A Multihop Building Monitoring System
Using Sun SPOT Sensor Nodes

Kim Thanh NGUYEN - 200429793
Parampal POONI

INFO3600
Major Software Development Project
Semester 2, 2006

Supervisors
Dr. Selvakennedy Selvadurai
Dr. Uwe Roehm
Dr. Cristina Cifuentes

Submission date
23 – October - 2006
A MULTIHOP BUILDING MONITORING SYSTEM USING SUN SPOT SENSOR NODES

Kim Thanh Nguyen - Parampal Pooni

School of Information Technologies
The University of Sydney

ABSTRACT
Wireless Sensor Network has emerged as a very hot topic in the networking area recently. It enables the monitoring of certain physical characteristics such as temperature or luminescence. In the effort to push Java into new applications including Wireless Sensor Network, Sun Microsystems Labs have been developing Sun SPOT (Small Programmable Object Technology) in terms of both hardware (Sensor Nodes) and software. This project focuses on developing a multi-hop routing protocol for communication among Sun SPOT sensor nodes and a user front-end (i.e. visualiser) to visualise the collected values from all the nodes. To test the routing protocol before deploying it to sensor nodes, a simulation using J-Sim is created.

ACKNOWLEDGEMENT
We would like to thank Sun Microsystems Labs for giving us a chance to work with Sun SPOT. Doing this project has been very exciting, especially when we worked with Sun SPOT sensor nodes.

Many thanks to our internal and external supervisors, Dr. Selvakennedy Selvadurai, Dr. Uwe Roehm, and Dr. Cristina Cifuentes, who have spent time with us, supported and encouraged us, and given us helpful feedbacks for the project. Regular meetings with Dr. Selvakennedy Selvadurai have brought us many useful ideas for solving problems. In addition, Dr. Cristina Cifuentes has been to Sydney to see the demo of our project and to comment on our work.

We are also thankful to our subject coordinator, Ms. Irena Koprinska, who has been encouraging us throughout the semester. Her enthusiasm and positive feedbacks for our early work did cheer us up a lot.
# TABLE OF CONTENTS

## TABLE OF CONTENTS

- **CHAPTER 1 - CONTEXT**..................................................................................................................................................................................2
  - 1.1 INTRODUCTION ..................................................................................................................................................................................2
    - A. Project problem ..................................................................................................................................................................................2
    - B. Project main goals .................................................................................................................................................................................2
  - 1.2 OVERVIEW OF THE COMPANY AND SIGNIFICANCE OF THE GROUP’S WORK ...............................................................................................2
    - A. Kim Nguyen’s part ..................................................................................................................................................................................2
    - B. Parampal Pooni’s part ..........................................................................................................................................................................3
  - 1.3 RELATED WORK – SUMMARY OF EXPLORATION OF COMPARABLE SYSTEMS ........................................................................................5
    - A. The simulator .......................................................................................................................................................................................5
    - B. The multi-hop routing protocol .......................................................................................................................................................5
    - C. The visualiser ....................................................................................................................................................................................5

## CHAPTER 2 – SYSTEM DESIGN

- 2.1 BACKGROUND TOOLS AND CONCEPTS ..................................................................................................................................................7
  - A. Tools and resources ................................................................................................................................................................................7
  - B. Coding and design methods .................................................................................................................................................................7
  - C. Algorithms ...........................................................................................................................................................................................8
  - 2.2 SYSTEM ARCHITECTURE .......................................................................................................................................................................9
  - A. Overview of different layers .................................................................................................................................................................9
  - B. Diagram showing the relationships between the main modules ......................................................................................................9
  - C. Modules and their interfaces ..............................................................................................................................................................11
  - 2.3 USER OPERATIONS ...........................................................................................................................................................................13
  - A. User interaction with the system .......................................................................................................................................................13
  - B. User scenarios ......................................................................................................................................................................................14

## CHAPTER 3 - EVALUATION

- 3.1 TESTING PROCEDURES, TOOLS AND RESULTS ...................................................................................................................................20
  - A. Testing procedures and tools ..............................................................................................................................................................20
  - B. Testing results .....................................................................................................................................................................................22
  - 3.2 INTERPRETATION OF THESE RESULTS ........................................................................................................................................24

