WIRELESS DATA ACQUISITION FOR APIOLOGY APPLICATIONS

A Thesis
by
LUKE ALDRIDGE RICE

Submitted to the Graduate School
Appalachian State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

August 2013
Major Department: Computer Science

WIRELESS DATA ACQUISITION FOR APIOLOGY APPLICATIONS

A Thesis
by
LUKE ALDRIDGE RICE
August 2013

APPROVED BY:

E. Frank Barry
Chairperson, Thesis Committee

Rahman Tashakkori
Member, Thesis Committee

James Wilkes
Member, Thesis Committee

Edelma D. Huntley
Dean, Research and Graduate Studies

ABSTRACT

WIRELESS DATA ACQUISITION FOR APIOLOGY APPLICATIONS

Luke Aldridge Rice, B.S., Appalachian State University

M.S., Appalachian State University

Chairperson: E. Frank Barry

Colony Collapse Disorder (CCD), a disease affecting honey bee colonies, is a problem threatening the food security and economy of the entire world. Discovering the cause of CCD is particularly difficult because of the variety of colony locations and environmental variables. In addition, CCD instances do not tend to follow an easily recognizable pattern with respect to apiary conditions, which is exacerbated by the subjective nature of manual apiary data recording methods. Traditional monitoring methods are typically too expensive for wide-scale deployment and often require manual collection of the data, reducing the quantity of data available for analysis. A general wireless data acquisition system was designed to improve the quantity and quality of data and to explore general issues related to wireless data acquisition systems. The system was constructed using off-the-shelf-components to reduce cost. The acquisition system and data management tools were programmed using freely available tools and software. Beehive data are transmitted to the Internet wirelessly through the use of a cellular GSM modem. Results show that it is feasible to build an economical, general purpose wireless data acquisition system that can gather quality data for an Apiology application with similar capabilities to higher-cost contemporary systems.

ACKNOWLEDGEMENTS

This thesis would never have been possible without the support of my parents, Candace and Steve Rice. I would like to thank them both for their unconditional support throughout my entire education experience. In particular I would like to acknowledge the financial assistance I was able to receive as a result of my father's military service, without which I would have in all likelihood not made it to where I am today.

I would also like to acknowledge my brother Tilson, who was my partner in education from childhood all the way through our master's degrees. Very few people have had a roommate and friend like him to count on through the ups and downs of life.

I would also like to thank all of my teachers for always encouraging me to go further than I thought myself capable of. In particular I would like to thank Michael O'Kane from ABTECH for igniting my initial passion for programming in a way that I would have not received in a traditional university setting.

I would like to specifically acknowledge my professors Dr. Rahman Tashakkori and Dr. Barry Kurtz for their mentorship during my undergraduate education, without which I would not have been as successful and without whom I would have likely not pursued a graduate degree.

TABLE OF CONTENTS

Chapter 1 - Introduction

In 2006 the honey bees began disappearing in record numbers. Beekeepers in the United States began reporting increased winter beehive losses from 30% to 90%. Many beehives were nearly if not entirely empty, leaving little evidence of what caused the calamity. This type of event has come to be known as Colony Collapse Disorder, or CCD. CCD has become a major threat to the livelihoods of beekeepers all over the world. In addition, as much as one third of all of the food for humans and livestock requires pollination by insects, of which 80% is done by honey bees [1].

A large amount of research on CCD has taken place since 2006, but progress has been slow. There are many possible causes, from systemic pesticides to mites. Most of the available data about beehive conditions are gathered manually by humans and self-reported, both limiting the available amount of data and adding subjective factors.

There are other systems that suffer from similar complications in observation, from environmental systems to household systems. Automated data gathering techniques have the ability to improve both the quantity and quality of data about such systems. Advances in technology related to data acquisition have made automated monitoring cheaper and more effective; however, many off-the-shelf solutions that rely on commercially available devices are cost prohibitive, limiting the scale of deployment. With a problem as diverse as CCD, wide-scale deployment would increase the quantity of the available data, improving the chances of determining the cause or causes of the problem.

The purpose of this research was to design a custom wireless data acquisition system that provides similar functionality to more expensive commercial or research-oriented systems. While some parts of a beehive monitor system are specific to a beehive (such as sensor placement), many of the design and implementation features are general and could be applied to many similar systems. As a result, this research also explores general issues related to wireless data acquisition systems.

The resulting prototype Remote Hive Monitor System (RHMS) is capable of monitoring internal beehive temperature and humidity from an array of sensors, beehive weight, and exterior weather conditions in the area of temperature, humidity, and light intensity. These data are transmitted wirelessly to the web using a cellular modem. These acquired data are aggregated using a number of different storage and presentation techniques to allow for flexible access.

This thesis explores the chosen application in contemporary systems, discusses relevant technological background information, outlines the research methodology, provides detailed discussion of the system implementation, and presents results, and analysis of the recorded data.

Chapter 2 - Background

This chapter provides details about the problem domain for which this research was conducted, prior related research in this area, and technical background information on the solution domain.

2.1 Problem Domain

There are many potential applications for remote data acquisition. Many examples of remote data acquisition involve applying data acquisition techniques to systems that were previously monitored by hand or not monitored at all [2] [3]. These systems are characterized by difficulties in observation; They are often obscured, difficult to access, or are in a remote location. Observation of a remote system would be time consuming and expensive to conduct manually. Many examples of these systems are spread over wide geographical areas and are used when a large quantity of information is deemed necessary to solve a problem. In some cases, these systems may not be remote, but are nevertheless difficult to observe, such as the conditions of an attic or basement home.

In some cases, the quantity of the acquired data is of greater importance than the relative quality of each individual measurement. For example, water level monitoring of rivers and creeks in remote locations could be indicators of drought conditions. Being able to observe changes in the water levels of many creeks and rivers could be more important than having highly precise measurements of the water level of a few bodies of water. In addition, attributes of these systems are not expected to change rapidly, allowing sample rates on the order of minutes or hours. Some potential systems might only monitor a few characteristics such as wetlands water level measurement [4], while others might monitor many characteristics such as a remote weather station [5]. Being able to detect changes in these systems can alert scientists and researchers automatically to exceptional conditions, signaling that precise measurements or human observation are needed. Hence no manpower is required for observing these systems manually. An area of agricultural monitoring in which a growing amount of research is being conducted is honey bee monitoring.

Much of the recent research on honey bees seeks to find solutions to or explanations for Colony Collapse Disorder (CCD) [1]. This research involves studying many different aspects of honey bees. Some research focuses on behavioral and environmental characteristics, such as tracking honey bees when they leave the beehive, where they forage, and the types of flora that the bees come into contact with [6]. Other research seeks to explain CCD by studying the conditions within a beehive as a large collection of bees as opposed to studying the behavior of individual bees [7]. The research discussed in this thesis focuses on the acquisition of data about honey bees by monitoring the conditions of a beehive.

There are several of beehive characteristics that classify the system as remote, including its geographic location and the obscuration of the physical beehive. One challenge in observing a beehive is the diversity of different potential environments in which beehives are located. Beekeepers can be grouped into four categories: commercial, honey producers, hobbyist, and research beekeepers. Commercial beekeepers control a large percentage of the total hives within the honey bee industry. Companies like BZ Bodies [8] and Pollination Contracting [9] transport beehives across the country during the growing season for crop pollination. Honey producers group beehives into an apiary in areas ranging from backyards to remote areas, depending on foraging quality [10]. Many beekeepers do not have a large number of beehives and do not rely on honey production for their livelihood; these can be termed hobbyists. Hobbyists often have beehives near their homes but can have them in remote areas as well. Researchers studying beehives may observe hives located in urban or rural environments. This diversity in geographic locations complicates traditional monitoring techniques.

Current techniques in beekeeping and honey bee research involve the observation of beehive conditions through beehive inspections, which require the beekeeper to manually open the beehive and make sensory observations such as appearance (sight), sound, and smell [11]. Despite extensive projects to aggregate large amounts of beehive inspection data such as the BeeInformed Partnership [12], little concrete progress has been made to find the cause(s) of CCD. Data acquisition techniques have the capacity to improve the quantity of data available and improve the quality of the large amount of available subjective data [13].

2.2 Existing Work

There has been a variety of research in applying data acquisition techniques to the study of honey bees [14] [15] [16]. Three contemporary systems were researched as the basis for the RHMS. Each of these systems seeks to improve knowledge about the internal conditions of a beehive using different techniques and levels of automation. The first studied system is a research system that gathers a number of beehive statistics, including video data, the second is a hobbyist system known as HiveTool that is designed for hobbyists of any level of technological skill, and the third is a commercially available system called BeeWatch that is accessible to many types of beekeepers.

2.2.1 Beehive Monitoring with Optical Flow

The Maximilian Michels honey bee monitoring system is a research system that strives to find an explanation for CCD. The research observes a modified observation beehive in climate controlled conditions in a non-remote location with ready access to internet and power utilities. It is, therefore, not subject to the same environmental challenges addressed by the RHMS. As a non-remote system, the Michels system can make use of general purpose PC systems and peripherals. The Michels system and the RHMS measure similar hive characteristics, such as temperature, humidity, and weight. The Michels system also records video data of the hive through a glass window in the side of the observation hive [14]. A high-level organization of the Michels system can be seen in Figure 2-1.

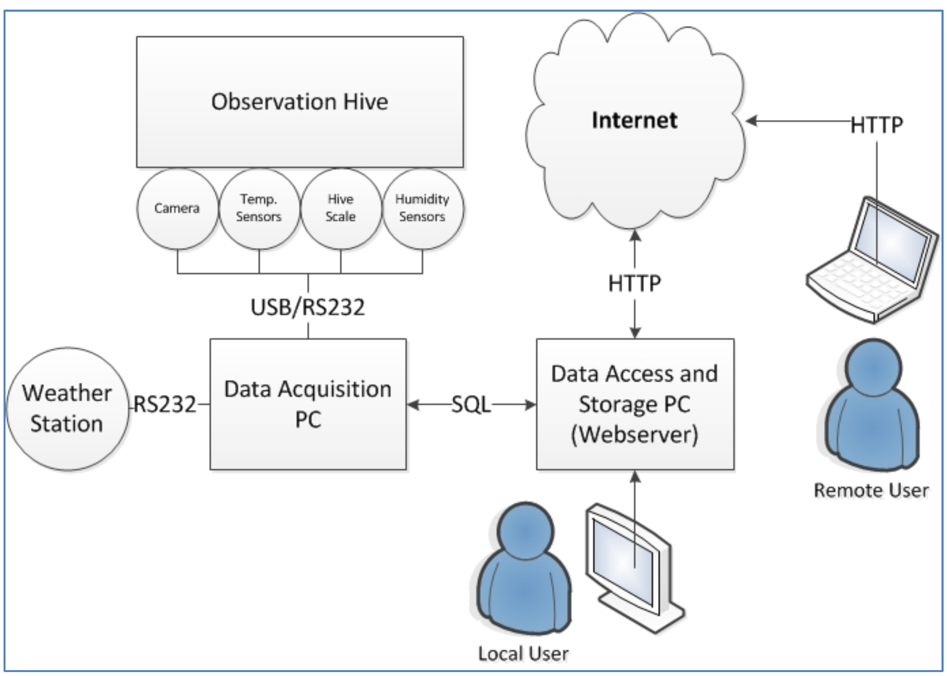

Figure 2-1 - Michels System Overview

The design of the Michels monitoring system is different from that of the RHMS and has advantages in some areas. Being in a non-remote, climate controlled environment, the system does not have to be concerned with environmental conditions. Additionally, the system is easy to access and perform maintenance on if necessary. A user may visit the hive to confirm data are being collected correctly. The measurement of optical flow using video data improves upon the diversity of data offered by the RHMS [14].

The Michels system has a number of disadvantages in operating in the environments in which the RHMS is to operate. The Michels system is composed of two PC systems for the acquisition of data and the aggregation and access of data, respectively. These are costly and power intensive systems that could not operate in a remote environment. As a research system, Michels' implementation could not be scaled to monitor many beehives and is restricted to areas where power and internet utilities are available. The Michels system utilizes PC components and off-the-shelf sensors which increase the price significantly. In addition, the climate controlled nature of the system, as well as light allowed by the translucent side of the observational beehive, introduce variables into the system that reduce the validity of honey bee behavioral information [14].

2.2.2 HiveTool

HiveTool is a project that is designed primarily for hobbyist beekeeping environments. The aims of the HiveTool project do not include finding the causes of CCD, but rather, gathering general information about the conditions of a beehive. HiveTool differs from other contemporary systems in that it describes a process to be implemented by other beekeepers rather than a complete system. Figure 2-2 shows a high level breakdown of the implemented example HiveTool system.

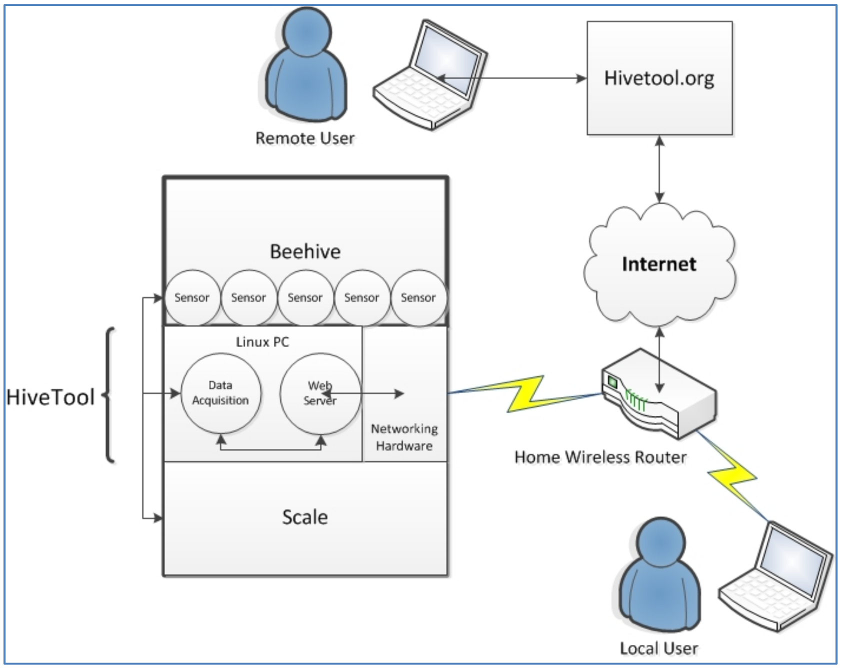

Figure 2-2 - HiveTool Overview

The example system detailed in the documentation is a collection of off-the-shelf sensors, a PC system, and consumer wireless technology. The core of the system is a Linux capable PC. The system is installed into a modified brood chamber on which the monitored beehive rests. This enclosure contains the PC, power supplies, and networking hardware. As a fully capable Linux computer, the HiveTool system is flexible. The example implementation includes a commercial scale and an array of temperature and humidity sensors. The PC system serves a dual purpose in both acquiring data and serving as a webserver through which the data is stored and accessed. The HiveTool documentation provides details on the process of constructing, installing, and configuring the system, along with the scripts and other software necessary. Each hive system locally hosts an individual webserver, while the HiveTool website acts as a hub for multiple implementations, providing links through which each implementation can be accessed. The HiveTool website acts as a hub for similar systems and the HiveTool site is hosted on a remote web server, but each site for individual implementations is hosted physically in a beehive [15]. This allows access over a local network as well as remote access using the IP aggregation system on http://hivetool.org.

The HiveTool system shares characteristics with both the Michels system and the RHMS. The core PC and off-the-shelf sensors make HiveTool a highly flexible and customizable system similar to that of the Michels system. The HiveTool documentation lists a vast array of sensors compatible with the system that are not implemented in the example system. An implementer could expand the system to monitor more attributes or multiple beehives in close proximity [15]. The HiveTool system has the advantage of flexibility over the RHMS but with the increased flexibility comes increased cost and power consumption. The HiveTool system would be more difficult to operate in remote locations.

2.2.3 BeeWatch

Among the capabilities and designs of the contemporary systems analyzed during this research, the BeeWatch system shares the most in common with the RHMS. BeeWatch is a commercial beehive data acquisition system that monitors a number of beehive characteristics such as a beehive’s weight and brood chamber temperature, as well as outdoor measurements with optional sensors such as a weather station and rain sensor [16]. BeeWatch uses similar hardware to the RHMS, making use of a common Global System for Mobile Communications (GSM) based cellular modem. Figure 2-3 shows a high level organization of the BeeWatch system. The BeeWatch hive scale is an integrated part of the system, as opposed to both Michels and the Hivetool systems in which the scales are independent components.

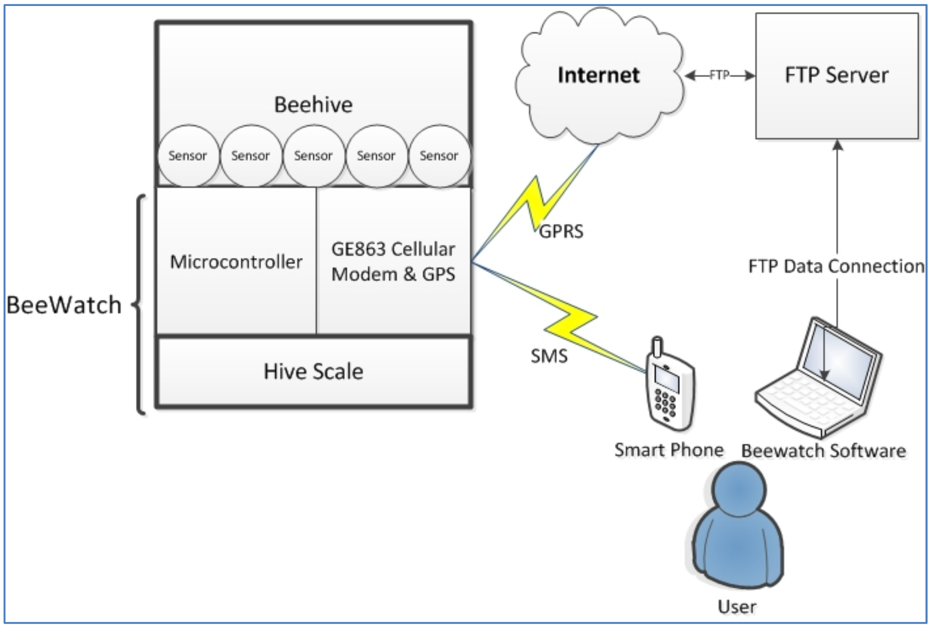

Figure 2-3 - BeeWatch Overview

The BeeWatch system has a wide array of configurations and parameters and allows for data to be transmitted wirelessly using the FTP protocol. Data and various alerts can be sent via SMS. Data from the system can be visualized in the software provided or the CSV data can be processed by an external system [16].

The main drawbacks of the BeeWatch system are cost and its proprietary features. The cost of the BeeWatch system is very high, making wide-scale beehive instrumentation unlikely, but beneficial for small sets of research beehives. The use of the GSM modem allows for geographic flexibility. The system uses proprietary sensors supplied by the manufacturer, making extension of the system difficult or impossible. Also, it is not possible to modify the source code or add modules [16].

2.3 Technological Background

The RHMS system operates within a number of emerging computing paradigms that are related to data acquisition. In addition there are many aspects of remote data acquisition that were studied as part of this research.

2.3.1 Paradigms

There are three paradigms specifically related to the research and development of the RHMS. They are Machine-to-Machine (M2M) Communication, the Internet of Things, and Wireless Sensor Networks.

2.3.1.1 Machine-to-Machine Communication

M2M refers to an emerging paradigm whereby machines can communicate information to other machines of similar capabilities. M2M has come to be used to define classifications of devices that facilitate this type of communication, such as cellular data modems [17]. One of the core components of the RHMS makes use of a cellular GSM modem that is marketed for M2M communication.

2.3.1.2 The Internet of Things

The Internet of Things (IoT) is an emerging paradigm where things, which are physical real-world objects, are the main content generators of the Internet. An IoT is compared to the current Internet in which human beings both generate and consume most information [17] [18]. This research shares attributes common to an IoT in that the RHMS models a physical system, in this case a beehive, and allows changes in the state of the beehive to be communicated automatically. An implementation of the IoT would be constructed using similar technologies to present in the RHMS as well as M2M techniques and technologies.

2.3.1.3 Wireless Sensor Networks

Wireless Sensor Networks (WSNs) represent another type of technology that works to gather sensor data in a distributed network. A WSN uses homogeneous nodes that can communicate with one another in a network [19]. WSN's can use M2M technology and techniques to communicate with each other and the Internet. WSNs could make up part of an eventual IoT. Many WSN's are implemented using commercially available devices called motes. Motes are very general-purpose wireless sensor devices that, through software, can be turned into a WSN [20]. WSNs built on motes can be highly customized but can be cost prohibitive for low-cost applications [20], reducing the potential scale of their deployment.

2.3.2 Technologies

The RHMS and the paradigms listed above make use of a number of basic technologies in the broad categories of Sensor Technology, Wireless Technology and Data Storage Technology.

