Solar-Powered Wireless Sensor Nodes with Dynamic Power Management for Indoor Use

Ethan Charles Humphrey

Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all

Part of the Computer Engineering Commons


https://corescholar.libraries.wright.edu/etd_all/702

This Thesis is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.
SOLAR-POWERED WIRELESS SENSOR NODES
WITH DYNAMIC POWER MANAGEMENT FOR INDOOR USE

A thesis submitted in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Engineering

By

ETHAN CHARLES HUMPHREY
B.S., Lipscomb University, 2011

2013
Wright State University
WRIGHT STATE UNIVERSITY
GRADUATE SCHOOL

April 22, 2013


Jack Jean, Ph.D.
Thesis Director

Mateen Rizki, Ph.D.
Chair, Department of Computer Science & Engineering

Committee on Final Examination

Jack Jean, Ph.D.

Yong Pei, Ph.D.

Meilin Liu, Ph.D.

R. William Ayres, Ph.D.
Interim Dean, Graduate School
ABSTRACT


Indoor environment monitoring, such as medical monitoring and home automation, is achieved through wireless sensor networks that collect data through wireless sensor nodes. These nodes are typically powered by exhaustible power sources that could be difficult or impossible to maintain regularly. A solution to the need for frequent maintenance is to collect the solar power in the environment.

This project shows how collecting the solar energy in the environment can extend the lifetime of a sensor node and limit the maintenance required. Along with charging the battery with a solar panel, a dynamic power management scheme can also be implemented. This scheme will monitor the power collected by the solar panel and adjust the duty cycle to a level where the sensor node power supply is self-sustaining.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>1.1 Thesis Motivation ........................................................................................................</td>
</tr>
<tr>
<td>1.2 Thesis Outline .............................................................................................................</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2.1 Wireless Sensor Network ............................................................................................</td>
</tr>
<tr>
<td>2.2 Rechargeable Batteries ...............................................................................................</td>
</tr>
<tr>
<td>2.3 Solar Panels ................................................................................................................</td>
</tr>
<tr>
<td>2.4 Dynamic Power Management .......................................................................................</td>
</tr>
<tr>
<td>2.4.1 Selective Switching of Hardware Components .......................................................</td>
</tr>
<tr>
<td>2.4.2 Dynamic Voltage and Frequency Scaling ..................................................................</td>
</tr>
<tr>
<td>2.4.3 Event Prediction Power Management .....................................................................</td>
</tr>
<tr>
<td>2.4.4 Discussion ..............................................................................................................</td>
</tr>
<tr>
<td>2.5 Related Work .............................................................................................................</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3.1 Hardware Components .................................................................................................</td>
</tr>
<tr>
<td>3.2 Hardware Connections ................................................................................................</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4.1 Introduction ...............................................................................................................</td>
</tr>
<tr>
<td>4.2 Arduino Development Environment .............................................................................</td>
</tr>
<tr>
<td>4.3 Sensor Readings .........................................................................................................</td>
</tr>
<tr>
<td>4.4 XBee Configuration and Communication ....................................................................</td>
</tr>
<tr>
<td>4.5 Dynamic Power Management Algorithm and Implementation ....................................</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>5.1 Hardware Components ...............................................................................................</td>
</tr>
<tr>
<td>5.2 System Performance ...................................................................................................</td>
</tr>
</tbody>
</table>
5.3 Improvements to the Dynamic Power Management Algorithm .................. 43
6 CONCLUSION ............................................................................. 45
   6.1 Summary ............................................................................. 45
   6.2 Future Work ......................................................................... 47
7 BIBLIOGRAPHY .......................................................................... 49
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Arduino Uno R3</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>XBee 802.15.4 (Series 1)</td>
<td>16</td>
</tr>
<tr>
<td>3.3</td>
<td>Power Supply</td>
<td>17</td>
</tr>
<tr>
<td>3.4</td>
<td>MAX756 Typical Application Circuit</td>
<td>18</td>
</tr>
<tr>
<td>3.5</td>
<td>TMP102 Digital Temperature Sensor Breakout Board</td>
<td>19</td>
</tr>
<tr>
<td>3.6</td>
<td>Typical Output Voltage VS Relative Humidity (At 25 degrees C and 5V)</td>
<td>20</td>
</tr>
<tr>
<td>3.7</td>
<td>HIH-4030 Humidity Sensor</td>
<td>20</td>
</tr>
<tr>
<td>3.8</td>
<td>PIR Motion Sensor</td>
<td>21</td>
</tr>
<tr>
<td>3.9</td>
<td>XBee Module and Explorer USB</td>
<td>21</td>
</tr>
<tr>
<td>3.10</td>
<td>XBee Series 1 to Arduino Uno Wiring Diagram</td>
<td>23</td>
</tr>
<tr>
<td>3.11</td>
<td>TMP102 to Arduino Uno Wiring Diagram</td>
<td>24</td>
</tr>
<tr>
<td>3.12</td>
<td>HIH-4030 to Arduino Uno Wiring Diagram</td>
<td>25</td>
</tr>
<tr>
<td>3.13</td>
<td>PIR Motion Sensor to Arduino Uno Wiring Diagram</td>
<td>25</td>
</tr>
<tr>
<td>3.14</td>
<td>Completed Prototype</td>
<td>26</td>
</tr>
<tr>
<td>3.15</td>
<td>Prototype without Boosting Circuit</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Arduino Development Environment Screenshot</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>X-CTU Modem Configuration Tab Screenshot</td>
<td>33</td>
</tr>
<tr>
<td>4.3</td>
<td>Sample Data Transmission via XBee using X-CTU</td>
<td>34</td>
</tr>
<tr>
<td>4.4</td>
<td>Solar Panel Collected Power by Hour, April 3, 2013</td>
<td>37</td>
</tr>
</tbody>
</table>
Figure 4.5 - Voltage and Power Relationship of Solar Panel ........................................... 38
Figure 5.1 - System Diagram of Proposed Design ............................................................. 43
LIST OF TABLES

Table 2.1 - Rechargeable Battery Comparison ................................................................. 5
Table 4.1 - Power Consumption of Wireless Sensor Node Components ........................ 38
Table 5.1 - Power Consumption of Optimized Design Components ............................. 42
ACKNOWLEDGEMENT

I would like to thank all the people involved in supporting me in the research, development, and defense of my thesis. The most influential person in this project was my thesis advisor, Dr. Jack Jean. I appreciate his support and knowledge which helped me throughout this project. I would also like to thank my committee members, Dr. Meilin Liu and Dr. Yong Pei, for their advice and evaluation of my thesis during my defense. Finally I would like to thank my family and my fiancé, Elizabeth, for their love and support throughout the entire course of completing my thesis.
1 INTRODUCTION

1.1 Thesis Motivation

Wireless sensor nodes are used indoors in many applications today including medical monitoring and home automation. Sensor nodes in many of the networks use an exhaustible power supply, which may require frequent maintenance or replacement of the sensor node. While connecting the sensor to the main power grid is a solution, it is often not feasible or possible in preexisting buildings.

