Smart Saver's Switch

Using Wireless Sensor Networks as a Power Control

Client: Dr. Qi Han, Alan Marchiori

Team: Arka Barman, Shanley Philip, and Cole Sobotka

Abstract

The “Saver Switch” program is currently offered by Xcel Energy to cut down on energy usage. The participants of the program have their air conditioners turned off by Xcel for 15 to 20 minute intervals depending on their respective power grid. This decreases peak energy demand, thus taking away the need to build additional power plants and transmission lines.

However, this method does not take into consideration individual household temperatures. An approach to this problem is to create “Smart” Saver Switches. By using wireless sensor networks (WSN’s), multiple air conditioners can be coordinated to reduce peak energy even further while keeping each household at a comfortable temperature. With a choice between two different algorithms, the wireless sensors are able to gather temperature data from the households and determine which air conditioners to turn on and shut off.

During the course of the CSM3 project, a distributed algorithm and a central algorithm have been developed to test against a regular thermostat program. These algorithms have been implemented on three environmental enclosures that mimic the average household. There are a total of ten environmental enclosures, but only three were needed for experimentation purposes. The results of the 13+ hour tests are compared and contrasted in the report.

Table of Contents

Introduction to the Project

Currently, Xcel energy employs a program called the "Saver's Switch." In this program, participants have their air conditioning system turned off for 10 to 15 minute intervals depending on the respective power grids. The goal of this program is to reduce the peak energy usage during the highest demand hours of the day. Although the current system aims to reduce peak energy, it does not consider the temperatures of the individual households in a power grid. This can lead to uncomfortably high temperatures in a participant's home.

In order to resolve this problem, wireless sensor nodes (within an AC unit or house) can be used to make smarter choices regarding the activation of the air conditioning in a grid of homes. As a result, the peak energy usage will be reduced while maintaining a comfortable temperature. The tasks of the wireless sensor nodes will include reading the current temperature, creating an information packet containing the node ID, the temperature reading, and other relevant information, and then sending the created packet over the radio. The goal of this project is to have two different possible algorithms that will reduce energy usage while maintaining the average temperature in the household. Since the wireless sensor nodes cannot be directly installed into households, experimental enclosures with similar heating and cooling curves will be created to allow for simple testing.

The two different algorithms that will be used are the distributed algorithm and the centralized algorithm. A third algorithm will also be created to mimic the basic thermostat (baseline) program used in regular households. Using the three separate algorithms, graphs of energy usage and temperature ranges can be made to determine effectiveness of the algorithms. The differences in the algorithms affect how the individual nodes control the temperature in their respective enclosures.

Requirements Specification

Functional Requirements

1. Develop environmental test enclosures
2. Implement & calibrate the Wireless Sensor Network (WSN) temperature sensors
3. Implement WSN controlled air cooler
4. Implement original thermostat program used in households and collect results
5. Implement several improved algorithms for intelligent control and collect results

Non-functional Requirements

1. Hardware must be programmed in NesC
2. Utilize the operating system XubunTOS (which comes with TinyOS) to program
3. Utilize TinyOS in order to install the program onto the sensor
4. Use experimental environmental enclosures to test efficiency of program
5. Upload program and proper documentation to the provided Wiki website and the Subversion server (SVN), respectively

High-Level Design

Process

In order to program the motes, a new operating system needed to be used. XubunTOS is a close relative of Ubuntu (Linux) that includes the TinyOS libraries. The TinyOS libraries consist of the necessary files to easily program the motes, including standard interfaces and components. The motes are programmed using nesC, a language very similar to C that is designed for network embedded systems. XubunTOS also includes the standard compilers for most programming languages, including nesC. This operating system was obtained through the XubunTOS live CD, which is provided by the Toilers Research Group.

The algorithms will be directly interfaced with the TinyOS libraries and compiler. The libraries provided in TinyOS allow the layering of interfaces between the software and hardware components. This makes it easier for the nesC language to use existing interfaces to communicate with the hardware. The nesC compiler automatically compiles the files and creates a binary executable. This executable can then be installed onto a mote if desired. The simple process (Figure 1) allows for quick coding and debugging of nesC code.

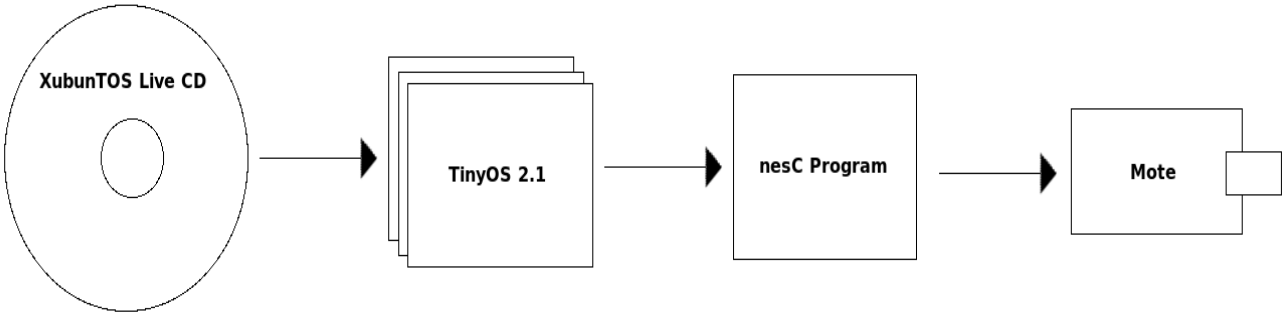

Figure 1 - This simple model represents the basic work flow in order to create and deploy a nesC program onto a mote.

Design Detail

Telosb Motes

The nodes utilized throughout this experiment are given the name of T-Mote Sky, but are referred to as telosb motes by TinyOS. This type of mote is developed by the University of California at Berkeley and designed with the following objectives: Minimal power consumption, ease of use, and increased software and hardware robustness. The telosb mote features a temperature sensor and an integrated onboard antenna with 50m indoor range. The CPU contains a MSP430 chip, stores 10KB of RAM, and 48KB flash.

Test Enclosures

In order to test the proposed hypothesis without implementing a mote system in actual households, test enclosures needed to be designed and built. The test enclosures needed to

mimic the heat gain and loss of an average household as best as possible. The test enclosures are approximately 12 inches wide and 10 inches tall. They are equipped with a small light fixture to act as the heat source and a computer fan to act as the cooling unit. The light sources varied in power for each box to represent different heat gains in different households. Some of the fans were also larger than others to represent different air conditioning efficiencies. The difference in power required between fans was recorded and accounted for in the results. Also, since real air conditioners provide relatively cool air, the air provided by the fan had to be a lower temperature relative to the actual room temperature (75^{o} F). To compensate for this, the "room temperature" in the boxes was calibrated higher than the actual room temperature so that when the fan was activated, cool air would be blown into the enclosure.

Active Messages

The nodes will communicate with each other using Active Messages. An Active Message is a predefined interface within the TinyOS library. The interface includes sending and receiving functions for the message_t abstract data type. The message_t data type is a structure that contains user defined data, which is referred to as payload, in addition to a header and a footer (both of which contain the size of information being sent). This is useful because multiple values can be stored in one payload.

The Active Message interface allows for simple communication between motes without routing. When a message is sent using Active Messages, the message is sent without ensuring that the data was delivered. The provided interface only checks if the message was sent from the host node. Although this would be an issue in an industrial setting, for the purpose of this experiment, range is not a concern.

Distributed Algorithm

One of the three algorithms being implemented is the distributed algorithm (Figure 2). The basic idea is that each node can make its' own decisions regarding the activation of its fan. After the node boots, the standard initializations take place (i.e. booting the radio and the temperature sensor). After the node has successfully initialized all of the necessary components, a periodic reader timer is started. Every time the reader timer fires, the temperature is recorded along with the node ID, the current status of the fan (on or off), the local time, and other relevant data used in the decision making process. This data is sent in a packet to all the nodes in the surrounding area that are utilizing the same radio frequency. Since all the nodes are also sending similar data, this algorithm needs to be able to read and parse the data from the surrounding nodes.

In order for the nodes to make reliable decisions based on the most recent data, they need to be synchronized. The synchronization can be achieved through the utilization of a "head mote." The head mote sends out unmodified time stamps that the other motes use to calibrate their local time. The head mote also periodically sends out a message that informs the other motes in the network to make a synchronized decision. If the head mote is reset or fails to communicate after a predetermined amount of time, a new head mote will be selected.

Once an individual node has received the decision message from the head mote, it will determine whether to activate its fan based on the temperature of the enclosure and the data received from the other nodes. Ideally, the nodes will decide what is best for the whole system and make different individual decisions regarding their respective fans.

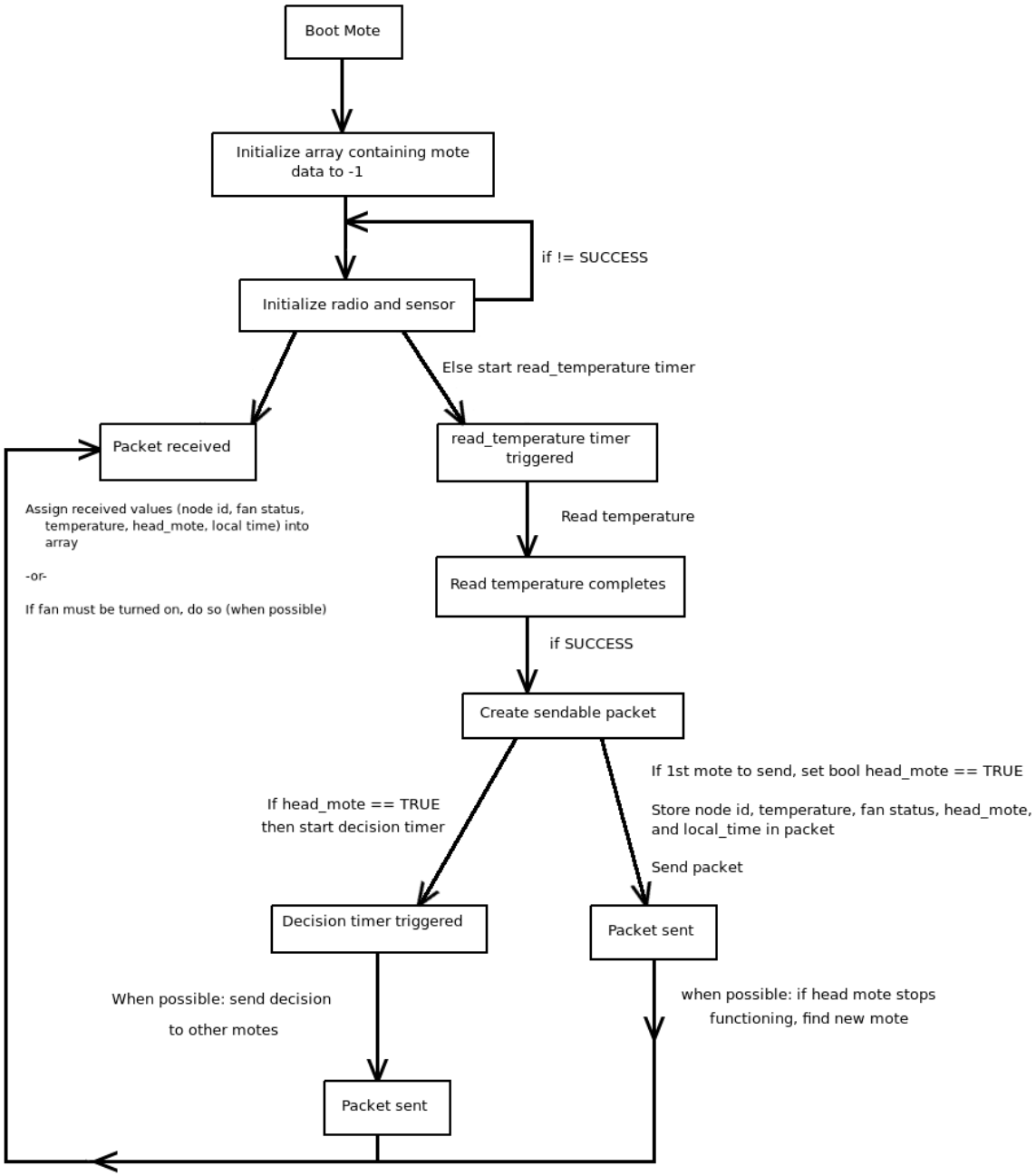

Figure 2 - This figure represents the events that occur and possible decisions that can be made by the event driven motes

Centralized Algorithm

The centralized algorithm is very similar to the distributed algorithm, except that the individual nodes do not make the decisions regarding the activation of their respective fans. The individual nodes will only read the temperature with the external sensor and then send the packet containing the node ID, temperature reading, and the fan status. Instead of sending the information to all the surrounding nodes on the network, the mote will send the packet to a mote acting as a base station. This base station store all the data from the network into an array and at a predetermined time, a decision will be made based on the available data.