## CHAPTER 4 – PROJECT MANAGEMENT

- 4.1 KIM NGUYEN’S PART ................................................................................................................................................................................26
  - A. Project plan and execution .................................................................................................................................................................26
  - B. Scope and schedule .............................................................................................................................................................................27
  - C. Tools and skills ......................................................................................................................................................................................27
  - D. Difficulties ...........................................................................................................................................................................................28
  - E. Group management .............................................................................................................................................................................28
  - F. Lessons learned ...................................................................................................................................................................................28
  - 4.2 PARAMPAL POONI’S PART ..............................................................................................................................................................29
  - A. Project plan and execution .................................................................................................................................................................29
  - B. Scope and schedule .............................................................................................................................................................................31
  - C. Tools and skills ......................................................................................................................................................................................31
  - D. Difficulties ...........................................................................................................................................................................................31
  - E. Group management .............................................................................................................................................................................32
  - F. Lessons learned ...................................................................................................................................................................................32

## CHAPTER 5 – CONCLUSION

- ..................................................................................................................................................................................................................33

## REFERENCES

- ..................................................................................................................................................................................................................34

## APPENDIX 1

- ..................................................................................................................................................................................................................35

## APPENDIX 2

- ..................................................................................................................................................................................................................38
CHAPTER 1 - CONTEXT

1.1 INTRODUCTION

A. PROJECT PROBLEM

Our project is under Sun Microsystems Labs’ sponsor and it involves the development of hardware (i.e. Sun SPOT nodes) specific code as well as an interface that allows users to monitor a network of Sun SPOT nodes. Hence, the first problem is developing the software for each sensor node which includes the higher layers of the communications protocol stack, namely routing, clustering and application layers. Secondly, we should also address the problem of developing a user front end (i.e. the visualiser) at the base station so that the collected values of sensor nodes in the network can be visualised.

B. PROJECT MAIN GOALS

The main goal of the entire project has been to develop a system that collects temperature and light data from a given area and report it back to the user in a simple user interface. This goal has been accomplished by completing three major tasks which included 1) Developing a simulation for a many to 1 data collection system, 2) Develop a visualiser (GUI) for monitoring the temperature/Light of an enclosure and 3) Migrate all code to the Squawk environment and then run on the required hardware (i.e. Sun SPOT nodes).

As a team our goals were simple, to accomplish all of these tasks mentioned to satisfy both our supervisors and client. But personally, as individuals in a group environment we each wanted to ensure that the code and documentation that we developed was at a high standard, so it could be actually useful for our client. We did this by adhering to all the coding practices mentioned in the requirements and planning report, such as code reviews, consistency in code layout, testing and so on.

1.2 OVERVIEW OF THE COMPANY AND SIGNIFICANCE OF THE GROUP’S WORK

A. KIM NGUYEN’S PART

- **Main interests and activities of the company**
  
  Our project is sponsored by Sun Microsystems Laboratories [7]. Sun Microsystems Labs was established in 1990 (8 years after the foundation of Sun Microsystems, Inc.) and is the applied research and advanced development arm of Sun Microsystems, Inc. Projects that researchers at Sun Labs are working on are of great significance to the evolution of technology and to our society’s future. Current Sun Labs research projects include exploration into:

  - Asynchronous and high-speed circuits, Proximity Communication
  - Next Generation Distributed Systems
- Security for the Network and the Internet
- Advanced Search for documents, music and speech
- **Wireless Sensors (the area that our project has been working on)**
- Expansion and enhancement of the JavaTM Technology Platform
- Next Generation clients and hardware platforms
- Digital Camera
- High productivity computing
- Programming Languages of the future

One interesting point which may distinguish Sun Labs from many other R&D groups is that it is of the highest rates of technology transfer. In addition, Sun Labs also pursues high-risk projects however they are of dramatic potential.

