, sets read timing, and collects the read information. SPOT 1 still performs the wireless transmission duties it

did in Figure 4.1. This modification was necessary due to insufficient documentation and support for controlling the reader device.

5.2 RFID Hardware

5.2.1 RFID Reader

The RFID hardware chosen for this implementation comes from DAILY RFID CO.,LIMITED. This company was selected for their product offerings and pricing. RFID reader model DL910 listed specifications that best suited the LCTracker application. This reader contains an integrated antenna and is weather sealed for exterior use. It has the ability to read multiple tags at once and has a maximum read distance of 8 to 49 feet, depending on the tags used and surrounding environment (Daily RFID, Limited, 2009). Choosing an integrated reader and antenna not only allows for a sealed and mountable unit, but it also reduces the cost over comparable separate units that provide similar performance. This reader has multiple connections for interfacing, including the RS-232 serial interconnect needed for LCTracker. An API with sample code is provided for interfacing with the unit.

The API included with the DL910 reader has an unadvertised feature of only working in a Microsoft Windows environment. In addition, the low-level communications code is not provided. This caused difficulty since either command codes or a Java based API are required for the SPOT to control the reader. The command codes provided in the DL910 documentation unfortunately are incomplete. To continue with prototype development, an intermediate C#.NET device was inserted to interface with the reader. This is shown as PC 1 in Figure 5.1. On this device, an intermediate software program was written in C# that is able to utilize the provided Windows API to

communicate with the reader. This application requests the reader to read and then writes to a MySQL database on the same PC as temporary storage for the read ID information. SPOT 1 is also connected to this intermediate PC and can read the ID data from this temporary storage and disseminate to subsequent SPOTs accordingly. As a proof of concept, this modified arrangement is fuctionally the same as the original design, but necessitates the addition of a second PC to communicate with the reader. This issue can be resolved in two ways. One, if the DL910 communication protocol is revealed then the reader SPOT software can be updated to control the reader directly itself. Two, an alternative RFID reader that offers a Java based software development kit can be employed.

Figure 5.2 shows a portion of the C# driver application for connecting to, disconnecting from, and reading data from the DL910 RFID reader. The OpenReader method establishes the initial connection to the reader based on the type and address of the connection. In this case, the communication takes place over a serial connection connected to COM port 1. Once connected, the internal station and connection speed is established. A connection rate of 19,200 baud was selected to provide sufficient speed while remaining low enough to reduce interference and cable length restrictions.

```
private void buttonConnect_Click(object sender, EventArgs e)
   ret = Demo.ReaderDll.OpenReader(ref m_hCom, linktype, com_port);
   if (ret == 0)
   {
      …
      Demo.ReaderDll.SelectStation(m_nSite);
      …
      ret = Demo.ReaderDll.SetBaudRate(m_hCom, nBaud);
      …
      strState3 = "Connection to " + strComm + " Established.";
   }
   else
   {
      strState3 = "Connection to " + strComm + " Failed!";
   }

   this.labelStatus.Text = strState3;
}

private void buttonDisconnect_Click(object sender, EventArgs e)
{
   if (m_bConnect == true)
   {
      Demo.ReaderDll.ResetReader(m_hCom);
      Demo.ReaderDll.CloseReader(m_hCom);
      …
      this.labelStatus.Text = "Disconnected";
   }

}

private void buttonRead_Click(object sender, EventArgs e)
{
   ret = Demo.ReaderDll.MultipleTagIdentify(m_hCom, nTagType, ref nCount, id);
   for (int rd = 0; rd < nCount; rd++)
   {
      for (int rdi = 0; rdi < 14; rdi++)
      {
         textBoxData.Text += id[(rd * 14) + rdi].ToString("X2");
         readID           += id[(rd * 14) + rdi].ToString("X2");
      }
      textBoxData.Text += Environment.NewLine;
      …
}
```

Figure 5.2. Windows RFID API Usage.

5.2.2 RFID Tags

The RFID tags that were utilized also came from Daily RFID Co., Limited.

These tags operate at the same UHF frequency as the DL910 reader. Three types of tags

were tested: labels, wristbands, and rigid plastic tags. The label tags, shown in Figure 5.3, are coated paper tags with adhesive on one side. If animals are acquired with existing ear tags, these tags can be applied over them. This saves the time and labor in removing the old tag and applying a new one, as well as providing cost savings over buying new RFID tags and discarding the old ones. The wristband tags can be attached to the leg or neck of an animal, depending on size. This type of tag is good for temporary attachment and can be moved from one animal to another. If animals are never tracked after leaving the premises, this allows for the reuse of tags. Since wristband and label style tags are sized similarly to the average animal ear tag, read performance is comparable. Figure 5.4 shows one example of a wristband style tag. The final type of tag used was a rigid plastic tag similar in construction to a display type ear tag. This type of tag is displayed in Figure 5.5. The UHF frequency that these tags use is an established standard, with multiple tag types available at many different RFID vendors.

Figure 5.3. RFID Tag Labels.

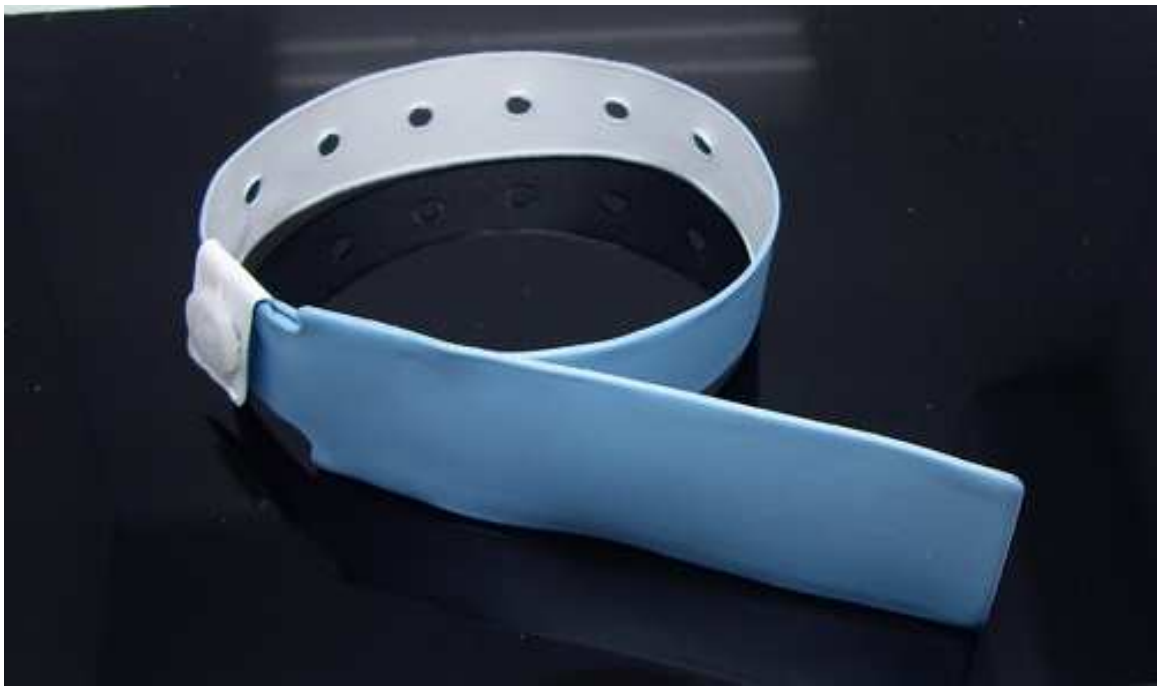

Figure 5.4. RFID Tag Wristbands.

Figure 5.5. Rigid Plastic Tag.

5.3 SPOT Communication

SPOTs from Sun Microsystems were chosen due to their wireless capabilities as well as their ability to function as a microcomputer and control other systems. These devices can be programmed in Java and can interface with a multitude of external systems. SPOTs have a serial interface that can provide RS-232 connections to external hardware. They also contain processor and memory subsystems to be able to handle computations based on input data and take appropriate action based upon it. This allows each sunspot to provide automation to a multitude of farm functions. This expandability was a main incentive for choosing this technology.