The main algorithm to determine the new status of the fans is located in the base station. Once the algorithm determines which fans to turn on or off, the base station will store the node ID and a Boolean fan status value into the payload and send the packet. The motes located within the enclosure will then parse the payload and determine if the decision made by the base station is applicable to that particular mote. If this is the case, then the simple Boolean value will notify the node whether to turn on or off its fan.

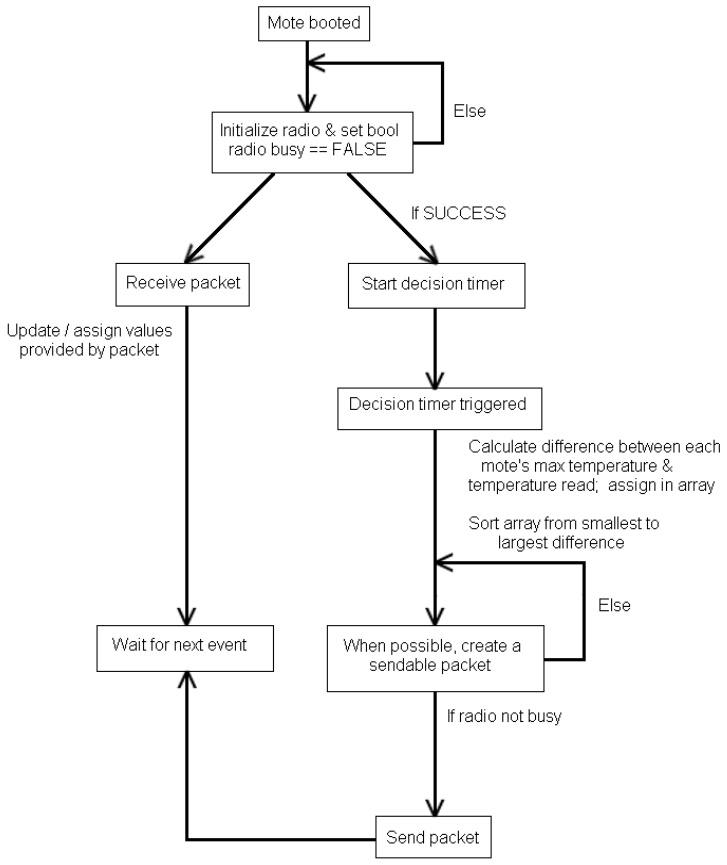

Figure 3 - This figure shows only the flow of events for the base station mote. The base station mote will send the decision via a radio packet, which will then be interpreted by the motes located in the environment enclosures.

Centralized and Distributed Output Interpretation

The TinyOS libraries come with simple programs that allow for easier monitoring and recording of data. The original Base Station program that comes with TinyOS can listen to radio frequencies and report what it receives to a processing client through the USB port. With a simple java program, the data can be seen on the screen and recorded to a file that can later be sorted. This is the method that was used to process and store the algorithm data. The data was then converted into a series of graphs using MiniTab, a statistics program.

Use Cases

1. Fan Activation

The fan turns on when the algorithm (distributed or centralized) determines that the temperature is exceeding the "comfortable" limit.

```
bool decision;
if(decision = = TRUE)
        Fan.set( );
```

Alternative Scenario 1: bool flag (decision) is set to false (another fan has higher priority over this fan). This results in the fan staying off.

2. Fan Deactivation

If the lower temperature threshold is greater than current temperature of the house, the fan will be turned off or kept off

```
bool decision;
if(decision = = FALSE)
        Fan.clr( );
```

Alternative Scenario 1: The bool flag (decision) is set to true (because the house needs to be cooled), the fan turns on

3. Send packets of data to another mote

The temperature data is sent as a packet (via Active Message) to another mote

```
AMSend.send(AM_BROADCAST_ADDR, &pkt, sizeof(msg));
```

Alternative Scenario 1: The packet sent is received as a bad packet. This will result in the packet being resent.

Implementation Details

The programming language utilized throughout this project is called nesC (an abbreviated form of nested embedded systems C, an extension to the C programming language). This language is primarily used in conjunction with the TinyOS platform, allowing the user to easily program at the hardware level. In order for a nesC program to operate correctly, two different components are required: 1) A module file (denoted by a 'C' appended to the end of the file name, i.e. fooC) and a 2) configuration file (denoted by an 'AppC' appended to the end of the filename, i.e. fooAppC). The module file contains the majority of the logic that allows the mote to operate. Similar to the "#include" of C++ and "import" of Java and Python, nesC employs the keyword "uses interface" to allow the programmer access to other functions. The configuration file, on the other hand, declares the library used (via the "components" keyword) and "wires" these libraries together with the interfaces declared in the module file. As a result, new code was only written to enable the algorithms to function.

There were a couple of issues associated with the development and implementation of this project. The single, largest issue was the construction of the environmental enclosures. The most difficult task in constructing the enclosures was choosing the proper light sources and fan sizes to best imitate an average household. In the end, we decided that it would be best to have three different types of light sources (18W, 11W, 7W light bulbs) to represent varying heat gains in different homes. Standard 12V computer fans were used to represent the air conditioning systems in the enclosures. The fans were powered by an external DC power supply.

The synchronization used in the Distributed algorithm also turned out to be a large issue. Since every mote in the network contained the same code, the code needed to be flexible enough to synchronize the motes without depending on one separately programmed mote, as in the centralized algorithm. The head mote protocol was designed such that every mote in the system would calibrate its time relative to the head mote. The head mote would also transmit a message that signaled all of the other motes to make their decisions. The head mote would also make its own decision when it sent the decision message. With every mote being programmed the same, the logic in each mote had to decide which mote was the head mote. The first mote to transmit in the network would become the head mote because it was the first or only mote. The logic also had to be able to adjust if the head mote died, or was reset. Because the times are synchronized, each mote determines if another mote is dead when it transmits its own data. If the motes find that the head mote is dead, or that there is no head mote, then the array logic used finds the next available mote to take on the head mote task.

Results

Baseline Algorithm

The first algorithm that was tested was the baseline algorithm. A total of four motes were used, three of them being in heated enclosures, and the fourth reading the ambient temperature of the room. Figures A.1, A.2, and A.3 in appendix A represent the temperatures throughout the test with respect to total time for each enclosure mote. Mote 1's enclosure was generally hotter because of the powerful heat source (18W) with Mote 2 closely behind (11W) followed by Mote 2 (7W). The differences in average temperatures represented different temperature preferences in households. In this algorithm, there is no dictation in how many fans can be on at a given time. This caused a generally high fan usage rate (Figure A.4), and consequently, a high power usage rate (Figure A.5 and Figure A.6). With this baseline thermostat program, the peak energy was quite high (4.775W) and the total Watt hours over the course of about 13 hours was 20.747 Wh.

Distributed Algorithm

The second algorithm that was tested was the distributed algorithm. Four motes were also used for this experiment, the fourth one reading the room temperature. Figures B.1, B.2, and B.3 in appendix B represent the temperatures throughout the distributed test with respect to total time for each enclosure mote. The same light sources as the baseline algorithm were used. In this experiment, the algorithm only allowed one fan to be on at a time. The peak energy for this algorithm was 1.785W, which occurred when the most powerful fan was on. The total power required for the distributed algorithm test was 7.09Wh.

Centralized Algorithm

The third algorithm that was tested was the centralized algorithm. A total of four motes were used in this test as well with the same light sources and fans as before. Figures C.1, C.2, and C.3 in appendix C represent the temperatures for each mote throughout the centralized test with respect to the total time. This algorithm also only allowed one fan on at a time for this experiment. The results were similar to the distributed algorithm, with a peak energy usage of 1.785W (power of largest fan), and a total power requirement of 9.25Wh.

Comparison

When comparing the three algorithms, it is easy to see that both the distributed and centralized algorithms were far more efficient and power conserving than the original baseline algorithm. Between the centralized and distributed algorithm, it is hard to tell the difference. Although it appears that the distributed algorithm used less total energy, there were many uncertainties between the separate experiments. Both the centralized algorithm and the distributed algorithm use very similar logic to control the status of the fans. In conclusion, it is hard to compare the two smart algorithms to each other, but easy to see that they are both better than the baseline algorithm.

Risks and Uncertainties

There were a couple of risks and uncertainties that might have thrown off or affected the gathered data. Using different motes for the different algorithms may have affected some of the data, as each mote does not read the same temperature. We attempted to compensate for the difference in readings by slightly changing the thresholds of the motes. Another factor that may have affected the data was the room in which the algorithms were tested. The algorithms were tested on different nights, each of which had different average room temperatures. The room temperature would also sporadically change at certain times. This most likely had a negative effect on the experiments, most likely making the algorithms look better than they really were. Another large uncertainty was the temperature readings based on battery power. We noticed that as the batteries in the motes ran low, the motes started to send maximum raw readings, which originally caused one of the tests to fail. A way to solve and correct these risks and uncertainties would be to have pre-calibrated motes, an environmental chamber to keep the ambient temperature constant, and another source to power the motes other than batteries or USB.

Conclusions and Future Direction

Summary

Overall, the goal of the project was attained. The goal was basically a proof of concept that wireless sensor nodes could be used to control energy usage and reduce overall peak energy. The graphs and statistics show that the two smart algorithms used about half the total energy of the baseline algorithm while keeping peak energy low.

Lessons Learned

One of the main lessons learned for this project was time management. It is always important to leave plenty of time for debugging and testing purposes. Another lesson learned was the lesson of proper documentation. It is always important to properly document code and data in order to prevent later confusion.