With the vision of "Everyone and everything participating on the network", Sun Microsystems, Inc. in general and Sun Microsystems Labs in particular have branches world-wide. For example in Australia they have Sun Australia and Sun Brisbane is part of Sun Australia. There are many people who work remotely for Sun Labs all over the world and our external supervisor, Dr. Cristina Cifuentes included.

- **The department that we were assigned in the company’s structure**
  The department that our project was assigned to is Sun SPOT project which is based in Menlo Park, California, with team members in Menlo Park, Boston, London, and Brisbane. As being mentioned before, Wireless Sensors is one important research area that researchers at Sun Microsystems Labs are working on, and Sun SPOT (Small Programmable Object Technology) is part of the effort to push Java into new applications (i.e. sensor nodes). This project could create innovative sensor applications and accelerate solutions for some of the lingering challenges in Wireless Sensor Network area.

- **Contribution to the company**
  Our project is one of student projects that are part of Collaborative Research agreements with Sun Labs on use of the Sun SPOTs for teaching and research purposes. These student projects are good proof-of-concepts for things that can be done with SPOTs. Students like us can make their projects (code) available and share with others.

  The main focus of our project is developing a multi-hop building monitoring system using Sun SPOT sensor nodes. If successful we would share our work in terms of (1) Building a simulation for Wireless Sensor Network using J-Sim, (2) Building a simple visualiser to monitor the network sensor nodes in an area, and (3) Building a multi-hop routing protocol for Sun SPOT nodes; hence people who make use of Sun SPOT for teaching and research purposes can make benefit from the work that we have done.

**B. PARAMPAL POONI’S PART**

- **Main interests and activities of the company**

  Sun Laboratories (the applied research and advanced development arm of Sun Microsystems) main interests lie in the research and development of new technologies. Furthermore Sun labs has been
responsible for many technology advancements and inventions which have made Sun a technology powerhouse. Below are some key areas of research that Sun is specifically interested in and is currently apart of.

Advanced Hardware Research Developing a novel system architecture for Java and New algorithms for multi-way partitioning, scheduling and packing of large graphs. Advanced Software Research Investigating and developing technologies that improve the ability of people to find and organize information in an enterprise setting and Java Virtual Machine for small devices. Clients Research: Investigating and developing mechanisms to ease the deployment of secure wireless sensor networks and Small Java powered Wireless Transducer Devices.

The bolded project in the client research area relates to our project of wireless sensor networks.

Specifically the research undertaken by our client Cristina Cifuentes consists of the "design, implementation and evaluation of compilation techniques for multiprocessor and uni processor machines, with an aim at ease of portability across a variety of machines." Her recent work includes partitioning techniques for parallelizing code, as well as static and dynamic compilation techniques for small virtual machines.

Sun Microsystems and Sun Microsystems Labs in particular has development branches throughout the globe, For instance our external supervisor, Dr. Cristina Cifuentes is apart of Sun Australia and is currently an active researcher with the Brisbane branch.

- The department that we were assigned in the company’s structure

The specific department that our project falls under is Sun's R&D labs and is associated with the Sun Spot project which is based in Menlo Park, California. The Sun SPOT project is built up of a team of group members from places all around the world such as Boston, London, and Brisbane. This project plays an important part in the company's research and development, as the research areas of wireless sensors and Sun Spot technologies is considered a hot emerging technology and is expected to lead a four-fold expansion of the Internet within the next five years. Furthermore Sun has begun to push forward in the development of efficient software to run its Sun SPOT nodes through projects such as the Investigating and developing mechanisms to ease the deployment of secure wireless sensor networks project mentioned in the client research area above. Sun believes that projects such as this one will continue to create progress and innovation in a challenging field such as WSN's.

- Contribution to the company

The work that we have produced may be adopted by Sun and be presented for use by there own clients. This however will not be its only use, according to Cristina (our industry supervisor) the main purpose of our and other similar student projects will be to use our work to uncover, further develop and implement new ideas in there own research. More specifically they are looking for new ways in which there Sun SPOT technology can be used, and its projects like ours that helps aid new innovations in there research.