Another reason for choosing Sun SPOT systems is the modularity inherent in their design. These hardware pieces can be built from scratch from open source hardware diagrams, and can be customized to the task at hand. If only a portion of the available capabilities are needed, then that is all that needs to be built. This provides the option of buying complete kits from Sun Microsystems, or building custom hardware specific to the duty assigned to a specific SPOT. This can provide significant cost savings when the full functionality of a prebuilt SPOT is not needed.

For the LCTracker system, there are three different roles played by the SPOT devices: reader SPOT, repeater SPOT, and base station SPOT. Referring to Figure 5.1, SPOT 1 is a reader SPOT, SPOT 4 is a base station and SPOT 2 and SPOT 3 act as repeaters. Repeater SPOTs only require the battery, radio, and proccssor components. Repeaters will also typically require weatherproof mounting. The base station SPOT does not even require a battery since it connects to a PC. The reader SPOT is the most complicated. In the design of Chapter 4, these SPOTs require a battery, processor board, weatherproof housing, an RS-232 communication cable, and an eDemo board. In the prototype LCTracker system, the battery is not required since it is connected to the extra computer.

Figure 5.6 is a code listing showing examples of the Java code necessary for controlling an RFID reader such as the DL910 via a byte code instruction set. These instructions are provided in communications protocol document available in the SDK. However, the initial communication instruction is missing from this document. This would provide the functionality of the OpenReader command in Figure 5.2. While utilizing the C# reader application enabled this proof of concept implementation to be

completed and evaluated, the SPOT controlled reader application was still tested. Since communication with the reader wasn't possible without the missing instruction, SPOT 1 from Figure 5.1 was connected to the serial port of a PC running a serial port monitoring program called portmon. By logging the serial output of the SPOT, the correct instructions were confirmed. Adding the missing initial connection instruction should complete the SPOT to RFID reader interface, allowing the removal of the extra PC. The next methods in Figure 5.6 initiate the serial interface on the eDemo board. The byte code instructions are written to the serial port, and read method will block until a predefined timeout occurs.

```
byte[] setBaudRate    = {(byte)0xA5, (byte)0x00, (byte)0x03, (byte)0x74, (byte)0x01, (byte)0xE3};
byte[] resetReader    = {(byte)0xA5, (byte)0x00, (byte)0x02, (byte)0x75, (byte)0xE4};
byte[] setPower       = {(byte)0xA5, (byte)0x00, (byte)0x03, (byte)0x7F, (byte)0x00, (byte)0xD9};
byte[] multiIdentify  = {(byte)0xA5, (byte)0x00, (byte)0x03, (byte)0x93, (byte)0x04, (byte)0xC1};

EDemoBoard demo = EDemoBoard.getInstance();

demo.initUART(EDemoBoard.SERIAL_SPEED_19200,
                    EDemoBoard.SERIAL_DATABITS_8,
                    EDemoBoard.SERIAL_PARITY_NONE,
                    EDemoBoard.SERIAL_STOPBITS_1);

demo.writeUART(setBaudRate);
demo.writeUART(multiIdentify);
returnData = demo.readUART();
demo.writeUART(resetReader);
```

Figure 5.6. Java RFID Reader Driver Code Listing.

5.4 MySQL Database

5.4.1 Database Design

The LCTracker system implementation utilizes two databases. One database is used for temporary storage of the tags read by an RFID reader and the other is the main storage for all tracking and setup information. The temporary storage database resides on

PC 1 in Figure 5.1 and is part of the additional requirements to complete this prototype without a SPOT compatible reader. It is not needed for implementations with a SPOT controlled reader. The database is called tmpStore and contains only one table, tmpLocation. As shown in Figure 5.7, this table contains the minimal information necessary to track animal tag identifications. Only basic information such as the animal's RFID number, date stamp, and which reader in a particular area picked up the tag is necessary. This is because the basic key information can join the main tables at the base station PC to retrieve or store all data linked to the animal (e.g., it's weight, etc.).

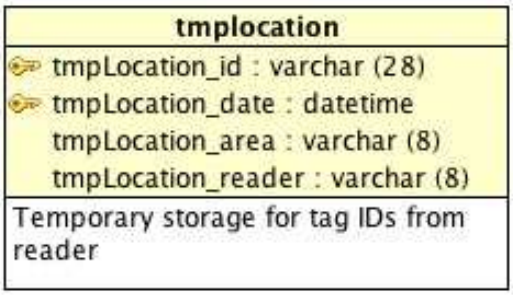

Figure 5.7. tmpStore Database.

The main system database shown in Figure 5.8 contains the core components of the LCTracker system. The initial setup of the system requires entering information about the farm itself. Different areas of the farm can be defined, and each area can contain multiple readers. Each of these tables can be added to on the fly with the acquisition of new land or restructuring of old. Animal information can be entered once and retrieved later via a single key. Animal keys can be one of three different identifiers. The animal_id in the animal table contains the RFID tag number attached to it, while the animal_easy_id is a unique number for that animal that is easier for a farm worker to remember and identify. These numbers are often displayed or written on the RFID tag itself, allowing both visual and electronic identification. The animal_national_id column

is for storing the ID number registered with the USDA. While not needed in daily use of LCTracker, this ID can reference any animal in the system if needed. It can also be transferred with the animal to maintain tracking information until time of slaughter.

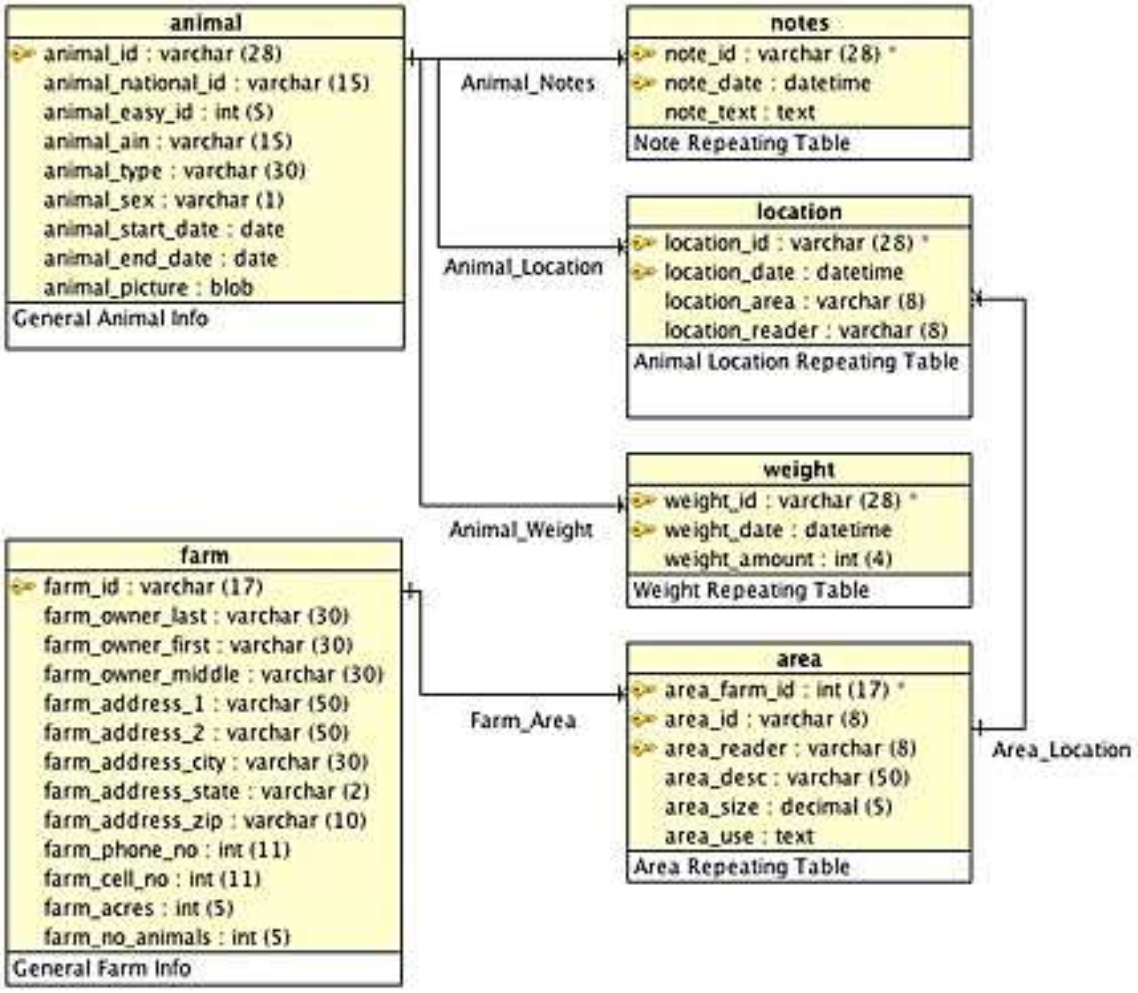

Figure 5.8. lctracker Database.