The solution to improving the longevity of these sensor nodes is to capture the solar energy in the environment. Collecting the solar energy from the environment where it is available can extend the life of the sensor node power supply substantially, and lower the need for maintenance of the battery. As technology grows, the improvement of battery, sensor, and solar panel equipment will increase the performance of these wireless sensor networks.

A dynamic power management algorithm can also significantly improve the power efficiency of the solar-powered wireless sensor node. Adjusting the power consumption of the sensor node based on the power collected by the solar panel allows for the sensor node to use the available power in a manner which will make the product sustainable over long periods of time without any required maintenance.
1.2 Thesis Outline

Chapter 1 introduces the motivation behind the solar-power wireless sensor node with dynamic power management. In chapter 2, the background information required for designing the sensor node is discussed, including, solar panels, rechargeable batteries, wireless sensor node components, dynamic power management schemes, and related work to the solar-powered sensor node.

Chapter 3 discusses the hardware design of the wireless sensor node prototype, including the descriptions of the components and the connections of the components in a completed system. The software development and implementation of the wireless sensor node is discussed in chapter 4. This chapter includes reading data from the sensors, the algorithm and implementation of the dynamic power management scheme, the methods of interfacing with the Arduino development environment, and the configuration and communication of the XBee modules.

Chapter 5 describes the proposed improvements in hardware design of the wireless sensor node from the original prototype. The conclusion is given in chapter 6, providing a summary of the project and possible improvements to the design and future work.
2 BACKGROUND

2.1 Wireless Sensor Network

The wireless sensor network is a system of wireless nodes equipped with various sensors which take readings of their surroundings and send the data back to a central node which is monitored by the user [1]. Wireless sensor networks can be used to monitor environmental and weather conditions, patient vital signs in hospitals, or in home automation systems.

Wireless sensor networks have advantages which make their use increasingly popular. A wireless sensor network can be deployed quickly in emergencies such as natural disasters without installation of large equipment [1]. Use of wireless sensor networks in dangerous environments will also keep humans out of harm’s way but can still get the needed environmental data. Wireless sensor networks also provide an economical method for long-term data collection [1]. Most of the cost of a wireless sensor network is in the installation and components. The cost of maintenance is lower than that of other data collection systems [1].

Sensor Node Hardware Components

1. **Sensors**: collects data specific to the wireless sensor node’s task such as temperature and humidity for outdoor and indoor environments, heart rate and
blood pressure in hospital patients, or many other conditions that need to be monitored.

2. **Analog-to-Digital Converter:** some sensors give an analog voltage value which will need to be converted to a digital value before the processor can interpret the data.

3. **Processor:** takes the readings given from the sensors and does any needed computation of the data.

4. **Memory:** stores temporarily the data collected by the sensors for any calculations and until it is sent to the central node. Also stores the needed program for the sensor node’s operation.

5. **Transceiver:** used to send and receive data and messages from other nodes and the central node.

6. **Power Supply:** supplies power to all of the components. The power supply typically consists of a battery which may be recharged by a solar panel or other power generator.

### 2.2 Rechargeable Batteries

Batteries all work in a similar way. The main components of a battery are the cathode, the anode, the separator, the electrolyte, and the collector. The cathode is connected to the positive terminal, the anode is connected to the negative terminal, and the separator is a barrier between the two that prevents them from touching while allowing charge to flow [2]. The electrolyte is the medium through which the charge flows. The final part, the collector conducts the charge to the outside of the battery [2].
A chemical reaction known as an oxidation reaction takes place between the anode and ions from the electrolyte releasing electrons. Another chemical reaction between the cathode, ions, and free electrons happens, known as a reduction reaction [2]. These two reactions create the flow of electricity. These reactions will continue until the materials of the anode and cathode are completely consumed [2].

Several characteristics are considered when choosing a battery for a wireless sensor node. The most important of these attributes are charge capacity, size, maintenance requirements, and cost. The characteristics best for a sensor node are high charge capacity, small size, no maintenance, and low cost. There are no battery technologies which meet all these criteria, so some compromises must be made. Each type of battery has its advantages and disadvantages in use with a wireless sensor node. The most common rechargeable batteries types are nickel cadmium, nickel-metal hydride, lead acid, lithium-ion, and lithium-ion polymer. The comparison of the different types of batteries can be seen in Table 2.1 [3] [4] [5] [6].

<table>
<thead>
<tr>
<th></th>
<th>Nickel Cadmium</th>
<th>Nickel-Metal Hydride</th>
<th>Lead Acid</th>
<th>Lithium-Ion</th>
<th>Lithium-Ion Polymer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy Density (Wh/kg)</td>
<td>45-80</td>
<td>60-120</td>
<td>30-50</td>
<td>110-160</td>
<td>100-190</td>
</tr>
<tr>
<td>Cycle Life</td>
<td>1500</td>
<td>300-500</td>
<td>200-300</td>
<td>500-1000</td>
<td>&gt;1000</td>
</tr>
<tr>
<td>Overcharge Tolerance</td>
<td>Moderate</td>
<td>Low</td>
<td>High</td>
<td>Very low</td>
<td>Low</td>
</tr>
<tr>
<td>Self-discharge / Month</td>
<td>20%</td>
<td>30%</td>
<td>5%</td>
<td>10%</td>
<td>~10%</td>
</tr>
<tr>
<td>Maintenance Requirement</td>
<td>30 to 60 days</td>
<td>60 to 90 days</td>
<td>3 to 6 months</td>
<td>None</td>
<td>none</td>
</tr>
</tbody>
</table>
1. **Nickel Cadmium (NiCd):** has a lower charge capacity per weight than all but the lead acid battery, can be charged quickly, is low cost, and has a high number of charge/discharge cycles. The NiCd batteries do require regular maintenance and has high self-discharge rate which makes it unlikely as a good battery for a wireless sensor node [4].

2. **Nickel-Metal Hydride (NiMH):** higher charge capacity, average cycle life. Still requires maintenance and has a high self-discharge. These characteristics and higher cost make this battery unlikely for use in wireless sensor networks [4].

3. **Lead Acid:** lowest charge capacity per weight, low cycle life and requires maintenance. The advantages are high overcharge tolerance and lowest cost but still not a good candidate for the sensor node [4].

4. **Lithium-Ion:** higher charge capacity than the previously given battery types, and have higher number of charging/discharging cycles. Lithium-ion batteries have a lower self-discharge rate than the other types and require no maintenance. This battery type does have low tolerance of overcharging and requires protection circuitry to cut off battery discharge once a minimum voltage is reached. The cost of these batteries is high than others. This type is a very good candidate for sensor nodes [4].