Future Direction

This project can certainly be improved on in the future. One possible direction to take would be to emulate the same "Saver's Switch" program used by Xcel, and compare different algorithms to that program specifically. Another way to improve upon this design in the future would be to test more environmental enclosures inside of a controlled environment. This would allow more reliable results than the results gathered in these experiments. Wireless sensor nodes can be used for many different control projects, and controlling temperature is only one of them. Other possibilities include utilizing the humidity sensor and factoring the data retrieved from this sensor into the temperature reading.

Glossary

AM – Active Message; an abstract interface provided by TinyOS to send data wirelessly

nesC – Network Embedded Systems C; Programming language used to interface with hardware

Telosb – USB enabled battery operated sensor node

TinyOS - Open source component-based operating system targeting wireless sensor networks

SVN – Subversion; a version control system

WSN – Wireless Sensor Network

XubunTOS – Linux operating system with TinyOS pre-installed

Appendix A

Baseline (Thermostat) Statistics

Test Conditions Description

Mote 1 placed in 18W Light bulb Test Enclosures – Fan: 11.9V, 0.15A
Mote 2 placed in 11W Light bulb Test Enclosures – Fan: 11.9V, 0 .11A
Mote 3 placed in 7W Light bulb Test Enclosures – Fan: 11.9V, 0.14A

Approximate period of time the experiment was conducted: ~ 13.236 hours and in each data point is taken with 256 millisecond interval

Individual Mote Statistics

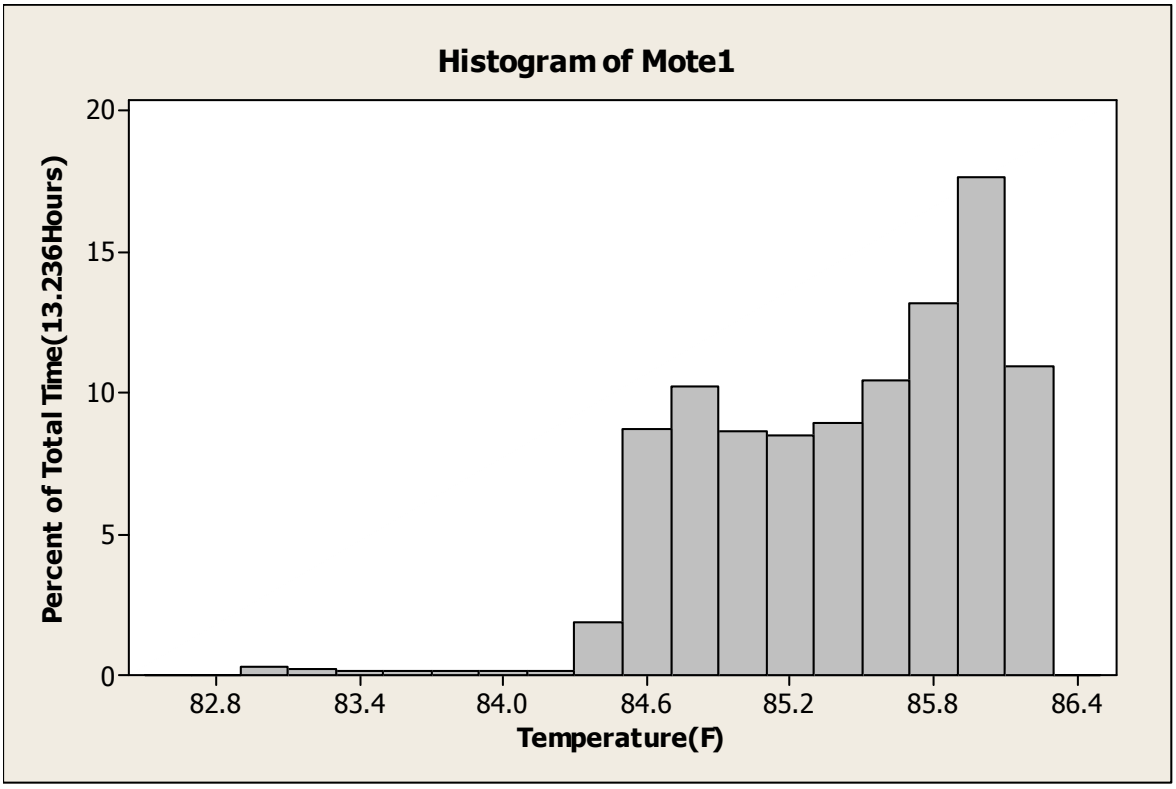

Figure A.1 – A histogram of mote 1's temperature readings for the baseline algorithm

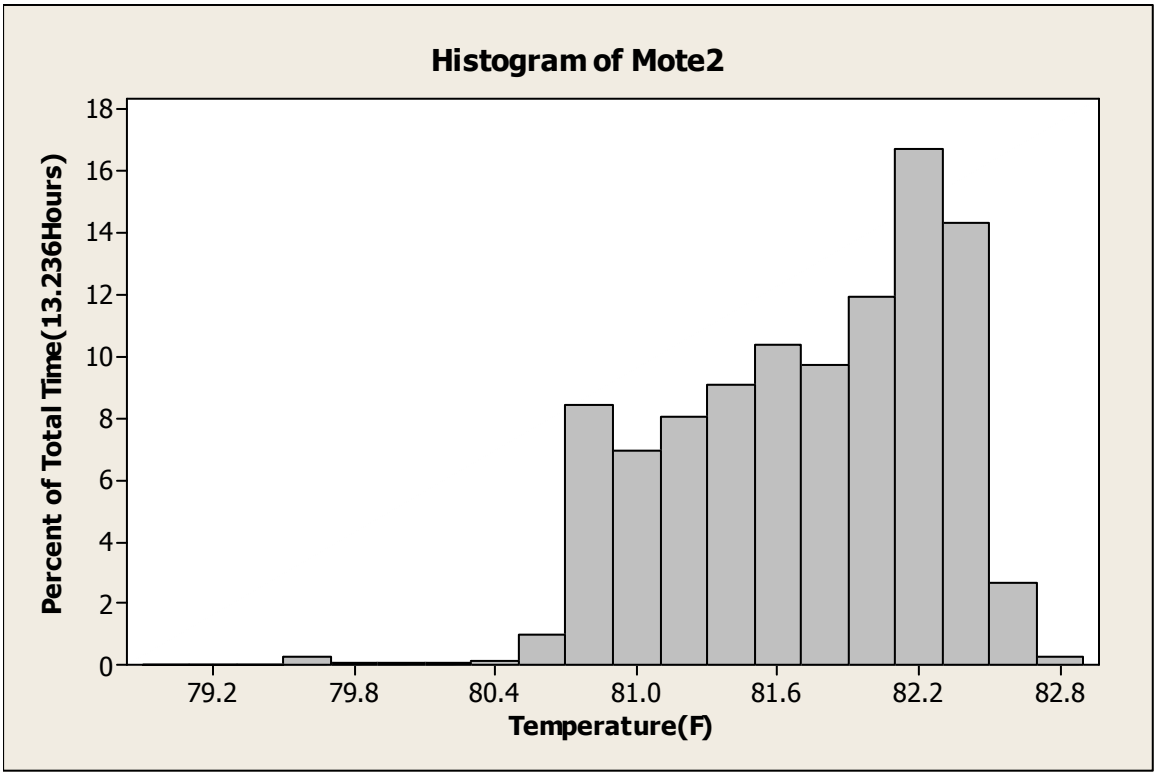

Figure A.2 - A histogram of mote 2's temperature readings for the baseline algorithm

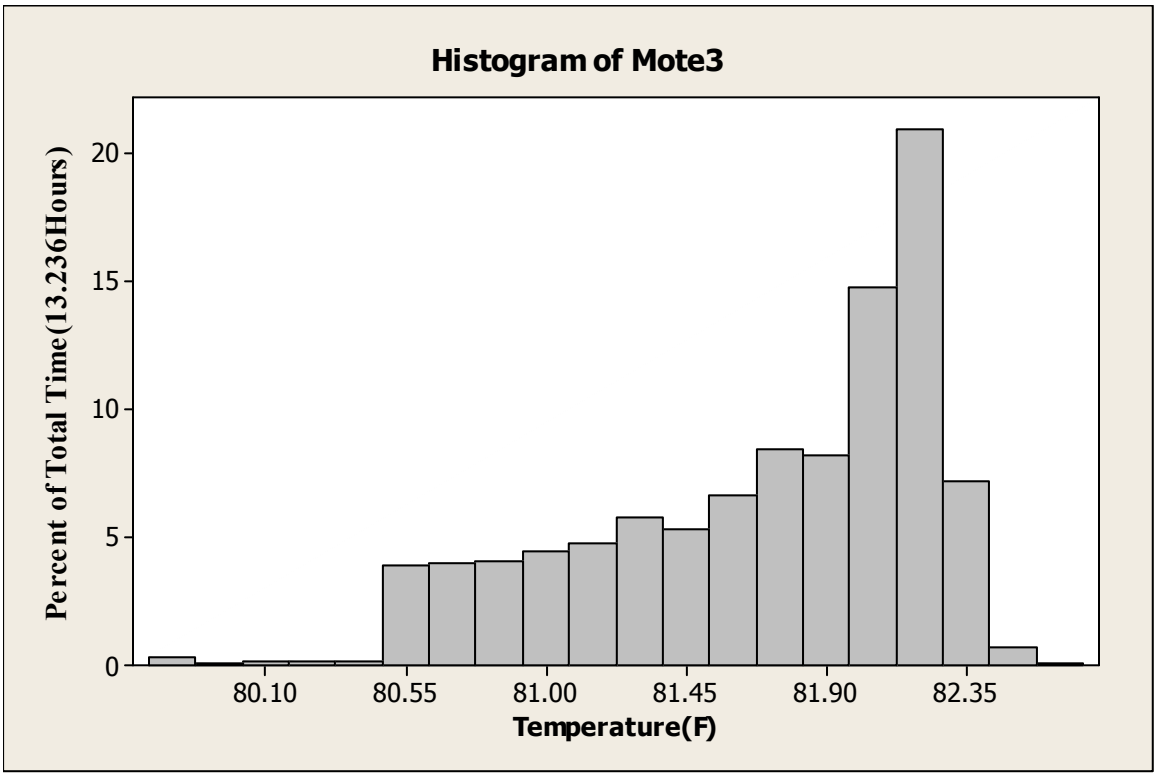

Figure A.3 - A histogram of mote 3's temperature readings for the baseline algorithm

Temperature vs Time Scatter plot (4 Hour Sample)