5.4.2 Database Connections

Connections to a MySQL database occur at the reader and at the base station PC. Figure 5.9 demonstrates how the MySQL connection is established in C# for the temporary storage of IDs collected from the reader by the intermediate PC-1. The MySqlClient connector is used to connect with the database and takes a connection string

with server, database, userid, and password. Once connected a MySqlCommand is created. This structure houses storage for the command itself, as well as execution methods.

```
using MySql.Data.MySqlClient;

string MyConString = "SERVER=localhost;" +
                     "DATABASE=tmpStore;" +
                     "UID=userid;" +
                     "PASSWORD=password;";
MySqlConnection connection = new MySqlConnection(MyConString);
MySqlCommand command = connection.CreateCommand();

connection.Open();

        command.CommandText = "insert into tmpLocation values ('" + readID +
"', NOW(), '3')";
        command.ExecuteNonQuery();

    connection.Close();
```

Figure 5.9. C# MySQL Connection.

Figure 5.10 shows how SPOTs access the MySQL database. SPOT 1 reads from the database, while SPOT 4 writes to the database during standard operating behavior. However, SPOT 1 and SPOT 4 can perform both duties when settings from the GUI have to be transferred to the remote readers. Once the appropriate Structured Query Language (SQL) packages are loaded, the Java Database Connectivity (JDBC) driver class is referenced. This allows a connection to be established with the provided location, username, and password. After preparing an SQL statement, the query is executed against the database and stored in a result set. This result set is then looped over, accessing values via column number indexing.

```
import java.sql.*;

String readRecord;

Class.forName("com.mysql.jdbc.Driver");
java.sql.Connection con = DriverManager.getConnection(
   "jdbc:mysql://localhost:3306/tmpStore", "USERNAME", "PASSWORD");
PreparedStatement statement = con.prepareStatement(
   "select * from tmplocation");
ResultSet result = statement.executeQuery();

while(result.next()) {
   readRecord = result.getString(1) + " " +
         result.getString(2) + " " + result.getString(3);
   System.out.println("readRecord = " + readRecord);
}
```

Figure 5.10. Java MySQL Connection.

The final database connection is made by the user interface. This dynamic web page is written in PHP and connects to the MySQL database running on the same workstation. As seen in Figure 5.11, the connection process in PHP is similar to the other methods covered. All connection variables are defined and then a connect function is called that returns a handle for that connection. Then the database is selected and the query is written. The query is executed and fetched one row at a time. A loop fetches rows and displays the information until no more rows exist, at which time the query is released from memory. This dynamic display of database information allows the GUI to reflect tracking changes in real time, requiring only a refresh of the page.

```
$hostname_lenovo = "MACHINE_ADDRESS";
$database_lenovo  = "lctracker";
$username_lenovo = "USERNAME";
$password_lenovo = "PASSWORD";
$lenovo = mysql_pconnect($hostname_lenovo, $username_lenovo, $password_lenovo) or
          trigger_error(mysql_error(),E_USER_ERROR);

mysql_select_db($database_lenovo, $lenovo);
$query_animalData = "SELECT * FROM location, animal WHERE location_id = animal_id
                     ORDER BY location_date";
$animalData       = mysql_query($query_animalData, $lenovo) or die(mysql_error());
$row_animalData   = mysql_fetch_assoc($animalData);

<?php do { ?>
  data.setCell(<?php echo $rowNum ?>, 0, <?php echo $row_animalData['location_easy_id']; ?>);
  data.setCell(<?php echo $rowNum ?>, 1, '<?php echo $row_animalData['location_date']; ?>');
  data.setCell(<?php echo $rowNum ?>, 2, '<?php echo $row_animalData['location_area']; ?>');
  data.setCell(<?php echo $rowNum ?>, 3, '<?php echo $row_animalData['animal_type']; ?>');
  data.setCell(<?php echo $rowNum ?>, 4, '<?php echo $row_animalData['animal_sex']; ?>');
  data.setCell(<?php echo $rowNum ?>, 5, '<?php echo $row_animalData['animal_start_date']; ?>');
  <?php $rowNum++; ?>
<?php } while ($row_animalData = mysql_fetch_assoc($animalData)); ?>

<?php
  mysql_free_result($animalData);
?>
```

Figure 5.11. PHP MySQL Connection.

CHAPTER 6: EVALUATION

6.1 RFID Hardware

The RFID hardware used in the implementation was chosen for its listed specifications and cost. To evaluate the hardware itself, it was placed in real world situations to compare advertised performance with performance in the field. The supplied software and documentation were assessed for their completeness and clarity, while vendor communication and support were examined. These evaluations ultimately led to a determination of the appropriateness of this hardware for the LCTracker application.

6.1.1 Reader and Tag Performance

Three different types of tags were used in these tests, labels, wristbands, and rigid tags. The label tags were easy to remove from their packaging and contained sufficient adhesive to properly secure them to a rigid surface such as existing ear tags. These tags measure 2 inches by 6 inches on the exterior, but the actual tag circuitry inside is only about three fourths of an inch by three inches. Rigid plastic tags attach the circuitry to hard plastic like the majority of large animal ear tags that are available. These tags measure 1 inch by 5 inches, with the RFID circuitry measuring .75 inches by 4.5 inches. The wristband tags can be removed and reattached due their soft plastic housing and contain a 1 inch by 3 inch RFID tag.

The first parameter evaluated was maximum read distance. This is the furthest separation a tag can have from the reader and still return the signal. The advertised maximum read range for the DL910 reader 26 to 49 feet, depending on the type of tag being read. Table 6.1 lists the maximum read ranges that resulted from this evaluation. With a maximum read distance for any tag of 30 feet, these read ranges fell into the bottom end of the manufacturer claims. Wristband and label tags fell even shorter of the goal, not reaching the claimed read distances. Given that the rigid type of tag most closely represents the average hard plastic animal ear tag, its maximum read range is still within the specifications of the LCTracker system.

Table 6.1

Maximum read distances

	Rigid Tag (ft.)	**Wristband Tag (ft.)**	**Label Tag (ft.)**
Open Air	30	21	19
Wood Door	25	17	15
Wood + Sheetrock	25	17	15
12” Cinder Block	6	4	4
Steel	0	0	0

Table 6.1 also shoes the maximum distances for tag readings to pass and return through different mediums. With many barns and other agricultural buildings being made of wood, the solid wood door test shows how much read distance is lost when a wooden structure lies between the animal and reader. With drops in distance of around 20% or less, wood causes a minimal range decrease. Not only does this mean that

livestock would have to be closer to the reader in order to be tracked, but it also means that animals inside and outside of a wood barn may be picked up. As each LCTracker installation is unique, this must be accounted for when placing the reader.

While the addition of sheetrock did not have an effect on RFID readings, they changed significantly with the change of obstruction material to cinder blocks. With read distances dropping an average of 78%, block structures significantly reduce the chance of identifying livestock through them. Again this fact has to be accounted for in reader installations. This may require additional readers on the opposite side of the wall, but it also gives the ability to restrict readings within the structure. Only animals immediately adjacent to the opposite side of the wall have a chance to be identified. As Table 6.1 indicates, steel can be used to completely impede RFID tag identification. This was the only medium in the evaluation to do so.