5. **Lithium-Ion Polymer (LiPo):** comparable to lithium-ion in charge capacity per weight, number of discharge cycles, self-discharge rate, and no maintenance requirements. LiPo does require protection circuitry to prevent the battery voltage dropping too low and becoming unsafe [7]. The main differences between the types are the construction of the battery and a small increase in cost. Lithium-ion
polymer batteries are able to be shaped to whatever is needed for the application because the electrolyte in the battery is gelled and does not need a rigid case to compress the electrodes together. This ability allows for thinner, smaller and lighter LiPo batteries than the lithium-ion [7].

2.3 Solar Panels

Solar panels are made up of groups of photovoltaic cells which convert photons from sunlight to free electrons. Each solar cell is made up of silicon which absorbs photons when placed in sunlight. The energy from the photons breaks the electrons free from their atoms creating hole. An electric field generated in the cell forces the free electrons to flow in a certain direction [8]. This electric field is created by adding impurities in the silicon. Some of the impurities give the silicon extra electrons, and some give the silicon extra holes, or lack of electrons. The result is two separate halves: one half positively charged, or P type, and one negatively charged, or N type [8]. The free electrons attempt to fill the holes but in the process create a barrier between the two halves of silicon and equilibrium is reached, resulting in the electric field [8].

Solar panels come in three different types. The three types come in many different sizes making it useful for many different applications. The cost of a solar panel is closely tied to its efficiency. The three types of solar panels are monocrystalline, polycrystalline, and amorphous thin fill. While these three types are all that are available currently, research is being done to create new solar panels with higher efficiency.

1. **Monocrystalline Solar Panel**: grown in a single, continuous crystal which is sliced into thin wafers. Monocrystalline are the most expensive to produce but...
are the most efficient, achieving up to 24.2 percent efficiency [9]. The higher efficiency allows for more power to be produced per area than any other type, helping to making up for the increased cost of production. This type of panel has a greater heat resistance, but is still fragile if hit by objects such as tree branches or debris carried by the wind. However, these solar panels can last for at least 25 years [10].

2. **Polycrystalline Solar Panel**: instead of being grown in single crystal, this type is formed by melting the silicon and pouring it into a mold. The molded block can then be cut into square wafers [10]. This type is similar to the monocrystalline in length of life, but also just as fragile. The production of these solar panels is much simpler to manufacture, giving them a significantly lower cost. The downside of this method of silicon molding is a decrease in efficiency, with a maximum efficiency of up to 19.3 percent [9].

3. **Amorphous Thin Film Solar Panel**: made by depositing silicon onto a sheet of glass or metal in a very thin layer. The separate cells in the solar panel can be deposited next to each other on the glass or metal instead of needing to be assembled as with the crystalline panels [10]. The use of the thin layers of silicon gives the ability to create a flexible solar panel by using plastic glazing [10]. The thin film panels can perform better in low light conditions, but in general are less efficient than crystalline types, achieving up to ten percent efficiency [9]. The other downside to these panels is they are less stable and can degrade over time [10].
2.4 Dynamic Power Management

The sensor node software contains both the code needed to take and perform computations of sensor readings and the code for managing the power used by the sensor node. The code for collecting data and completing computations changes for different applications. The goal of the power management part of the code is to use each component of the sensor node in the most efficient manner. This can be accomplished by controlling the power consumed by each component.

The need for improvement in the power efficiency of wireless sensor nodes has led to several techniques that increase the lifetime of the power supply of the wireless sensor node. Among these techniques are selective switching of hardware components, dynamic voltage and frequency scaling, and event prediction power management.

2.4.1 Selective Switching of Hardware Components

Wireless sensor nodes have a limited power supply and must work efficiently to conserve available power. The components in the sensor node only need to be active when they are needed for the current operation. When not being used, components can be put into a sleep mode in which they consume less power. Dynamic power management looks for the components of the sensor node that are idle or not being used to their full capacity and switches the components on and off to conserve power. The dynamic power management scheme must take into account the system’s workload and performance constraints to keep the sensor node operating at the minimum requirements [11].
Dynamic power management can be expanded to give a sensor node multiple modes of operations that gradually reduce the amount of power consumed. In the active state, all components are fully powered. In the lowest power state, only the absolute necessities are still active and the rest of the components are powered off or in a sleep state. Many of the components also have the capability to be put into a sleep state, which uses much less power and can be put back into active mode in a much shorter time period than if completely shut down.

2.4.2 Dynamic Voltage and Frequency Scaling

This method of increasing power efficiency regulates the voltage supplied to the different components, giving each component the optimal voltage for its operation. This can be done with fixed or variable regulators from the power source to the different components [12]. The consideration for these different types of regulators is the power savings for the different duty cycles. Another approach to the dynamic voltage and frequency scaling conserves energy based on the tasks. Each task of the sensor node has a priority which will determine how much each task will be scaled. The voltage and frequency are scaled in a way which will maintain the deadlines required for each task. The idle time of the node is eliminated and each task is stretched to take its place [13].

2.4.3 Event Prediction Power Management

The main goal of the event prediction power management is to provide each subsystem in the wireless sensor node with just enough power to carry out the task at hand. The strategy is to adjust the power used by the idle and underutilized hardware.
To adjust the power used by each component, the events of the surrounding environment are predicted by the sensor node in order to maximize efficiency. One algorithm to predict the events is the grey model. The grey model uses a small set of data to calculate the close mean generated series [14]. The series is recalculated continuously to adjust the predictions as time goes on [14]. The disadvantage to this method comes from large fluctuations in the data, since the data set used is small.

The prediction of events can also be done with wavelet neural networks. This uses five sleep states for each component. The most efficient method would be to put each component into the lowest power state, but it takes longer to get back to active mode and consumes more power to transition between those states. The way in which the sensor node transitions from one state to another uses set threshold time values for each state so the amount of time spent in each state is at least enough to make the energy saved greater than the energy spent to transition [15]. The network consists of three layers: the input, the wavelet layer, and the output layer. Learning samples are used to calibrate the neural network and the error is reduced by repeating the second and third level calculations until the mean squared error satisfies a given threshold. From this point on, the sensor node uses the predictions to put each component into the corresponding sleep state [15].

2.4.4 Discussion

The event prediction power management will not apply to this indoor wireless sensor network application because there is no need to predict events. The properties being measured are measured at a known regular interval. The methods of selective
switching of the components and voltage scaling can be combined in the indoor application. Each component in the node could possibly be put into a sleep mode when not collecting data and the voltage from the battery can be scaled to only supply the needed power to each component. Frequency scaling can be more difficult to implement in the indoor situation because adjusting the length of the task operation time adds complexity to the sensor node programming which may in the long run increase the power consumption of the sensor node. The tasks of the collecting data in the sensor node can be prioritized to only give power to the necessary sensors in cases of lower power input from the solar panel.