This project relates to existing systems and processes through the use of Sun labs Collaborative Research agreement. This agreement is essentially a system that has been put in place to allow any work produced by our group to be used for future Sun SPOTs teaching and research. Also if we
chose to, we can make our entire project available to others in the WSN community.

The project that we have completed for Sun may not be of great significance in terms of what we developed, but small innovations that we have uncovered during development could very well be looked into. For instance the use of mapping and selection of areas in the visualiser GUI and the use of J-Sim as our simulator were two aspects in which our industry supervisor was very pleased with and will follow up in the future.

1.3 RELATED WORK – SUMMARY OF EXPLORATION OF COMPARABLE SYSTEMS

There are three main products in my project, they are: (1) the simulator which simulates a wireless sensor network, (2) the multi-hop routing protocol which is deployed in Sun SPOT nodes, and (3) the visualiser which visualises collected data from all the nodes in the area.

A. THE SIMULATOR

In this project, J-Sim was chosen as the simulation tool to simulate a Wireless Sensor Network. And the one that can be considered to be our competing product is bSPOT [1]. Although bSPOT is strictly Sun SPOT-oriented and easy to use, it is quite simple and does not support many factors of a Wireless Sensor Network like signal attenuation, energy consumption as well as batteries and antennas.

A general method to develop a simulator includes: 1) Choose the most suitable simulation tool that is able to fulfill the project's requirements well. 2) Make suitable changes to the simulation's code if necessary. 3) Configure the input to make the simulation run.

B. THE MULTI-HOP ROUTING PROTOCOL

There are several routing protocols have been developing for wireless networks of Sun SPOT sensor nodes and we will consider one-hop routing protocols (radio protocol and radiogram protocol) that were provided in Sun SPOT SDK package are our competing products. There are many limitations of these one-hop routing protocols. Firstly these protocols only provide simple data transmission scheme like sending data directly to the sink. However each sensor node has only a fixed communication range, thus nodes which are far from the sink might not communicate with the sink. Secondly, using one-hop routings my lead to overhead at the sink since it might have to process too much data at a time.

To build a multi-hop routing protocol for Wireless Sensor Network first you should need a suitable shortest-path algorithm. Then you should test the built protocol using a simulation. After having the protocol work on the simulation you can deploy the code to real nodes to test whether the protocol works properly and as expected or not.

C. THE VISUALISER

MoteView Monitoring Software [4] is considered to be our competing product. MoteView was designed to be the primary user interface between a user and a deployed network of wireless sensors. This is a great tool that allows users to easily monitor the network as well as to connect to the database, to analyse, and to graph sensor readings. However, MoteView only supports the MICA platform of wireless sensor network...
hardware and some sensor boards manufactured by Crossbow. Hence we could not use MoteView to
monitor a network of Sun SPOT sensors.

The general method to make a data visualiser includes two main steps. The fisrt step is designing the
visualiser's interface so that information can be displayed plausibly and kept track of. Then the second step
is using a programming language that well supports GUI (Graphic User Interface).
CHAPTER 2 – SYSTEM DESIGN

2.1 BACKGROUND TOOLS AND CONCEPTS

A. TOOLS AND RESOURCES

The main tools that were used during the project included Java and its libraries, Squawk SDK and J-Sim. The hardware involved in the project included Sun SPOT nodes and a basic PC with the packages mentioned above installed.

Since our project requires our code to be written in Java to run on the Sun SPOT nodes, we have chosen Java and its related packages to develop the entire system. For the user interface, an open source Java graphing library (J2D) was used to present line graphs of the temperature and light recorded for selected nodes. This library was used for two key reasons, an external graphing library was needed since Java's library doesn't provide an easy way to develop graphs, and it did exactly what we needed, while still being simple to learn and use.