Since tagged livestock can move freely, the angle of the tag to the reader is always changing. As there is no front or back of an RFID tag, read angles were read based on the tag being perpendicular to the radio frequencies emitted by the reader. The rigid tag performed the best, triggering reads at all angles. Wristband tags produced reads up to 80 degrees from perpendicular, while label tags stopped reading at 65 degrees. As with the other tests of tag functionality, the rigid plastic tags performed the best. Wristband tags would cause reads with most any movement, but label tags had a large angle window where no readings took place. With the higher rate of missed readings, label tags are not suggested for use in the LCTracker system.

The final evaluation tests for the RFID hardware determined how far apart each tag must be and how many can be read at a time. All three types of tags functioned

similarly in the spacing tests. The only condition that caused misreads was two RFID tag antennas touching the ends of each other. This situation generated an average 12% failure rate over 20 reads. Overlapping tags that were not touching were still readable. Since tag antenna ends touching each other when on an animals ear is unlikely and would only be a temporary situation it is not seen as a significant issue. The DL910 reader does not have a listed maximum for number of tags read at a given time. It will attempt to read every tag that is energized by the antenna. Without the volume of tags to explore this upper ceiling, smaller tests were completed. It was found that at least seven tags in a one square foot area could be read simultaneously. Concentrations that high are virtually impossible when in use on the farm, so the number of tags read at a time was determined to not be an issue. Due to tag angle, proximity, barriers, and other external factors, every tag was not read every time. This shows the need for multiple reads, disregarding duplicates.

6.1.2 Reader Interface

The DL910 reader advertises interface connections for RS-232, RS-485, Weigand, and TCP/IP ports. The RS-232 connection was used and evaluated with LCTracker. This type of connection is an industry standard, but is commonly considered a legacy interface. It does not provide the speed and ease of use of more modern connection methods, but is more than adequate for use in this system. The RS-232 connections with a PC worked as advertised.

The SDK supplied with this reader caused most of the difficulty in setting up a proof of concept implementation identical to the initial design. SPOTs require a platform independent Java interface, and the DL910's APIs are Microsoft Windows only. A

communications protocol was provided, but was missing the initial connection instruction codes. This prevented the SPOT from controlling the reader, and required the extra PC show in Figure 5.1. While the actual function of the system was not compromised, this extra hardware increases cost and complication. It should be noted that the SDK is not advertised as Windows only, but is not promoted to be platform independent either. RFID reader options do exist for platform independent Java interfacing, and should be utilized for future LCTracker implementations.

6.2 SPOT Hardware

The SPOT mobile devices that were utilized in the implementation are advertised to be self-powered wireless development devices. In the LCTracker system, they are utilized for wireless data transmission, RFID reader control, and database access.

Serving as wireless transmitters to connect RFID readers and a base station PC, the SPOTs performed as advertised when within range. Built-in error checking ensured that data sent was received and persistent connections were sustained. However, the publicized maximum communication of 328 feet was not obtained in the LCTracker prototype evaluation. To test actual radio range, two programs were written. One SPOT was programmed to periodically broadcast a transmission, and the other was programmed to receive transmissions. Radio strength settings were set to maximum, and one of the internal LED lights of the receiving device was set to blink when it was able read the transmission. In this setup, the light will quit blinking when it is out of range.

The radio range can be affected by external conditions and placement angle. Outside, with no obstructions, a maximum distance of 57 feet was reached before signal failure occurred. This distance was even less if obstructions came between the antennas.

The maximum range indoors was 87 feet, still far short of the claimed value. This range shortfall compromises the LCTracker design, remaining useful for short-range wireless transmission only. An RFID reader could be placed outside of a barn, for instance, without running wires. However, connecting to distant fields would require too many repeater spots to be cost effective.

Rage tests were also conducted with clear acrylic enclosures sealing the SPOTs from the weather. This weather sealing is required to protect any spot subjected to moisture. Since maximum range can vary slightly from test to test, three samples were taken without the protective enclosure and three with. The averages for these tests show no appreciable loss of range from this thin clear box.

6.3 Cost

One of the main hurdles that the USDA saw with the acceptance of their NAIS program was the cost incurred by the livestock producers (Jeffries, 2006). Since price was a prohibiting factor, a major goal of the LCTracker system was cost. Every step in the design evaluated cost vs. features for that particular function. To evaluate the performance of LCTracker in the respect, pricing for tracking systems was divided between software and hardware.

6.3.1 Hardware Costs

Table 2.4 showed RFID hardware costs for the equipment popular with commercial tracking systems in use today. Allflex and Destron RFID systems are specialized for livestock identification, therefore limiting their market. The hardware utilized in LCTracker not only uses different radio frequencies to increase performance, but is also less specialized in function. This drives down the cost of the equipment.

Table 6.2 shows the costs for panel readers from two manufacturers. Daily RFID Co., Limited produces the DL910 reader used in the implementation. This reader provides the necessary specifications for frequency and range, but lacked the platform independent API necessary to complete an LCTracker installation without an extra PC. While slightly more expensive, the Alien reader model 9650 offers similar performance and has the necessary Java API to enable SPOT control of the reader. It should be noted that shipping prices significantly affect total purchase price. The lowest shipping option for the Alien reader is around $10. The lowest option for the Daily product is $147, bringing the actual cost difference to around $130.

Table 6.2

Implementation RFID hardware costs

	Daily	**Alien**
Panel Reader/Antenna	$510	$778
Tags	$0.90 - $3.00	$2.00 - $5.00

RFID tag prices, as seen in Table 6.2, can vary depending on type and quantity purchased. The actual RFID components inside a tag are quite inexpensive, and only make up a small portion of tag costs. Alien produces RFID inlays that are available for less than $0.30 when bought in quantities of 500 or more. These inlays must be placed in the chosen tag medium, such as an ear tag. Small farms may be unwilling to invest in the 10,000 or more quantity of tags to get the best pricing. Orders of as little as 250 are available for around $2.00 per animal.

SPOT technology by Sun Microsystems is only sold in development kits at this time. Each development kit contains two free-range SPOTs, one base station SPOT, development tools, a USB cable to connect to a PC, and quick-attach mounts. The base kit provides the equipment necessary to create two RFID reader installations at up to 200 yards from the base workstation. SPOT to SPOT communication is limited to a maximum of 100 yards[1], but automatic transmission hopping will allow any spot to communicate with any other spot as long as a network of connected SPOTs exists between them.

SPOTs are connected to RFID readers via an RS-232 serial connection. To make this connection, a 9-pin serial cable is required. These cables can be obtained from reputable retailers for as little as $2, as shown in Table 6.3. One cable is required for each reader that is used. The SPOT to PC connection is made via a USB cable that is provided in the development kit.

The protective housing item in Table 6.3 is necessary when mounting a SPOT where it will be vulnerable to the weather. Clear acrylic boxes suitable for this purpose can be found for as little as $3. If custom enclosures are desired, large sheets of acrylic can be cut and assembled with silicone caulk. These sheets are available for around $20 for two feet by four feet sections.

Table 6.3

SPOT Hardware Costs

	SPOT Dev Kit	**Serial Cable**	**Protective Housing**
Price	$399	$2	$3

[1] Refer to Section 6.2 for wireless

6.3.2 Software Cost

The software components of the LCTracker system generate a large cost savings over commercial systems. Freely available open source software was used for the system interface and all source code of the software created for the proof of concept implementation is available. The cheapest commercial system reviewed costs at least $245 to be able to track an unlimited number of livestock. This minimum savings of $245 can grow much higher with version upgrades, multi-user licenses, and the addition of wireless support. Some of the features of LCTracker are either not available, or only available as add-ons at an extra cost.

The LCTracker software also saves on hardware costs. Government tracking systems such as NAIS and its forthcoming successor require a unique ID for all livestock. In previous systems, this meant the purchase of specialty RFID tags with a specific number range. LCTracker enables the use of any RFID tag by coupling the actual tag number with the government provided number in the database. A lookup by either number is specific to a single animal. This also broadens the type of tag available for use.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Conclusion

Livestock tracking is a common and valuable practice. Modern tracking involves the use of semi-automated techniques such as barcodes or RFID tags. The USDA has proposed and is revising a policy for livestock tracking, particularly for disease management and traceability. Many consumers are requesting knowledge of animal product lifecycle. While larger livestock operations may be able to afford the costs of current commercial livestock tracking systems, there is a need for a low cost livestock tracking system.