2.3.2.1 Sensor Technology

Sensor technology is central to data acquisition. Sensors allow for the observation of physical quantities and their conversion into electrical signals. Selection of sensors for data acquisition must be based on a number of characteristics such as cost, accuracy, precision, and interface. These attributes determine the requirements of the data acquisition system and have a large effect on cost. Sensors can range from cheap RFID technology to expensive space-based remote sensing arrays [22]. Sensors come in an array of construction types, including resistive, capacitive, MEMS, and piezoelectric. Sensor packages can also be either analog or digital. In the case of analog, sensors are read by an analog-to-digital converter built into many microcontroller systems. Digital sensors contain analog-to-digital converters and produce a digital signal that must be sampled [23].

2.3.2.2 Wireless Technology

Wireless technology developments have been central to the development of remote acquisition systems, the IoT and WSNs. There are a number of standards and protocols that have driven technological advancements in the area of wireless communication. One of the most popular standards is the IEEE 802.11 standard, which drives Wi-Fi. An increasingly popular standard is IEEE 802.15.4, which serves as the basis for the Zigbee protocol and many others. IEEE 802.15.4 is the subject of much discussion in the low-power wireless community as a means for integrating low-power wireless devices into the TCP/IP network stack by the implementation of a new network protocol. One such effort is known as 6LoWPAN, which would allow the IPv6 protocol to operate on IEEE 802.15.4 wireless hardware for low-power devices like sensor networks [17].

2.3.2.3 Data Storage Technology

Data storage for distributed sensor networks incurs challenges not faced by many traditional data applications. The heterogeneous nature of sensor data and the lack of rigid schemas have driven the need for flexible, scalable storage systems. Traditional relational database storage techniques are not well suited to the unstructured, changing nature of sensor data. These storage challenges are discussed heavily in the context of the IoT but are also relevant to remote sensing and remote data acquisition systems [24].

Chapter 3 - Methodology

The research methodology that resulted in the RHMS took place in five stages: Requirements Analysis, System Architecture Selection, Component Selection, System Implementation, and Data Acquisition.

3.1 Requirements Analysis

Before the RHMS system could be designed or implemented, a concrete set of system requirements and constraints was gathered. To determine the requirements of the system informal interviews were conducted with beekeepers from various backgrounds. Represented were commercial beekeepers, researchers studying honey bees related to CCD, honey producers, and hobbyists.

Through this brainstorming session, sets of functional and non-functional requirements were generated.

3.1.1 Functional Requirements

The functional requirements identified through the informal interviews related to the overall functions that the system must perform, such as monitoring beehive attributes and transmitting the measurements to the web. The functional requirements determined based on the interviews were as follows:

The System must be able to monitor the following beehive characteristics:

- Weight
- Internal Hive Temperature
- Internal Hive Humidity
- External Temperature
- External Humidity
- External Light Intensity

- The system must transmit acquired data to the web where it can be accessed
- The system must allow for the sending of alerts when exceptional conditions are detected

3.1.2 Non-Functional Requirements

The non-functional requirements of the system represent a class of requirements that do not directly relate to a function the system must perform but are important and must be realized within the system. These non-functional requirements are represented by constraints placed on the system by the operation environment or as non-functional feature requests by the potential user(s). Non-functional features of the system are more difficult to quantify and must be explained in more detail than functional requirements. The main non-functional requirement is system cost, with additional constraints of sampling frequency, power consumption, and environmental flexibility.

3.1.2.1 System Cost

System cost is affected by almost all attributes of the system making it difficult, if not impossible, to set a target price in a research prototype. However, cost must be considered in most component and software decisions and might impose a review of components and software if choices in these areas prove more expensive than anticipated.

The cost of the system affects its ability to be scaled to monitor a large number of beehives. The benefits to large scale monitoring were discussed in Chapter 2.

3.1.2.2 Sampling Frequency

Sampling frequency refers to how often the system samples its array of sensors. Sampling frequency affects power consumption and storage/data plan needs. If attributes are not sampled often enough, significant events might not be detected, but if samples are taken too often it can increase cost by increasing data plan needs or storage space requirements. Increased sample rate increases the ratio of the time the system is active, which can increase power consumption.

3.1.2.3 Power Consumption

As a remote system that is likely to not be easily accessible, power consumption is an important constraint. The length of time the system can run without having batteries replaced affects where the system can be used geographically and how much maintenance the system requires. Power consumption must be considered when determining functional requirements and during all hardware component choices. Given that power consumption is affected by so many other parameters of the system, it is difficult to predict consumption but if optimization cannot yield suitable system runtime, lower power components might have to be considered. Building a prototype system like the RHMS makes it possible to identify areas of power inefficiency for improvements in future versions.

3.1.2.4 Environmental Flexibility

One of the main constraints placed on the system comes from the variety of geographic locations in which the system must be able to operate. This constraint is a part of the overall environmental flexibility of honey bees and the variety of markets discussed in Chapter 2. These environments vary in terms of topography, proximity to accessible internet and power sources, temperature ranges, and weather conditions. These constraints had an effect on many of the component choices of the system. Geographic flexibility had the largest effect on the uplink hardware used, which is described in detail in Chapter 4.

3.2 System Architecture Selection

After Requirements Analysis, a system architecture was selected that could fulfill the determined requirements. The selection of the system architecture identified the main subsystems that would make up the RHMS. The high level system organization can be seen in Figure 3-1. The main subsystem components are represented by rectangles and are organized into three subsystems: the bee hive node (Node), supervisory node (Supervisor), and the Web System. This three tiered approach allows the most costly hardware to be concentrated in the Supervisor, thereby minimizing the cost of an individual Node and increasing the likelihood of wide-scale deployment. Power efficiency is maximized by placing the most computationally intensive processes on the web system, allowing the Supervisor and Node systems to minimize power consumption.

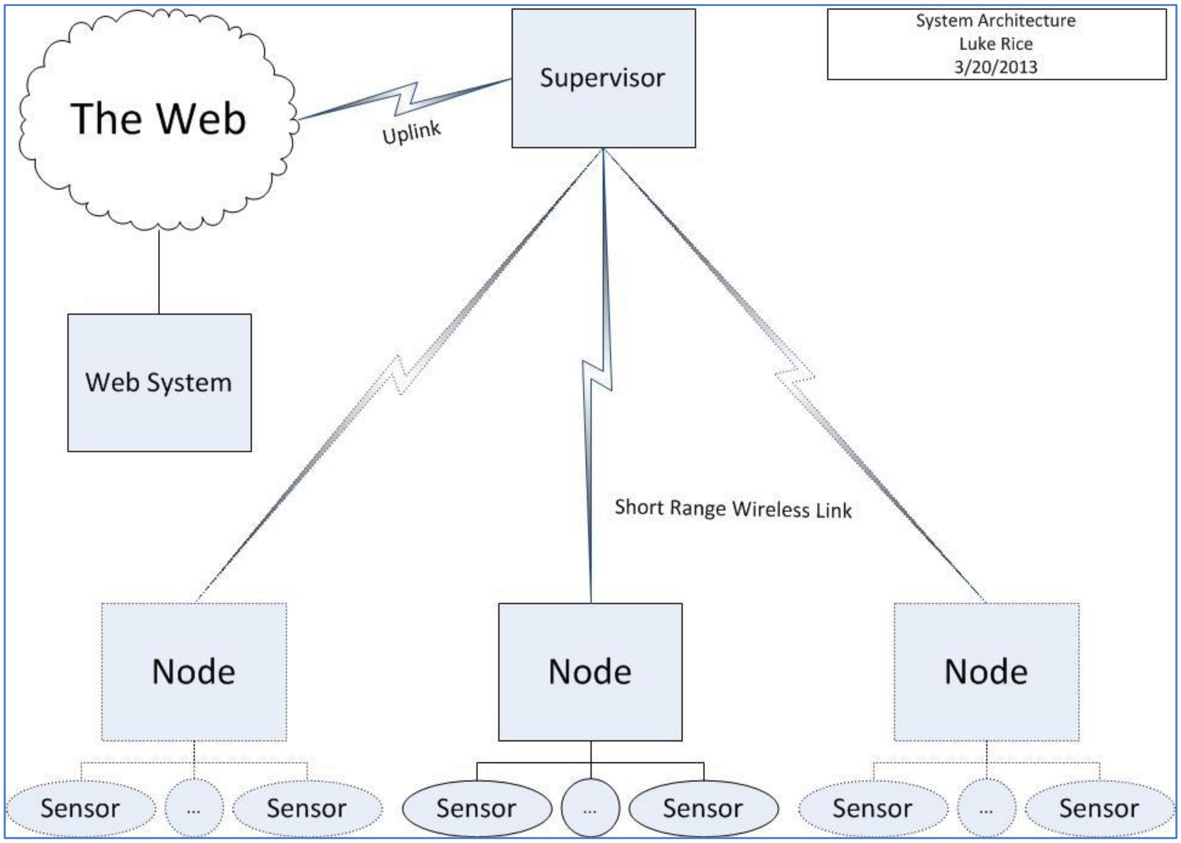

Figure 3-1 – System Architecture

3.2.1 Node

The Node in the RHMS models and monitors a single beehive. The Node is made up of sensors and processing hardware needed to acquire the data and the wireless hardware needed to transmit the data it to the Supervisor node. The Node samples the array of sensors on an interval defined in the software before transmitting the data to the Supervisor.

3.2.2 Supervisor

The Supervisor takes sensor input from the Nodes and transmits these data to the Web System through a piece of uplink hardware. For the RHMS prototype, the Supervisor is constructed of two wireless systems and does not run any intelligent software. This was done as a result of only having a single Node constructed for testing. The uplink hardware has the capability to run programs, and future developments for a more robust Supervisor are detailed in 7.1.1.

3.2.3 Web System

The Web System encompasses all the software needed to receive data from the Supervisor, process and store the data, and make the data available to the end user(s). This subsystem is made up of a number of different software and data storage components. The Web System comprises all of the software that converts sensor data to final units, allowing modifications to be made to conversion parameters and constants, allowing changes without requiring recompilation of embedded software at the Supervisor or Node.

3.3 Component Selection

After selecting the overall system architecture, hardware and software components to implement the architecture were chosen. Component selection is an iterative process wherein each subsystem is realized by a set of components to satisfy the operations of the specific subsystem. Component selection is mainly relevant to the Supervisor and Node systems; the Web System has greater flexibility and fewer constraints placed on it.

Component selection is done with awareness of the system constraints. The main constraints to consider in any hardware component choices are (in order of importance):

1. Cost/Performance Tradeoff
2. Power consumption
3. Environmental reliability

Hardware components come in a wide array of configurations with a number of different interface protocols. During component selection, extra care must be taken to ensure that components are compatible. This can cause a cascade effect whereby a component is chosen that has one type of interface that affects all the other components that can be chosen later, possibly affecting the constraints listed above. These issues make component selection an iterative process.

3.4 System Implementation

After suitable components were selected, system implementation was conducted. Each subsystem was implemented incrementally and iteratively and each subsystem was unit tested.

Following unit testing, the subsystems were combined and integration testing was conducted. This involved first bench-top testing with components assembled loosely in such a way that hardware and software modifications could be made easily. The Node sensor components were integrated into an empty beehive, and the Node hardware was installed in a waterproof enclosure for the final testing.

3.5 Data Acquisition

Following system implementation, the data acquisition phase of the research was initiated. The goal of the data acquisition phase was to gather data about the application domain to analyze the functioning of the system itself and to determine the quality of the acquired data for future analysis. The data acquisition process took place in three stages: Indoor Testing, Outdoor Calibration Testing, and Live Data Acquisition.

3.5.1 Indoor Testing

The first phase of data acquisition involved running the fully implemented system indoors in reasonably controlled (office) environment. This was done to ensure that the system was stable enough for installation in a live beehive and that the system would not require significant maintenance when gathering real data. This phase also ensured that the sensor readings were stable enough to be able to produce reliable results in live conditions. Errors detected during this phase were easily addressed due to the proximity of the prototype to the development tools.

3.5.2 Outdoor Calibration Testing

After ensuring that the system was stable enough to be moved from the development setting, the fully assembled system was installed outdoors for calibration testing. This phase established the effects of the uncontrolled environment on the system. Calibration weights were placed on the empty beehive to gather sufficient data for calibration of the hivescale based on temperature changes.

3.5.3 Live Data Acquisition

After enough calibration data were gathered, a honey bee colony was installed in the empty beehive and live data acquisition was conducted.

Chapter 5 will list the results of the data acquisition and Chapter 6 will contain an analysis of the acquired data.

Chapter 4 - System Implementation

The system architecture of the RHMS was discussed in Chapter 3. This chapter details the Implementation and Physical Construction of the final system.

4.1 Implementation

Each subsystem in the RHMS was implemented independently before defining the software interfaces that allow the subsystems to communicate. The Node subsystem was implemented first given that the majority of the hardware configuration and the most complex software were implemented on the Node. The Supervisor was then assembled and tested so the Node could communicate with the web. The Web System was implemented last as it was the easiest to modify. Minor modifications were made to each system after integration as bugs were found and assumptions were tested.

4.1.1 Node

The Node subsystem is a combination of hardware, software, and device configuration and constituted the bulk of this research. Figure 4-1shows a high level block diagram of the Node and its constituent components. Detailed schematics of the Node including all connections and device pin numbers can be seen in Appendix A. Datasheets for all Node hardware components can be found in Appendix A Table A-1. The implementation of the Node subsystem will be described in terms of the hardware components used and the software and configurations needed to utilize these components.

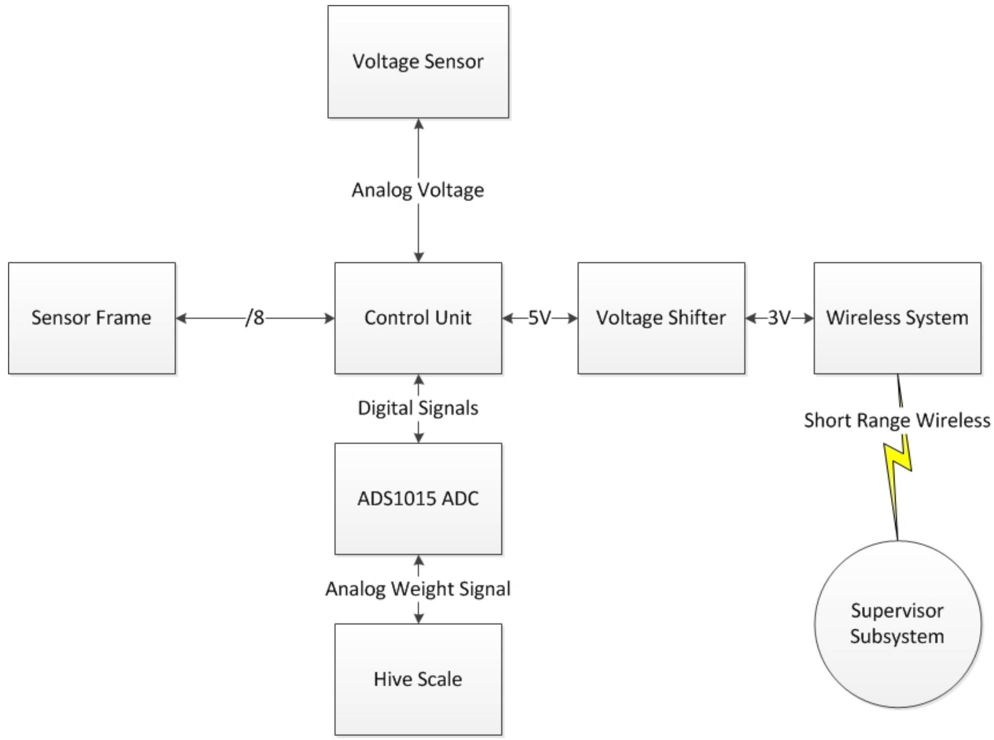

Figure 4-1 – Node Organization

Both the Sensor Frame and Hivescale components reside at the beehive and are separate physically from the rest of the components. The Sensor Frame contains the majority of the sensor components, including internal temperature and humidity as well as external weather conditions of light intensity, temperature, and humidity.

4.1.1.1 Hardware Components

The Node system is made up of three groups of hardware: the Control Unit, the Sensors, and the Wireless System. The full list of hardware components and their associated datasheets are listed in Appendix A.

4.1.1.1.1 Control Unit

The Control Unit component of the Node is the Atmega169 microcontroller, combined into a prototyping board known as the AVR Butterfly. The AVR Butterfly extends the capabilities of the Atmega169 microcontroller with features such as a bootloader allowing for simple serial programming, a Real Time Clock (RTC), speaker, and Liquid Crystal Display (LCD) as well as port headers for attaching I/O connections. The Atmega169 is a low-power microcontroller from the AVR family of microcontrollers. It has a wide array of I/O capabilities suitable for the RHMS, including a serial UART and an array of analog and digital GPIO ports. In addition, the Atmega169 platform was a familiar platform with previously developed code for some basic sensors and I/O functionality. The use of an AVR family microcontroller provides upgradability or miniaturization capacity for a potential production version of the RHMS. Datasheets 1 and 2 in Appendix A give detailed descriptions of the AVR Butterfly and datasheet 3 details the design and use of the Atmega169.

4.1.1.1.2 Sensors

To satisfy the functional requirements of domain attributes to be monitored, a variety of sensors was selected. Diagnostic measurements in the form of battery voltage and the internal temperature of the node enclosure were added for system testing.

4.1.1.1.2.1 Temperature

The RHMS includes two types of digital temperature sensor, the DS18B20 and RHT03 sensors. As can be seen in datasheet 4, the DS18B20 can measure temperatures from -55°C to +125°C with an accuracy of ±0.5°C when temperatures are between -10°C and +85°C. The DS18B20 has a programmable resolution of up to 0.0625°C when used at the full 12 bit resolution. The DS18B20 can communicate over a shared data line, allowing many DS18B20 sensors to be connected without increasing the I/O requirements on the control unit. The RHT03 is another digital temperature sensor with similar capabilities to the

DS18B20. The RHT03 has a range of -40 to +80°C, accuracy of ±0.5°C and resolution of 0.1°C, as can be seen in datasheet 5. The RHT03 sensor does not have the ability to use a shared data line, requiring an I/O pin for each RHT03 sensor used. These digital sensors are not the most cost effective, but are easy to use and require less calibration than analog sensors, making them suitable for a research context. A commercial system utilizing many temperature sensors would require further cost considerations of the number of digital temperature sensors used. See datasheets 4 and 5 for detailed information on the DS18B20 and RHT03 respectively.

In addition to the digital temperature sensors used to monitor the beehive, the AVR Butterfly contains an analog Negative Temperature Coefficient (NTC) thermistor temperature sensor that is used to track the temperature of the hardware component enclosure. The thermistor can measure temperatures between -10°C and +60°C but with an accuracy of only ±1°C. This sensor is measured using an analog to digital converter (ADC). The ADC available on the AVR Butterfly has a resolution of 10 bits, giving the thermistor a resolution of 70.0°C/1024, or 0.068°C. The details of this sensor and the conversion calculation can be seen in datasheet 2.

4.1.1.1.2.2 Relative Humidity

The humidity sensors used in the RHMS are the RHT03 sensor. The RHT03 doubles as a temperature and relative humidity sensor. The RHT03 can measure relative humidity to within 2% accuracy with a resolution of 0.1%. Datasheet 5 contains more detailed information on the relative humidity measurement capabilities of the RHT03.

4.1.1.1.2.3 Hive Weight

The scale used to measure beehive weight is a custom system designed by Wayne Esaias [13], a NASA engineer well known for his research into honey bees and their relation to CCD and climate change. It is capable of resolutions of 0.086 lb resolution with an approximately 350 lb maximum weight as used in the RHMS. The beehive scale is accessed using an ADS1015 12 bit Analog to Digital Controller (ADC) that is not a part of the AVR Butterfly system. The ADS1015 can sample up to four analog inputs and is accessible by an industry-standard I2C interface. The details of these components can be seen in datasheets 6 and 7.

4.1.1.1.2.4 Light Intensity

The measurement of light intensity makes use of a very simple sensor known as a Light Dependent Resistor, or LDR. It is also known as a photoresistor. It operates like a typical resistor or potentiometer but its internal resistance is dependent on the light intensity. It is operated using a simple voltage divider circuit, seen in Appendix A, Figure A-2. Datasheet 8 shows the electrical details of the LDR.

4.1.1.1.2.5 Battery Voltage

Battery voltage is monitored through a simple voltage divider circuit that scales the 12V nominal battery pack to 0-5V for sampling by the microcontroller’s analog to digital converter (ADC). The circuit can be seen in Appendix A, Figure A-1. The monitoring of battery voltage allows for diagnostic analysis of system runtime, temperature effects on battery performance, and prediction of system shutdown for battery replacement schedules.