The Sun SPOTs SDK was essential to the project as we needed to use its libraries to develop code for the Sun SPOT nodes. By reading simple examples found with the SDK resource we were able to develop the needed code for the system. J-Sim, a simulation tool, was used for the first task of the project which was to simulate the path taken from each node to the sink and ensure it was the shortest. J-Sim was chosen over other possible candidates such as bSpots for a number of reasons such as ease of use, ability to do what is required and familiarity with the tool. (This is thoroughly discussed in this Wiki document: [10])

The use of Sun SPOT Nodes were essential for the successful completion of the project. As mentioned in section one of this document, the code that we developed in the earlier stages of the project needed to be migrated to the Squawk environment and then run on Sun SPOT nodes, hence showing the necessity of the nodes.

Access to a computer which has all the packages mentioned installed meant that each member in our group could develop at home within their own free time. This made development on the project more convenient for all members in our group.

Although not essential to the project's completion, we made use of a document management tool (ie. wiki) which allowed our group to collaborate on issues such as choosing a simulation tool or writing reports, outside scheduled meeting times. We also used the Wiki to record meeting summaries for each meeting that took place with our supervisor. (See Wiki here: [10])

B. CODING AND DESIGN METHODS

During and after development, our group made use of a number of coding and design methodologies, such as inheritance, minimization of code coupling and refactoring of code.
Our group chose to use inheritance to abstract classes within the user interface and simulation code. From our previous studies in computer science, we knew that inheritance would contribute to improved code design, extensibility and also its re-usability in the future, so for these reasons we decided to use it whenever needed.

A small concern that played a part during coding was to ensure that each class we developed had low dependency/coupling between other classes. Initially some of the user interface classes showed to have high coupling as many of the classes depended heavily on the VisualiserGUI class. This was spotted during a weekly code review by Kim and was quickly fixed by dividing the VisualizerGUI class into many classes by Parampal. We chose to ensure our code had low coupling because this is a key sign of a well structured computer system and makes code a lot easier to understand and maintain.

As mentioned previously, our group held code reviews on a weekly basis to ensure design errors/flaws were detected early and that any code that was developed was at a high standard. As a result when issues were detected, for example in the codes design, we would refactor the code to ensure the issue was resolved. For instance, Parampal noticed an issue in the simulation code where, although not essential, inheritance could be used. After a short group discussion, this was added appropriately by Kim.

Due to the nature of the project we felt there were no common design patterns that we could make use of, except those forced upon us, ie. Java's swing library follows a structured way to add components to a window.

To improve the look and readability of our code in general, our group decided to implement a standard coding strategy, which meant we decided on how our code should be written, in terms of spacing between braces etc. This was done to make our entire code uniform and provide future users of our code an easier time understanding what we have implemented. This issue was discussed on our wiki [10] and was later set in our IDE to ensure we follow the decided standards.

C. ALGORITHMS

In this project, we need an algorithm to build the multi-hop routing protocol. Among several available routing algorithms we chose Bellman-Ford [2], a distance-vector algorithm, to find the shortest path from each sensor node to the sink. The routing algorithm should be simple to avoid overhead since each sensor node has only a limited memory capacity. Using Bellman-Ford, each node estimates distance to every other node in the network (distance vectors) and transmits the information to its neighbors, then it receives similar distance-vectors from its neighbors. Bellman-Ford algorithm can adapt to traffic changes and link failures as well as is suitable for networks with multiple administrative entities. Hence, Bellman-Ford was a good choice for building a multi-hop routing protocol in our project.
2.2 SYSTEM ARCHITECTURE

A. OVERVIEW OF DIFFERENT LAYERS

There are four main layers to our application as a whole, the database layer, visualiser (GUI Layer), J-Sim (Simulation Layer) and Sensor nodes. Each layer can function separately, but only when placed together will the complete application work as desired.

1. The Gui Layer is used to present the user with statistical data that has been retrieved by each node. This includes the name, id, status, temperature, light and location of each node. All this data is presented in tables, but only the most current temperature and light data for each node is shown. To show how temperature and light performs over time, the visualiser also presents the user with two line graphs. The temperature graph clearly indicates how the temperature of selected nodes varies over time. Similarly the Light Line graph performs the same function with light over time.