This thesis describes the design and prototype implementation of the LCTracker low cost livestock tracking system. Wide-area, panel-style RFID readers can be positioned at critical livestock locations, such as feed stations, water stations, gates, barns, etc. Smart devices can control these readers and transmit RFID tag readings via wireless radio signals. Repeating devices are smart devices that can inexpensively propagate readings from distant readers to a base station. The base station is a wireless receiving device attached to a computer and typically located in a barn or house. Readings reaching the base station can be stored locally in the LCTracker database or communicated to an offsite LCTracker hosted database via Internet connection. The offsite feature means that farmers do not have to maintain the computer system and perform data backups.

The LCTracker prototype implementation used a low-cost panel RFID with Sun SPOT devices as the controller, repeater, and base station devices. All components could easily be replaced with other hardware that provides equivalent functionality. The thesis also evaluates the prototype for cost and performance. The cost is low, primarily due to the exclusive use of open-source software. Reading and wireless transmission ranges were found to be lower than specified by the manufacturers.

7.2 Future Work

The implementation of LCTracker proved that such a system can be created and could be a cost effective solution to more expensive commercial systems. As with any system, however, a continued refinement is necessary to create a polished end product.

7.2.1 RFID Reader Controller

The main future work for SPOTs and RFID readers in the LCTracker system is the complete operation of a reader by the SPOT. Having the extra PC requirement, as is shown in Figure 5.1 compared to Figure 4.1, increases complexity and cost significantly. The SPOT hardware was specifically chosen because it has the components necessary to control the reader without the assistance of a PC. This can be resolved by using a Java API enabled RFID reader, such as the Alien 9650. Removing the extra PC also removes the tmpStore MySQL database from the system design. The connected device receives the read tag ID information directly and can disseminate accordingly.

The SPOT's role as a reader controller is enabled by their processing and external device integration abilities. SPOTs can be replaced in this role with similar products capable of performing the same function. Small, low-power devices based on the ZigBee protocol could be used, as well as mobile Arduino devices. ZigBee devices run on the

same 802.15.4 radio standard as SPOTs and many device options exist (ZigBee, 2010). Arduino devices are open source, and can be custom built for a specific task (Arduino, 2010). Cost and performance evaluations would determine the best option for this task.

7.2.2 Repeaters

The current task for repeater SPOTs is simply to relay wireless data in route to a specific destination through any SPOTs in range. In the expansion of this system, these SPOTs can provide additional functionality. To handle the possibility of data loss when the destination cannot be reached, a store and forward technique can be implemented. When the destination SPOT is unreachable, the data is stored on the device for later transmission when the original is back online. This would increase the fault and delay tolerance of the network, allowing it to handle a dead battery or other SPOT disruption.

To avoid the disruption possibility of a dead SPOT battery, battery monitoring could be built into the system. SPOTs can check their own power level, which could be reported and monitored in the system interface. Alerts could notify farmers when battery levels are getting low so that they can be recharged before a failure occurs. The ability to periodically check their power level and report back to the base station could be built into every free-range spot in use.

Similar to periodic battery level checks, a system integrity check could alert system operators about problems before they occur. This would be a recurring system check ensuring that every SPOT in the network was reachable. If a SPOT is not reachable, notifications could be presented in the user interface. These can be initiated from an otherwise idle repeater spot, making use of all available hardware.

Full free-range SPOTs have the ability to do much more than is required to repeat a wireless signal. These devices can control other systems such as servos and motors while still repeating wireless traffic. However, if the extra capabilities of a SPOT are not needed for a repeater node, custom built SPOTs with only the required components could be built. If other wireless devices are used to control the reader, then they could also relay a data transmission to nearby devices.

7.2.3 LCTracker System Software

There are many expansion opportunities for the user interface of LCTracker. This software could offer the ability to set up automated integrity checking, as well on-demand checks when a problem is suspected. Additional system settings in the interface could allow for system adjustment from one location. Providing adjustments such as read and wireless transmission frequencies, as well as putting all or parts of the system asleep from within the user interface allows for remote management opportunities.

Reporting options could also be added to the system software. NAIS compliance, daily activity, movement, and feeding reports would provide farmers with data collections that would require hours to compile individually. Custom fields and reports would allow the farmer to dictate the exact information they wanted assembled and how it is presented.

Another area of expansion for the user interface is notifications. The end user would determine the best method or methods for receiving them, and enter those into the system. PCs with an Internet connection can automatically send emails and texts, but notifications can also be presented within the interface. Livestock that haven't been to a feed bin in a set amount of time can signal a problem, so the notification setup would

allow the farmer to set the amount of time and notification means for this issue. Farmers could also be notified when there are too many animals in an area, or when an animal hasn't been located by a reader in an established period of time. This type of data can be presented in a report, but is more time sensitive in nature. Being proactively notified by the system frees the user from having to constantly run this type of report, knowing they will be alerted about urgent situations.

All data for the LCTracker system reside in a MySQL database. This database can either be located on the base station PC or stored by a hosting provider. If the database is stored locally, a method for data backup is needed. This can be as simple and cost effective as an external USB key or hard drive. These are available as low as $50 in sizes large enough to store all the data for a moderate sized farm. Options built into the system software could provide one-touch backups or restores from these external storage devices. If the database is hosted, these companies typically offer backup as part of the service price. Online backup sites are another option for either setup. There are many options available that offer as much as 50GB of storage, adding no cost to an LCTracker implementation. Integration with these backup solutions should be easy for a layman to setup, with scheduling them automatically being ideal.

7.2.4 Additional Automation Possibilities

There are many possibilities for increasing automation in the current system. One such method is to utilize the processing and interfacing abilities of Sun SPOTs. Free-range SPOTs contain their own temperature, light, and gyroscopic sensors, and can interface with many other types of external devices via onboard I/O ports. The automatic

collection of data from these subsystems can be stored and can trigger other actions within the system.

Since SPOTs contain a photo sensor, controls can be activated based on available light, such as motors that open or close gates at dawn and dusk. The temperature sensor can monitor daily highs and lows, both indoors and out. Weight scales can be connected to the external ports of a SPOT to weigh an animal. Multiple sensors can be connected to one spot at a time, so the SPOT could automatically sense an animal on the scale and trigger the reader to read the tag of that animal. This data can be compared to the animals frequency of location at feed bins or other locations to determine which is producing more animal growth. Sensors can also be placed in feed bins to determine when supply is low, reporting this to the end user.

Adding multiple RFID stations at different areas of a farm allows livestock to be tracked more accurately. This data can then be used to determine land use efficiency. Areas of little use can either be repurposed or can handle a higher head count, and overused areas can be identified so that animals can be moved to another area. This information can also be used to control gates or connectors between different areas based on their current population. This could mitigate the possibility for overgrazing and control herd movement without human direction.

REFERENCES

Apache HTTP Server Project. (2009). Retrieved July 6, 2010, from http://httpd.apache.org/

APHIS. (2010). *Questions and Answers: New animal disease traceability framework.* Retrieved February 16, 2010, from http://www.aphis.usda.gov/publications/animal_health/content/printable_version/faq_traceability.pdf

Arduino. (2010). Retrieved July 29, 2010, from http://www.arduino.cc/

Barbari, M., Conti, L., & Simonini, S. (2010). *Spatial identification of animals in different breeding systems to monitor behavior*. Retrieved July 25, 2010, from http://www.diaf.unifi.it/upload/sub/attivitadiricerca/progetti/costruzioni/RFID/RET_paper0209_Barbari.pdf

Cattlesoft. (2010*). Cattle software - record keeping made easy by cattlemax*. Retrieved July 7, 2010, from http://www.cattlemax.com/

CattleStore.com. (2008). Retrieved July 19, 2010, from http://www.cattlestore.com/

Daily Rfid Co., Limited. (2009). Retrieved July 11, 2010, from http://www.rfid-in-china.com/2008-08-25/products_detail_2119.html

DHIA Services. (2008). *Ear tag central.* Retrieved July 19, 2010, from http://www.eartagcentral.com/home.php

Domdouzis, K., Kumar, B., & Anumba, C. (2007). Radio-frequency identification (RFID) applications: A brief introduction. *Advanced Engineering Informatics*, *21*(4), 350–355.