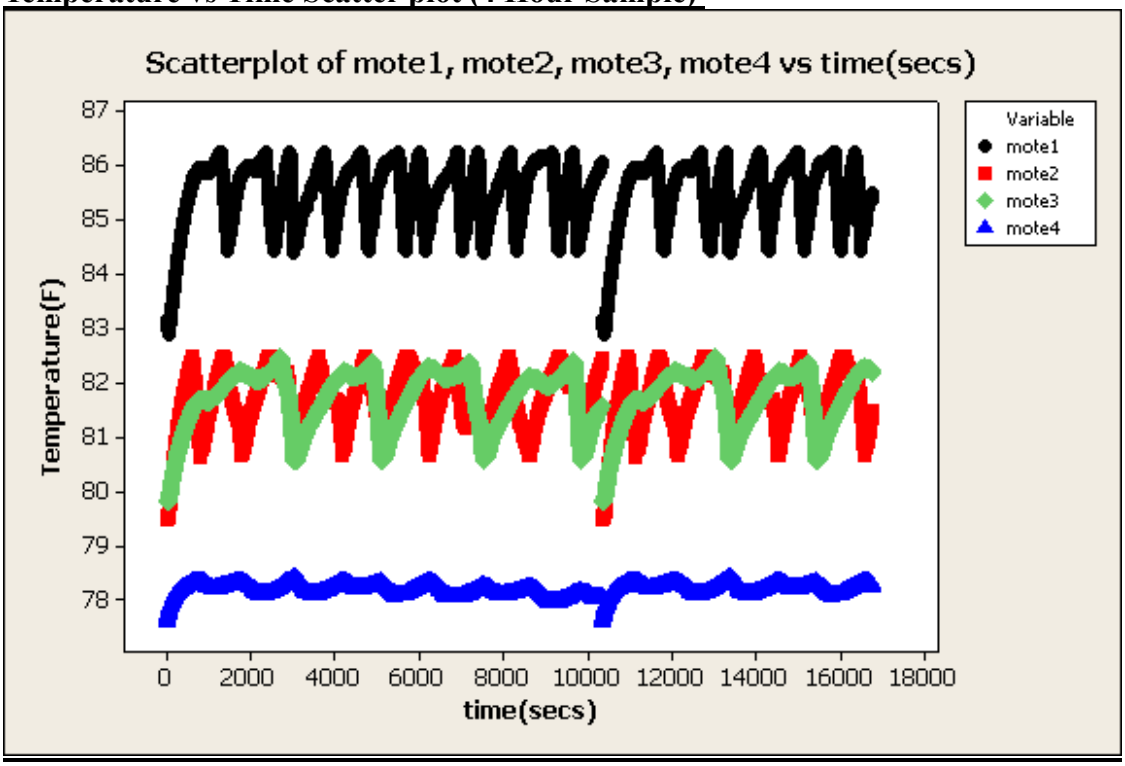

Figure A.4 – A four hour sample period of temperature data using the baseline algorithm

Fan Usage Histogram

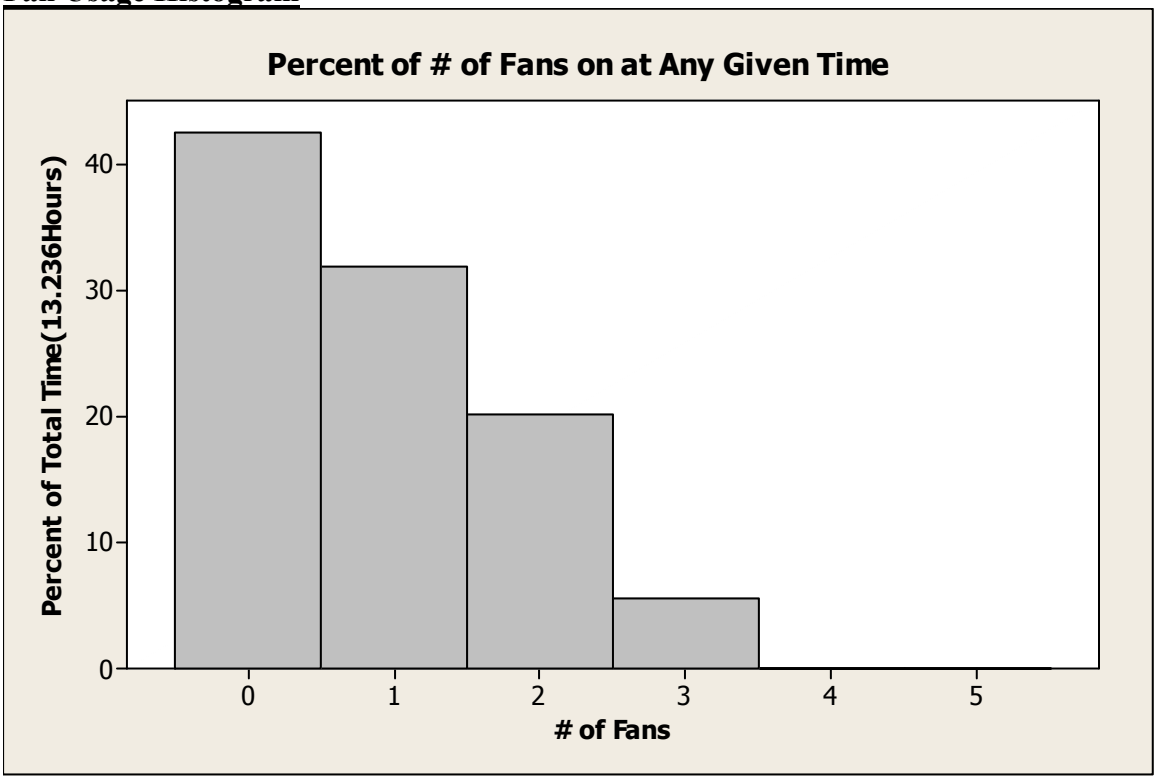

Figure A.5 – A histogram of the number of fans on as a percentage of total time (~13.24 hours)

Power vs Time Line Plot for Fans(2 Hour Sample)

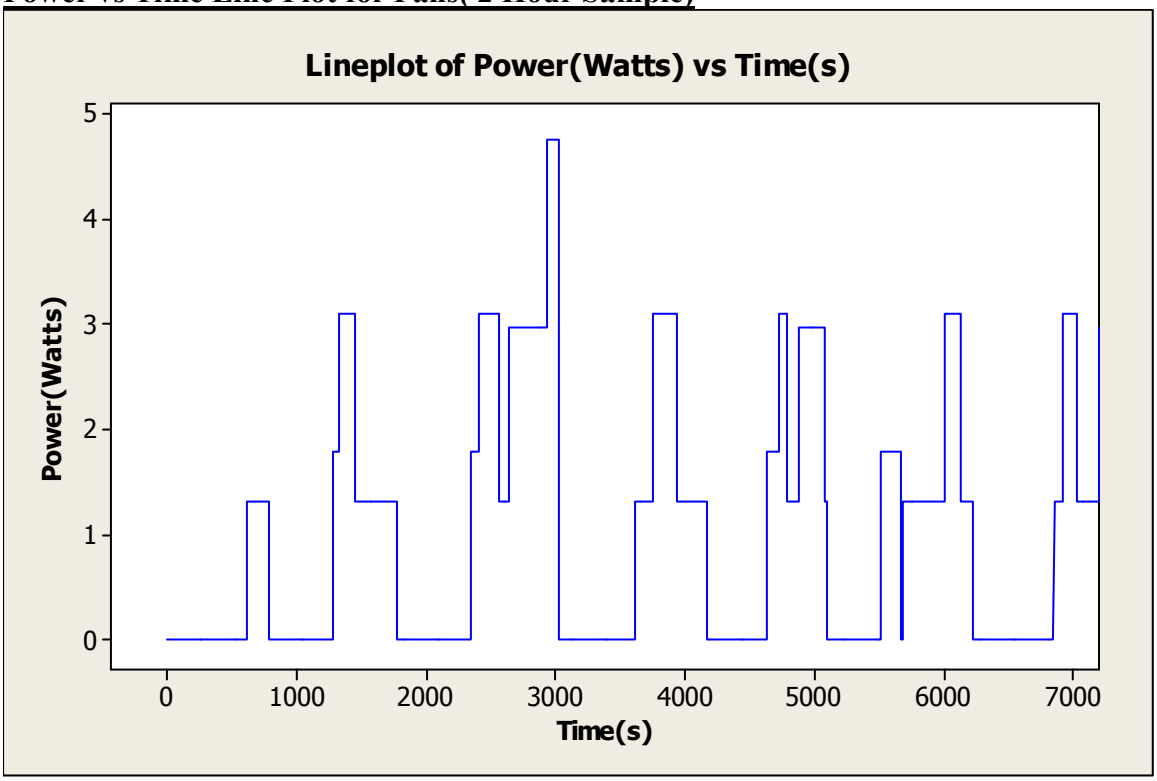

Figure A.6 – A scatter plot of power usage with respect to time

Histogram for Power Usage by the Fans

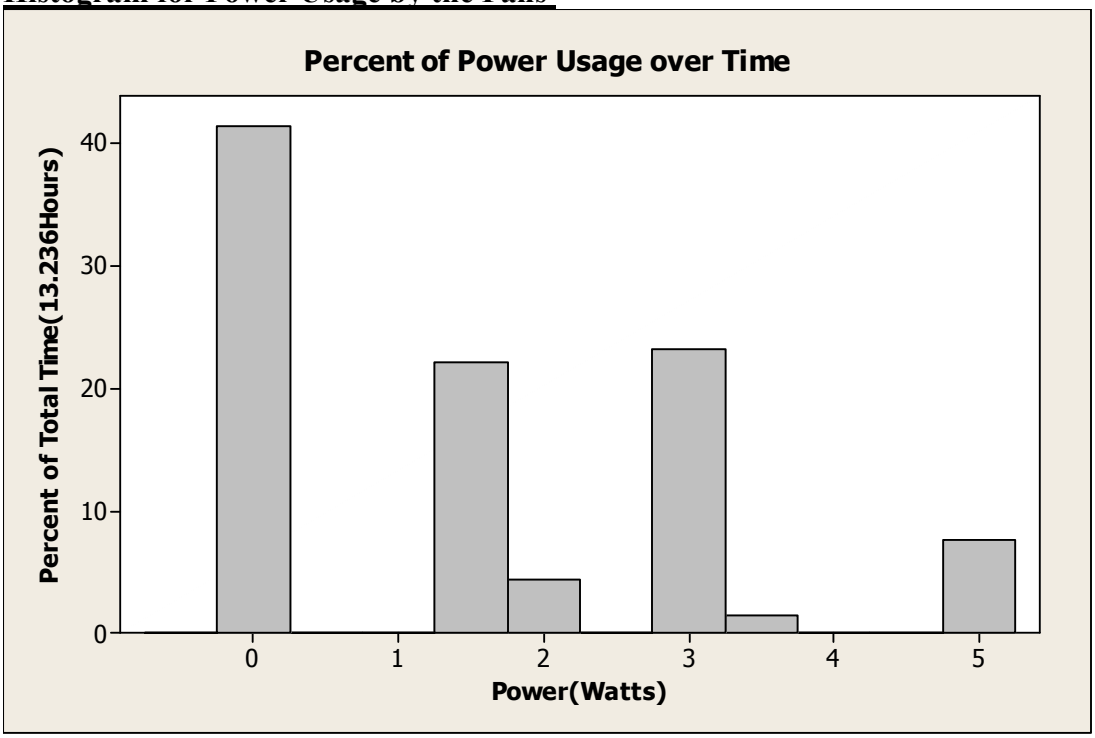

Figure A.7 – A histogram of the power usage as a percentage of the total time (~13.24 hours)

Descriptive Statistics: Power(Watts)

Variable	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
Power(Watts)	186132	0	1.4838	0.00355	1.5317	0.00000	0.00000	1.3090

Variable	Q3	Maximum	Range
Power(Watts)	2.9750	4.7600	4.7600

Approximate Power Usage:
Mote 1 – 3.564 Wh
Mote 2 - 0.288 Wh
Mote 3 – 1.030 Wh
Mote 2 + Mote 3 – 5.109 Wh
Mote 2 + Mote 1 – 5.348 Wh
Mote 1+ Mote 3 – 0.633 Wh
Mote 1 + Mote 2 + Mote 3 – 4.775 Wh
Total – 20.747 Wh
Approximate Mean Power: 1.4838 Watts