4.1.1.1.3 Short Range Wireless

The short range wireless device utilized is the Xbee Pro. The Xbee Pro is an 802.15.4 compliant device that has a possible outdoor line of sight (LOS) range of up to a mile. It has low-power states, range, and data rates suitable for our system. In addition, the Xbee Pro has "virtual-wire" capabilities that allow for digital I/O signals to be passed wirelessly between two devices. This allows for wireless manipulation of programming, reset and run pins on the AVR Butterfly. The Xbee family of low-power wireless systems shares a common physical interface and command set, allowing for interchangeable RF power, antenna types and frequencies. Datasheet 9 contains the full operational manual and command set for the Xbee Pro.

4.1.1.2 Software and Configurations

The software of the Node subsystem resides on the Control Unit (Atmega169) and is composed of driver software, kernel software and library routines provided by avr-libc libraries [25]. In addition to system software there are also configuration settings for the Xbee pro wireless component of the Node.

4.1.1.2.1 Kernel Software

The kernel software component of the Node contains the software necessary to initialize system hardware components such as the Real Time Clock (RTC) and System Timers. The kernel provides a simple signal interface that allows drivers to access the system timer to recover from error states in which the driver code may become stuck in an interrupt service routine, never returning control to the kernel. In addition, the kernel contains the Kernel Control Loop that controls the system startup and data acquisition algorithm. The location of the source code for the kernel software can be found in Appendix A. The state machine that drives the kernel control loop can be seen in Figure 4-2. During system startup, system components are configured during the Init stage. After initialization, the system alternates between sensor data acquisition (Data Gathering), transmitting data (Open Connection), and entering the Low Power sleep state. Data gathering and transmission are separated into multiple states to allow the transmission (wireless) hardware to be enabled for the shortest possible time. It is possible there are sensor errors during the Gather Data state. When data have been gathered correctly from all the sensors, the wireless hardware is enabled and an attempt to transmit the data is made. The system ideally and in practice spends the majority of running time in the low power sleep state.

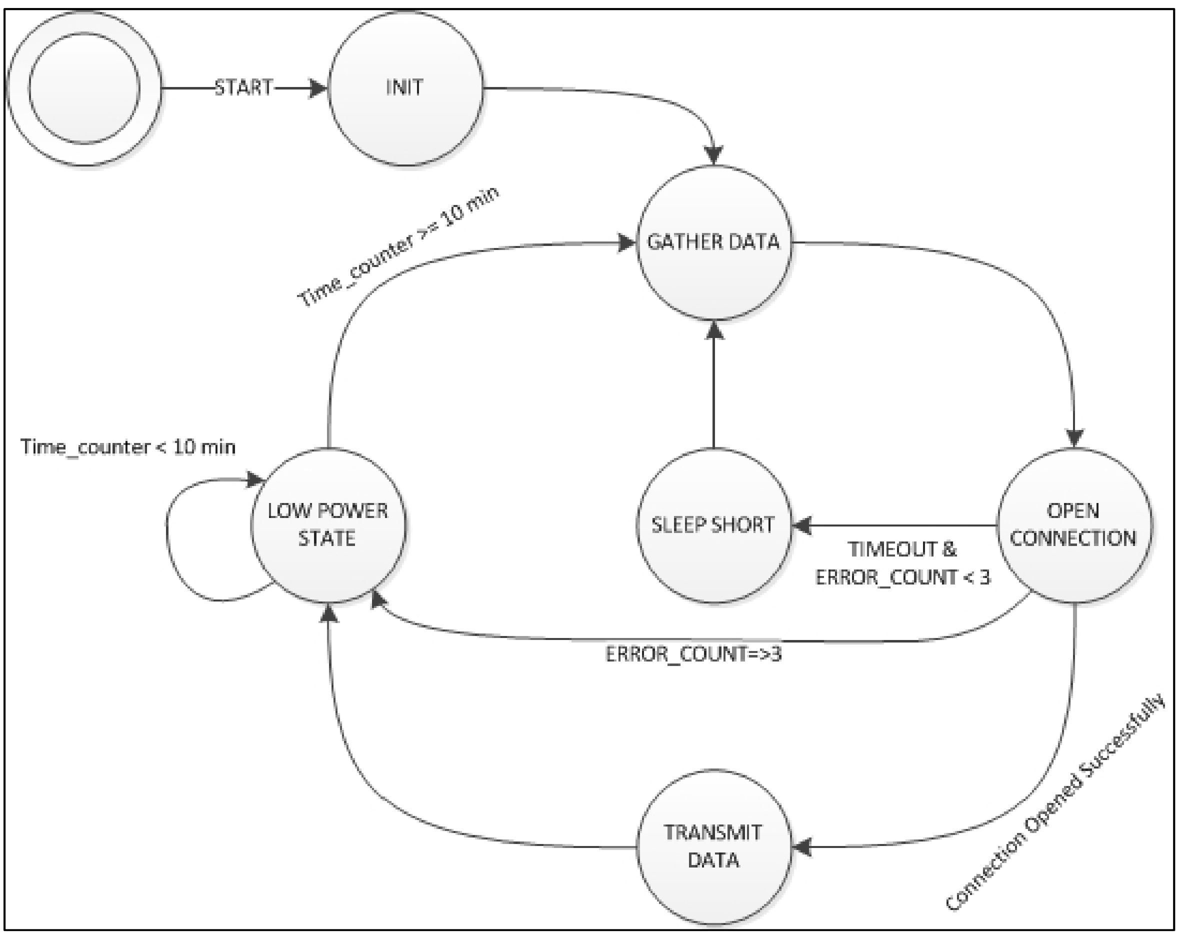

Figure 4-2 – Kernel Control Loop State Machine

4.1.1.2.2 System Drivers

The system drivers allow for the abstraction of complex parts of the system such as serial communication and digital sensor data acquisition from the main kernel. This allows the kernel to be as minimal and simple as possible, allowing the addition of new components and sensors to the system to be conducted with minimal modification of the kernel. System drivers may contain interrupt routines, data buffers, and interface functions necessary for the kernel to communicate with the components.

The drivers used in the RHMS Node and the location of their source code are listed Appendix A. Drivers are used for serial (UART) communication, hivescale access (through the ADS1015 ADC), interfacing with the GE865 cellular modem, and for acquisition of data from the DS18B20 and RHT03 sensors.

4.1.1.3 Device Configuration

The only device configuration required for the Node subsystem is necessary for the use of the Xbee Pro short-range wireless device. The configuration pairs the Xbee to the Xbee devices within its network and configures the network topology. The network topology for the single-node prototype system is point-to-point but a future multi-node system would be a star configuration with the Supervisor as the root. Both of these configurations are detailed in datasheet 6. In addition to network topology, the configuration settings also control virtual-wire management, transmission power, sleep state management, and serial port configurations. Device configurations are controlled and accessed through the use of the built-in AT command interface. The AT command set is a standard command language using ASCII character strings and is used for many M2M devices. These configuration settings and commands are shown in Table A-3 in Appendix A.

4.1.2 Supervisor

The Supervisor subsystem is a combination of a short-range wireless device for supervisor to node communication and an uplink device for communication with the web. The internal makeup of the Supervisor can be seen in Figure 4-3. The Supervisor in the prototype system is made up of two components, the Uplink hardware and the Short Range Wireless hardware. These pieces of hardware allow the Node to access the web without each node having to contain the Uplink hardware, which makes up a significant portion of the system cost. In the prototype system, there is no software or processing and storage operating at the Supervisor level, due to the fact that the Supervisor is not fully implemented for this prototype. A fully implemented Supervisor would involve supervisor-level software, as discussed in Chapter 7.

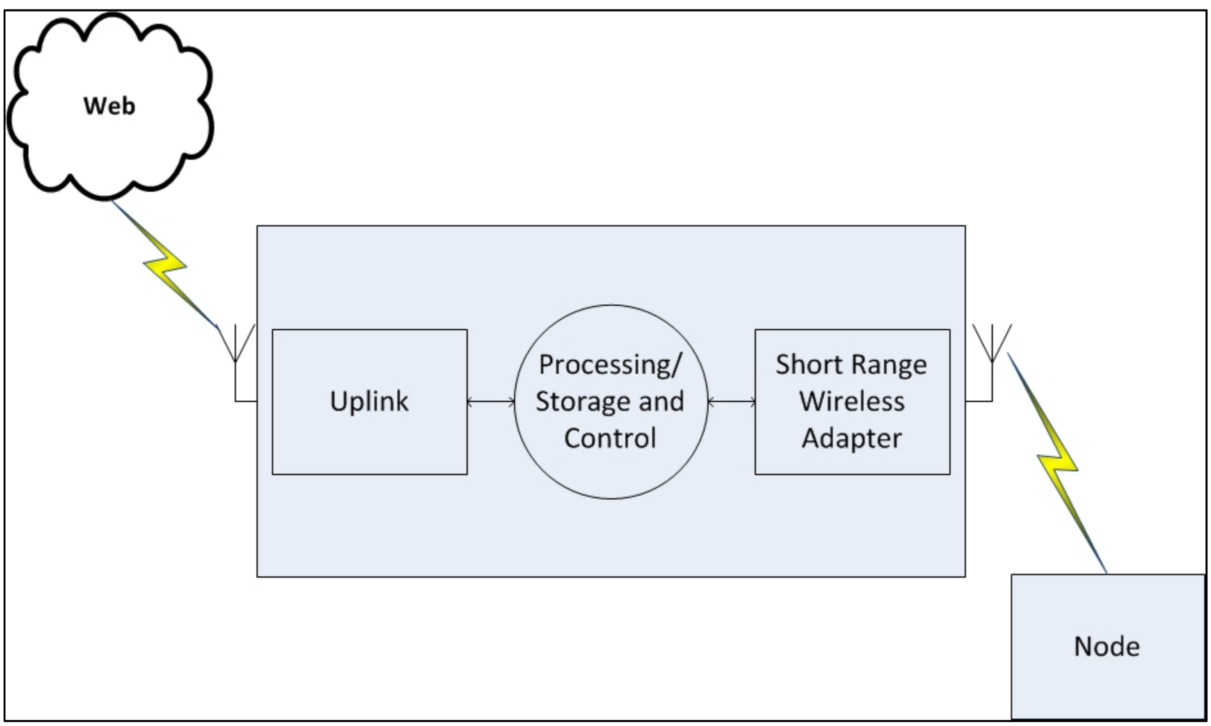

Figure 4-3 - Supervisor Internal Organization

4.1.2.1 Uplink

The uplink hardware used in the RHMS prototype is the Telit GE865 GSM modem. The GE865 is capable of data, voice, and SMS. For data communications, the GE865 uses the General Packet Radio Service (GPRS) protocol. GPRS has a maximum upload bandwidth of approximately 40 kilobits/second, which, while slow compared to modern 3/4G networks, is adequate for a system like the RHMS. Appendix B Table B-1 contains the datasheets for all Supervisor hardware. See datasheet 1 in for the GE865 user guide and datasheet 2 for the command set reference and 3 for the hardware reference. Table B-2 in Appendix B shows the configuration settings needed to operate the modem. The modem is accessed from the Node, whose source code is discussed Appendix A.

4.1.2.2 Short Range Wireless

The short range wireless device used in the supervisor is the same as that used in the Node, the Xbee Pro. The Xbee Pro used at the supervisor is able to transmit to and receive data from any number of other Xbee devices (at individual Nodes), but the Nodes would only communicate with the supervisor Xbee device using point-to-multi-point configuration. For the RHMS prototype, only point-to-point communication is required. The main difference in configuration of the Supervisor Xbee device is its virtual-wire configuration. Four virtual-wires are used in the RHMS for suspending, programming, resetting and executing the system. These capabilities are used for debugging and diagnostics, though some method of remote reprogramming would be desired in a production system. The configuration information can be seen in Table B-3 in Appendix B. The virtual-wire connections can be seen in the system schematic in Appendix B.

4.1.3 Web System

The Web System in the RHMS is implemented as three layers: Middleware, Processing, and Storage and Access. All of the software for the Web System resides on a single server for the RHMS prototype but each component communicates using standard TCP/IP protocols allowing the software to be distributed across multiple machines. The Web System hierarchy can be seen in Figure 4-4. The Web System is made up entirely of software. The locations for the source code for the components of the Web System can be seen in Appendix C.

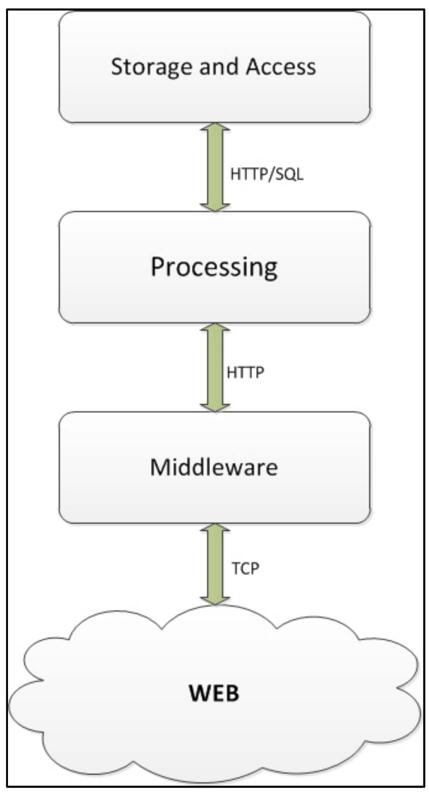

Figure 4-4 - Web System Hierarchy

4.1.3.1 *Middleware*

The middleware server takes the initial connection from the Supervisor node (cellular modem) and accepts the data. The middleware server is implemented using the Java programming language. The server spawns a thread on an incoming connection from the Supervisor. This was because in some cases it can take longer than a minute to receive a response from the processing script. Short sample rates could result in multiple simultaneous requests. The middleware server logs the request, checks it for validity by ensuring that it is CSV data, and forwards the data to a PHP processing script using a GET request. The use of a middleware server between the cellular modem and the processing script allows the data transferred by the cell modem to be minimal (a single CSV string as opposed to a HTTP GET or POST request with headers) and avoids an HTTP response that would otherwise get sent back to the cellular modem from the processing script, tripling in some cases the total data transfer for a single sample. The flow of control in the middleware server can be seen in Figure 4-5. The middleware server also allows for the logging of errors whose cause cannot be detected otherwise. Errors are discussed in Chapter 5.

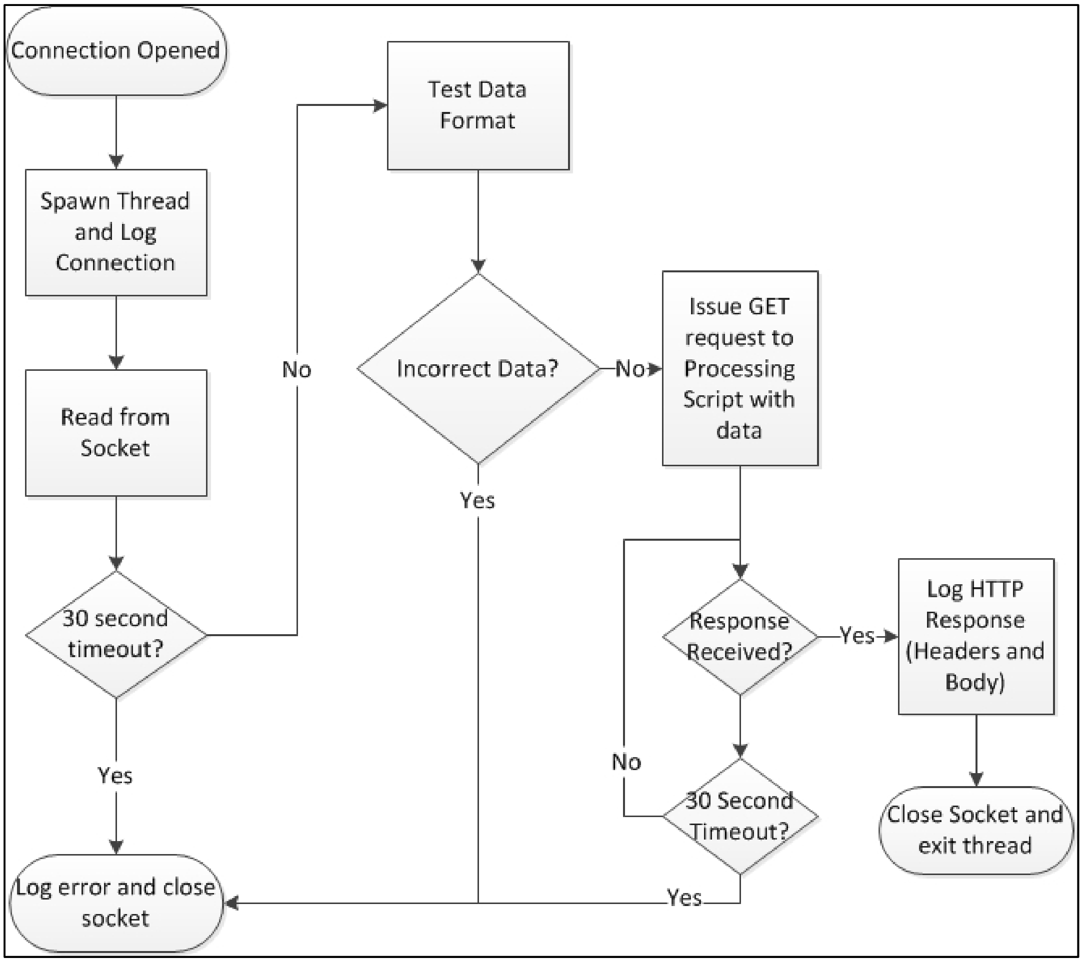

Figure 4-5 - Middleware Control Flow

4.1.3.2 Processing

The Processing component of the Web System is implemented using PHP. The PHP processing script is the destination of the GET request from the middleware server. The PHP script logs the full request in a MYSQL database before processing the data. The script control flow can be seen in Figure 4-6. The processing script stores the raw sensor data in a table in a MYSQL database before processing the sensor data and submitting it to a free IoT data aggregation and visualization too known as COSM. The purpose of storing data in the raw format (that is, unprocessed and in unit-less integer format) is to maintain the capability to re-process the data with different constants at a future date. When a sufficient amount of data is collected, statistical analysis might allow for more accurate conversions that require the original values. This is particularly important with the beehive scale which is subject to temperature affects that are counteracted with calibrations. These calibrations depend on both the temperature magnitude as well as the slope (rate of change) of temperature. Table 4-1 shows the calculations used to process the raw sensor data. In this table, raw refers to unprocessed data and value refers to the processed data in the correct units.

Table 4-1 - Sensor Processing Calculations

Sensor	**Calculation**	**Notes**
RHT03 (temperature and humidity	Value=raw/10.0	Must maintain sign
DS18B20	Value=raw*0.0625	12 bit resolution is in 0.0625°C increments, must maintain sign
Hive Scale	If raw > 0x800 then value = (raw-0x8f8)*0.089 Else value=(raw+0x708)*0.089	Weight of 0 outputs 0x800 (max negative), highest positive weight outputs 0x7ff. Empty scale weight is 0x8f8.
Internal Temperature (NTC Thermistor)	See 0datasheet 2 Figure 3-13	
Battery Voltage	Value=(raw/1024)*5.0/0.33637	ADC has 1024 steps (10 bits) and a full scale voltage of 5V. Ratio of input to output voltage was calcluated to be 0.33637.

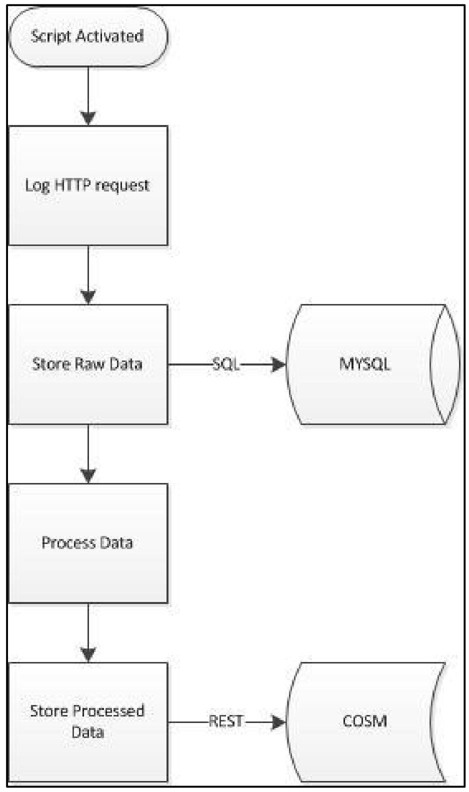

Figure 4-6 - Processing Control Flow

4.1.3.3 Storage and Access

Due to the variety of potential users requiring access to gathered data a variety of storage and access systems were utilized. These systems were a Relational Database, a free storage service known as COSM (note: COSM was renamed Xively during the writing of this thesis) [26] and a custom Access and Graphing script. Figure 4-7 - Storage System Access gives an overview of the different storage system components and potential usage connections based on the user of the system.