Farren, L. (2008). *GPS collars track cattle on range.* Retrieved February 17, 2010, from

http://hayandforage.com/grazing/0501-gps-collars-track-cattle/

Frost, A. R., Schofield, C. P., Beaulah, S. A., Mottram, T. T., Lines, J. A., & Wathes, C. M. (1997). A review of livestock monitoring and the need for integrated systems. *Computers and Electronics in Agriculture*, *17*(2), 139–159.

Grow Systems. (2006). *TrackLivestock.net - Web-based livestock tracking software.* Retrieved July 7, 2010, from http://www.tracklivestock.net/default.aspx

Halverson, Gary Don. (2008). *RFID animal identification in the U.S. beef industry: A study of actual costs incurred and price premiums received at the producer level.* Unpublished dissertation, Utah State University, Logan UT.

Herd-Pro Software. (2008). *StocKeeper 2003 -- Dairy and livestock software.* Retrieved February 7, 2010, from http://www.herd-pro.com/default.asp?IncPage=default_content.asp

Hospital RFID system stops baby abduction. (2005, July 19). RFID Gazette. Retrieved July 24, 2010, from http://www.rfidgazette.org/2005/07/hospital_rfid_s.html

Jeffries, W. (2006). *About NoNAIS.org.* Retrieved February 23, 2010, from http://nonais.org/about/

Johnston, L. (2003). *New mad cow rules are leftovers.* CBS News. Retrieved February 23, 2010, from http://www.cbsnews.com/stories/2003/12/23/national/main590039.shtml

Midwest MicroSystems. (2008). *Cow sense store - cattle products, cow ear tags, and cattle software for cow calf producers*. Retrieved July 7, 2010, from http://www.shopcowsense.com/default.aspx

NAIS Benefit-Cost Research Team. (2009). *Benefit-cost analysis of the national animal*

identification system. Retrieved July 7, 2010, from http://www.naiber.org/Publications/NAIBER/BC.analysis.NAIS.pdf

Netcraft. (2010). *June 2010 web server survey.* Retrieved July 6, 2010, from http://news.netcraft.com/archives/2010/06/16/june-2010-web-server-survey.html

QC Supply. (2009). Retrieved July 19, 2010, from http://www.qcsupply.com/qcsupply/index.jsp

Oracle. (2010a). *MySQL :: Dispelling the myths*. Retrieved June 12, 2010, from http://dev.mysql.com/tech-resources/articles/dispelling-the-myths.html

Oracle. (2010b). *Squawk.* Retrieved July 15, 2010, from https://squawk.dev.java.net/

Rasmussen, W. (1962). *The impact of technological change on American agriculture, 1862-1962.* The Journal of Economic History, *22*(4), 578 - 591.

RFID Tribe. (2010). *Livestock Management.* Retrieved February 22, 2010, from http://www.rfidtribe.com/index.php?option=com_content&view=article&id=472&Itemid=102

Rossing, W. (1999). Animal identification: Introduction and history. *Computers and Electronics in Agriculture*, *24*(1-2), 1–4.

Sun Microsystems. (2010). *SunSPOTWorld - Our vision.* Retrieved May 20, 2010, from http://www.sunspotworld.com/vision.html

The Apache Ant Project. (2010). *Frequently asked questions.* Retrieved July 15, 2010, from http://ant.apache.org/faq.html#what-is-ant

The PHP Group. (2010). *PHP.* Retrieved July 6, 2010, from http://php.net/index.php

U.S. Department of Homeland Security Smart Border Alliance. (2005). *RFID feasibility study final report, Attachment D.* Retrieved from

www.dhs.gov/xlibrary/assets/foia/US-VISIT_RFIDattachD.pdf

USDA. (2007). *NAUS user guide.* Retrieved February 23, 2010, from http://docs.google.com/viewer?a=v&q=cache:ZOCw5cDGwG4J:animalid.aphis.usda.gov/nais/naislibrary/documents/guidelines/NAIS-UserGuide.pdf+usda+nais+guidelines&hl=en&gl=us&pid=bl&srcid=ADGEEShDtvztzIrMhdf0Ms-B74mtvv4fT9wTSGv3Ju1b4wnGFTiFow4nka0dmMGU4Lgu7Wk-qivqOFwyNL1SljLlDSuaTVeN_d--DSGl2qp_sFG43UMJWqIINVvHWsOlDbGw_wdxZC40&sig=AHIEtbRY8iESA9Ppb0eyfJjVkQo3A7vIUg

USDA. (2010a). *Animal disease traceablility home.* Retrieved July 7, 2010, from http://www.aphis.usda.gov/traceability/forum/index.shtml

USDA.(2010b). *Animal identification information.* Retrieved February 23, 2010, from http://www.aphis.usda.gov/animal_health/animal_diseases/animal_id/

Voulodimos, A. S., Patrikakis, C. Z., Sideridis, A. B., Ntafis, V. A., & Xylouri, E. M. (2010). A complete farm management system based on animal identification using RFID technology. *Computers and Electronics in Agriculture, 70*(2), 380–388.

ZigBee Alliance. (2010). Retrieved July 29, 2010, from http://www.zigbee.org/

APPENDIX A

A.1 Reader PC C# Code Listing:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
using System.Timers;
using System.Threading;


namespace Demo
{
    public partial class Form2 : Form
    {
        int  m_hCom     = 0;
        byte m_nSite    = 0;
        bool m_bConnect = false;
        byte nTagType   = 4;
        int  nCount     = 0;
        byte[] id;
        byte ret        = 1;
        const int IDENTIFYSINGLE    = 1;
        const int IDENTIIFYMULTIPLE = 2;
        const int GEN2TAG           = 2;
        System.Timers.Timer myTimer = new System.Timers.Timer();

        public Form2()
        {
            InitializeComponent();
        }

        private void buttonConnect_Click(object sender, EventArgs e)
        {
            byte linktype    = 1;
            String com_port  = "COM1";
            String strState3 = "Error State: ";
            String strComm   = com_port;

            ret = Demo.ReaderDll.OpenReader(ref m_hCom, linktype,
                                            com_port);
            if (ret == 0)
            {
                bool bSucce = true;
                m_nSite     = (byte)0;
                Demo.ReaderDll.SelectStation(m_nSite);
                int nBaud   = 1;

                ret = Demo.ReaderDll.SetBaudRate(m_hCom, nBaud);
```

```
        if (ret == 0&&(Demo.ReaderDll.StopRFwork(m_hCom)) == 0)
        {
            strState3 += " Stopped RF Work";
        }
        else
        {
            strState3 += " Error stopping RF work";
            bSucce     = false;
        }

        if (bSucce == true)
        {
            byte[] major = new byte[1], minor = new byte[1];
            ret = Demo.ReaderDll.GetFirmwareVersion(m_hCom,
                  major, minor);
            if (ret == 0)
            {
                m_bConnect = true;
                String strVer = String.Format(", V{0:D}.{1:D}",
                                major[0], minor[0]);
                strState3 += strVer;
                this.buttonConnect.Enabled    = false;
                this.buttonDisconnect.Enabled = true;
            }
        }
    }
    else
    {
        strState3 = "Connection to " + strComm + " Failed!";
    }

    this.labelStatus.Text = strState3;
}

private void buttonDisconnect_Click(object sender, EventArgs e)
{
    if (m_bConnect == true)
    {
        Demo.ReaderDll.ResetReader(m_hCom);
        Demo.ReaderDll.CloseReader(m_hCom);
        this.buttonConnect.Enabled    = true;
        this.buttonDisconnect.Enabled = false;
        this.labelStatus.Text         = "Disconnected";
    }

}

private void buttonRead_Click(object sender, EventArgs e)
{
    id              = new byte[150];
    string   readID = "";
    // Build in enough overhead for expected head count:
    string[] prevID = new string[20];
    // tmpStore MySQL connection
    string MyConString = "SERVER=localhost;" +
                         "DATABASE=tmpStore;" +
                         "UID=REPLACE_WITH_USER_ID;" +
```