Figure A.8 – The general statistics of power usage for the baseline algorithm

Appendix B

Distributed Algorithm Statistics

Test Conditions Description

Mote 1 placed in 18W Light bulb Test Enclosures – Fan: 11.9V, 0.15A
Mote 2 placed in 11W Light bulb Test Enclosures – Fan: 11.9V, 0 .11A
Mote 3 placed in 7W Light bulb Test Enclosures – Fan: 11.9V, 0.14A

Approximate period of time the experiment was conducted: ~ 13.236 hours and in each data point is taken with 5 second intervals

Individual Mote Statistics

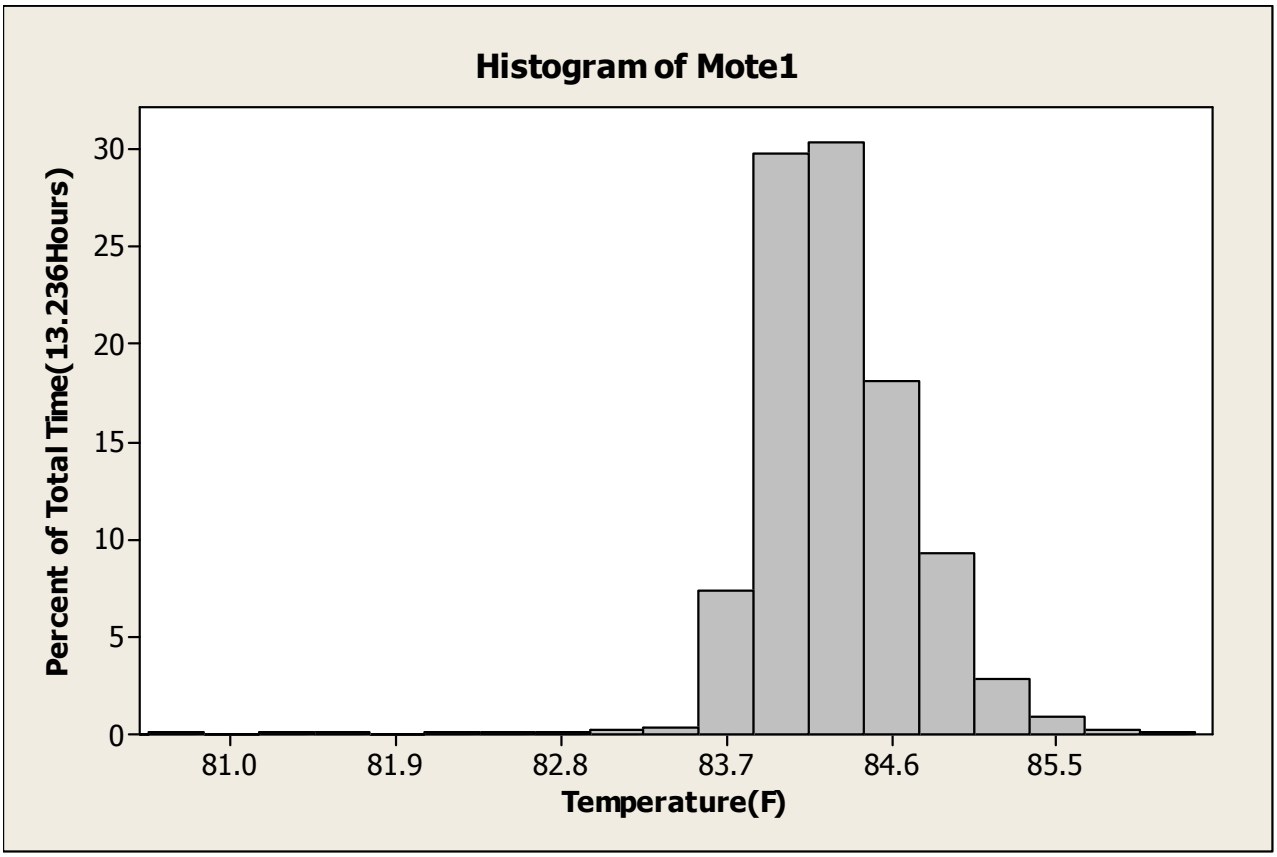

Figure B.1 - A histogram of mote 1's temperature readings for the distributed algorithm

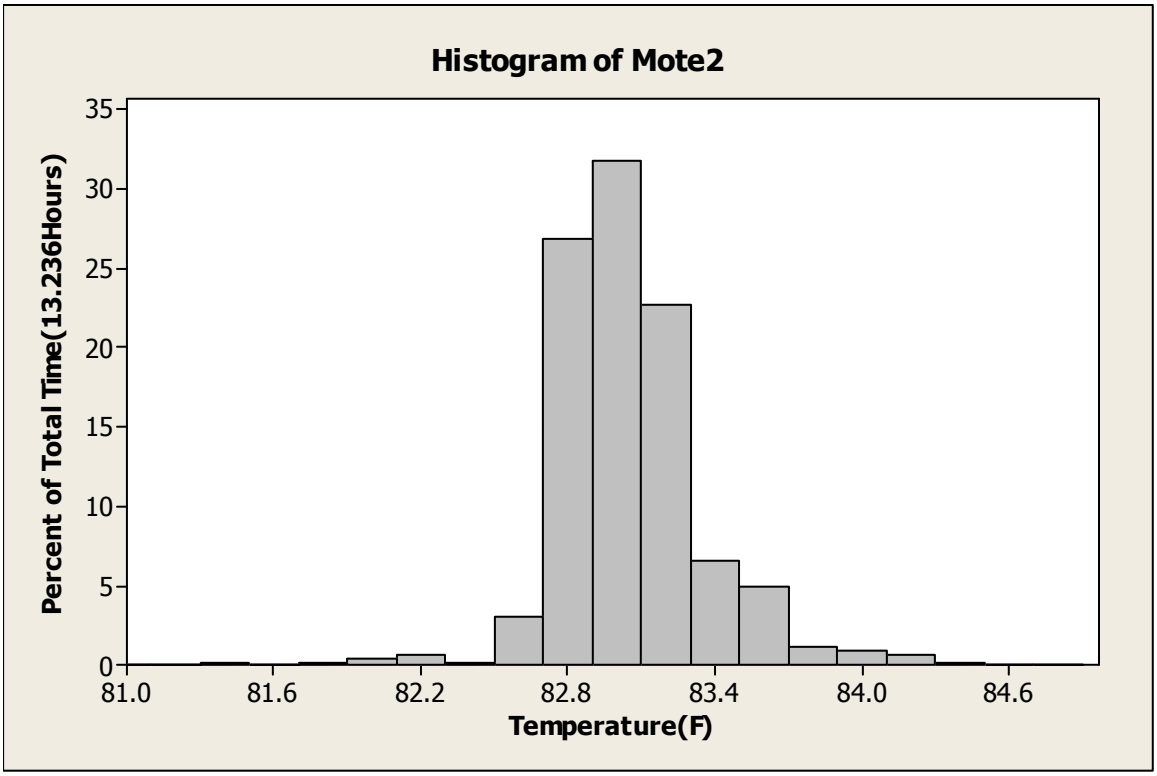

Figure B.2 - A histogram of mote 2's temperature readings for the distributed algorithm

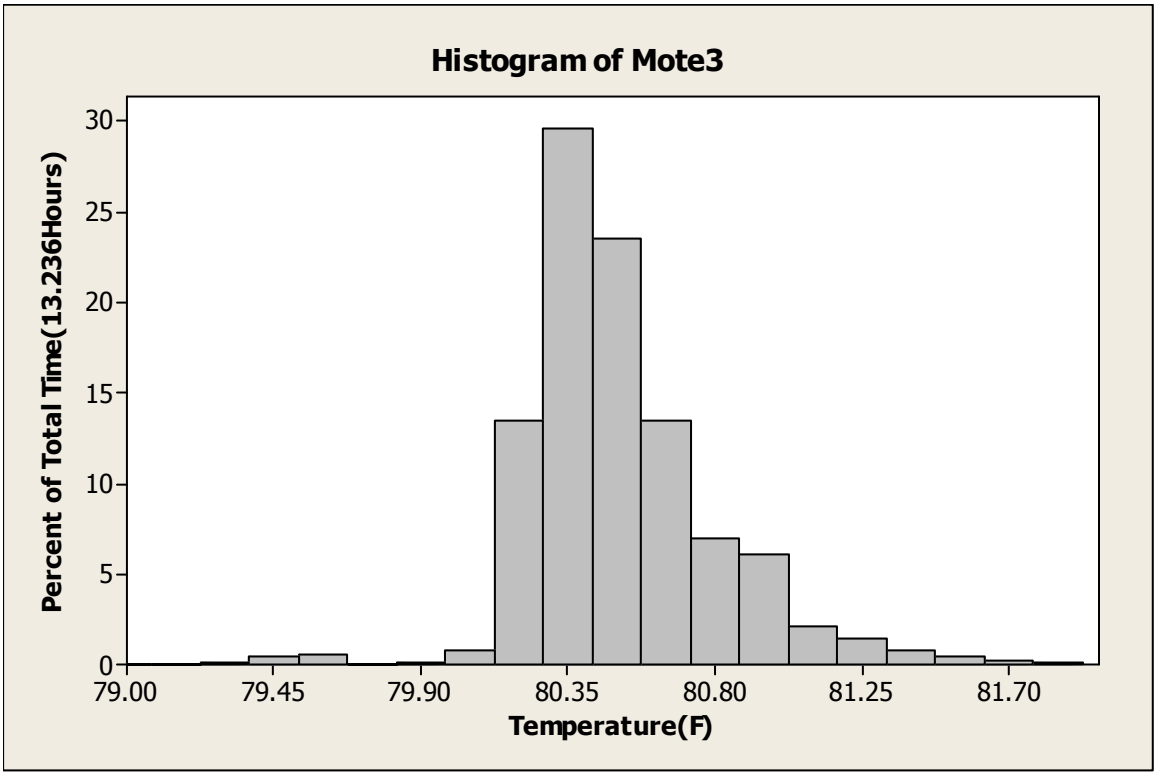

Figure B.3 - A histogram of mote 3's temperature readings for the distributed algorithm

Temperature vs Time Scatter plot (4 Hour Sample)