2. The database layer essentially allows other layers to store and retrieve data from the database. The database used is MySQL, and it was chosen due to its light weight nature and it being open source. There are 3 tables in the database, Node, DataNode and NeighborNode. As each name suggests, Node stores all data referring to each node such as each the node’s id, name, location, power and data rate. The DataNode table keeps the statistical data gathered by each node such as the temperature, light, status and the time the data was gathered. The NeighborNode table is used to keep track of each nodes closest neighbor, so it simply stores two node ids, one of the node and the other of its neighbor.

3. The simulation layer is built using J-Sim [3] is used to simulate a many-to-one data collection system of wireless sensor networks. The main purpose of developing the simulation is to test the simple many-to-one routing protocol before migrating all the codes to J2ME/Squawk environment. To support simulating Wireless Sensor Networks, J-Sim provides object-oriented definitions of (i) target, sensor and sink nodes, (ii) sensor and wireless communication channels, mobility model and power model. Input data such as network’s configuration that includes network’s size, locations of all the nodes in the area and the range of each sensor node is written in the form of a Tcl/Java file. The output of the simulation will be stored into the database which is connected to the visualiser so that the simulation’s results could be visualised.

4. The sensor nodes layer involves working with Sun SPOT sensor nodes. There are two main types of hardware that are involved in this layer: the base station and target sensor node. Each type of hardware requires a specific type of code to run. The base station's code allows it to receive data from target nodes and to store that data into the database. The target sensor node's code allows it to read data (temperature/light) from the sensor node through an interface (provided in Sun SPOT SDK) and send that data to the base station.

B. DIAGRAM SHOWING THE RELATIONSHIPS BETWEEN THE MAIN MODULES

Below is a simple diagram showing the layers mentioned above interacting. We can see that the three layers, Visualiser, J-Sim and Sensor node all refer to the database, which is essentially how we integrated our application. The direction of the lines indicate the way the data is moving, so we can clearly see that the visualiser is retrieving data from the database and both the sensor nodes and J-Sim simulation is storing data to the database.
Relationship between J-Sim, Database and Visualiser modules

The following sequence diagrams and the layers diagram seen above, clearly show the interaction between components/modules in our application. The first diagram shows how the simulation and the visualiser work together to report the necessary data to the user. As you can see the J-Sim module, goes through each node, generates random temperature and Light data, sends it to the sink (via the shortest path through other nodes in the network) and then stores it in the database. This is repeated every 10 seconds until the simulation is stopped by a user. At the same time the visualiser is running and it gathers temperature, light and other data from the database for each node and displays it to the user. This is also repeated every 10 seconds.
C. MODULES AND THEIR INTERFACES

The main modules for each layer mentioned above are listed below.

1. Visualiser Modules

   See Appendix 2 for class diagram.

   - **DataNodeGui**: This module handles all the components that are placed in the data tab. The main method used to gather the information required from the database is `getTableData`, and is called everytime a user clicks the refresh button to update the details.

   - **LineGraphGui**: This module handles all the components that are placed in the Temp Line graph and Light Line graph tabs. `addNewTrace` is the key method used, as it is called everytime a user selects a checkbox next to a node in the node table.
NodeGui: This module handles all the components that are placed in the node table located to the left of the screen. The method getTableData is used to retrieve all necessary node details from the database.

TopologyGui: This module is the most complex as it handles all the components that are placed in the topology tab. There are three key methods used, makeNodes, makeNodeNeighbors and paintComponent. As their name suggests makeNode creates the circles which represent nodes, makeNodeNeighbors create lines which represent links between a node and its neighbors and paintComponent draws both of these sets of objects to the screen. paintComponent also draws, delete's, moves and resizes the selected areas (rectangles) to the screen based on a user's mouse clicks.

2. Database Modules
See Appendix 2 for class diagram.

DatabaseConnection: This module gives database access to other classes through its setupConnection method.

NodeTable/DataNodeTable/NeighborNodeTable: Each of these modules provides access to their respective table in the database. Each of these classes contains a getTableById and getAllTable method which do as each name suggests. DataNodeTable however contains a method which gets a node's data when given the node's name, called getNodeDataByName. This method contains multiple queries to achieve the required result.