2.5 Related Work

Solar panel-powered wireless sensor nodes have been tested in varying outdoor light environments. Some different design strategies have been proposed to improve the lifetime of the sensor node. One method uses super-capacitors as the main storage unit of the sensor node, which provides a longer charge/discharge cycle life than rechargeable batteries [16] [17] [18]. The main problem with this design is the much lower energy density of super-capacitors, which is compared in [18]. This design provides the needed lifetime of systems used outdoors, where maintenance could be difficult or impossible. For indoor applications, a higher energy density would be preferable to the longer lifetime because of the easier maintenance and lower solar energy available and high need for increased storage capacity in darker environments.

The other design improvement proposed was a series-parallel combination of the solar panel cells [19]. The combination of the solar cells in parallel and series give an
increase in power produced for the sensor node. The increase in size of the solar panel through the parallel and series combination of the solar cells is acceptable in outdoor applications. This increase in size is not tolerable for indoor applications because of the limited space available for the sensor node and the visual appeal of the design which is important for home automation and other indoor applications.
3 HARDWARE DESIGN

3.1 Hardware Components

The main goal of the project was to develop a wireless sensor node powered by a battery and solar panel combination that uses a dynamic power management scheme to adjust the duty cycle of the sensor node and conserve power, increasing the lifetime of the sensor node. Due to time and cost constraints, the sensor node prototype was constructed from components readily available in small quantities and low cost. While more efficient components could be purchased, the higher efficiency components would only be available in large quantities, making that design only suitable for mass production. The readily available components will still demonstrate the main goal of the project.

The prototype used an Arduino Uno development board to allow for a simpler design and easier connection of sensors and the radio. The sensors used in this design are common for home automation, giving readings of temperature, humidity, and motion detection. The sensors and radio components were mounted on an Arduino shield, allowing for easy mounting on the Arduino development board. To power the Arduino Uno, a combination of solar panel and battery were used via a charging circuit.

1. **Arduino Uno Development Board**: the board is developed by Arduino, an open-source electronics prototyping platform, providing the designer with easy to use
software and hardware. The microcontroller is programmed using the Arduino programming language and the Arduino development environment.

The Arduino Uno R3 board is designed around the ATmega328 microcontroller. The microcontroller has fourteen digital input/output pins, six of which can be used as pulse width modulation (PWM) outputs, six analog inputs, a 16 MHz clock, a USB connection, a power jack, an in-circuit serial programming (ICSP) header, and a reset button. Everything needed to provide full functionality to the microcontroller is included [20].

2. **XBee 802.15.4 (Series 1):** uses the IEEE 802.15.4 networking standard for fast point-to-multipoint or peer-to-peer networking. The data rate for the XBee Series 1 module is 250kbps and has a range of 100 feet indoors or 300 feet in line-of-sight range. The module operates in the 2.4GHz frequency band. The supply
The voltage needed is 3.3V and requires 45mA to transmit and 50mA to receive. The module can also be configured for sleep mode, which uses less than 10uA [21]. This module was chosen because of its low power consumption. The data rate is sufficient for the sensor node application.

![XBee 802.15.4 (Series 1)](image)

3. **Power Supply**: consists of a battery, solar panel, and charging circuit. The battery used is a Lithium Polymer battery with 2000mAh capacity, with a nominal voltage of 3.7 volts. The battery contains built-in protection from overvoltage, undervoltage, and overcurrent. The max discharge rate of the battery is 4000 mA per hour, but the connectors are rated at 1A, so the adjusted discharge rate would be 1000 mA per hour [22]. This battery was chosen because of its small size and high storage capacity. The 2000mAh should provide enough power for nighttime conditions and still be operational by the next day when it can recharge. This battery was also readily available. The solar panel is monocrystalline with an
efficiency of 17 percent. The typical operating voltage is 5.5V, depending on the light intensity. The typical current in full sunlight is 540mA [23].

The choice of the solar panel was made based on the availability. This solar panel was readily available for use with the project, whereas many other applicable solar panels would need to be custom made and would take longer to receive and increase the cost for small quantities. The charging circuit used is the Lipo Rider v1.1. It has connections for the solar panel, battery, and load. The circuit provides a stable 5V to the load. The supply current maximum is 350mA and the charge current maximum is 600mA. The battery charge voltage is 4.2V [24]. The required input voltage from the solar panel is 4.4 to 6V.

The Lipo Rider has a limited input range, and with lower light environments the solar panel drops below 4.4V and the battery will no longer
charge. The power that the solar panel collects below that voltage is not used. To increase charging capabilities of the solar panel, a DC-DC converter circuit was added between the solar panel and the Lipo Rider to increase the voltage level, allowing the charging to continue. The DC-DC converter used was the MAX756. This integrated circuit requires a startup voltage of 1.8V and once started the voltage minimum is 0.7V. The stated efficiency of the IC is 87 percent with a load current of 200mA. Figure 3.4 shows the circuit used with the MAX756 [25]. The SHDN pin is connected to the input because as long as the input voltage is 1.6V or above, the SHDN pin registers a high signal which keeps the circuit active. When the SHDN pin goes low the IC shuts down. The 3/5 pin is attached to ground to set the output voltage of the MAX756 to 5V.

![MAX756 Typical Application Circuit](image)

**Figure 3.4 - MAX756 Typical Application Circuit**

4. **TMP102 Digital Temperature Sensor**: measures temperatures with 12-bit accuracy and 0.0625 degree Celsius resolution. The sensor requires a 1.4 to 3.6V
supply and approximately 10µA in active mode. Communication with the
TMP102 sensor is achieved through a two-wire serial interface [26]. This sensor
was chosen for its very low power consumption and high accuracy.

![Figure 3.5 - TMP102 Digital Temperature Sensor Breakout Board](image)

5. **HIH-4030 Humidity Sensor:** requires a 5V supply and consumes approximately
200µA. The output is a near linear analog signal. The reading of the relative
humidity sensor is within 3.5 percent of the actual value. Figure 3.6 shows the
typical output compared to relative humidity of two HIH-4030 sensors and the
best linear fit [27]. The humidity sensor requires little power and the data is easy
to interpret with the almost linear response, which is why it was chosen for the
sensor node.
6. **Passive Infrared (PIR) Motion Sensor**: requires 5V to 12V supply. The signal pin is an open collector, requiring a pull-up resistor on the pin and connected to 5V. On motion detection the signal is pulled to ground [28]. The sensor works with a combination of two sensors that takes images at two different times and compares the two images. If changes are present, then motion has been detected [29].
7. **XBee Explorer USB:** used to connect a second XBee Series 1 module to a computer for communication. This module gives access to the serial and programming pins on the XBee unit. Using the X-CTU software available from Digi International Inc. the board can be configured and the incoming data can be read.
3.2 Hardware Connections

The connection of each sensor and the radio was first tested on a solder-less breadboard for correct connections and proper operation. Once each component was working properly the circuits required were assembled and soldered onto an Arduino shield, which allowed for easy mounting and removal on the Arduino Uno microcontroller board. For the XBee module, a breakout board with sockets to adapt the 2mm spaced pins to the standard 0.1 inch (2.54mm) spaced pins was soldered to the shield, removing the need to solder the more expensive component to the Arduino shield and allowing for changes to the programming on the XBee module.