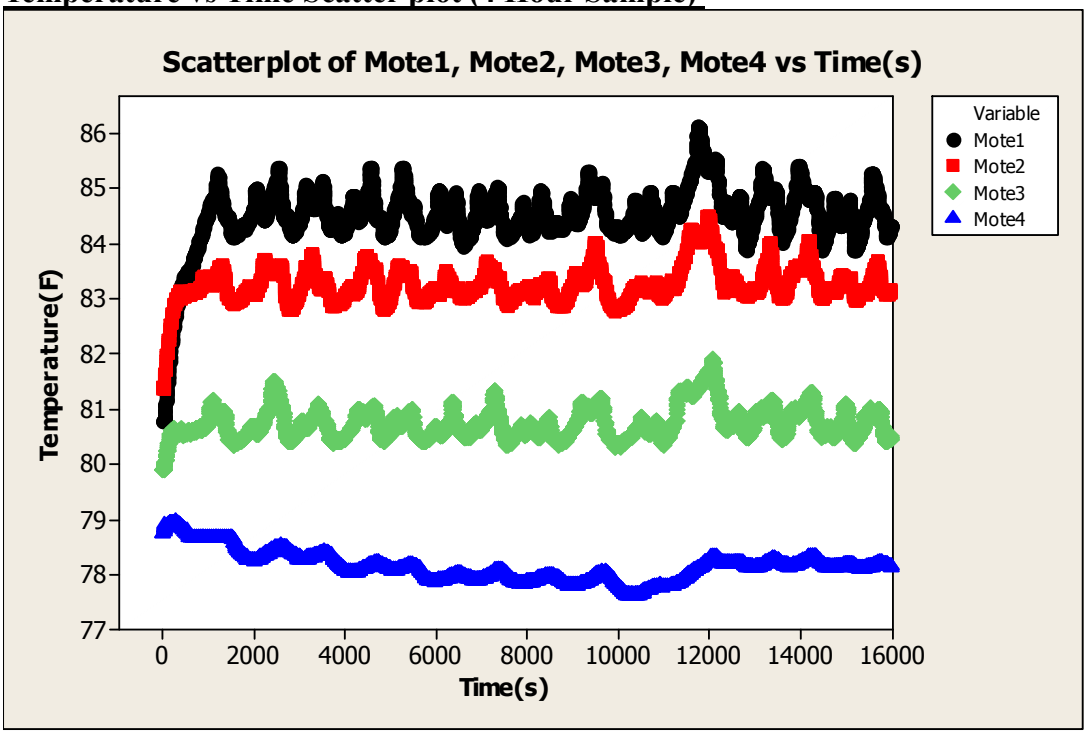

Figure B.4 – A four hour sample period of temperature data using the distributed algorithm

Fan Usage Histogram

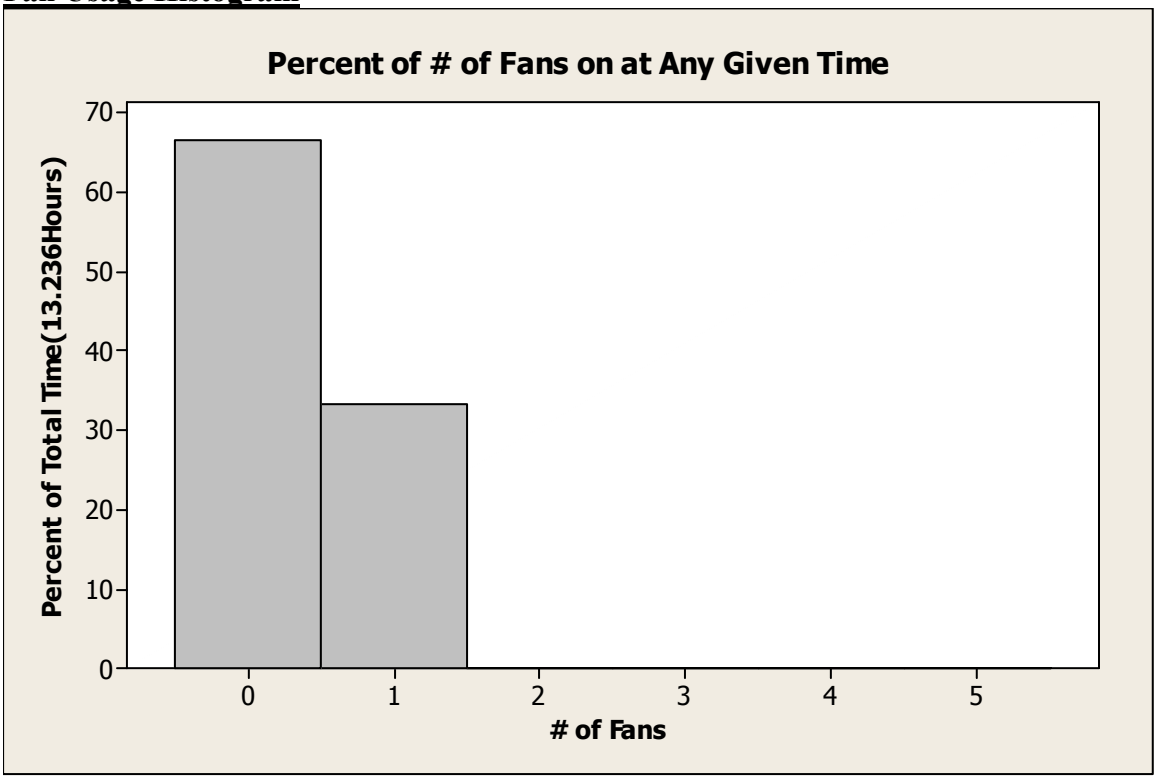

Figure B.5 - A histogram of the number of fans on as a percentage of total time (~13.24 hours)

Power vs Time Line plot for Fans (2 Hour Sample)

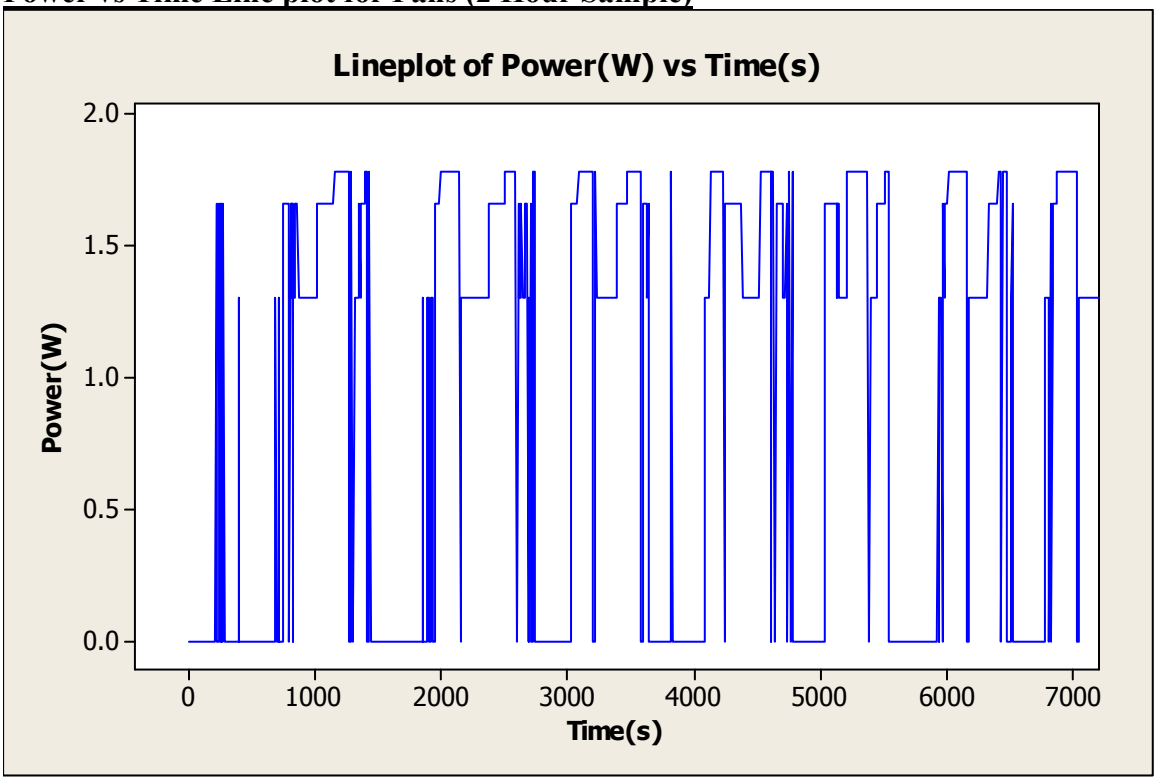

Figure B.6 - A scatter plot of power usage with respect to time

Histogram for Power Usage by the Fans

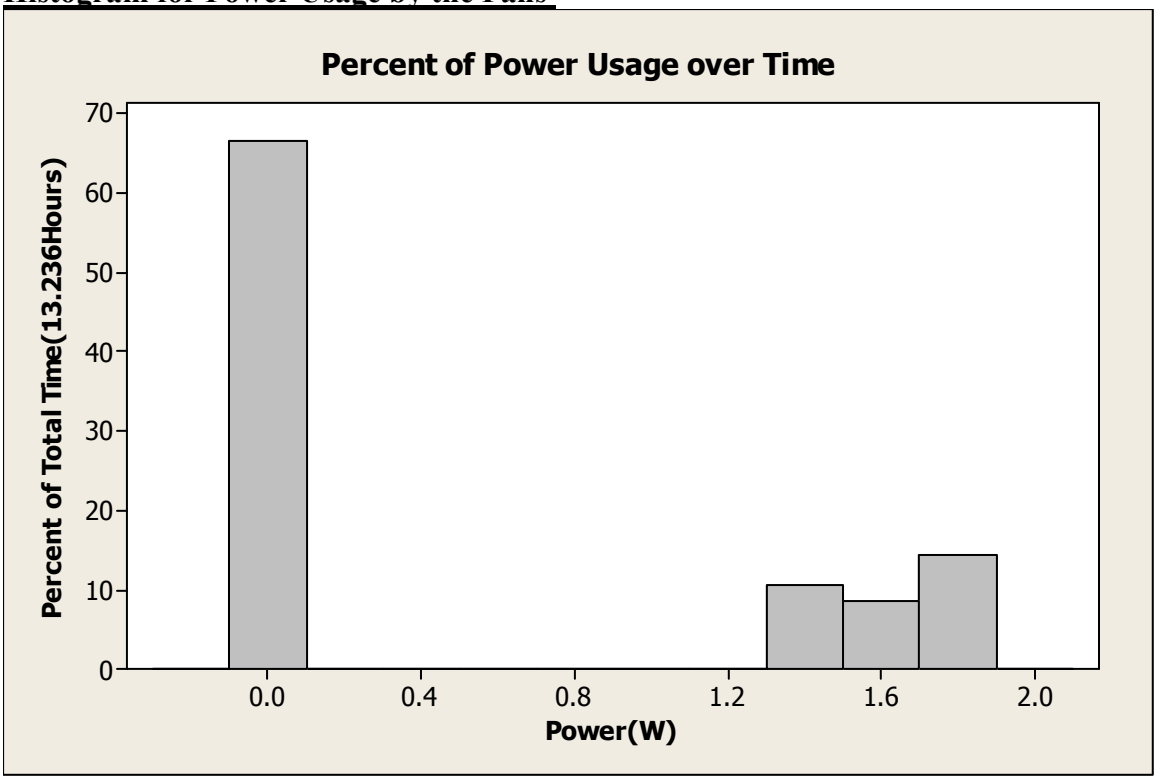

Figure B.7 - A histogram of the power usage as a percentage of the total time (~13.24 hours)

Descriptive Statistics: Power(w)

Variable	N	N*	Mean	SE Mean	StDev	Minimum	Q1
Power (w)	9530	0	0.53577	0.00785	0.76610	0.00000	0.00000

Variable	Median	Q3	Maximum
Power (w)	0.00000	1.30900	1.78500

Approximate Power Usage:
Mote 1 Total Wh = 3.3915 Wh
Mote 2 Total Wh = 1.824 Wh
Mote 3 Total Wh = 1.877 Wh
Total for the system = 7.0925 Wh
Approximate Mean Power: 1.64 Watts