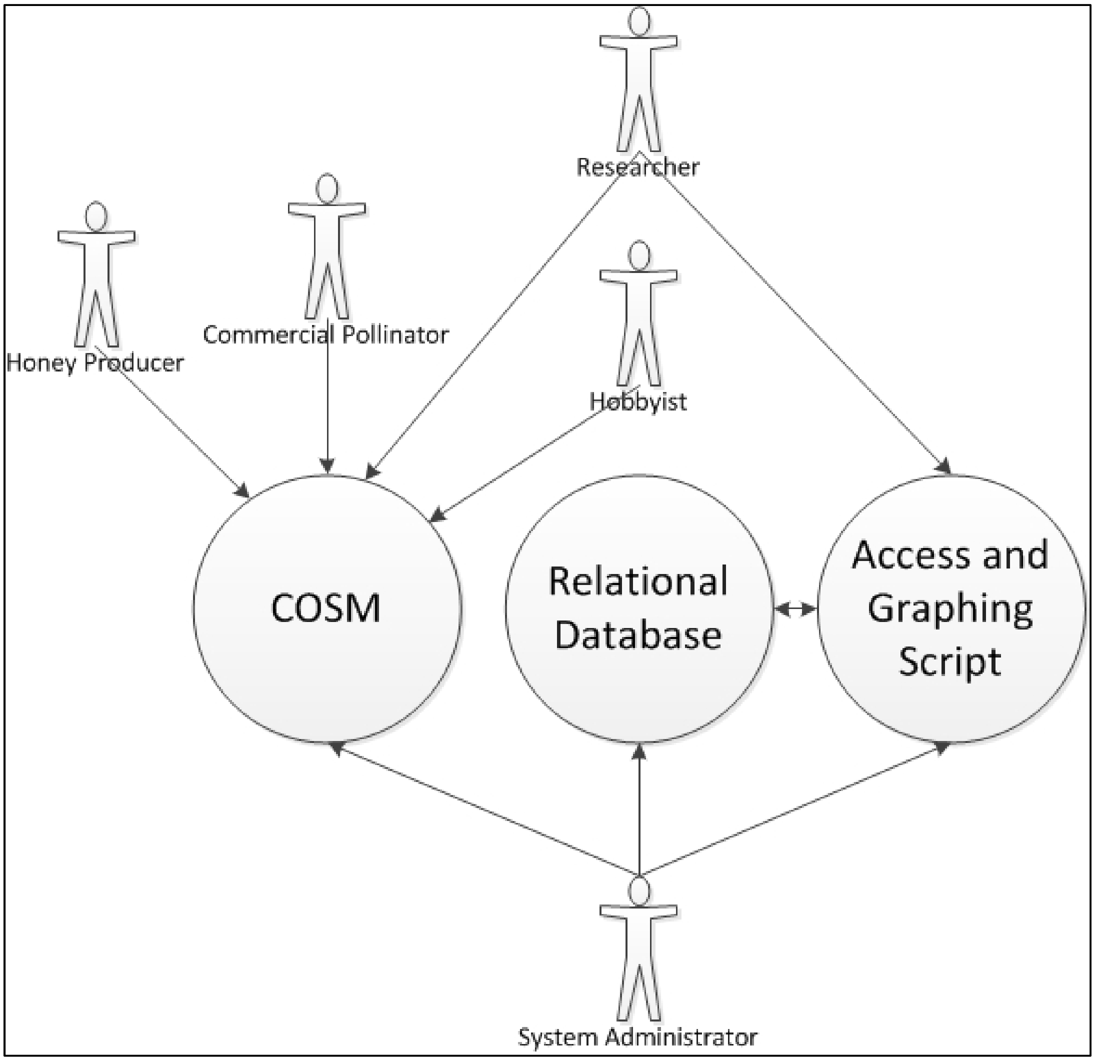

Figure 4-7 - Storage System Access

4.1.3.3.1 Relational Database

The relational database software used was the open source database MySQL. MySQL was chosen due to its ubiquity and ease of use with PHP. The challenges to using relational databases were discussed briefly in 2.3.2.3. Due to these challenges, the relational database was structured as simply as possible to accommodate unforeseen changes in sensor data. The data is stored in a single table named Sensor_data whose fields can be seen in Table 4-2. A sample set of data from a single sensor reading can be seen in Table 4-3. The values in the

Value field depend entirely on the sensor itself and have undergone no conversions and therefore have no units to track. The use of a loosely-structured schema allows sensors to be added or removed from the system without requiring modifications to the database. As can be seen from Figure 4-7 - Storage System Access, system administrators would typically be the only users with access to the relational database.

Table 4-2 - Sensor_data Table Structure

Field Number	Field Name	Field Type
1	Sensor_name	Varchar(255)
2	Datetime	Date
3	Value	Int

Table 4-3-Sample Data

Sensor_name	Datetime	Value
battery_voltage	2013-05-29 14:53:49	746
DS18B20_3000000349A7B428	2013-05-29 14:53:49	457
DS18B20_C80000034999A328	2013-05-29 14:53:49	446
DS18B20_E10000034996C428	2013-05-29 14:53:49	470
DS18B20_EC00000349A49B28	2013-05-29 14:53:49	452
Hivescale	2013-05-29 14:53:49	2794
internal_temp	2013-05-29 14:53:49	502
light_sensor	2013-05-29 14:53:49	925
outdoor_humid	2013-05-29 14:53:49	576
outdoor_temp	2013-05-29 14:53:49	260
ths1_humid	2013-05-29 14:53:49	563
ths1_temp	2013-05-29 14:53:49	316
ths2_humid	2013-05-29 14:53:49	606
ths2_temp	2013-05-29 14:53:49	313
Uptime	2013-05-29 14:53:49	889

4.1.3.3.2 COSM

COSM was a free data storage service designed to be a data repository for IoT data (the IoT is discussed in 2.3.1). COSM had built in visualization tools that allowed for the observation of incoming data during the testing phases of the system. COSM was accessed through a Representational Stateless Transfer (REST) based API that simplifies access across different systems. The COSM service was renamed to Xively during the writing of this thesis and some of the visualization tools were deprecated. Xively maintains a legacy API allowing the RHMS to continue to function. Xively also introduced a number of new debugging and testing mechanisms that were not fully researched. The remainder of this section will continue to refer to Xively as COSM, given that specific technical details relevant to COSM may only be relevant to "legacy" feeds in Xively. COSM and Xively also provide trigger functions to allow custom alerts when different conditions were met. These conditions ranged from sensor value thresholds to the feed becoming inactive due to no recently received data. This could allow beekeepers to set thresholds for characteristics such as the beehive weight being above 150lb, and a system administrator could get alerts that the system has not uploaded any data in fixed length of time.

COSM data is organized into feeds which can be used to represent a single device, which in the case of the RHMS is a single Node. Within the feed is a number of datastreams which represent individual sensor measurements over time. Datastreams are made up of datapoints which have an associated value and timestamp. Each datastream has an associated unit along and other parameters [26]. An example of two COSM datastreams can be seen in Figure 4-8.

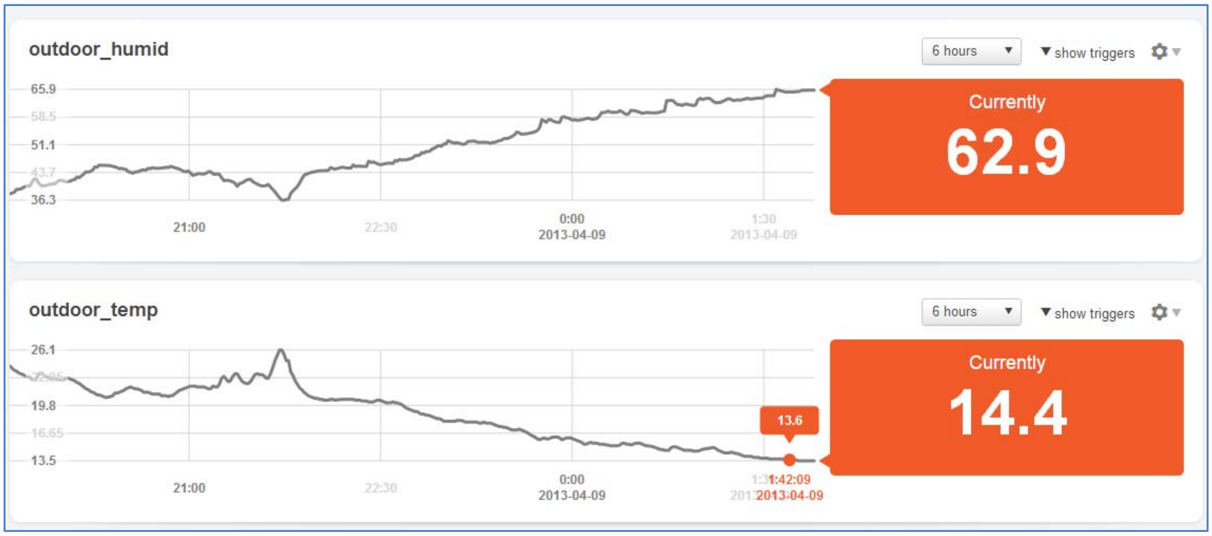

Figure 4-8 - COSM Datastreams

The COSM service is accessed through an open source 3rd party PHP API that allows for Creation, Read, Update, and Delete (CRUD) operations on COSM feeds. This API is accessed from the processing script described in 4.1.3.2. The source code for the COSM API and processing script can be seen in Appendix C.

4.1.3.3.3 Access and Graphing Script

To provide a convenient method of downloading sensor data for analysis, a data access script was written that allows users to select the desired sensors and time-frame; the data is then visualized in overlapping line graphs to let the user preview the dataset before downloading the data in CSV format. Figure 4-9 shows a screenshot of the interface of the access and graphing script.

Figure 4-9 - Access and Graphing Script

4.2 Physical Construction

Previous sections described the Node in the RHMS as all of the components necessary to acquire data about a single beehive, including sensors, a control unit, and wireless component. For the purposes of this section the Node is described in terms of physical construction, meaning not all of the components that are part of the Node subsystem are physically located in the same place. The physical reality of the RHMS is that the majority of the sensor components are resident in, on, or under the beehive. These components do not typically need to be physically accessed. The remainder of the components, including the control unit, wireless component, batteries, and other miscellaneous components, are installed in a weatherproof enclosure away from the beehive such that maintenance tasks like restarting the system and replacing the batteries do not disturb the bees. The remainder of this section will describe the typical beehive called the Langstroth Hive, the process of integrating the sensors into the hive, and the construction of the Node enclosure.

4.2.1 The Langstroth Hive

Figure 4-10 shows the components of a typical beehive. The most common type of beehive is known as the Langstroth beehive. The beehive used for the installation of the RHMS resembles this typical Langstroth beehive with a couple modifications. The beehive in Figure 4-10 - Langstroth Hive features a queen excluder, two honey supers, and a stand which are absent in the RHMS installation. The beehive into which the RHMS integrated is made up of a bottom board, deep super (often called a brood chamber), inner cover, and outer cover. There are a number of different types of bottom board available, one of which is screened to theoretically allow mites to fall out of the hive. This is the type of bottom board used for the RHMS test hive. This type of bottom board opens the hive to outside air more than a solid bottom board, which may affect interior temperature and humidity readings differently than a solid bottom board.

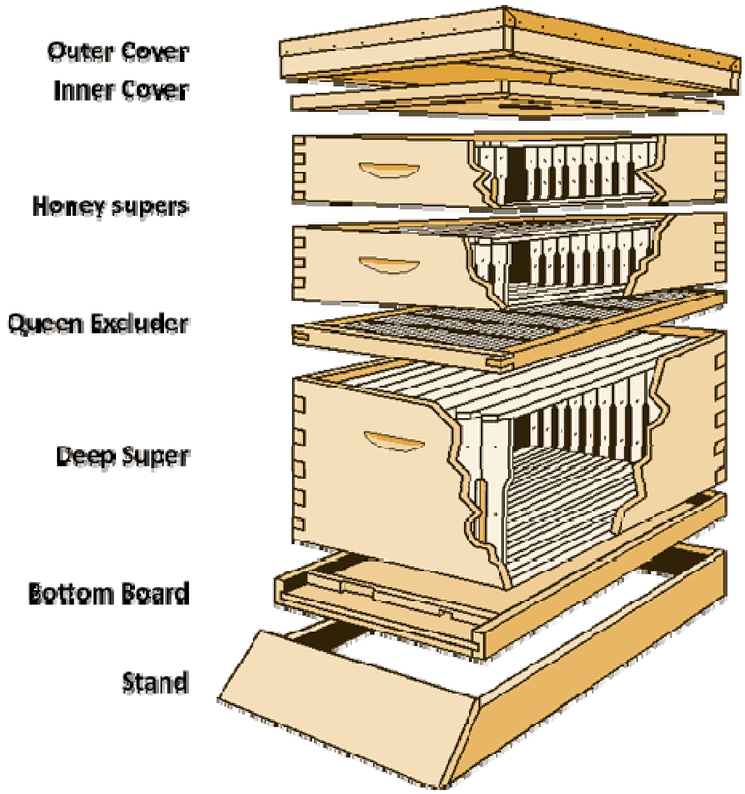

Figure 4-10 - Langstroth Hive [27]

4.2.2 Enclosure Construction

The Nod subsystem is physically separated into the Node enclosure, which houses most of the hardware components, and the sensors that are located in and around the beehive. Enclosure construction involved installing the control unit, Xbee, and power components into a weatherproof outdoor enclosure to be placed near the hive. This enclosure can be seen in Figure 4-11. Figure 4-12 shows the internal wiring of the enclosure. The only sensor component physically located within the enclosure is the battery voltage sensor discussed in 4.1.1.1.2.5. The enclosure also houses waterproof connectors for connecting and disconnecting the external sensor components at the beehive, allowing for the system to be moved more easily.

Figure 4-11 - Node Enclosure

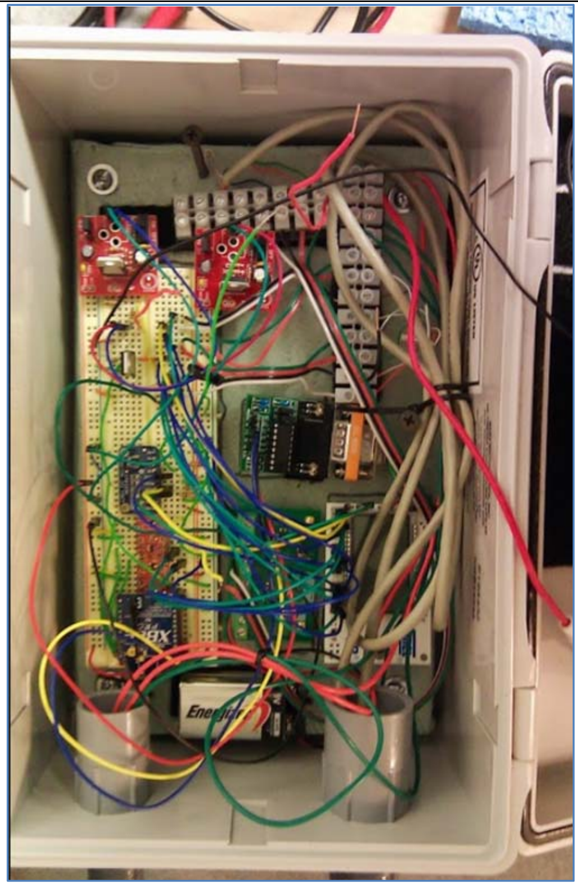

Figure 4-12 - Internal Node Layout

4.3 Sensor Integration

The majority of the sensor components used in the RHMS are physically located at the beehive. The majority of the sensors used are installed into the Sensor Frame, which is a modified inner cover similar to the inner cover shown in Figure 4-10. The additions to this inner cover include a notch on the back side for the cables used for conveying sensor signals to the rest of the Node hardware. In addition, cutouts were made in six locations for the six sensors installed. The sensors installed in the inner cover are four DS18B20 temperature sensors and two dual temperature and relative humidity RHT03 sensors. Figure 4-13 shows the sensor layout of the Sensor Frame. The wiring diagram for the electrical connections made in the inner cover can be seen in Figure A-2 in Appendix A.

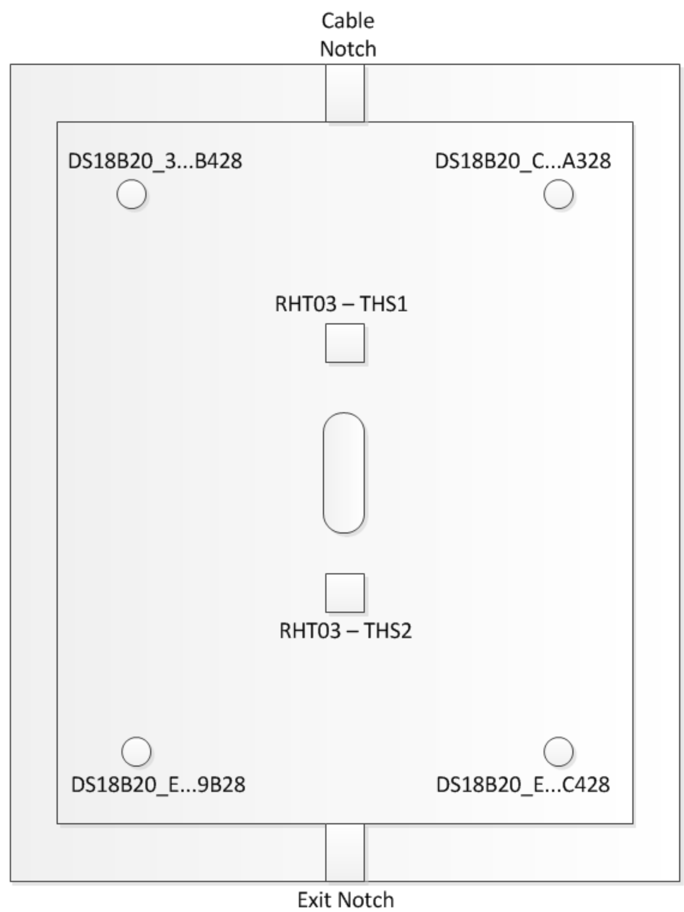

Figure 4-13 - Sensor Frame Layout

In addition to the sensor frame, there are also two sensors that reside on the outside of the beehive for monitoring weather conditions. These are an RHT03 temperature and relative humidity sensor as well as a photoresistor for light intensity measurement.

The Sensor Frame sensors are connected to the remainder of the Node hardware by a shielded, UV rated Category 5 (CAT5) Ethernet cable. This data cable carries a mix of digital and analog signals in the range of 0 to 5V.

The hivescale sensor that measures hive weight sits under the hive and utilizes a separate outdoor rated wire to carry the necessary signals to and from the load cell component of the scale. The hivescale interface board is located with the rest of the hardware in the Node enclosure. Figure A-3 in Appendix A shows a schematic of the inputs and outputs needed to operate the load cell.

4.4 Final Installation

The RHMS in operation can be seen in Figure 4-14. The following chapter details the testing results and acquired data from the implemented system in the varying test environments.

Figure 4-14 - Final Installation

Chapter 5 - Results

This chapter presents and describes the data acquired as part of the RHMS research prototype. The data acquisition process for this research was described in section 3.5. The data acquisition phase involved gathering data for system testing and evaluation as well as future analysis. Included in this chapter are meta-data about the system functioning and cost, and samples of relevant sensor data acquired.

Table 5-1 and Table 5-2 show an approximate cost breakdown of the Node hardware. These prices are all at consumer costs. See “DeviceCostBreakdown.xlsx” on the included CD to review purchase locations and device information.

Table 5-1 – Current Node Hardware Costs

Component	**Name**	**Cost**
Control Unit	Atmel Butterfly	$21.25
Wireless	Xbee Pro	$37.95
Temperature Sensors	DS18B20x4, RHT03x3	$46.85
Weight Sensor	Custom Hivescale	~$200
	Total	$306.05

Table 5-2 – Potential Node Hardware Costs

Component	**Name**	**Cost**
Control Unit	Atmega169 Chip	$4.95
Wireless	Xbee Pro	$22.95
Temperature Sensors	TMP36x7	$10.50
Weight Sensor	Custom Hivescale	~$200
Total		$238.40

Table 5-3 shows the cost breakdown of the Supervisor, both the current cost using research hardware and future cost using lower cost alternative hardware. See “DeviceCostBreakdown.xlsx” on the included CD to review purchase locations and device information.

Table 5-3 – Current and Potential Supervisor Hardware Costs

Component	**Current Device**	**Current Cost**	**Potential Device**	**Potential Cost**
Uplink	Janus-RC GE865 Development Kit	$300	Janus-RC Terminal GE865	$150
Wireless	Xbee Pro	37.85	Xbee 1mW	22.95
Total		337.95		172.95

Table 5-4 indicates the timeframes in which the RHMS was acquiring data with descriptions of the circumstances of the acquisition.