1. **XBe 802.15.4 (Series 1) Wiring**: the connections from the Arduino to the XBee module are shown in Figure 3.10. The red and black wires show the 3.3V supply and the ground connections between the two, respectively. The TX pin on the Arduino is connected to the DIN pin on the XBee module and the RX pin on the Arduino is connected to the DOUT pin on the XBee. The SLEEP_RQ pin or pin 9 on the XBee is connected to digital pin 13 on the Uno board. This pin is active low, so when the pin is asserted, the XBee enters sleep mode and when the pin is de-asserted the module returns to active mode.
2. **TMP102 Temperature Sensor Wiring:** the connection of the TMP102 temperature sensor to the Uno board is shown in Figure 3.11. The 3.3V supply of the sensor is connected to the Arduino via the red wire and the ground pin is the connection from the bottom pin to the Uno ground. The TMP102 is I2C device, meaning it uses a two wire connection. There is a data line (SDA) and a clock (SCL). The SDA and SCL lines of the sensor are connected to the SDA and SCL pins on the Arduino Uno board, shown as the blue and yellow wires, respectively. The black wire connected to the top pin of the TMP102 sensor sets the address. By grounding the wire, the address is set to 0x48. If the wire was connected to the 3.3V of the Uno, the address would be set to 0x49.
3. **HIH-4030 Humidity Sensor Wiring**: the wiring of the humidity sensor is shown in Figure 3.12. The red wire connects the 5V of the Arduino Uno board to the supply input pin on the sensor. The black wire connects the ground of the sensor to the ground pin of the Arduino. The blue wire is the signal wire for the sensor, which is connected to analog pin zero on the Uno board. A 100k resistor is placed between the sensor ground and signal pins for a load on the signal pin.
4. **PIR Motion Sensor Wiring**: the wiring of the PIR motion sensor is shown in Figure 3.12. The red wire connects the 5V supply on the Uno board to the supply wire of the sensor. The yellow wire connects the grounds of the Uno and the sensor. The signal pin, shown by the black wire of the motion sensor is connected to digital pin 12 of the Arduino Uno. The signal pin is an open collector, so the
digital pin 12 is configured as an input with an internal pull-up resistor. When motion is detected, the signal pin is connected to ground and the signal goes low.

5. **Power Supply**: the solar panel connects to the MAX756 circuit via JST connector with another JST connector to connect to the Lipo Rider. The solar panel with the MAX756 circuit and the battery connect via JST connectors on the Lipo Rider board. Each connector on the Lipo Rider is marked to signify where to connect the solar panel and battery. The Lipo Rider board connects to the USB jack on the Arduino Uno via a USB 2.0 A male to B male cable. The monitoring of the solar panel voltage for the dynamic power management is achieved by connecting a wire from the output of the MAX756 circuit to analog pin 2 on the Arduino board.

The initial completed prototype can be seen in Figure 3.14. The shield is mounted on the Arduino Uno board by seating together the female headers on the Uno board and the male headers on the shield. The power supply is connected to the Arduino Uno via the USB cable. After testing the boosting circuit with the MAX756 integrated circuit to

![Figure 3.14 - Completed Prototype](image)
boost the voltage of the solar panel up to a usable level for the battery charger, it became apparent that the boosting circuit was very inefficient. The power coming from the solar panel was reduced by as much as 75% when the boosting circuit was connected. The problem with the circuit was that it was attempting to pull more current from the solar panel than available, causing the voltage to drop to approximately one volt and resulting in a drop in efficiency. The remainder of the testing of the sensor node was completed with the boosting circuit removed and the solar panel connected directly to the Lipo Rider board as seen in Figure 3.15.

Figure 3.15 - Prototype without Boosting Circuit
4 CODE DEVELOPMENT AND IMPLEMENTATION

4.1 Introduction

This section includes the description of the code used with the sensor node, including the reading of the sensors and the dynamic power management algorithm and implementation. Configuration of the XBee radio and reading of the data sent is also described. For the Arduino, the uploading process will be discussed. The Arduino development software is used to write and upload the code to the Arduino Uno board. The X-CTU software, provided by Digi, is used for configuring and reading the data from the receiving radio.

4.2 Arduino Development Environment

The development environment provided by Arduino uses a simple text editor for entering code for the Arduino. Setting up the environment requires the user to set the type of Arduino board being used and the communication port being used by the Arduino. A USB 2.0 A male to B male cable is used to connect the Arduino Uno to the computer. The development environment also includes a serial monitor to monitor the serial communication of the board when a program is being run, making testing of the program easier because there is no need to use the radio each time the code is tested.
Once the code is compiled and verified, the code can then be uploaded to the Arduino board. A screenshot of the Arduino development environment can be seen in Figure 4.1.

![Arduino Development Environment Screenshot](image)

**Figure 4.1 - Arduino Development Environment Screenshot**

### 4.3 Sensor Readings

The TMP102 temperature sensor uses the I2C two-wire interface to communicate with the Arduino Uno. The sensor has its own 7 bit address which is used to read and write data to the sensor. The addresses of the devices on the I2C two-wire interface allow for multiple devices to be connected to the two lines and each device can be
accessed with its own unique address. The Arduino software includes the Wire library, which contains all the functions needed to communicate with devices on the I2C lines.