The primary author of all the above modules and any code related to the Visualiser and Database layers is Parampal Pooni.

3. Simulation Modules
See Appendix 2 for class diagram.

Sensor node: This module represents a sensor node which is equipped with (1) a Sensor Protocol Stack which allows it to capture signals generated by target nodes through a sensor channel, and (2) a Wireless Protocol Stack which enables it to send reports to the sink nodes through a wireless channel.

Target node: Each target node will generate signals which are then sent to sensor nodes through the sensor channel. A target node includes (1) Target Agent Layer and (2) Sensor Physical Layer.

Sink node: A sink node is built in a plug-and-play fashion using a Sensor Application Layer (SensorApp), an Interface Layer (WirelessAgent) with the Wireless Protocol stack: network layer (AODV protocol (drcl.inet.protocol.aodv.AODV)), MAC layer (MAC 802.11 (drcl.inet.mac.Mac_802_11)), and physical layer (drcl.inet.mac.WirelessPhy).

Sensor channel: The Sensor Channel receives a signal from a target node and sends a copy of that signal to all sensor nodes within the transmission radius of that target node.
• **Wireless channel**: Through the Wireless Channel, data from sensor is forwarded to the sink.

• In addition, the simulation also contains a Battery model, CPU model and Radio model.

4. SensorNode Modules

See [Appendix 2](#) for class diagram.

• **Bellman-Ford**: the module contains the implementation of a multi-hop routing protocol, finding the shortest path using a distance-vector algorithm (Bellman-Ford)

• **Spot's code**: the code is deployed in each sensor node so that each node can communicate with each other as long as with the base station.
  - **Sender**: open connection to another node; read data from hardware and send that data through the opened connection.
  - **Startup**: initialize and start the sender.

• **Base station's code**: the code is deployed in each sensor node so that each node can communicate with each other as long as with the base station.
  - **Display**: open connection to another node; read data through the opened connection and process that data.
  - **Startup**: initialize and start the sender.

The primary author of the simulation and SensorNode modules listed above is Kim Nguyen.

### 2.3 USER OPERATIONS

#### A. USER INTERACTION WITH THE SYSTEM

The main way that users interact with the system is through the use of the visualiser, as it displays all the important data such as temperature and light retrieved from each node.

**Viewing Statistical Data**

The visualiser separates the main statistical data into page tabs, so it is easy for a user to view large amounts of data without being overwhelmed. The visualiser tabs include:

1. **Data**: Which shows the most current data (temp, light, location etc) retrieved from the database for each node. This is placed inside an adjustable table.

2. **Topology**: This tab shows graphically the location of each node on a given map. Each node is represented by a red circle, with its id in the middle. A node’s neighbors are represented by direct lines drawn from the node to each neighbor node. This tab is talked about further in the User interactions section, showing other ways a user can interact it.

3. **Temperature Line graph**: This tab shows selected nodes in a line graph and how there temperature varies over time.

4. **Light Line graph**: This tab shows selected nodes in a line graph and how there light varies over time. Further interactions with the line graph are talked about in the User interactions section.

Apart from the tabs mentioned above, on the LHS of the screen is a fixed table which shows vital stats of each node, i.e. its name, status, id and whether or not it is being graphed.
At the top of the screen is a button named Refresh which allows a user to manually refresh all tabs and the node table. There is also a simple menu system above the button which can be used to exit or also refresh the node data.

At the bottom of the screen a user can see response messages from the system in a small text box. It is populated whenever a user performs simple system operations, for instance if a user clicks refresh, the word refreshing will appear in the message box.

**User interactions**
Within the visualiser the user can perform a number of operations to view this statistical data in different ways. For instance the user may select up to three nodes at a time to display their temperature and light results in separate line graphs. This can be done by selecting or deselecting a checkbox found next to each node in the node table (The table found at the LHS of the screen).