Table 5-4 - Data Acquisition Timeframes

Sample Timeframe	**Date start**	**Date end**	**Description**
1	2/20/2013	2/25/2013	First full week indoor test
2	3/6/2013	3/7/2013	Second indoor test
3	4/5/2013	4/13/2013	Outdoor calibration testing (empty hive)
4	4/15/2013	6/3/2013	Data acquisition using live colony of honey bees. Acquisition continues after 6/3/2013 but data will not be included in this thesis.

The different codes used to identify each data sample as either correct (properly formed), or an error are detailed in Table 5-5. The errors are detected by the Middleware Server discussed in 4.1.3.1.

Table 5-5 - Transmission Codes

Code Number	Transmission Code	Description
0	CorrectData	Correctly Formed Data (no error)
1	SocketInputTimeout	Timeout after the incoming connection is detected but no data
2	ATCommandDetected	An error occurred where the system missed a response code causing the node to re-send the AT command to open a socket
3	HTTPRequestTimeout	Timeout in HTTP request made to PHP portion of the server system. This could be a result of accessing the Cosm system
4	UnknownData	Data is received but is not of the correct CSV format (not sensor specific, just that the data is CSV).

Table 5-6 shows the transmission statistics for the time periods shown above in Table 5-4. Column 2 combines the first two timeframes due to the fact that they occurred in the same location with the same sample frequency. This table gives the total samples taken during the timeframe(s) and a breakdown of how many attempted transmissions were successful (Code 0) and how many were erroneous (Code 1 through Code 4). In addition, the table indicates the total runtime of each time period and total runtime.

Table 5-6 - Transmission Summary

	Timeframes 1 and 2	**Timeframe 3**	**Timeframe 4**	**Totals**	**Averages**
Total Number of Samples	11471	10805	5780	28056	
Code 0	10200	10490	5369	26059	
Code 1	261	285	408	954	
Code 2	683	1	0	684	
Code 3	326	26	3	355	
Code 4	1	3	0	4	
Total Runtime (hours)	128.74	186.55	990.70	1305.98	
Sample Rate (minutes)	1	1	10		2.8541
Error Rate (%)	11.0801	2.9153	7.1107		7.0354
Transmission Rate (%)	88.9199	97.0847	92.8893		92.9646

Table 5-7 details the differences between the desired sample rate (Sample Rate) and the minimum, maximum, average, and mean sample rates recorded during testing.

Table 5-7 - Transmission Frequency Summary

(minutes)	**Timeframe 1**	**Timeframe 2**	**Timeframe 3**	**Timeframe 4**
Sample Rate	1	1	1	10
Min	0.630	0.667	0.350	9.733
Max	10.300	14.000	31.400	70.383
Average	1.070	1.068	1.098	11.018
Median	1.030	1.033	1.033	10.033

Table 5-8 details the highest and lowest temperatures withstood by the system during testing and the accumulated time at those temperatures.

Table 5-8 - Temperature Extreme Summary

	Temp. (Celsius)	Total Elapsed Time (hours)
Min	-0.6°	3.589166667
Max	38.9°	167.3038889
Average	22.5°	

Figure 5-1 gives a visual representation of the most common error, the SocketInputTimeout error, over time to indicate changes in the rate of this error. The date ranges can be compared to the sample timeframes from Table 5-4.

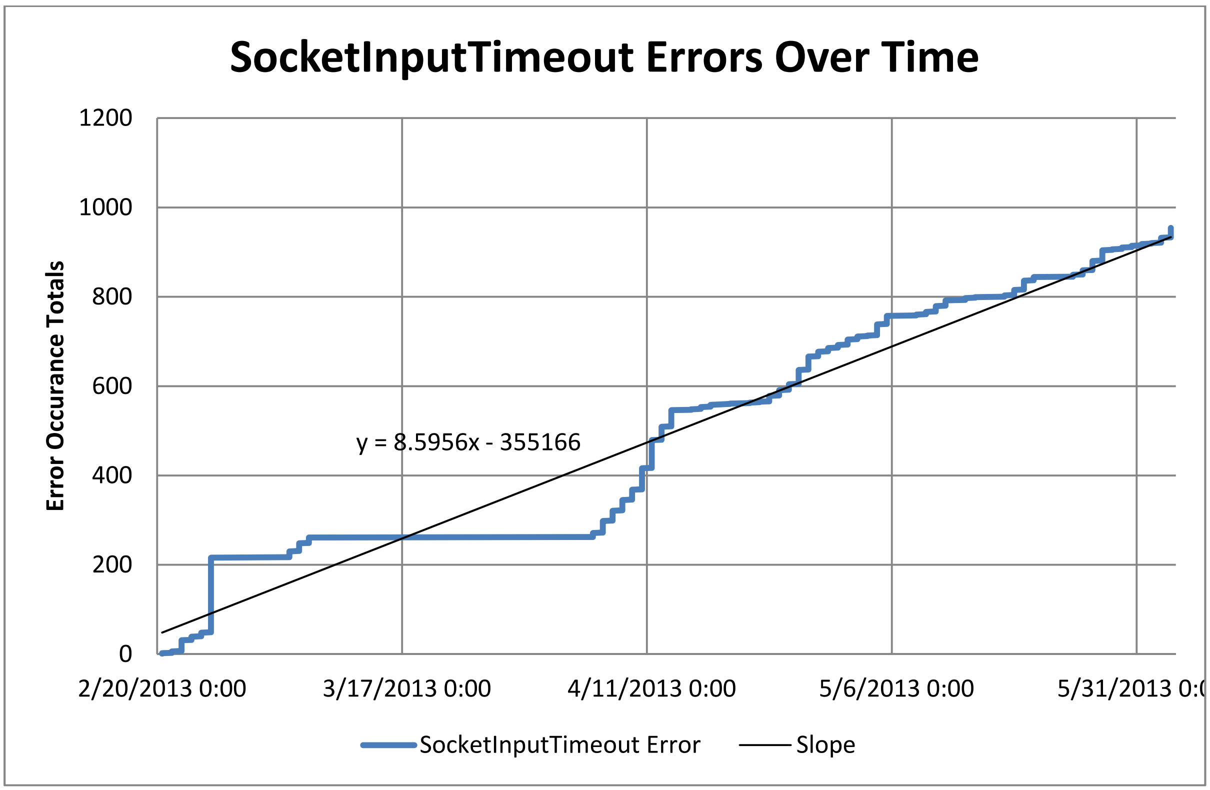

Figure 5-1 SocketInputTimeout Errors Over Time

Figure 5-2 shows the differences in the internal temperature of the beehive compared to the external temperature as measured by an internal and external RHT03 sensor.

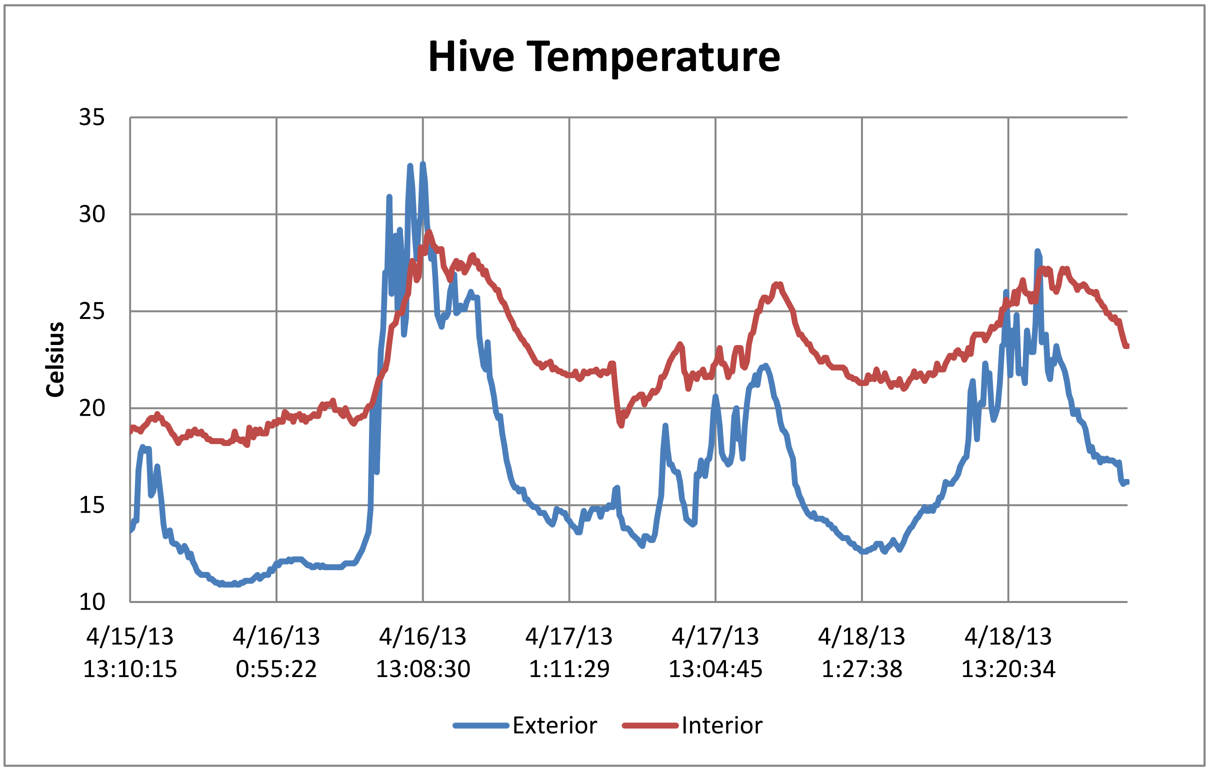

Figure 5-2 - Hive Temperature Effects

In Figure 5-3, a comparison of the interior and exterior relative humidity measurements can be seen. These measurements were taken by internal and external RHT03 sensors.

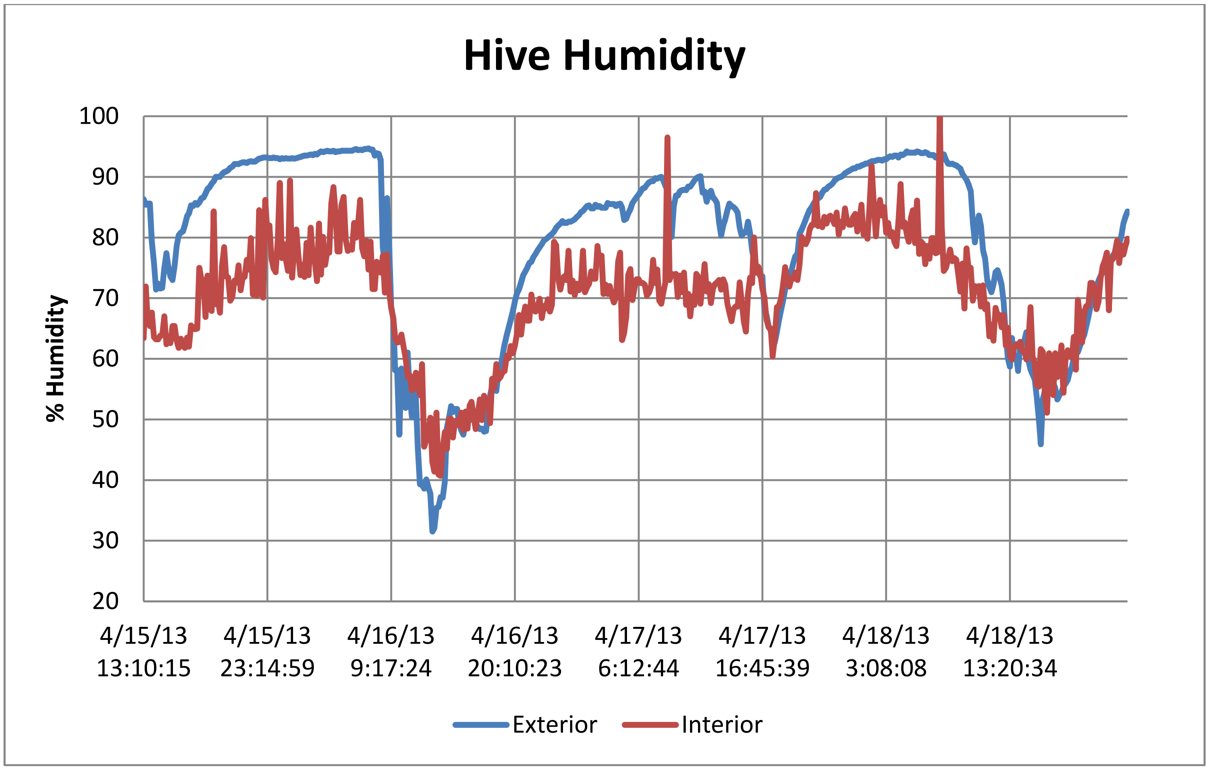

Figure 5-3 - Hive Humidity Effects

Figure 5-4 and Figure 5-5 show a comparison of the effects of temperature on the relative humidity inside and outside the beehive over the same time period.

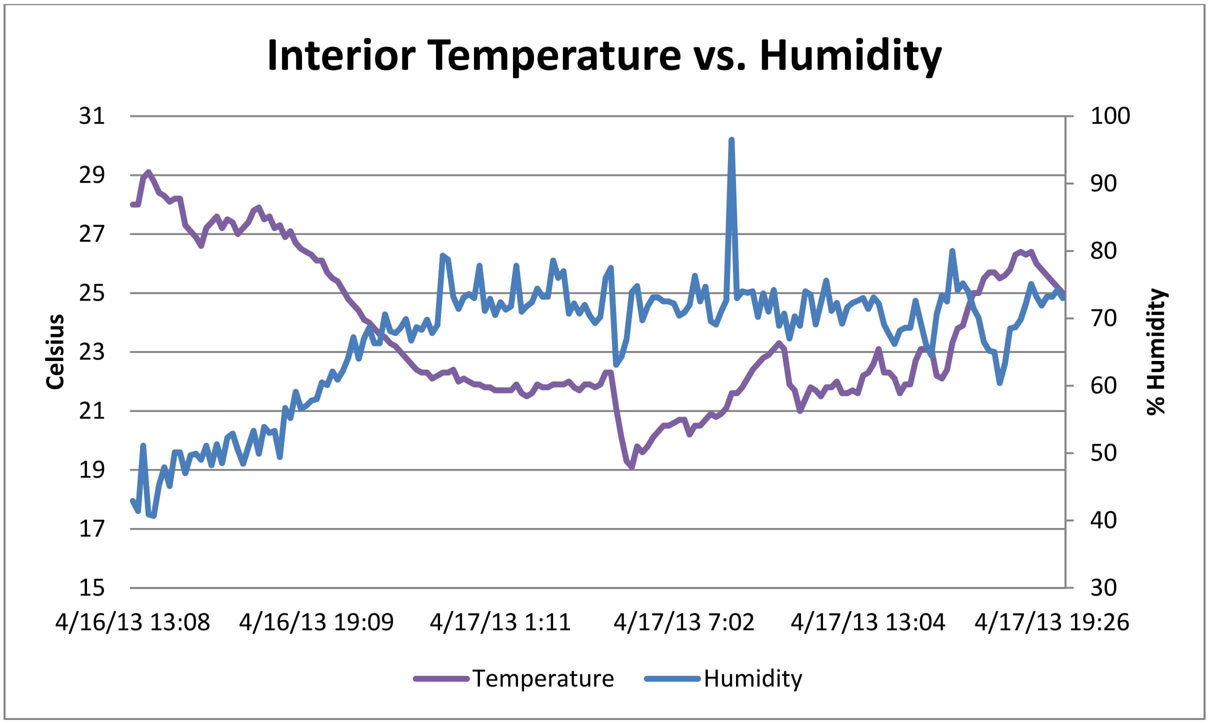

Figure 5-4 - Interior Temperature vs. Humidity

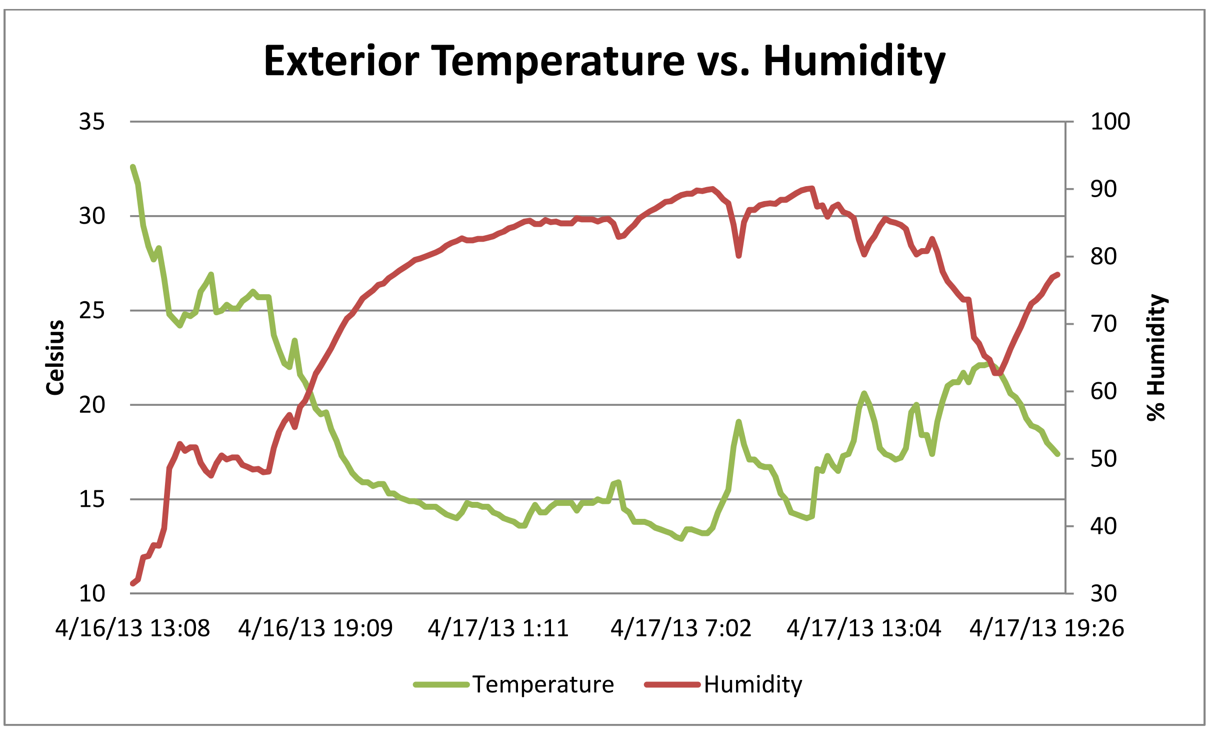

Figure 5-5 - Exterior Temperature vs. Humidity

Figure 5-6 shows a comparison of the main battery voltage of the system compared with temperature. The temperature sensor measured is located inside the enclosure in which the batteries are located.

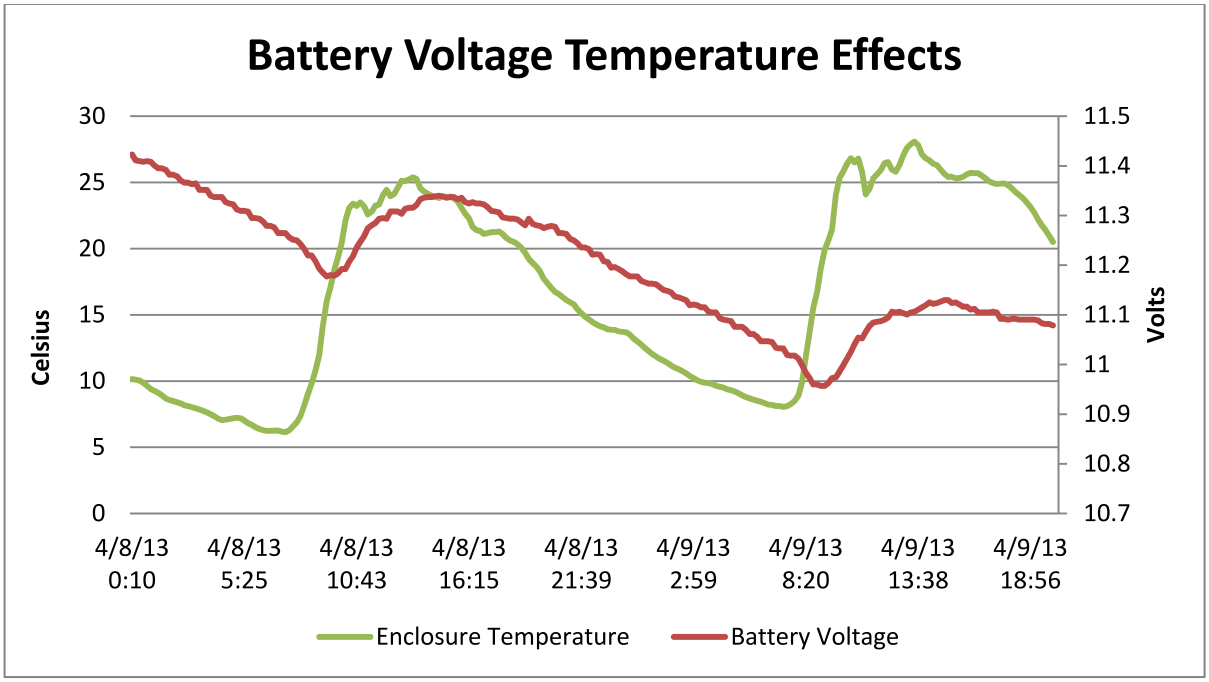

Figure 5-6 - Battery Voltage Temperature Effects

Figure 5-7 shows the rate of battery voltage decrease from system startup to the system shutting down due to low battery.