```
                        "PASSWORD=REPLACE_WITH_PASSWORD;";
    MySqlConnection connection =
        new MySqlConnection(MyConString);
    MySqlCommand command = connection.CreateCommand();

    try
    {
        connection.Open();
    }
    catch
    {
        MessageBox.Show("Unable to open connection to
                        Database!\nNo tags will be stored.");
    }
    textBoxData.Text += Environment.NewLine +
        "**** Read Values ****" + Environment.NewLine;
    ret = Demo.ReaderDll.MultipleTagIdentify(m_hCom, nTagType,
                                             ref nCount, id);
    for (int rd = 0; rd < nCount; rd++)
    {
        for (int rdi = 0; rdi < 14; rdi++)
        {
            textBoxData.Text += id[(rd * 14) +
                                   rdi].ToString("X2");
            readID           += id[(rd * 14) +
                                   rdi].ToString("X2");
        }
        textBoxData.Text += Environment.NewLine;

        if (!((IList<string>)prevID).Contains(readID))
        {
            command.CommandText = "insert into tmpLocation
                                   values ('" + readID + "', "+
                                   “NOW(), '3', '1', 'N')";

            try
            {
                command.ExecuteNonQuery();
            }
            catch
            {
                MessageBox.Show("Unable to execute table " +
                    insert!\nNo tags will be stored.");
            }
        }
        prevID[rd] = String.Copy(readID);
        readID = "";
    }

    connection.Close();
}

private void buttonClear_Click(object sender, EventArgs e)
{
    textBoxData.Clear();
}
```

```
private void buttonTimedStart_Click(object sender, EventArgs e)
{
    myTimer.Elapsed += new ElapsedEventHandler(timedRead);
    myTimer.Interval = 10000;
    myTimer.Start();
}

delegate void SetTextCallback(string text);

private void SetSomeText(string text)
{
    if(textBoxData.InvokeRequired)
    {
        SetTextCallback d = new SetTextCallback(SetSomeText);
        this.Invoke(d, new object[] { text });
    }
    else
    {
        this.textBoxData.Text += text;
    }
}

public void timedRead(object source, ElapsedEventArgs e)
{
    id = new byte[150];
    string readID = "";
    // Build in enough overhead for expected head count.
    string[] prevID = new string[20];
    // tmpStore MySQL connection
    string MyConString = "SERVER=localhost;" +
                         "DATABASE=tmpStore;" +
                         "UID=root;" +
                         "PASSWORD=dinanm3;";
    MySqlConnection connection =
        new MySqlConnection(MyConString);
    MySqlCommand command = connection.CreateCommand();

    try
    {
        connection.Open();
    }
    catch
    {
        MessageBox.Show("Unable to open connection to “ +
            “Database!\nNo tags will be stored.");
    }

    SetSomeText(Environment.NewLine + "**** Read Values ****" +
                Environment.NewLine);

    ret = Demo.ReaderDll.MultipleTagIdentify(m_hCom, nTagType,
                                             ref nCount, id);
    for (int rd = 0; rd < nCount; rd++)
    {
        for (int rdi = 0; rdi < 14; rdi++)
        {
            SetSomeText(id[(rd * 14) + rdi].ToString("X2"));
```

```
                    readID += id[(rd * 14) + rdi].ToString("X2");
                }

                SetSomeText(Environment.NewLine);

                if (!((IList<string>)prevID).Contains(readID))
                {
                    command.CommandText = "insert into tmpLocation “ +
                     “values ('" + readID + "', NOW(), '3', '1', 'N')";
                    command.ExecuteNonQuery();
                }
                prevID[rd] = String.Copy(readID);
                readID = "";
            }

            connection.Close();
        }

        private void buttonTimedEnd_Click(object sender, EventArgs e)
        {
            myTimer.Stop();
        }
    }
}
```

A.2 Reader PC SPOT Code Listing

```
package org.sunspotworld.demo;

import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.util.Utils;

import com.sun.spot.peripheral.ota.OTACommandServer;
import java.text.DateFormat;
import java.util.Date;
import java.io.DataInput;
import javax.microedition.io.*;
import java.sql.*;

/**
 * This application is the 'reader PC' portion of the LCTracker Demo
 *
 * @author: Jason Grubb
 */

public class SendDataDemoHostApplication {
    // Broadcast port on which we listen for sensor samples
    private static final int HOST_PORT = 67;

    private void run() throws Exception {
        RadiogramConnection rCon;
        Datagram dg;
        boolean resultsFound = false;
        byte[] b = {};
        byte   x = (byte)0xFF;
        DateFormat fmt = DateFormat.getTimeInstance();
```

```
    final String HEXES = "0123456789ABCDEF";
    String readRecord;

    try {
        // Open up a server-side broadcast radiogram connection
        // to listen for sensor readings being sent by different
        // SPOTs
        rCon = (RadiogramConnection)
            Connector.open("radiogram://broadcast:" + HOST_PORT);
        dg = rCon.newDatagram(rCon.getMaximumLength());
    } catch (Exception e) {
         System.err.println("setUp caught " + e.getMessage());
         throw e;
    }

    Class.forName("com.mysql.jdbc.Driver");
    java.sql.Connection con = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/tmpStore", "root",
            "dinanm3");
    PreparedStatement getStatement = con.prepareStatement(
            "select * from tmplocation where tmpLocation_sent " +
            "= 'N'");
    ResultSet result = getStatement.executeQuery();
    PreparedStatement putStatement = con.prepareStatement(
            "update tmplocation set tmpLocation_sent = 'Y' " +
            "where tmpLocation_sent = 'N'");

    while(true) {
        while(result.next()) {
            readRecord = result.getString(1) + " " +
                         result.getString(2) + " " +
                         result.getString(3) + " " +
                         result.getString(4);
            System.out.println("readRecord = " + readRecord);
            resultsFound = true;
            try {
                dg.reset();
                dg.writeUTF(readRecord);
                rCon.send(dg);
            } catch (Exception e) {
                System.err.println("Caught " + e + " while " +
                                   "sending " +  record.");
                throw e;
            }
        }

        if (resultsFound) {
            putStatement.executeUpdate();
            resultsFound = false;
        }
        Utils.sleep(20000); // Check for new tags every 10 seconds.
        result = getStatement.executeQuery();
    }
}

/**
 * Start up the host application.
```

```
     *
     * @param args any command line arguments
     */
    public static void main(String[] args) throws Exception {
        // register the application's name with the OTA Command server
        // & start OTA running
        OTACommandServer.start("SendDataDemo");

        SendDataDemoHostApplication app = new
            SendDataDemoHostApplication();
        app.run();
    }
}
```

A.3 Base Station PC SPOT Code Listing

```
package org.sunspotworld.demo;

import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.peripheral.ota.OTACommandServer;
import java.text.DateFormat;
import java.util.Date;
import java.io.DataInput;
import javax.microedition.io.*;
import java.sql.*;

/**
 * This application is the 'reader PC' portion of the LCTracker Demo
 *
 * @author: Jason Grubb
 */

public class SendDataDemoHostApplication {
    // Broadcast port on which we listen for sensor samples
    private static final int HOST_PORT = 67;

    private void run() throws Exception {
    System.out.println("Starting...");
    RadiogramConnection rCon;
    Datagram dg;
    String readData;

    Class.forName("com.mysql.jdbc.Driver");
    java.sql.Connection con = DriverManager.getConnection(
        "jdbc:mysql://152.10.146.52:3306/lctracker", "USERID",
        "PASSWORD");

    try {
        // Open up a server-side broadcast radiogram connection
        // to listen for sensor readings being sent by different SPOTs
        rCon = (RadiogramConnection) Connector.open("radiogram://:" +
                HOST_PORT);
        dg   = rCon.newDatagram(rCon.getMaximumLength());
    } catch (Exception e) {
        System.err.println("setUp caught " + e.getMessage());
        throw e;
```

```
    }

    // Main data collection loop
    while (true) {
        try {
            rCon.receive(dg);
            readData = dg.readUTF();

            System.out.println("readData = " + readData);
            String patternStr = "[ ]+";
            String[] fields = readData.split(patternStr);

            PreparedStatement putStatement = con.prepareStatement(
            "insert into location (" +
            "location_id, location_date, location_area,” +
            “location_reader) " +
            "values ('" + fields[0] + "','" + fields[1] + " " +
             fields[2] + "','" + fields[3] + "','" +
             fields[4] + "')");