Figure B.8 – The general statistics of power usage for the distributed algorithm

Appendix C

Central Algorithm Statistics

Test Conditions Description

Mote 1 placed in 18W Light bulb Test Enclosures – Fan: 11.9V, 0.15A
Mote 2 placed in 11W Light bulb Test Enclosures – Fan: 11.9V, 0 .11A
Mote 3 placed in 7W Light bulb Test Enclosures – Fan: 11.9V, 0.14A

Approximate period of time the experiment was conducted: ~ 13.236 hours and in each data point is taken with 5 second intervals

Individual Mote Statistics

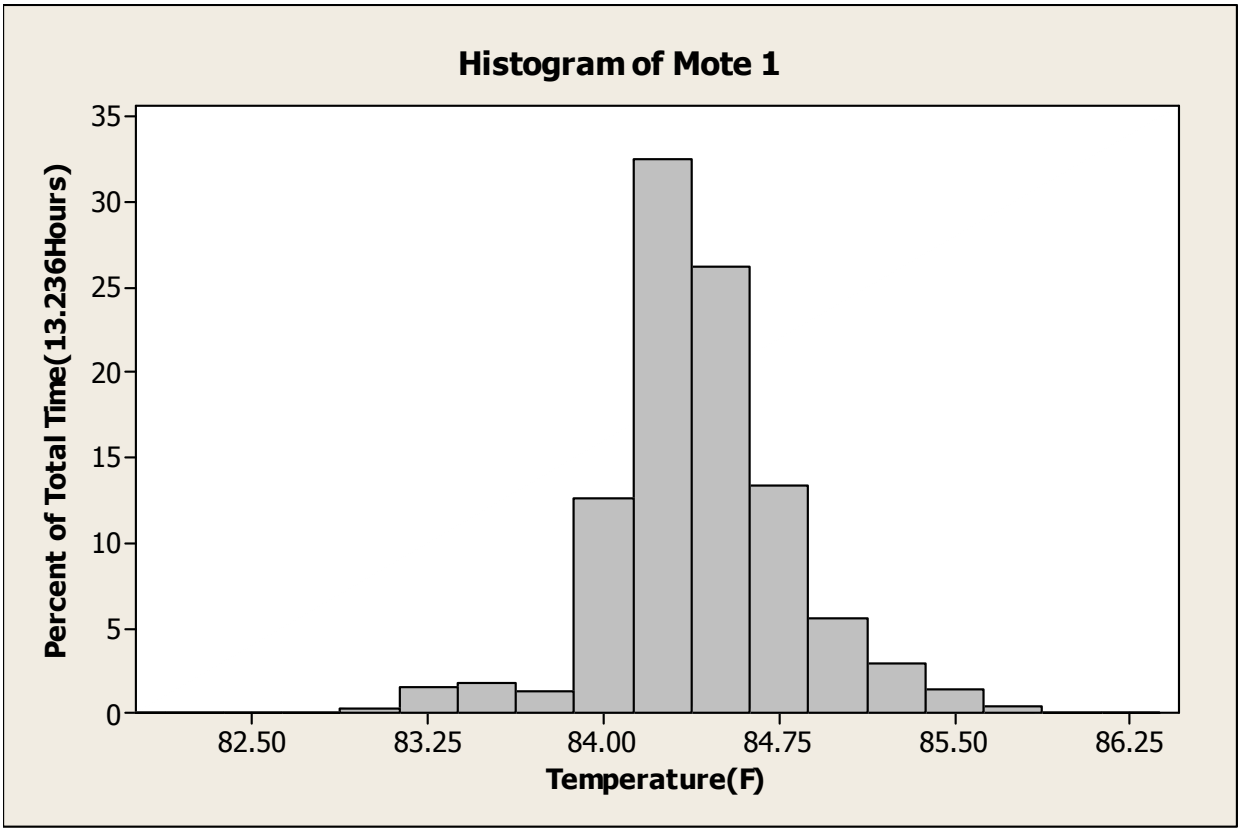

Figure C.1 - A histogram of mote 1's temperature readings for the centralized algorithm

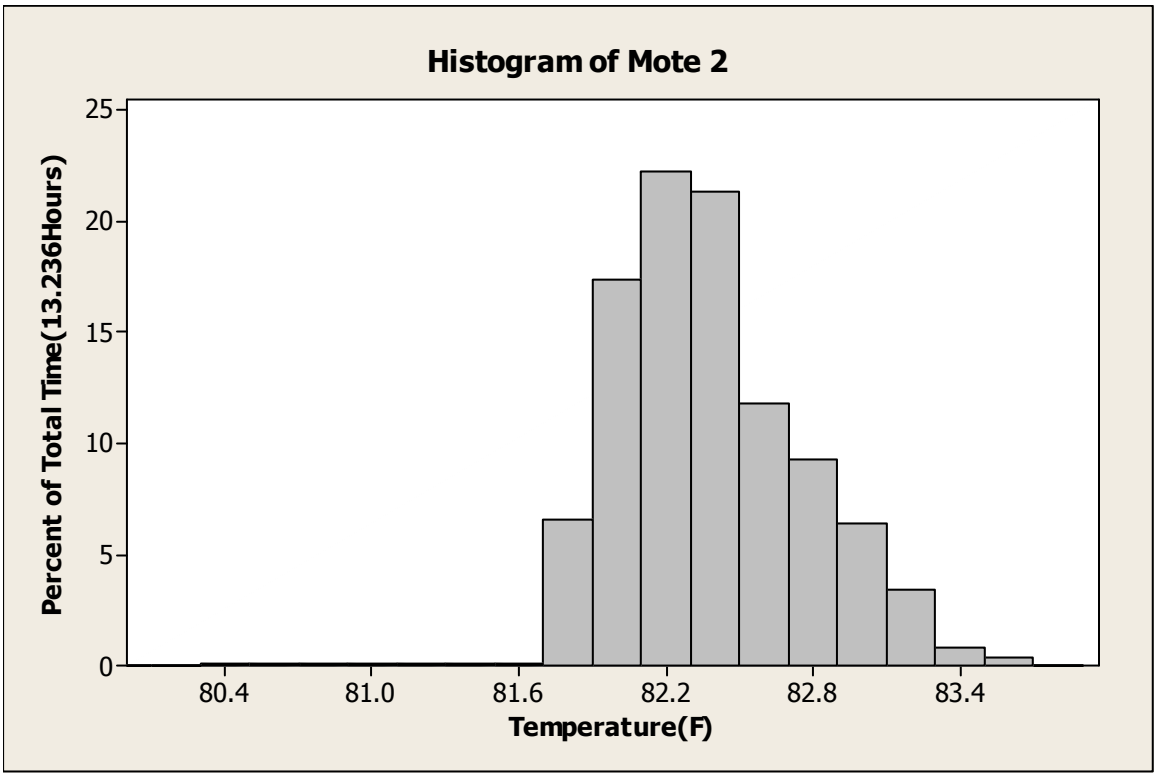

Figure C.2 - A histogram of mote 2's temperature readings for the centralized algorithm

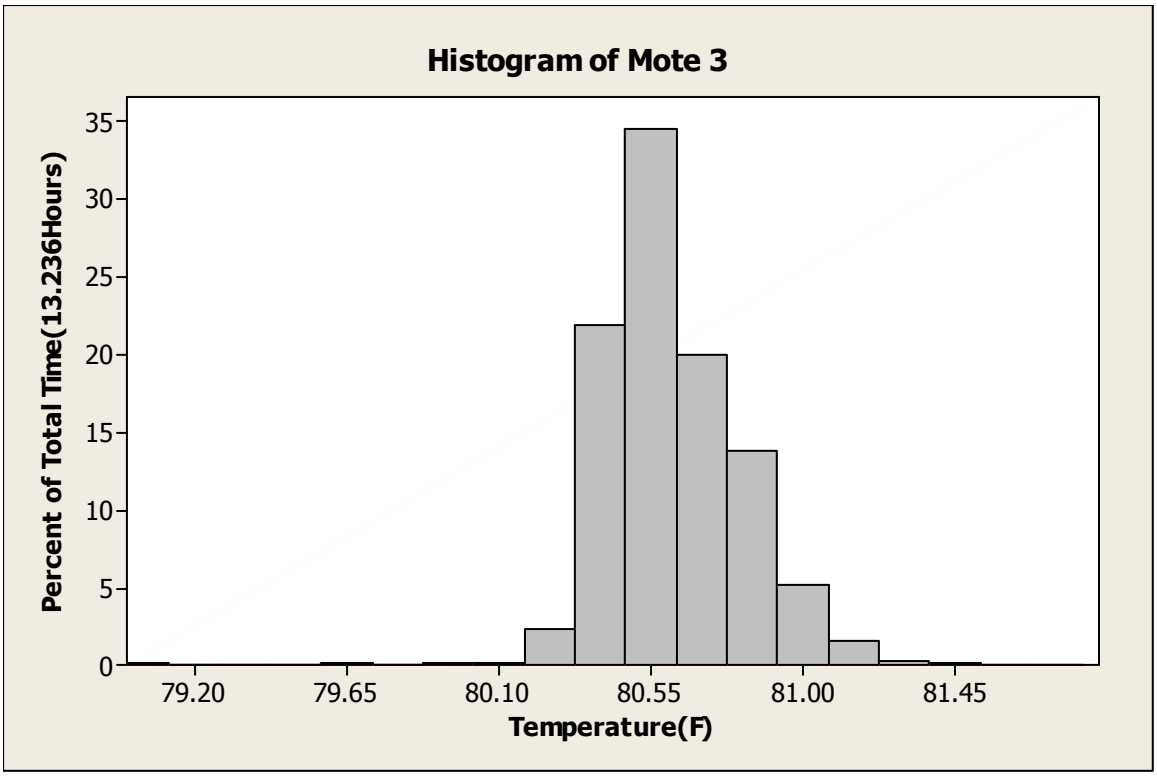

Figure C.3 - A histogram of mote 3's temperature readings for the centralized algorithm

Temperature vs Time Scatter plot (4 Hour Sample)