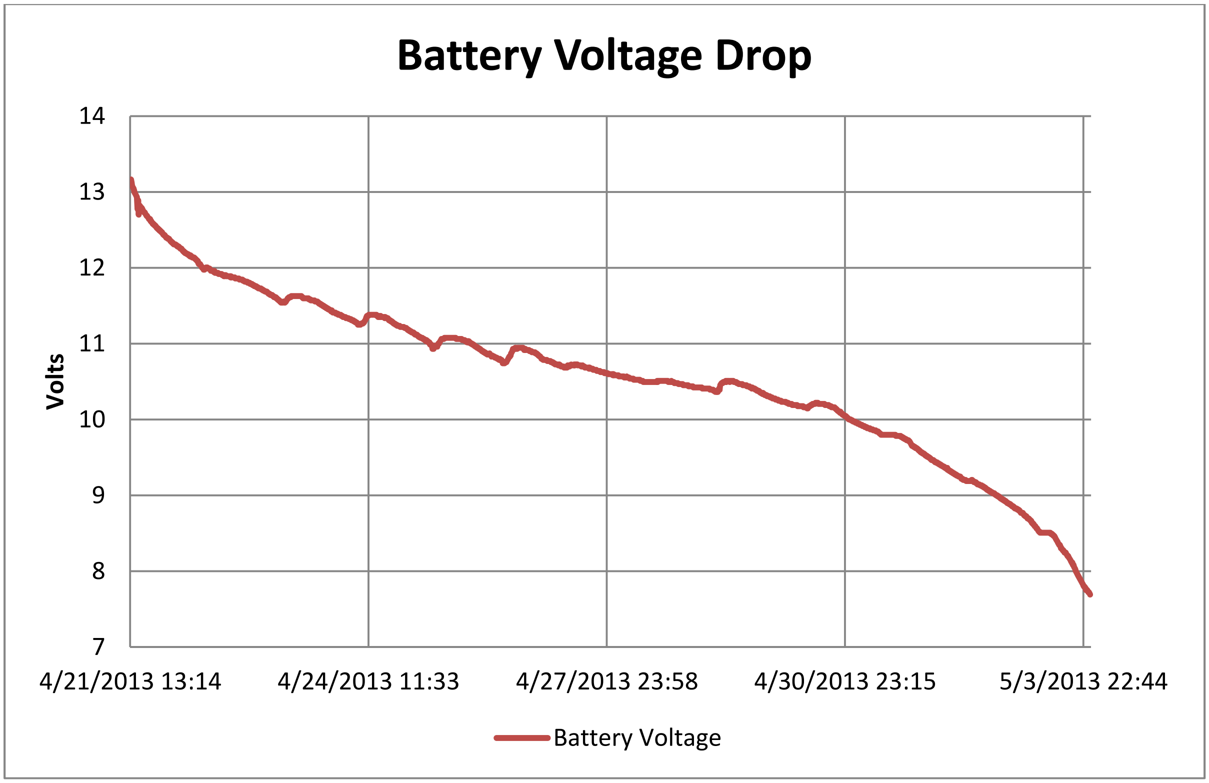

Figure 5-7 - System Battery Voltage Drop

Figure 5-8 shows a comparison of battery voltage and weight when the battery voltage reaches near-cutoff levels. Figure 5-9 shows a comparison of the beehive weight and temperature at low battery voltages.

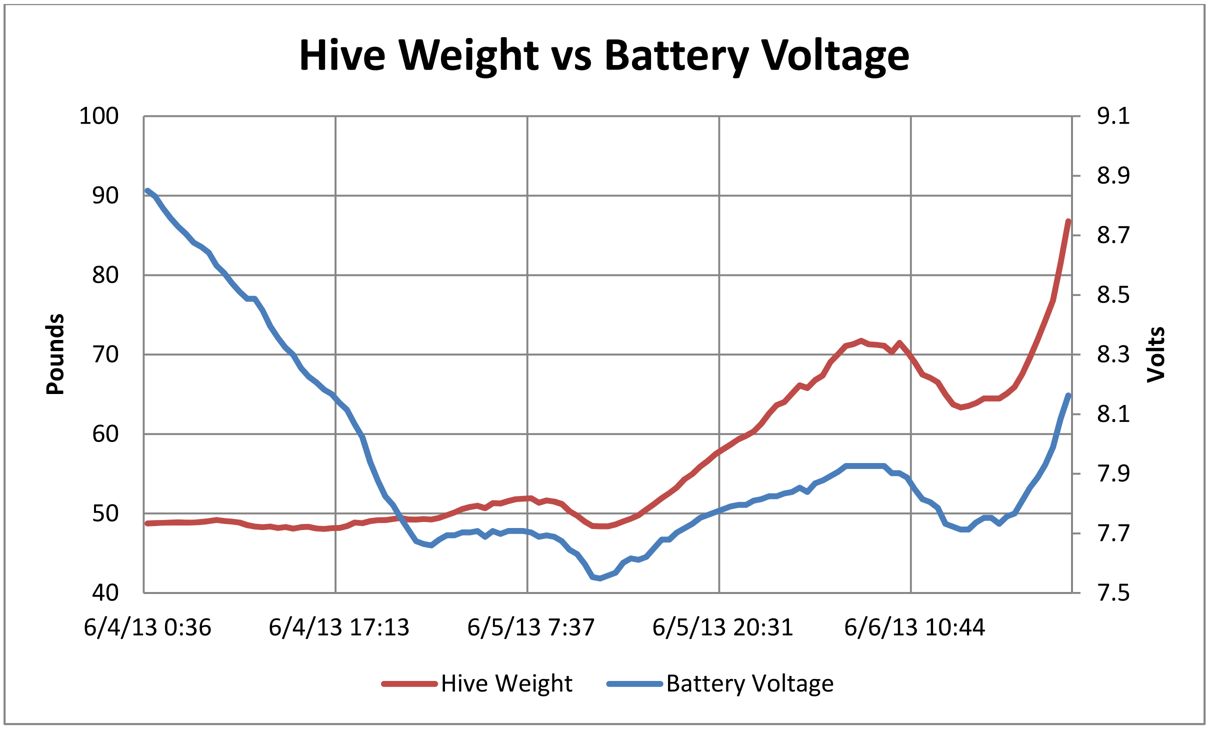

Figure 5-8 - Hive Weight Battery Voltage Effects

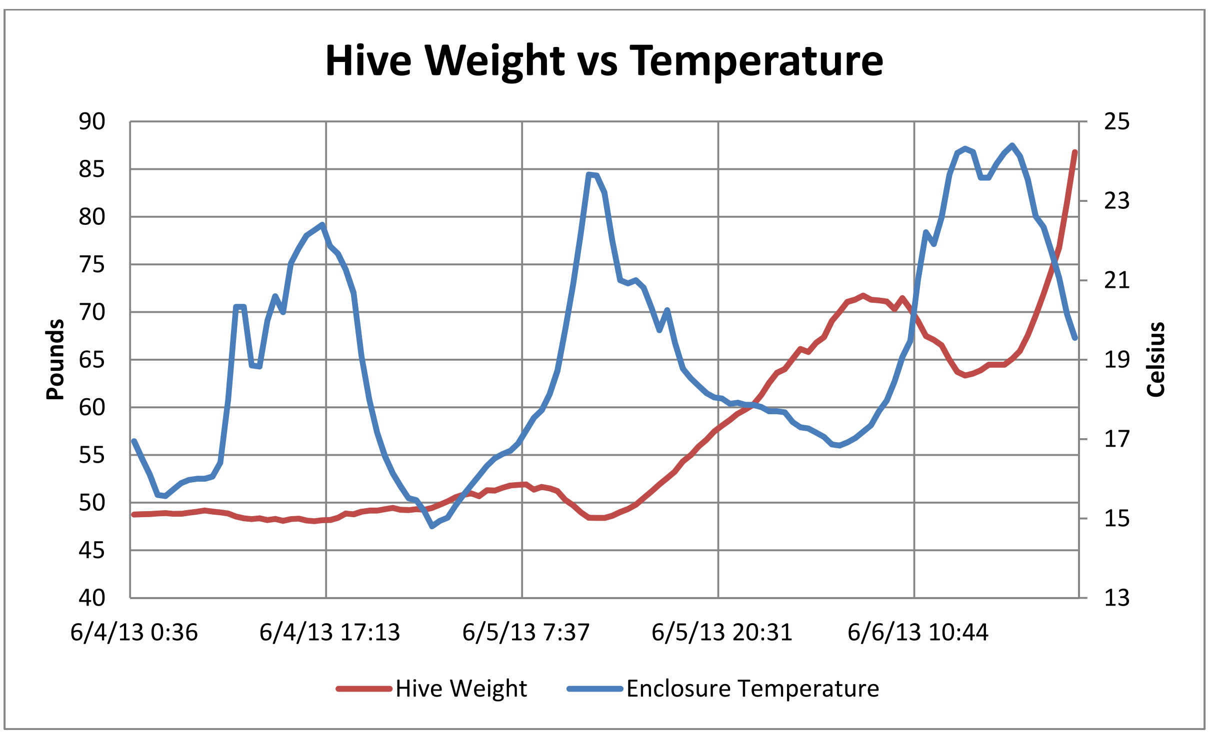

Figure 5-9 - Hive Weight vs. Temperature at Low Voltage

Figure 5-10 shows a comparison of beehive weight measurements and outdoor temperature at normal battery voltage levels (as opposed to above in Figure 5-8 - Hive Weight Battery Voltage Effects). The outdoor temperature measurements were taken with the outdoor RHT03 sensor.

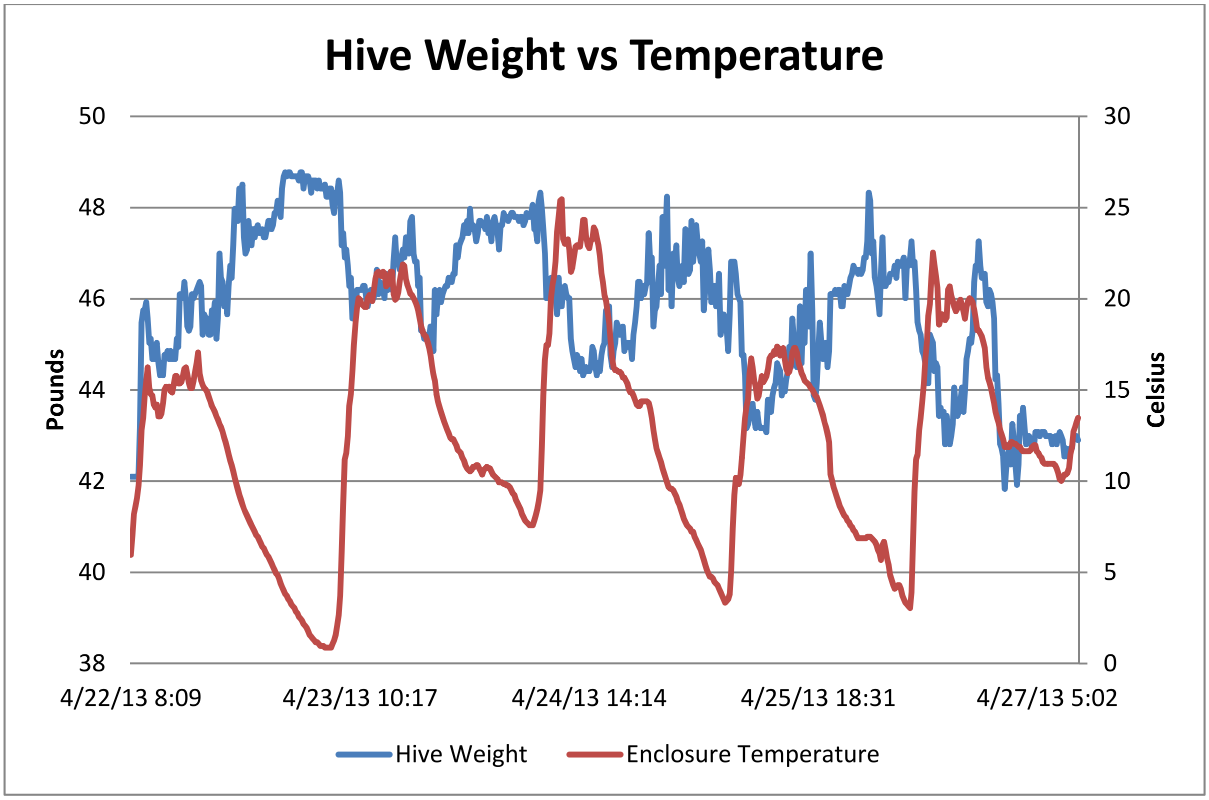

Figure 5-10 - Hive Weight Temperature Effects

Figure 5-11 shows a comparison of recorded light intensity and outdoor temperature over a 10 hour period beginning at 8 am.

Figure 5-12 shows a comparison of the beehive weight and outdoor temperature during a time period of high foraging for the hive.

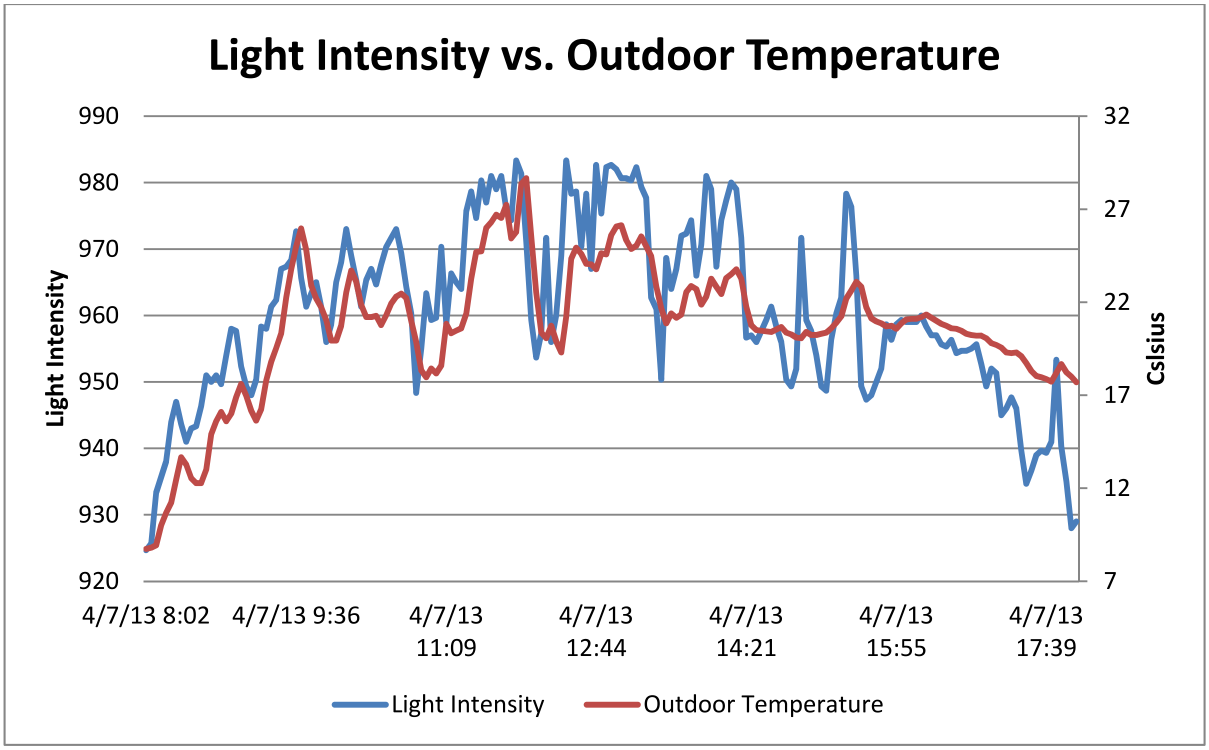

Figure 5-11 - Light Intensity vs. Outdoor Temperature

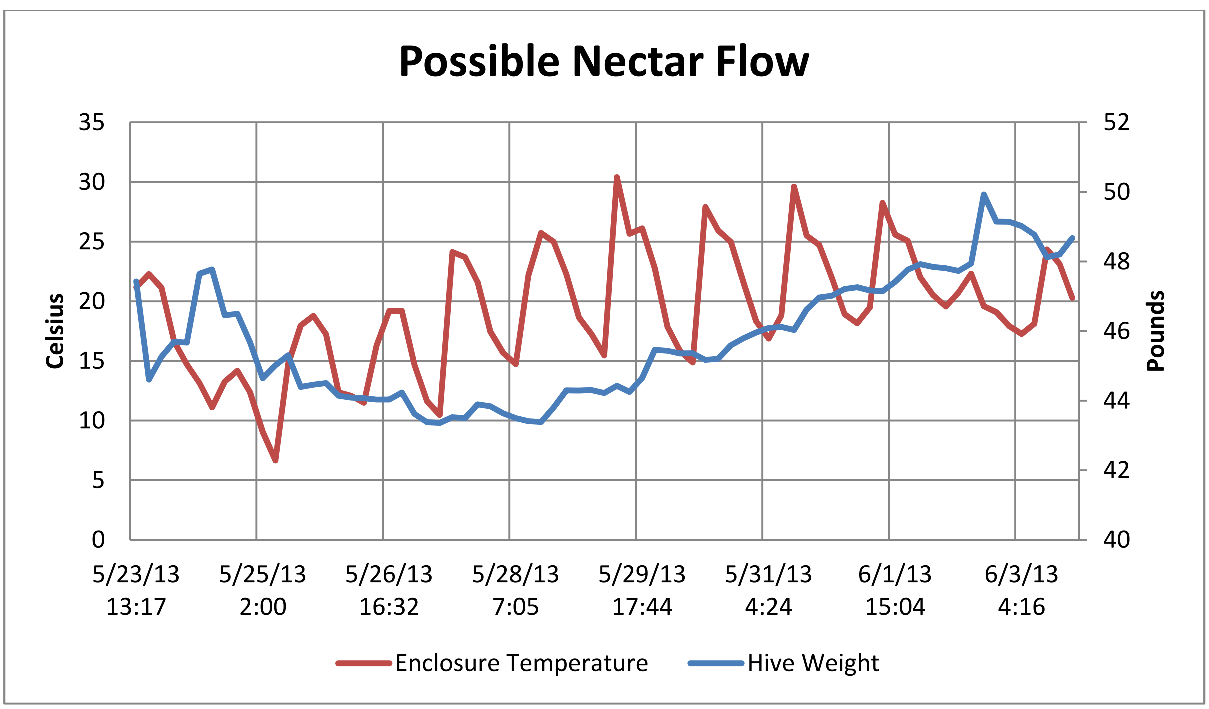

Figure 5-12 - Possible Nectar Flow

Chapter 6 - Conclusion

The ability to wirelessly transmit data from a remote device to the web at a reasonable cost was the primary function of the system and the main goal of this research. The secondary goal of this research was to gather data of sufficient quality about a beehive to ensure that the system can continue acquiring data that can be analyzed to make useful contributions to the field of honey bee research. These metrics were used to evaluate the effectiveness of this system.

6.1 System Cost

A goal of the RHMS was to achieve comparable data acquisition performance at a competitive cost to contemporary systems. As a research prototype, the cost of the RHMS was not minimized to the optimal level; however, it was important to identify areas of the system that could be optimized in the future. This was accomplished by selecting hardware components that offered a minimization-path for a future, lower cost system.

A cost breakdown for the hardware used in the Node and Supervisor can be seen in Table 5-1, Table 5-2 and Table 5-3. As these tables show, the RHMS operates at comparable cost to commercially available systems such as the BeeWatch system that can cost as much as $1500 [16], or customizable mote driven systems which can cost more than $100 per node without any sensors or uplink capabilities [20]. As Table 5-1 and Table 5-2 show, the hive scale component of the Node makes up a significant portion of the total cost. Producing a low cost and accurate scale was not a goal of this research; however for a future wide-scale system to be viable, a lower cost alternative must be found. Other avenues of reduced cost in the node can be found by reducing the number of digital sensors or by replacing them with analog alternatives such as the TMP36. In addition, the Xbee Pro can be reduced to a lower power 1mW version to reduce cost. Additional cost reduction could be attained through the creation of a custom circuit board containing a discrete microcontroller such as the Atmega169, as opposed to a prototyping tool like the Atmega Butterfly. The cost of the Supervisor node could be reduced significantly through the use of a lower-cost version of the same cellular module, as can be seen in Table 5-3.