The address set on the TMP102 sensor is 0x48. The function I wrote to read the temperature is shown below. The requestFrom() function from the Wire library uses two inputs: the address and the number of bytes to be sent back. Using another Wire library function, the two bytes are read back, first the most significant byte of data and then the least significant byte. Theses bytes are then concatenated giving the number of 0.0625 degree intervals in the read temperature. To get the temperature all that needs to be done is to multiply the number of intervals by 0.0625 and the result is the temperature in degrees Celsius.

```c
//Function to read temperature
float getTemperature(){
    //Tell the sensor to send back 2 bytes
    Wire.requestFrom(tmp102Address,2);

    //Read two bytes
    byte MSB = Wire.read();
    byte LSB = Wire.read();

    //Concatenate the two bytes together
    int tempReading = ((MSB << 8) | LSB) >> 4;

    //Calculate Celsius temperature
    float celsius = tempReading*0.0625;
    return celsius;
}
```

The next sensor that needed to be read from is the humidity sensor. This sensor is much simpler to read, where the output of the sensor is a voltage that can be read on an analog pin on the Arduino board. The Arduino board returns a value from 0 to 1023 based on the voltage on the analog pin. This value can then be converted to a voltage by dividing by 1023 and multiplying by the supply voltage, which in this case is five. To calculate the sensor relative humidity the following equation [27] provided in the datasheet is used:
\[ V_{out} = (V_{supply})(0.0062(sensor \ RH) + 0.16) \]  \hspace{1cm} (1)

The equation must be solved for the sensor relative humidity. The voltage \( V_{out} \) is the voltage read on the Arduino board and the voltage \( V_{supply} \) is the voltage supplied to the sensor, or five volts. This result must then be entered into another equation [27] provided by the datasheet to calculate the true relative humidity using the temperature reading from before:

\[ True \ RH = (Sensor \ RH)(1.0546 - 0.00216T), \text{T in degrees C} \]  \hspace{1cm} (2)

The code showing the function which reads and interprets the humidity sensor data is shown below.

```c
//Function to get relative humidity
float getRH(float celsius)
{
  //Read digital value
  int analogVal = analogRead(A0);

  //Convert to voltage
  float voltage = analogVal * 0.0049;

  //Calculate the sensor relative humidity
  float sensorHumidity = ((voltage / 5.0) - 0.16) / 0.0062;

  //Calculate the actual relative humidity
  float humidity = sensorHumidity / (1.0546 - 0.00216 * celsius);
  return humidity;
}
```

The final sensor used in the wireless sensor node is the PIR motion sensor. The output of this sensor is a one bit digital signal. When motion is detected the pin is grounded, resulting in a zero being read. When no motion is detected, the pin stays high. Reading this value on the Arduino board is simple. The digitalRead() function is used by passing a pin number as a constraint. The sensor must be frequently checked to get
accurate readings of motion detection. The motion sensor takes one to two seconds to take initial infrared pictures of the environment, during which the output pin could go low. Each time motion is detected this process will happen again. The true data cannot be taken until after this time has expired. Another method to read the pin without having to wait the two seconds is to ignore the false readings. This is done by keeping a counter of the number of time motion is detected and if the number exceeds a certain limit, then motion was truly detected.

4.4 XBee Configuration and Communication

The wireless sensor node uses an XBee Series 1 module configured for sleep mode and the computer end XBee module is used with the default configuration. The software used to configure the XBee module, X-CTU, is provided by Digi International, Inc. and can be downloaded from their website for free. Using the XBee USB Explorer and a male mini-USB to male USB cable, the XBee module can be connected to the computer. From the PC Settings tab, the correct communication port for the XBee module must be selected to set the configuration of the XBee module. Also in this tab the baud rate, parity bits, and flow control can be selected. These settings are left as default for both XBee modules. In the Modem Configuration tab, the configuration of the XBee takes place.

Reading the current settings on the XBee module can be done by clicking the Read button. A sample screenshot of the software in the Modem Configuration tab can be seen in Figure 4.2. There are several settings options available with the XBee module, but the only one needing to be changed for the XBee connected to the wireless sensor...
node is the Sleep Mode setting. This setting will be changed to setting 1 – Pin Hibernate. The setting allows the module to go to sleep when the SLEEP_RQ pin on the XBee module is asserted high. When the pin is asserted low, the XBee module wakes up. The XBee module on the computer side of the wireless sensor network is kept with the default setting of no sleep mode. Writing of the settings to the XBee module is done by clicking the Write button in the Modem Configuration tab.

![Figure 4.2 - X-CTU Modem Configuration Tab Screenshot](image)
Once the two XBee Series 1 modules are configured, communication can begin. For this prototype, the XBee on the wireless sensor node is kept in sleep mode until there is data to send. The data being sent to the XBee connected to the computer can be read out in the Terminal tab of the X-CTU software. First, the correct communication port for the XBee module must be chosen in the PC Settings tab. Back in the Terminal tab, the XBee is ready to receive data. A sample view of the Terminal tab with data sent can be seen in Figure 4.3.

![Sample Data Transmission via XBee using X-CTU](image)

**Figure 4.3 - Sample Data Transmission via XBee using X-CTU**
4.5 Dynamic Power Management Algorithm and Implementation

The dynamic power management algorithm designed for the wireless sensor node uses the selective switching of hardware components, because no events are being predicted and the data sampling of the environment takes place at predefined times. The algorithm uses the voltage from the solar panel to adjust the duty cycle of the wireless sensor node. The change in duty cycles cause the sensor readings to be taken and the radio sending data more or less frequently depending on the change in duty cycle. Between the sensor reading and sending of the data the idle components are put into a sleep mode to conserve power.

The algorithm acts based on the voltage of the solar panel, so a relationship between the voltage and total power collected by the solar panel must be established. The voltage and current of the solar panel can be measured at regular intervals throughout the daylight hours giving the amount of power collected by the solar panel throughout the day. Using a program such as Microsoft Excel, a trend line can be created correlating the voltage to the power being collected at any particular point in time.

The next step was to develop an equation which will give us the required duty cycle based on the voltage read from the solar panel. The total solar panel power collected throughout the day must be equal or greater than the power used during the daylight hours plus the power used at night to sustain the wireless sensor node. The duty cycle at night is a constant value, because the solar panel will no longer be producing enough power to charge the battery. This nighttime duty cycle is set to around five to ten percent, because at this point the change in power consumed as the duty cycle goes lower is negligible. During the daylight hours, the duty cycle changes based on the solar panel
power, which is calculated from the voltage using the previously discussed trend line equation. The duty cycle is calculated based on the specified nighttime power, the active power during the day, the sleep power during the day, the power collected by the solar panel and the hours of light and dark in the day, $T_{light}$ and $T_{dark}$, respectively. The following equation shows the relationship:

\[
P_{solar\ panel} \times T_{light} = \left((duty\ cycle)(P_{active}) + (1 - duty\ cycle)(P_{sleep})\right) \times T_{light} + P_{night} \times T_{dark}
\]  

The calculation of the duty cycle can be done using this equation. Solving for the duty cycle gives the following equation:

\[
duty\ cycle = \frac{P_{solar\ panel} - P_{sleep} \times T_{light} - P_{night} \times T_{dark}}{(P_{active} - P_{sleep}) \times T_{light}}
\]  

When calculating the duty cycle, a minimum duty cycle around five to ten percent should be set because at that point, the difference in the power consumed is very little. The final part to the algorithm is to give each sensor priorities which will determine if the sensor will be used or not at a certain duty cycle. This priority can be based on a number of requirements. For some applications, the priority can be set based on the need of the data from that sensor. In other applications, the priority can be based on the type of sensor. Some sensors can take a single reading and be finished, while others have to sample continuously in order to get accurate readings. With different types of sensors, the ones with a single reading required have a higher priority than the other.