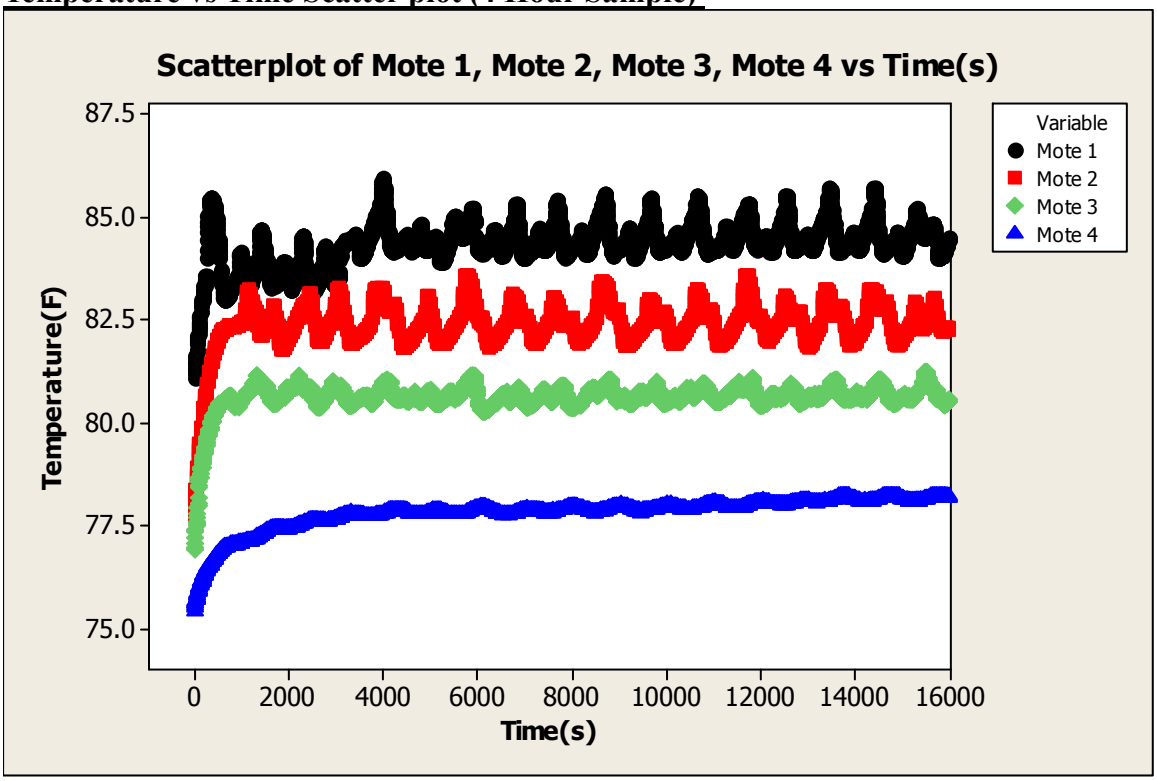

Figure C.4 – A four hour sample period of temperature data using the centralized algorithm

Fan Usage Histogram

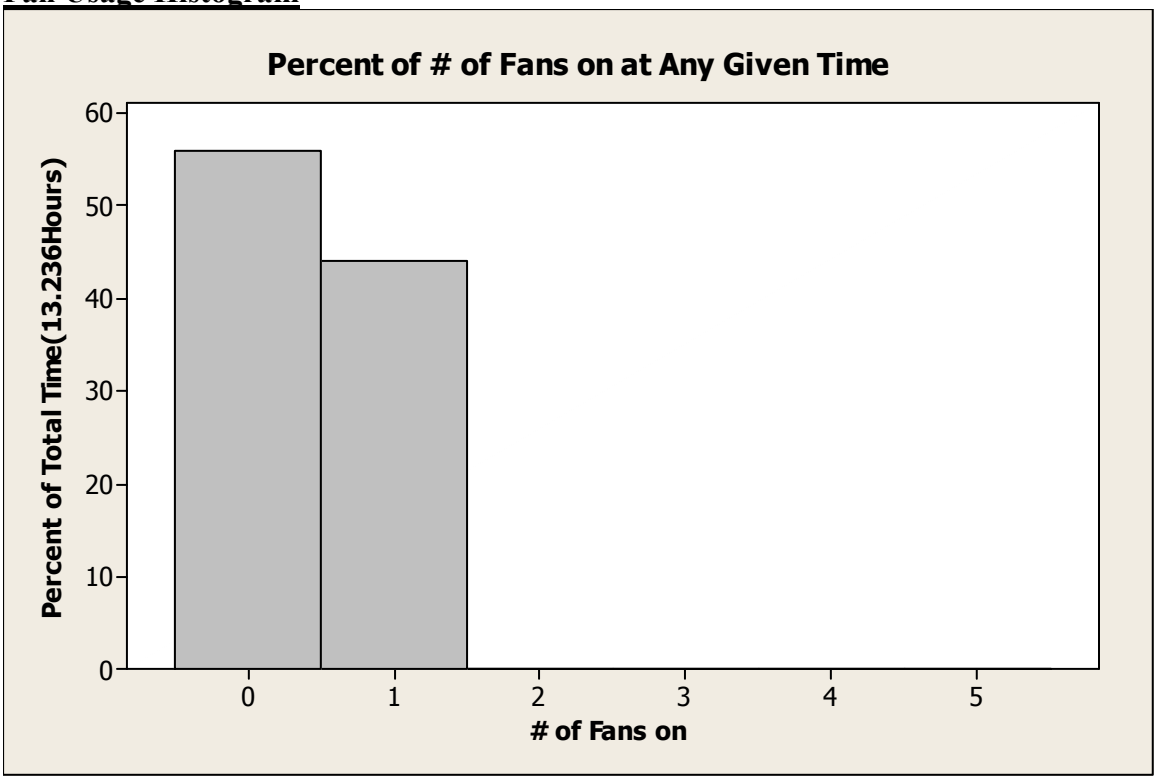

Figure C.5 - A histogram of the number of fans on as a percentage of total time (~13.24 hours)

Power vs Time Line plot for Fans (2 Hour Sample)

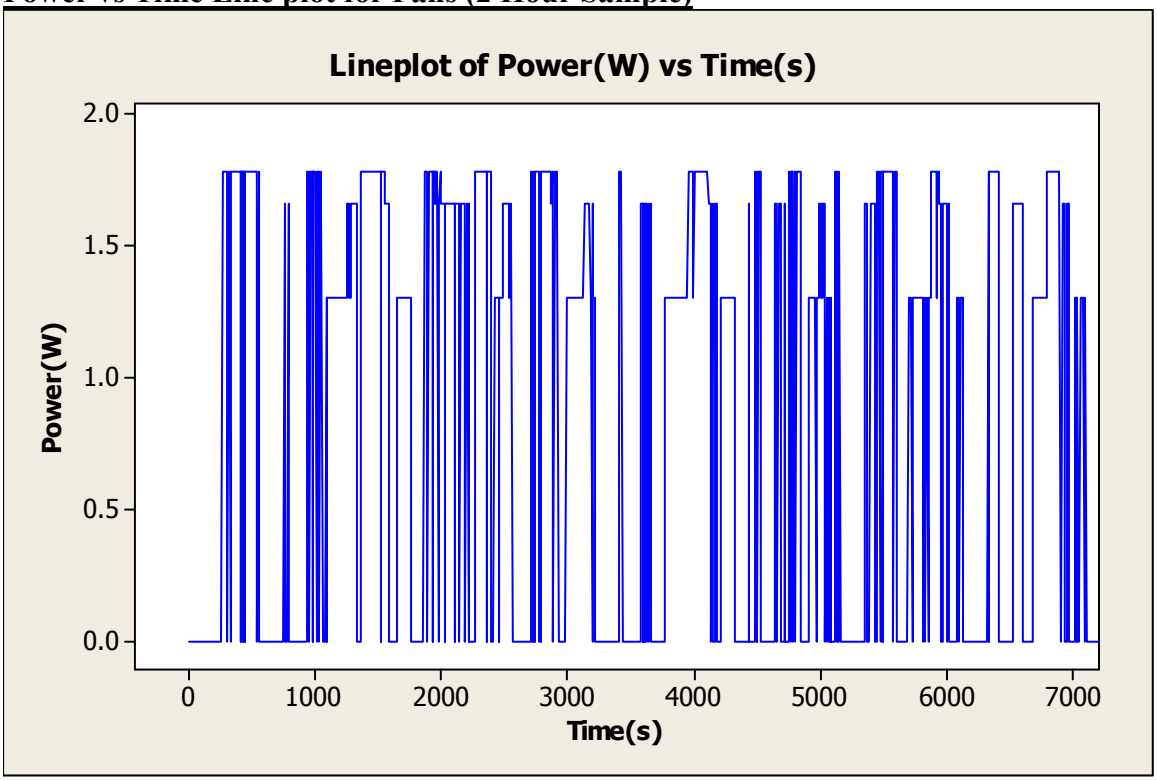

Figure C.6 - A scatter plot of power usage with respect to time

Histogram for Power Usage by the Fans

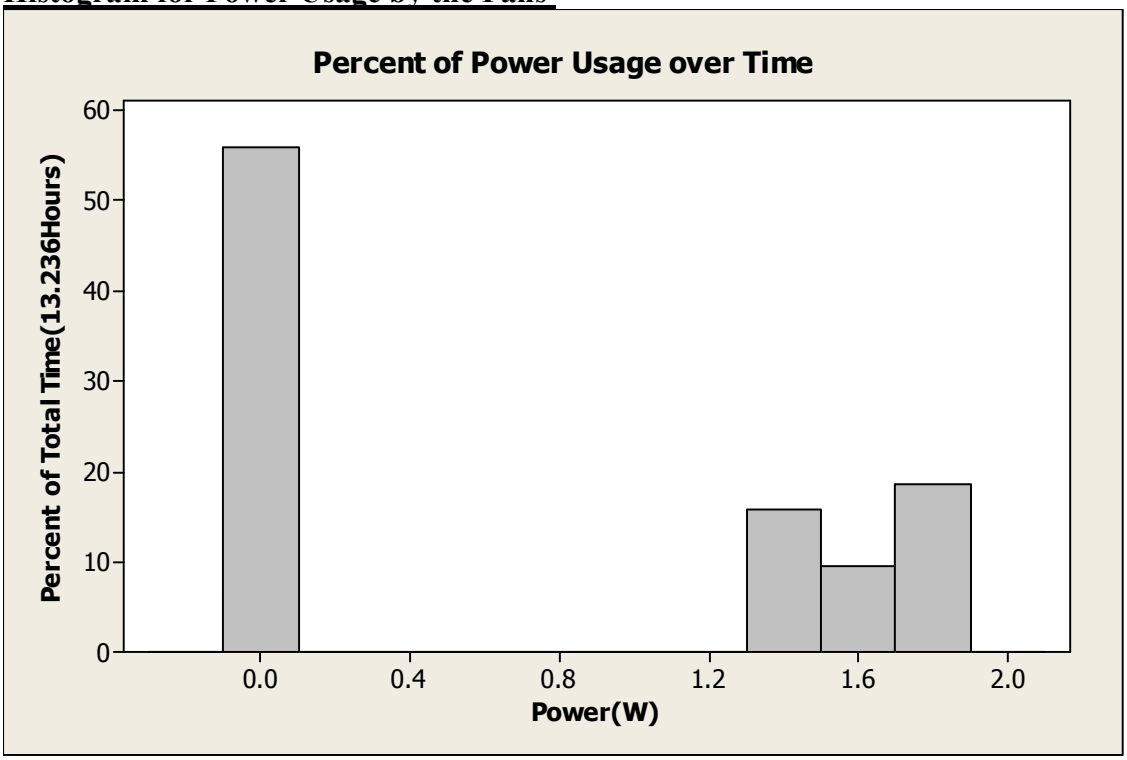

Figure C.7 - A histogram of the power usage as a percentage of the total time (~13.24 hours)

Descriptive Statistics: Power(W)

Variable	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
Power(W)	9530	0	0.69873	0.00821	0.80101	0.00000	0.00000	0.00000

Variable	Q3	Maximum
Power(W)	1.66600	1.78500

Approximate Power Usage:
Mote 1 Total Wh = 4.40 Wh
Mote 2 Total Wh = 2.74 Wh
Mote 3 Total Wh = 2.11 Wh
Total for the system = 9.25 Wh
Approximate Mean Power: 0.699 Watts

Figure C.8 – The general statistics of power usage for the centralized algorithm