6.2 Wireless Transmission Performance

Table 5-6 shows a summary of the transmission statistics of the system during testing. The RHMS prototype ran for a combined 1305.98 hours for the collection of these data. The only cause of system failure during this time was battery depletion and server issues resulting in the shutdown of the Middleware server on two occasions. Otherwise the system ran continuously without requiring any maintenance or manual reset. In total, 28056 samples were taken, with the majority of those taking place before deployment in a live honey bee colony. This is due to the increased sample rate during indoor testing and outdoor calibration testing. The error rate seen during the combined timeframes 1 and 2 is higher than the later timeframes and can be attributed to errors in the source code present at the Node. Modifications to the Node software all but eliminated these errors in timeframe 3 and 4, reducing the error rate significantly, leading to correct transmission rates of >90%. Errors of type code 2-4 were not significant factors in timeframe 3 or 4.

An increase in Code 1 in timeframe 4 errors from the same table can be attributed to the change in location of the Supervisor wireless hardware from the testing environment to the deployed environment and a decrease of preventable errors. The signal strength during timeframes 1 through 3 was -85 dBm, compared with -109 dBm in the deployed location during final testing. Figure 5-1 shows the accumulated SocketInputTimeout errors over the course of testing. Calibration testing outdoors around 4/5/2013 caused a significant increase due to the increased sample rate, the slope of these errors decreases when the sample rate is decreased. In addition to other factors, tree cover increases during the period of increased code 1 errors that would not have been a factor to RF signal quality during timeframe 1 and 2.

Table 5-7 shows that, despite poor cellular reception, the average sample frequency achieved during in timeframe 4 was within less than 10% of the desired sample frequency. The algorithm used favors power efficiency over error detecting, choosing to enter a low power state after several attempts to upload a sample. In times of fluctuating RF reception due to local interference or environmental conditions can improve battery life, but at the expense of missed samples. With a sample rate of 10 minutes, this can explain the observed sample average.

The system was exposed to the typical outdoor environment that many beehives are exposed to. The system was installed when weather conditions were such that it was safe to install a live honey bee colony without risk to the colony, which required temperatures to remain >=10°C for several days following installation of the colony. Temperature statistics can be seen in Table 5-8. The system did encounter significant rainfall, which is common in the mountains of North Carolina in the spring, without any adverse effects, showing that the RHMS prototype is sufficiently robust for the geographic area.

Figure 5-11 shows correlations between the outdoor temperature and light intensity. This is to be expected. However, due to the interface circuitry for the LDR, this measurement will most likely not be useful. The resistor values in the circuitry properly range the input signals between the minimum and maximum values, but to be useful, more magnitude of change needs to be seen in times of high light intensity. The result would be light intensity measurements of near zero until mid-morning, and large variations observable during times of cloud cover or clear sky.

Power consumption is important to monitor to estimate and model system runtime. Monitoring battery voltage was also conducted to track the effect of the environment on the batteries and to determine what effect battery voltage has on other parts of the system. Figure 5-6 shows the effect of temperature during the day/night cycle on the battery voltage. Figure 5-7 shows the maximum and minimum voltage of the batteries during a full system run (from fresh batteries to dead batteries). This period takes roughly two weeks. It can be observed that battery voltage fluctuates once the batteries reach approximately 8 volts. The voltage appears to “bounce” and even seemingly increase. This does not appear to have an effect on the temperature and humidity sensors; however as Figure 5-8 shows, this low voltage begins to affect the weight measurements. This indicates that for accurate weight measurements, the system should not be allowed to fully shut down before batteries are replaced. The use of COSM (Xively) allows for alerts when battery voltage levels fall below a threshold. Figure 5-10 shows a comparison of hive weight to outdoor temperature during this same time period. The effect of temperature on hive weight is discussed in the next section.

6.3 Data Quality

Data analysis was not a major goal of this research but it was important to ensure that the acquired data was of sufficient quality that future analysis could prove useful. To do so, a number of sensor measurements were analyzed to ensure that the acquired data were reasonable and some basic expected results could be seen.

Given that honey bees are living creatures that generate heat, it was expected that interior temperatures would be different from outdoor temperatures. Honey bees will also ventilate the beehive in hot weather. Figure 5-2 corroborates these by showing that the interior temperature, while tracking the day/night fluctuations of the outdoor temperature, maintains a more consistent temperature without the drastic swings of the outdoor temperature. Temperatures did not reach sufficient levels during the data acquisition period to show definitive examples of hive ventilation.

In addition to temperature, hive humidity is also tracked. Hive activity was expected to have an effect on interior humidity but that effect was not known. Figure 5-3 shows a comparison of interior and exterior humidity. Nectar is evaporated into honey in the hive after being collected by the foragers [28]. This process could be responsible for the fluctuations of humidity within the beehive.

Temperature and relative humidity are closely linked. Figure 5-5 shows that, typically, relative humidity inversely mirrors temperature very closely outdoors. This trend continues inside the hive as can be seen in Figure 5-4; however, the humidity does not mirror temperature as closely as observed outdoors in Figure 5-5. In addition, the humidity fluctuations can be seen inside the hive that do not seem to correspond to changes in temperature.

One of the most desired beehive attributes to monitor is hive weight. Hive weight is also the most difficult and most expensive measurement. Weight measurements with a load cell such as the one used in the RHMS are affected by temperature, as can be seen in Figure 5-10. These effects are difficult to account for and require post processing techniques to negate. These temperature effects were most notable at lower temperatures. Figure 5-10 shows temperature effects at higher temperatures during the same time as Figure 5-8 discussed previously, showing that at temperatures above 15°C, temperature affects are less significant. Figure 5-12, shows similar temperature fluctuations at higher average temperatures with less fluctuation in weight. This figure also shows potential weight increases as a result of honey production, despite temperature interference, as the gradual increase is not consistent with observed temperature fluctuations.

Chapter 7 - Summary and Future Work

The results of the testing of the RHMS show that the system was able to acquire valid data about the conditions of a beehive wirelessly at a competitive cost for a research prototype. However are a number of areas for improvement in the prototype system. In addition, there are a number of future areas of research in analyzing the currently available and future data.

7.1 System Modifications

Several areas of improvement were identified during the design and construction of the prototype that could be implemented in a future version of the RHMS. These include scaling the system to use multiple Nodes, expanded power efficiency and capabilities, more uplink methods and flexibility, and local configuration options.

7.1.1 Multiple Nodes

The current RHMS prototype consists of only a single Node, reducing the need for a fully-implemented Supervisor node. The next iteration of the RHMS would involve multiple Nodes capable of monitoring multiple hives. This would require a fully implemented Supervisor that would be responsible for coordinating transmission times among each Node. The Supervisor would be installed into an enclosure similar to the one in which the current prototype Node is installed in a central location in the bee yard.

This process would involve converting the schematics generated during the prototype to a custom PCB design for the construction of multiple identical Nodes. The Node software would not require significant modification aside from a scheduling technique that allows the Nodes to share a single Supervisor. This would most likely utilize some form of time-division technique.

7.1.2 Renewable Power

Another area of improvement in the current system is in power consumption and runtime. A number of inefficiencies were identified in components utilized in the Node system. These included peripheral components such as power supplies and logic-level-converters, as opposed to central components such as the wireless device or control unit. These over-consuming units could be disabled when the system was in sleep mode, greatly reducing the overall power consumption.

In addition, renewable energy sources were considered for the RHMS prototype, but were not central to the research and were therefore not implemented. A solar energy array for the Supervisor and Nodes could allow the system to run for significantly longer periods of time, requiring even less maintenance and attention.

7.1.3 Flexible Uplink

Several uplink technologies were considered for the RHMS prototype and the cellular GSM modem was selected due to its geographic flexibility. Other wireless uplink methods investigated were Wi-Fi and Xbee. Wi-Fi requires more configurations on the user-side in terms of selecting SSID, channel, and security parameters. The Xbee devices used in the prototype would be a viable solution but would require a PC system with software capable of redirecting transmissions to the web. A future commercial version of the system could feature a flexible uplink interface that could facilitate a number of client-specific interfaces for different environments.

7.1.4 Local Configuration

There are a number of system configurations that are, in the prototype version, “hard-coded” that could be made to be user-configurable. Examples of these configurations include sample-frequency, enable-debug mode, and other software settings. In addition there are hardware settings that the user could select. One example of these is the maximum weight measurement. The current weight measurement configuration relies on a “voltage divider” that provides a “bias voltage” for the analog to digital converter that samples the load cell used for weight measurement. This bias voltage has an effect on the maximum possible weight that can be measured, in addition to the accuracy of weight measurements.

Through the use of Dual Inline Package (DIP) switches or magnetic switches, the user of the system could adjust these settings locally to the hives being monitored. This would reduce the need for node re-programming or resetting.

7.1.5 Cost Reduction

As discussed in Chapter 6, there are a number of avenues for cost reduction in the RHMS. A future version of the RHMS would focus specifically on the identification of a lower-cost hive scale alternative, but other reductions in terms of sensor costs and wireless hardware cost would be also be explored.

7.2 Data Analysis

An area of future work that does not involve modifications to the system is in the area of data analysis. The RHMS prototype during initial testing gathered more than 20,000 individual samples of its array of sensors. Analysis of these data could find correlations between sensors that were not predicted or expected.

Many areas of analysis involve relating hive activity to other measurements. The main indicator of hive activity comes from changes in hive weight, indicating that honey bees are leaving the hive to forage. To be able to quantify this, calibration of the hive scale would be required for accurate weight measurement. This process involves taking acquired weight and temperature measurements and performing statistical analysis to determine temperature coefficients. With this process completed, temperature effects can be negated in real-time or near real-time, allowing accurate measurement of weight to be conducted.

In addition to the data gathered by the RHMS prototype, another area of data analysis research would involve the analysis of outside datasets relevant to atmospheric and natural conditions. There are many weather data sources available from the National Weather Service (NWS) [29] that could be used to extend the RHMS's weather measurements such as atmospheric pressure, rainfall and cloud cover. These data could be combined with natural measurements such as pollen index and pollution measurements. The combination of multiple rich datasets could yield meaningful results from seemingly unrelated projects. The expanding use of web API's allowing flexible access to growing amounts of data will continue to add to and be improved by the type of data gathered by systems like the RHMS.

BIBLIOGRAPHY

[1] J Kim Kaplan. Colony collapse disorder. *Bee Culture* (Aug. 2012), 63-65.

[2] John D. Lea-Cox, Andrew G. Ristvey, David S. Ross, and George F. Kantor. eployment of wireless sensor networks for irrigation and nutrient management in nursery and greenhouse operations. In *Proc. Southern Nursery Assoc. Res. Conf.* 2009.

[3] A. Cano, E. Lopez-Baeza, J. L. Anon, C. Reig, and C. Millan-Scheiding. Wireless sensor network for soil moisture applications. *International Conference on Sensor Technologies and Applications* (2007), 508-513.

[4] Jiang Peng, Huang Qingbo, Wang Jianzhong, Dai Xiaohua, and Lin Ruizhong. Research on wireless sensor networks routing protocol for wetland water environment monitoring. *Innovative Computing, Information and Control*, 3 (Sep. 2006), 215-254.

[5] M. Popa and C. Iapa. Embedded weather station with remote wireless control. *Telecommunications Forum (TELFOR), 2011 19th* (Nov. 2011), 297-300.

[6] Kirk P. Visscher and Thomas D. Seeley. Foraging strategy of honeybee colonies in a temperate deciduous forest. *Ecology*, 63, 6 (Dec. 1982), 1790-1801.

[7] Ernesto Guzmán-Novoa, Leslie Eccles, and Yireli Calvet. Varroa destructor is the main culprit for the death and reduced populations of overwintered honey bee (Apis mellifera) colonies in Ontario, Canada. *Apidologie*, 41, 4 (2010), 443-450.

[8] BZBODIES. Retrieved May 3, 2013 from http://www.mybzbodies.com/Pages/default.aspx.

[9] Pollination Contracting. Retrieved May 4 from http://www.beepollination.com/.

[10] Sammataro Diana and Avitabile Alphonse. In *The Beekeepers Handbook (4th ed.).* Comstock Publishing Associates, New York, 2011.

[11] S Keith Delaplane, Jozef van der Steen, and Ernesto Guzman-Novoa. Standard methods for estimating strength parameters of apis mellifera colonies. *Journal of Apicultural Research*, 52 (Jan. 2013), 1-12.

[12] Dennis VanEngelsdorp. 2012. Retrieved May 11 from http://beeinformed.org/.

[13] Wayne Esaias. 2010. Retrieved May 3 from http://honeybeenet.gsfc.nasa.gov.

[14] Maximilian Michels. 2011. Retrieved April 13 from http://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/Betreute_Arbeiten/Bachelorarbeit-Michels.pdf.

[15] HiveTool. Retrieved May 7 from http://www.hivetool.org.

[16] Beewatch. 2013. Retrieved April 3 from http://beewatch.biz/index.php/stockwaage-41.html.

[17] V. Gazis, K. Sasloglou, N. Frangiadakis, and P. Kikiras. Wireless Sensor networking, automation technologies and machine to machine developments on the path to the Internet of Things. *16th Panhellenic Conference on Informatics* (2012), 276-282.

[18] Luigi Atzori, Antonio Iera, and Morabito Giacomo. The Internet of Things: a survey. *Computer Networks*, 54, 15 (Oct. 2010), 2787-2805.

[19] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52, 12 (2008), 2292-2330.

[20] M. Johnson, M. Healy, P. van de Ven, M. J. Hayes, J. Nelson, T. Newe, and E. A. Lewis. A comparative review of wireless sensor network mote technologies. *Sensors, 2009 IEEE* (2009), 1439-1442.

[21] Samuel R. Madden, Joseph M. Hellerstein, and Wei Hong. The sensor spectrum: technology, trends, and requirements. *ACM SIGMOD Record*, 32, 4 (Dec. 2003), 22-27.

[22] Volker Grober, Detlef Heydenbluth, Ralf Moos, et al. Sensor trends 2014. *AMA Association for Sensor Technology*.

[23] Joshua Cooper, and Anne James. Challenges for database management in the Internet of Things. *IETE Technical Review*, 26, 5 (2009), 320-329.

[24] Free Software Foundation. 2013. Retrieved Febuary 6 from http://www.nongnu.org/avr-libc/.

[25] LoMeIn, Inc. 2013. Retrieved April 11 from https://xively.com/?from_cosm=true.

[26] Heather Brown. 2012. Retrieved June 3 from http://www.almanac.com/blog/beekeeping/beekeeping-101-building-hive.

[27] C. P. Dadant. First Lessons in Beekeeping. Dadant Publications, Hamilton, 1997.

[28] National Weather Service. 2012. http://graphical.weather.gov/xml/rest.php.

Appendix A - Node Resources

Table A-1 - Node Datasheets

#	**Device**	**URL**
1	AVR Butterfly User Guide	http://www.atmel.com/Images/doc4271.pdf
2	AVR Butterfly Quick Start	http://www.atmel.com/Images/doc4249.pdf
3	Atmega169 Datasheet	http://www.atmel.com/Images/doc2514.pdf
4	DS18B20 Datasheet	http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf
5	RHT03 Datasheet	http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Weather/RHT03.pdf
6	ADS1015 Datasheet	http://www.ti.com/lit/ds/symlink/ads1015.pdf
7	Hivescale Datasheet	**TBD**
8	LDR Datasheet	https://www.jameco.com/Jameco/Products/ProdDS/202403.pdf
9	Xbee Pro Datasheet	https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf

The source code for the Node System can be found on the attached CD or on github at the URL: https://github.com/ricela1/MastersThesis2013. $ROOT refers to either the root of the CD resources or the top level directory of the github repository. The Node System code is located at $ROOT/Node_System/

Table A-2 - Node Source Code Reference

Filename and Location	**Description**
kernel_v6.c	Contains kernel control loop, device initialization and main().
adc.c	Driver for the ADC on the ATMEGA169
gsm_comms.c	Driver for transmitting data with the GE865
RHT_sensor.c	Driver for the RHT03 sensor
usart_comm_driver.c	Driver for UART (RS232) communication
ds18b20.c	Driver for the DS18B20 sensor
hivescale.c	Driver for the hivescale sensor
USI_TWI_Master.c	Driver for using I2C on the ATMEGA169

Figure A-1 shows the connections for the control unit. The labels connected to each pin can be traced to schematics later in this chapter. This schematic also shows the battery sensor voltage divider.

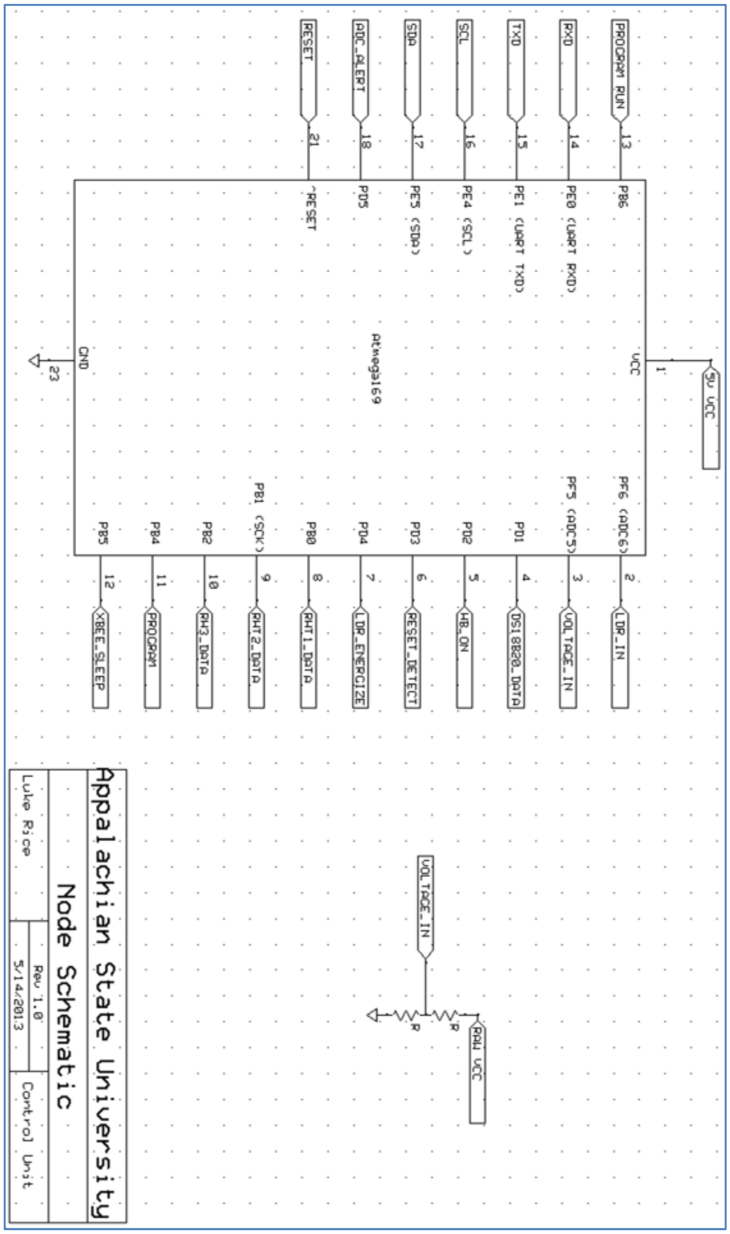

Figure A-1 - Control Unit Connections

Figure A-2 shows the layout of the sensor frame and the sensor connections. This schematic also includes the exterior RHT03 sensor and the LDR light sensor, as well as the 4 DS18B20 sensors and 2 RHT03 sensor s built into the inner cover.