The implementation of this algorithm on the prototype started with the testing of the solar panel to find the power collected during the daylight hours. Samples were taken
every hour between sunrise and sunset for one day. Figure 4.4 shows the power collected over time by the solar panel. This sampling was done in Cedarville, OH on April 3, 2013. The solar panel was placed facing a southern facing window indoors.

The voltage of the solar panel compared to the power collected can be seen in Figure 4.5. The calculated trend line, a fourth order polynomial, is shown on the graph. The resulting equation of the trend line is:

\[ P = 54.709V^4 - 978.81V^3 + 6571.3V^2 - 19596V + 21892 \]  

(5)

Using the voltage read from the analog input on the Arduino board, the current power from the solar panel can be calculated. The average power collected by the solar panel throughout the day is 57.23 mW. The nighttime, sleep, and active power consumption values are constant values.
These values for the each component of the prototype can be found in Table 4.1. For the nighttime power, I will be using a five percent duty cycle. The minimum duty cycle for the daytime operation will also be set to five percent. The amount of time between the sunrise and the sunset is approximately twelve hours per day in Dayton, OH [34], so the $T_{light}$ and $T_{dark}$ would both be set to twelve hours. The duty cycle of the system can then be calculated again after each sleep period of the sensor node.

<table>
<thead>
<tr>
<th>Component</th>
<th>$P_{active}$</th>
<th>$P_{sleep}$</th>
<th>$P_{night}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino Uno</td>
<td>248.46 mW</td>
<td>209.01 mW</td>
<td>211.07 mW</td>
</tr>
<tr>
<td>XBee Series 1</td>
<td>147.15 mW</td>
<td>0.03 mW</td>
<td>7.39 mW</td>
</tr>
<tr>
<td>PIR Motion Sensor</td>
<td>12.25 mW</td>
<td>12.25 mW</td>
<td>12.25 mW</td>
</tr>
<tr>
<td>HIH-4030 Humidity Sensor</td>
<td>0.98 mW</td>
<td>0.98 mW</td>
<td>0.98 mW</td>
</tr>
<tr>
<td>TMP102 Temperature Sensor</td>
<td>0.03 mW</td>
<td>0.03 mW</td>
<td>0.03 mW</td>
</tr>
<tr>
<td>Total</td>
<td>408.87 mW</td>
<td>222.30 mW</td>
<td>231.72 mW</td>
</tr>
</tbody>
</table>
Comparing the power collected by the solar panel and the power consumed by the wireless sensor node components, the sensor node consumes almost four times as much power as the solar panel produces. With the 2000 mAh battery operating at 3.7V, the system running at five percent duty cycle will only be able to operate approximately one and a half days. The problem with this prototype is the use of the Arduino Uno board. While the processor itself has a sleep mode which only consumes less than 5 mW, the Uno board has extra electronics that are not being used, but are still consuming power. A new prototype design can solve this issue with high efficiency components and no extra unused components to waste power. In section five, a new, more efficient prototype is proposed.
5 PROPOSED WIRELESS SENSOR NODE DESIGN

5.1 Hardware Components

The prototype used in the initial testing showed that the dynamic power management algorithm works, but because of the all-purpose Arduino Uno board with extra components not needed by the wireless sensor node, the power consumption of the first prototype was too high for a sustainable product. Components with higher efficiency can be used to replace some of the less efficient components in the initial design. The components needing to be replaced are the solar battery charger and the microcontroller board. The same sensors can be used because they use very little power. Another improvement comes with the battery. The combination of more efficient components can improve the sustainability of the wireless sensor node power supply.

1. **Solar Battery Charger:** the old charger had a limited input voltage range, which left the system wasting power from the solar panel when below that range. The replacement for this product is an integrated circuit, the BQ24210, produced by Texas Instruments. This IC allows for input from 3.5 V to 7 V, so the design will charge the battery effectively. When the voltage drops below that point, the solar panel is only able to collect less than 1mA, which is not a significant loss. The higher voltage range also allows for higher voltage solar panels to improve the collected power of the solar panel. The IC also includes the needed circuit protection for charging lithium-ion batteries [30].
2. **Microcontroller:** the Arduino board included many unused components that were wasting power during the normal operation of the wireless sensor node. The improvement to the system is to remove the ATmega328P used on the Arduino and use it on its own. The microcontroller operates at 3.3V or 5V. For this design, 3.3V is used. At 3.3V, the microcontroller clock works at 4 MHz, and consumes 2.5mA in active mode and 0.9μA in power-save mode [31]. The other components needed from the Arduino Uno board are the 5V and 3.3V regulators. The 5V regulator to use is the LT1173, a DC/DC Converter. This IC will take the voltage from the battery and converts to 5V using only 110μA of the supply current [32]. Only three external components are required for operation. The 3.3V regulator is a TLV70033, which will take the 5V from the other regulator and output 3.3V. The quiescent current of the IC is only 31μA, making it good for power-sensitive applications [33].

3. **Battery Capacity:** the battery capacity of the wireless sensor node can be optimized in order to reduce size and weight of the battery. The battery only needs enough power capacity to run during the time that the solar panel cannot collect enough power to offset the power consumed by the wireless sensor node. In order to find the optimized battery size, the active, sleep, and nighttime power values need to be calculated for the new components. The new values are shown in Table 5.1. The amount of time between the sunrise and the sunset is approximately twelve hours per day in Dayton, OH [34], where the solar panel was tested, so there are also twelve hours of dark. To keep the wireless sensor node operating in the dark, the battery needs to supply 25.42 mW per hour for
twelve hours, or a total of 305 mWh capacity. With the change in weather and seasons, and also loss of capacity of a battery over time, a higher capacity battery will need to be used, around 500 to 600 mWh.