            System.out.println("insert into location (" +
            "location_id, location_date, location_area, “ +
            “location_reader) " +
            "values ('" + fields[0] + "','" + fields[1] + " " +
             fields[2] + "','" + fields[3] + "','" + fields[4] + "')");

            putStatement.executeUpdate();
        } catch (Exception e) {
            System.err.println("Caught " + e +
                               " while reading tag IDs.");
            throw e;
        }
    }
    }

    /**
     * Start up the host application.
     *
     * @param args any command line arguments
     */
    public static void main(String[] args) throws Exception {
        OTACommandServer.start("SendDataDemo");

        SendDataDemoHostApplication app = new
            SendDataDemoHostApplication();
        app.run();
    }
}
```

A.4 Web Display Demo Code Listings

```
lenovo.php:
<?php
# FileName="Connection_php_mysql.htm"
# Type="MYSQL"
```

```
# HTTP="true"
$hostname_lenovo = "IP_ADDRESS";
$database_lenovo = "lctracker";
$username_lenovo = "USERNAME";
$password_lenovo = "PASSWORD";
$lenovo = mysql_pconnect($hostname_lenovo, $username_lenovo,
$password_lenovo) or trigger_error(mysql_error(),E_USER_ERROR);
?>

layout.css:
#wrapper {
    width: 90%;
    margin: 0px;
    padding-right: 5%;
    padding-left: 5%;
    padding-top: 5px;
}
body {
    margin: 0px;
}
#header {
    height: 60px;
    width: 100%;
    font-family:Arial, Helvetica, sans-serif;
    font-size: 14px;
    line-height: 22px;
    font-weight: bold;
}
#navbar {
    float: right;
    height: 22px;
    width: 300px;
    font-family:Arial, Helvetica, sans-serif;
    font-size: 14px;
    line-height: 22px;
    font-weight: bold;
    border: 1px solid #000;
    background-image:url('../images/navbar_gradient.jpg');
    background-repeat:repeat-x;
    border-radius: 5px;
}
#body {
    /*
    padding: 20px;
*/
    border-right-width: 5px;
    border-left-width: 5px;
    border-right-style: solid;
    border-left-style: solid;
    border-right-color: #252525;
    border-left-color: #252525;
    margin: 0px;
    }
#footer {
    background-color: #252525;
    height: 60px;
    width: 100%;
```

```
}
#button a{
    float: left;
    width: 100px;
    height: 25px;
    color: #000;
    text-decoration: none;
    text-align: center;
}
#button a:hover{
    float: left;
    width: 100px;
    height: 25px;
    color: #FFF;
    text-decoration: none;
    text-align: center;
    background-image:url('../images/navbar_gradient_rollover.jpg');
    background-repeat:repeat-x;
}
#logo {
    width: 600px;
    height: 100%;
    float: left;
}
```

lctracker.php:

```
<?php require_once('Connections/lenovo.php'); ?>
<?php
if (!function_exists("GetSQLValueString")) {
function GetSQLValueString($theValue, $theType, $theDefinedValue = "",
$theNotDefinedValue = "")
{
  if (PHP_VERSION < 6) {
    $theValue = get_magic_quotes_gpc() ? stripslashes($theValue) :
$theValue;
  }

  $theValue = function_exists("mysql_real_escape_string") ?
mysql_real_escape_string($theValue) : mysql_escape_string($theValue);

  switch ($theType) {
    case "text":
      $theValue = ($theValue != "") ? "'" . $theValue . "'" : "NULL";
      break;
    case "long":
    case "int":
      $theValue = ($theValue != "") ? intval($theValue) : "NULL";
      break;
    case "double":
      $theValue = ($theValue != "") ? doubleval($theValue) : "NULL";
      break;
    case "date":
      $theValue = ($theValue != "") ? "'" . $theValue . "'" : "NULL";
      break;
    case "defined":
      $theValue = ($theValue != "") ? $theDefinedValue :
$theNotDefinedValue;
```

```
      break;
  }
  return $theValue;
}
}

$maxRows_animalData = 20;
$pageNum_animalData = 0;
if (isset($_GET['pageNum_animalData'])) {
  $pageNum_animalData = $_GET['pageNum_animalData'];
}
$startRow_animalData = $pageNum_animalData * $maxRows_animalData;

mysql_select_db($database_lenovo, $lenovo);
$query_animalData = "SELECT * FROM location, animal WHERE location_id =
    animal_id ORDER BY location_date DESC";
$query_limit_animalData = sprintf("%s LIMIT %d, %d", $query_animalData,
$startRow_animalData, $maxRows_animalData);
$animalData = mysql_query($query_limit_animalData, $lenovo) or
    die(mysql_error());
$row_animalData = mysql_fetch_assoc($animalData);

if (isset($_GET['totalRows_animalData'])) {
  $totalRows_animalData = $_GET['totalRows_animalData'];
} else {
  $all_animalData = mysql_query($query_animalData);
  $totalRows_animalData = mysql_num_rows($all_animalData);
}
$totalPages_animalData =
    ceil($totalRows_animalData/$maxRows_animalData)-1;
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; />
<title>Untitled Document</title>
<link href="css/layout.css" rel="stylesheet" type="text/css" />
    <script type='text/javascript'
     src='http://www.google.com/jsapi'></script>
    <script type='text/javascript'>
      google.load('visualization', '1', {packages:['table']});
      google.setOnLoadCallback(drawTable);
      function drawTable() {
        var data = new google.visualization.DataTable();
        data.addColumn('string', 'location id');
        data.addColumn('string', 'location date');
        data.addColumn('string', 'location area');
        data.addColumn('string', 'location reader');
        data.addColumn('string', 'animal type');
        data.addColumn('string', 'sex');
        data.addColumn('string', 'start date');
        data.addRows(<?php echo $maxRows_animalData; ?>);
        <?php $rowNum = 0; ?>
        <?php do { ?>
          data.setCell(<?php echo $rowNum ?>, 0, '<?php echo
```

```
            $row_animalData['location_id']; ?>');
          data.setCell(<?php echo $rowNum ?>, 1, '<?php echo
            $row_animalData['location_date']; ?>');
          data.setCell(<?php echo $rowNum ?>, 2, '<?php echo
            $row_animalData['location_area']; ?>');
          data.setCell(<?php echo $rowNum ?>, 3, '<?php echo
            $row_animalData['location_reader']; ?>');
          data.setCell(<?php echo $rowNum ?>, 4, '<?php echo
            $row_animalData['animal_type']; ?>');
          data.setCell(<?php echo $rowNum ?>, 5, '<?php echo
            $row_animalData['animal_sex']; ?>');
          data.setCell(<?php echo $rowNum ?>, 6, '<?php echo
            $row_animalData['animal_start_date']; ?>');
          <?php $rowNum++; ?>
        <?php } while ($row_animalData =
                    mysql_fetch_assoc($animalData)); ?>
       var table = new
        google.visualization.Table(document.getElementById('new_div'));
       table.draw(data, {showRowNumber: true});
      }
    </script>
</head>

<body>
<div id="wrapper">
    <div id="header">
        <div id="logo"><h1>LCTracker Data Collection Demo</h1></div>
    </div>
    <div id="body">
        <p><div id="new_div"></div></p>
    </div>
</div>
</body>
</html>
```

VITA

Jason T. Grubb was born in Boone, NC on July 15, 1978. He attended elementary schools there and graduated from Watauga High in June 1996. Later that year, he entered Appalachian State University to study Computer Science. After working part-time for the university, he accepted a full-time position in the library's computer support center in August 1999. In September 2000, he accepted a position with the Instructional Technology Center's hardware repair center. He received his Bachelor of Science degree in Computer Science the following year. Having gained experience with software and hardware systems, he accepted an Applications Analyst Programmer position with Appalachian State University's Information Technology Services division in August 2001. He began studying for his Master of Science degree part-time in spring of 2003. In April, 2007, Mr. Grubb was promoted to his current position as Business and Technology Applications Specialist. In August, 2010, he was awarded a Master of Science degree in Computer Science from Appalachian State University.