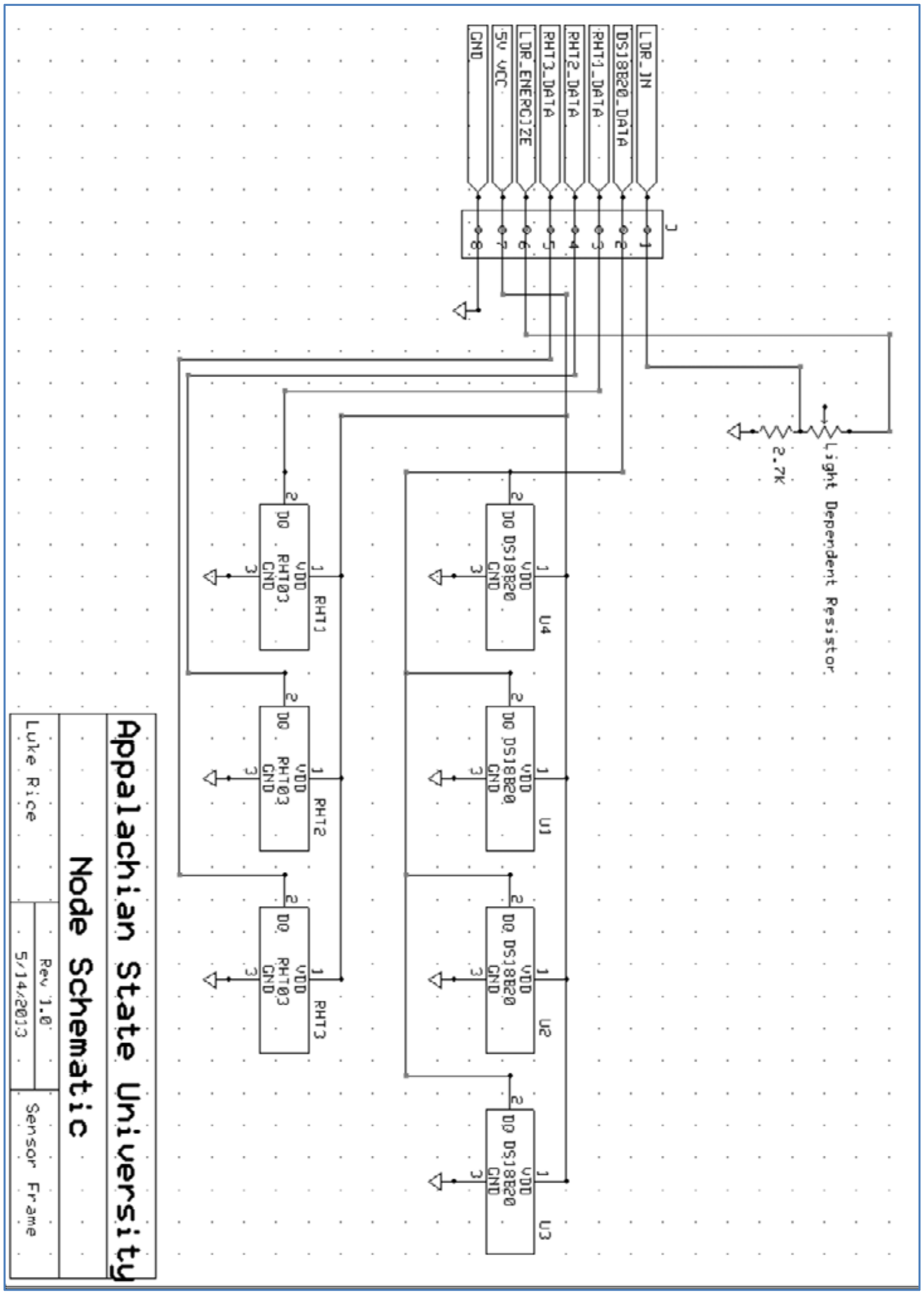

Figure A-2 - Sensor Frame Layout

Figure A-3 shows the composition of the hivescale electronics and the ADC circuitry needed to acquire data from the hivescale. The Ain1 pin on the ADS1015 is the reference pin for the Ain0 input, producing a bias voltage of ~1.024V allowing for maximum precision. The load cell itself is not located with the circuitry, the LC_5V, LC+, LC- and LC_GND are carried over a wire from the enclosure to the physical scale.

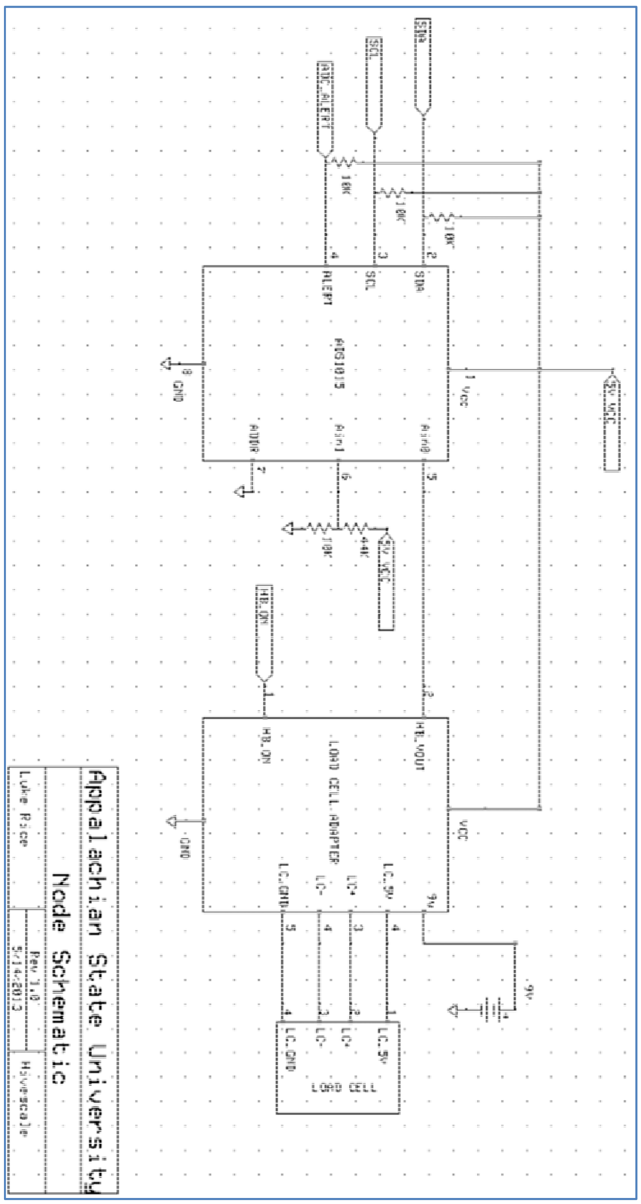

Figure A-3 - Hivescale Schematic

Figure A-4 shows the layout of the wireless hardware for the Node. The Xbee Pro is powered by 3.3V, compared to the 5V of the Atmega169 and sensors, requiring a separate power supply and logic level converters for the virtual wire connections. Additionally, the Atmega169 serial port is not logic level, requiring a MAX232 converter to convert the standard RS232 voltages to the 3.3V logic level voltage used by the Xbee Pro.

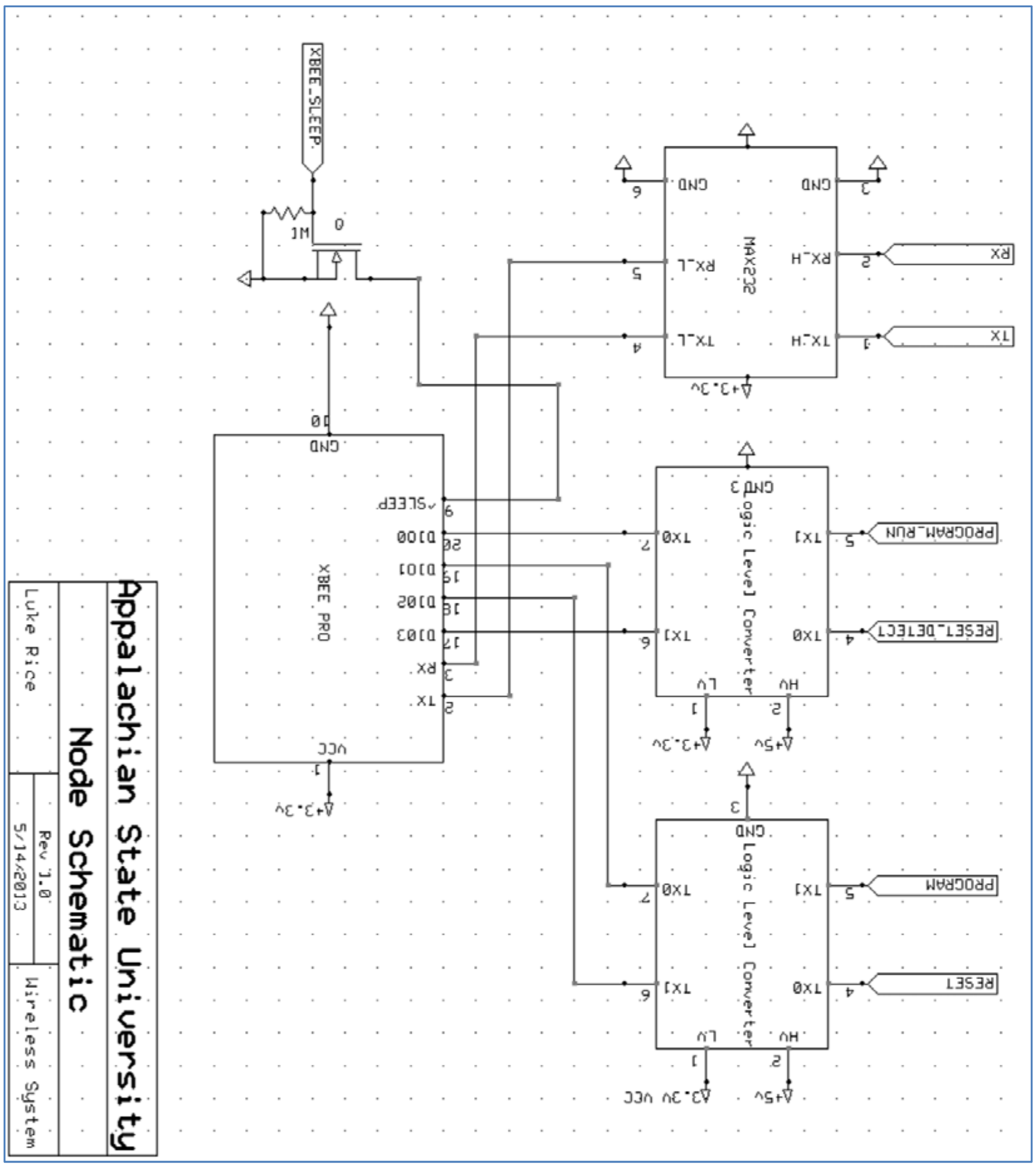

Figure A-4 - Wireless System

Figure A-5 shows the wiring of the power systems used by the node. The Xbee Pro requires 3.3V whereas the rest of the components use 5V, requiring a second 3.3V power supply. Testing was conducted with 2x8AA battery packs providing 4000mAh at 12V.

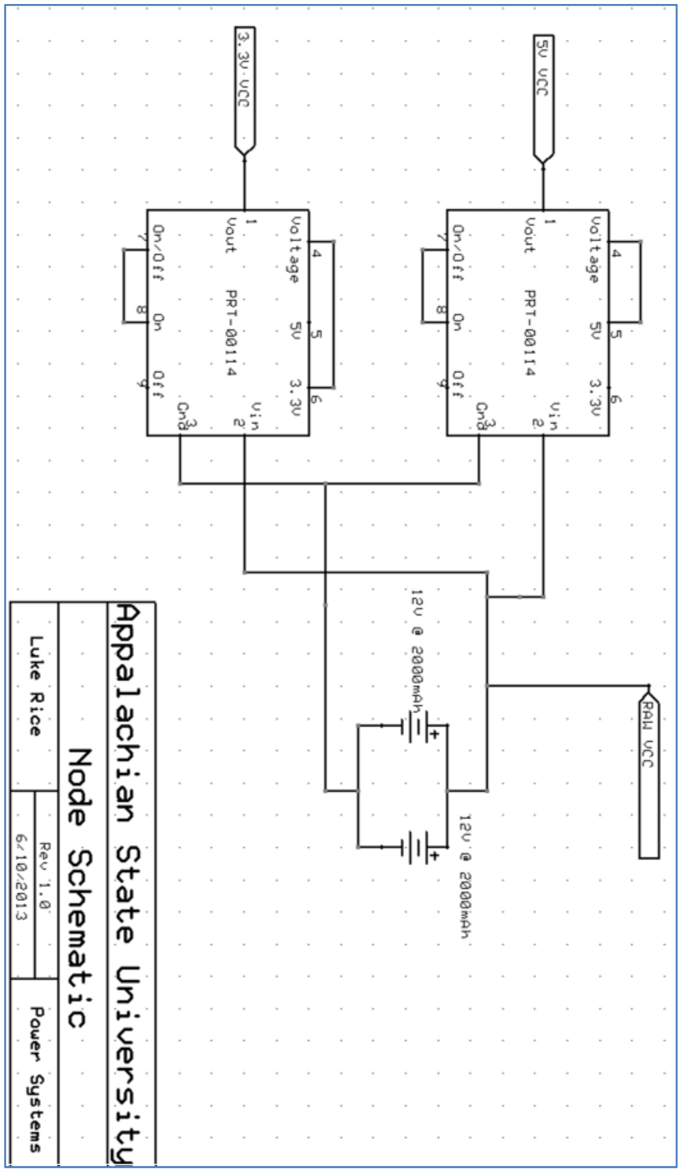

Figure A-5 - Power Systems

All AT commands for the Xbee Pro are preceded by the text “AT”. Table A-3 - Node Xbee Configuration shows the configuration of the Xbee used at the node with the command required to recreate the current configuration.

Table A-3 - Node Xbee Configuration

Command	**Description**
MY=2	ID of Node Xbee
DL=1	ID of Supervisor Xbee
CH=0x0E	Channel to transmit and receive on.
PL=4	Set power level to maximum
SM=1	Sleep mode to pin hibernate
BD=4	Set baud rate to 19200 (default is 9600)
D0-D4=3	Set virtual pins to digital input
IT=1	Digital pin samples before transmit =1
IC=0	Monitor changes on pins D0-D7
IR=1000	Digital Input sample rate = 1 second
DB	Test signal strength from last good packet

Appendix B - Supervisor Resources

Table B-1 - Supervisor Datasheets

#	**Device**	**URL**
1	GE865 Software User Guide	http://www.telit.com/module/infopool/download.php?id=522
2	GE865 Hardware User Guide	http://www.telit.com/module/infopool/download.php?id=1666
3	GE865 Command Reference	http://www.telit.com/module/infopool/download.php?id=542
4	Xbee Pro Datasheet	https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf

All AT commands are in the form “AT<command><args>”. In the below table, the preceding AT is omitted. Detailed descriptions of arguments can be seen in datasheet 3 above. Table B-2 - GE865 Configuration shows the commands needed to recreate the current configuration of the GE865.

Table B-2 - GE865 Configuration

Command	Description
+CMEE=1	Enable error reporting using numeric format
#SELINT=2	Use most recent AT command set
#ENS=1	AT&T/Cingular Configuration
+CGDCONT=1,"IP","broadband","0.0.0.0",0,0	Define PDP context (for GPRS). The first argument defines the context number (allowing for multiple contexts to be saved), the 2nd specifies the IPv4 protocol. The third argument specifies the APN, which is basically the "router" that GPRS data goes through to the web. The 4th parameter is a static IP, empty if not used. The last two parameters disable compression.
+CGQMIN=1,0,0,3,0,0	Set minimum Quality of Service (QoS)
+CGQREQ=1,0,0,3,0,0	Set desired QoS
#SGACT=1,1,"",""	Activate PDP context 1 with blank username and password
#SD=1,0,IPP,IPA,0,0,0	Open a socket using PDP context 1,TCP protocol, over port IPP to address (IP or hostname) IPA.
V0	Enables numeric response codes (0 for OK, 4 for ERROR)

Table B-3 - Supervisor Xbee Configuration shows the commands needed to replicate the current supervisor Xbee configuration. All AT commands for the Xbee Pro are preceded by the text "AT".

Table B-3 - Supervisor Xbee Configuration

Command	**Description**
MY=1	ID of Supervisor Xbee
DL=2	ID of Node Xbee
CH=0x0E	Channel to transmit and receive on.
PL=4	Set power level to maximum
SM=1	Sleep mode to pin hibernate
BD=4	Set baud rate to 19200 (default is 9600)
D0-D4=3	Set virtual pins to digital input
IT=1	Digital pin samples before transmit =1
IC=0	Monitor D0-D7
IR=1000	Digital Input sample rate = 1 second
DB	Test signal strength from last good packet

Figure B-1 shows the wiring schematic of the supervisor, with the most important connecting sbeing the wiring of the virtual wire switches for remote control of the Node.

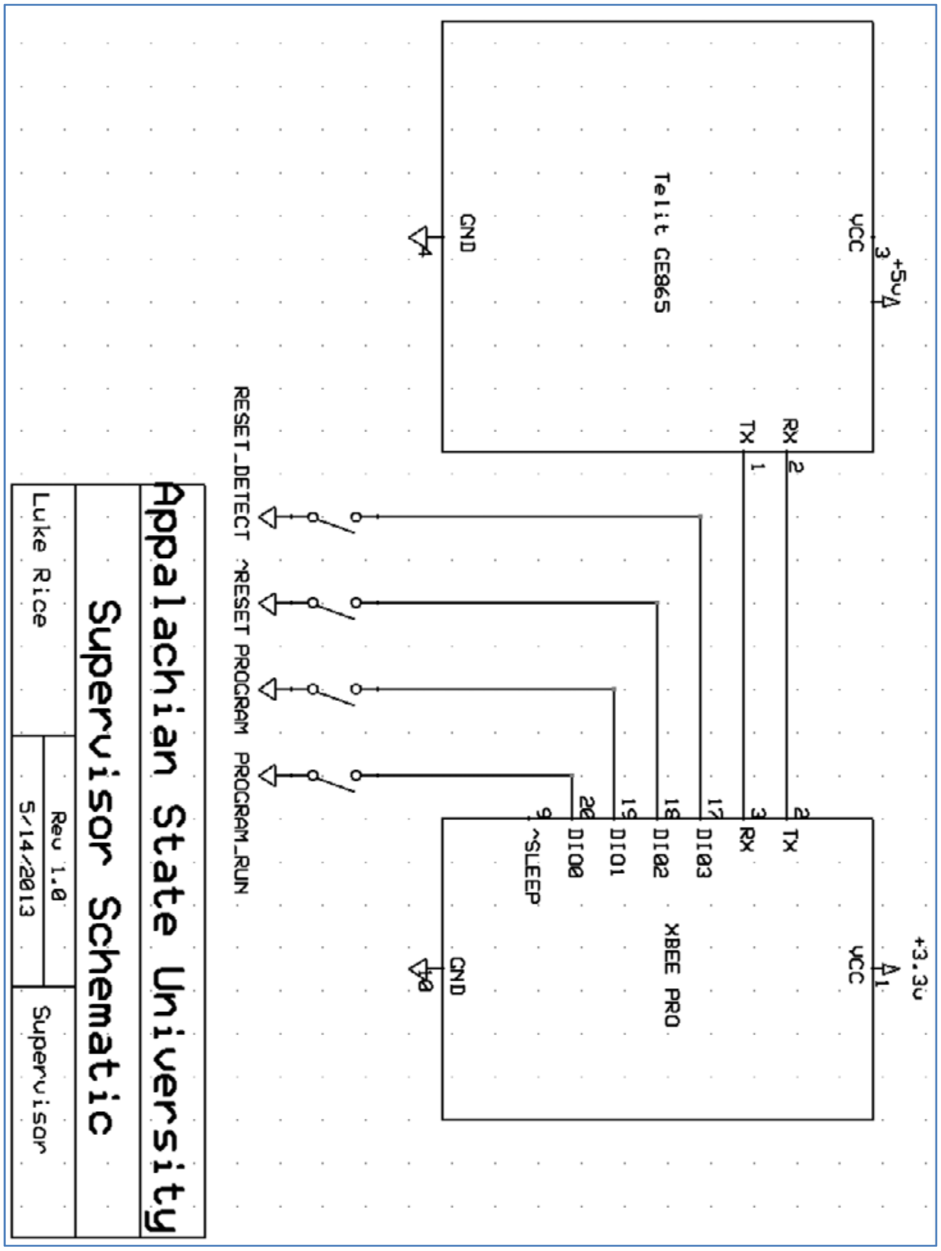

Figure B-1 - Supervisor Schematic

Appendix C - Web System Resources

The source code for the Web System can be found on the attached CD or on github at the URL: https://github.com/ricela1/MastersThesis2013. $ROOT refers to either the root of the CD resources or the top level directory of the github repository. The source code for the Web System is located at $ROOT/Web_System/. Table C-1 shows the filenames and descriptions of each source file used in the Web System.

Table C-1 - Web System Source Code Reference

Filename and Location	Description
processing_script.php	Processing script – Responsible for receiving GET request with sensor data, parsing sensor data and storing it in COSM and database
MiddlwareServer.java	Middleware Server – Responsible for receiving initial connection from the Supervisor and accepting string of CSV data, checking the data for errors and then transmitting it to the Processing script.
data_access.php	Data Access and Graphing pt 1 – Responsible for the front-end and graphing functionalities – mostly HTML and javascript with some minor PHP.
data_handler.php	Data Access and Graphing pt 2 – Responsible for responding to AJAX request from data_access.php and returning a formatted JSON object with the requested data.

VITA

Luke Rice was born in Marshall, North Carolina to Candace and Steve Rice in 1985. He holds an A.A.S. degree in Computer Programming from Asheville Buncombe Technical Community College, a B.S. in Computer Science from Appalachian State University in 2011 and a M.S. in Computer science from Appalachian State University in 2013. Luke will pursue a career in the computing field after graduation.