<table>
<thead>
<tr>
<th>Component</th>
<th>$P_{active}$</th>
<th>$P_{sleep}$</th>
<th>$P_{night}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATmega328P</td>
<td>5.61 mW</td>
<td>0.003 mW</td>
<td>0.28 mW</td>
</tr>
<tr>
<td>XBee Series 1</td>
<td>147.15 mW</td>
<td>0.03 mW</td>
<td>7.39 mW</td>
</tr>
<tr>
<td>PIR Motion Sensor</td>
<td>12.25 mW</td>
<td>12.25 mW</td>
<td>12.25 mW</td>
</tr>
<tr>
<td>HIH-4030 Humidity Sensor</td>
<td>0.98 mW</td>
<td>0.98 mW</td>
<td>0.98 mW</td>
</tr>
<tr>
<td>TMP102 Temperature Sensor</td>
<td>0.03 mW</td>
<td>0.03 mW</td>
<td>0.03 mW</td>
</tr>
<tr>
<td>BQ24210 Battery Charger</td>
<td>4 mW</td>
<td>4 mW</td>
<td>4 mW</td>
</tr>
<tr>
<td>5V Regulator</td>
<td>0.33 mW</td>
<td>0.33 mW</td>
<td>0.33 mW</td>
</tr>
<tr>
<td>3.3V Regulator</td>
<td>0.16 mW</td>
<td>0.16 mW</td>
<td>0.16 mW</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>170.51 mW</strong></td>
<td><strong>17.78 mW</strong></td>
<td><strong>25.42 mW</strong></td>
</tr>
</tbody>
</table>

5.2 System Performance

The new components are connected as in Figure 5.1. The solar panel and battery connect to the BQ24210 solar battery charger. The 5V regulator is connected in parallel with the battery, boosting the 3.7V of the battery up to 5V. From the 5V line the PIR motion sensor and HIH-4030 humidity sensor are powered. The 3.3V regulator is also connected to the output of the 5V regulator. The output of the 3.3V regulator is then connected to the TMP102 temperature sensor and the ATmega328P microcontroller.

The calculated power used at five percent duty cycle is much lower in this new design. With the average solar panel collected power at 57.23 mW per hour, the average duty cycle of the system during the day can be calculated. Using the five percent duty cycle at night and twelve hours for both $T_{light}$ and $T_{dark}$, the average duty cycle during the
day is 9.19%, meaning that the wireless sensor node sustainable. For a frame of reference, if the active time required taking samples of the temperature, relative humidity, and motion is approximately 50 ms, then samples will be taken about every 550 ms, which is much faster than is necessary for many applications. Adjustments can be made to the duty cycle to increase the efficiency.

![System Diagram of Proposed Design](image)

**Figure 5.1 - System Diagram of Proposed Design**

### 5.3 Improvements to the Dynamic Power Management Algorithm

The hours of daylight changes throughout the year, so using a constant value for the light time and dark time may not allow the wireless sensor node to be self-sustaining. The improvement that can be made with this algorithm is to allow the light and dark time to change. The values can be determined by creating another trend line equation that
correlates to the length of time between sunrise and sunset as the time of year changes to
the day of the year, from zero to 365. This would require that the time of year can be
determined by the wireless sensor node and while the ATmega328P microcontroller can
keep track of time elapsed, it has no way of synchronizing to the current time and date
without external hardware. The date can be sent from the central node to the wireless
sensor node to help the sensor node determine how long the time is between sunrise and
sunset.

The other improvement proposed for the algorithm comes from the estimated
power consumed. The calculation of the duty cycle does take into account any extra
expenditure of power by the components. The error in the calculation of the active and
sleep power can cause the wireless sensor node to prematurely exhaust its power supply.
The solution to this problem is simple. The error can be considered by lowering the
calculated duty cycle by a certain percentage that will allow for variations in the power
consumed by the sensor node.
6 CONCLUSION

6.1 Summary

Applications such as home automation where wireless sensor networks are used in indoor environments need products such as the prototype designed in this paper. When installing home automation systems, a connection to a power grid may not be available or feasible or cost effective to connect with. In these situations, a solar powered wireless sensor node can be the solution. With just batteries, regular maintenance is required to recharge or replace batteries, but with the solar powered system, the only maintenance required is possible replacement of batteries after the long period of time when the batteries capacity has decreased so much so that it no longer supports regular operation. The lithium-ion batteries can last for over one thousand charge and discharge cycles before they are reduced to 80% capacity.

The design of the wireless sensor node prototype included the sensor node hardware design and a dynamic power management algorithm to conserve power. The initial design used an Arduino Uno microcontroller board. The sensors used were a temperature, humidity, and motion detection sensor. Along with an XBee radio module, the sensors are soldered onto a shield which can be easily mounted on the Arduino board.

The initial design made from readily available, inexpensive components showed the improvement made in efficiency using the dynamic power management algorithm, which uses the voltage of the solar panel to adjust the duty cycle of the sensor node. The
relationship of the voltage and the power collected by the solar panel was found by sampling voltage and power values and using a program such as Microsoft Excel to find an equation of a trend line fitting the data. The duty cycle is adjusted by measuring the voltage of the solar panel and use the trend line equation to find the corresponding power collected. Using the power value, the duty cycle can be set in a manner that the amount of power consumed by the sensor node is low enough to allow for the wireless sensor node to be sustainable. This sensor node design was not sustainable because of the extra components on the Arduino Uno board that were not needed but still consumed significant amounts of power. The solution provided for this is a new proposed design for the wireless sensor node.

The next step in this project was proposing a new hardware design that uses less power than the original and makes the wireless sensor node a sustainable solution. The replaced parts were the microcontroller board, the battery charging circuit, and the battery. The microcontroller used in the proposed design is the same one used in the Arduino Uno. The ATmega328P can be used without the extra components used on the Uno board. Along with the microcontroller, a 5V regulator and 3.3V regulator are needed to power the microcontroller, radio, and sensors. The battery charging circuit in the new design has a wider charging voltage range, allowing for a longer charging time from the solar panel. The charger also includes the circuit needed to properly charge a lithium-ion battery.

The battery used in the built wireless sensor node design was just a typical single-cell battery, with a 2000m Ah capacity. A new battery can be used with only enough capacity needed for the design, lowering the space and weight of the final design. The
required capacity need for the new proposed design can be in the range of 500-600 mWh, or with a battery of 3.7 V, 150 to 200 mAh. The battery capacity gives enough power to supply the power needed when no solar power is present and allow for a decrease in battery capacity over time.

6.2 Future Work

The prototype proposed in section five has some improvements that must be made before this product is ready for commercial use. The list below contains some of the improvements for the proposed design.

- The proposed design needs to be tested and built using the dynamic power management algorithm. A custom circuit board can be designed for this product and enclosed in a plastic case for circuit protection. The components in this design can be purchased in surface mount IC which will make the proposed design a small board to enclose.

- The battery capacity can be optimized for the proposed design also. A smaller and lighter battery will improve the size of the case need to enclose the device. The battery can also be replaced with a super-capacitor based power supply if the energy capacity of super-capacitors is improved in the future. The super-capacitor design will allow for more charge/discharge cycles before maintenance is required.

- The solar panel of the design can be changed as well. Solar panels can be combined in series and parallel to increase the power supplied to the wireless
sensor node. The solar panel tested in this project worked at 17% efficiency and more efficient solar panels are available.

- The sensors used in this design were typical sensors that can be used in home automation applications. Other sensors can be added or removed from this design for any applications.
7 BIBLIOGRAPHY