As mentioned in tab 3 the user can view the topology of the network of nodes. Additionally the user may also choose to turn on / off a map of the networks location via a checkbox found near the top of the tab. The map allows users to clearly see where each node is located. An additional interaction is provided by allowing a user to create resizable and movable rectangular areas which will report back the selected areas current temp/light statistics. This can be achieved by right clicking to create the rectangle, left clicking the bottom right corner and dragging it for resizing and left clicking the center of the rectangle and dragging to move it. The user can also choose to delete a rectangle by middle clicking on it anywhere.
B. USER SCENARIOS

The following are all common scenarios that could be carried out by a user, when operating the visualiser.

**S1:** Open the visualiser, go to the data tab, view the current temperature and light for node 2, press refresh after 15 seconds, inspect the values have changed and then close the visualiser.

*A screenshot showing the data tab and refresh button*
S2: Open the visualiser, go to topology tab, show the map as the background, create a selection rectangle, move it to the desired location (the university tennis courts), resize the rectangle if needed, delete the selected region and then close the visualiser.

A screenshot showing the topology
A screenshot showing the topology with map
A screenshot showing the topology with map and a selected area.
S3: Open the visualiser, go to the Temp Line graph tab, check the checkboxes next to nodes 1, 2 and 3 in the nodes table, view the temp line graph for 1 minute, go the Light line graph tab and then close the visualiser.

*A screenshot showing the temperature line graph*
A. TESTING PROCEDURES AND TOOLS

1. Unit testing
During development we have used unit tests to validate each module in our source code and to ensure the system was working correctly. We also made use of test driven development, which essentially involves writing tests before the source code and only writing code to make the tests pass. This technique is apart of extreme programming and has had numerous benefits in our application, such as building the system faster, increased trust in code and catching defects earlier.

The unit tests have been particularly useful as they have helped prove to our supervisors that our code works correctly and is ready for migration to the Sun Sensor Nodes. In particular, the test cases have shown to cover all methods of each class, so each class can guarantee correctness according to the tests provided to our supervisors.

To assist us with writing and running these tests effectively, we have made use of JUnit, a popular testing framework for Java. Some benefits that JUnit has provided us include a reduction in redundant test code, Unit tests can be composed into a hierarchy of test suites and it has allowed us to write code faster while increasing quality. Specifically, JUnit tests were used to test the database modules.

Also as previously mentioned in the progress report, to ensure old bugs are not re-introduced each time new functionality is implemented, we have been using regression testing throughout development. To carry this out we ran every test that had been written whenever new code was implemented. This has allowed us to detect if code that had been written previously got broken from a later change.

To test the simulation and sensor node code, several network scenarios were constructed, both simple and complex. The best way to test whether the simulation and the whole network of sensor nodes functions as expected is setting up a sample network with predefined behaviors, then run the code to test whether the actual output is the same as expected.

2. Acceptance testing
As adapted from the planning report, the following shows a list of acceptance tests that have been used to evaluate the needs of our clients/supervisor. So essentially, if these tests pass, the supervisor can be sure that the entire system is doing what they require.

<table>
<thead>
<tr>
<th>Acceptance Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
</tr>
<tr>
<td>In the J-Sim</td>
</tr>
<tr>
<td>simulator, send data from a node (X) to the sink from 50 meters away, via the shortest path</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Using the Sun Sensor nodes, send data from a node (X) to the sink from 50 meters away, via the shortest path</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>View temperature results periodically every 10 seconds</td>
</tr>
</tbody>
</table>

3. Usability testing

Usability tests were used to validate the correctness and usability of the visualiser layer. To perform these tests, specific scenarios were created to ensure all functions of the visualiser were examined. Then 4 individuals aged 20, 21, 20 and 22 carried out these scenarios and there results have been documented in the following section.

Below are the scenarios in which the 4 individuals carried out.

S1: Open the visualiser, go to the data tab, view the current temperature and light for node 2, press refresh after 15 seconds, inspect the values have changed and then close the visualiser.

S2: Open the visualiser, go to topology tab, show the map as the background, create a selection rectangle,
move it to the desired location (the university tennis courts), resize the rectangle if needed, delete the selected region and then close the visualiser.

**S3**: Open the visualiser, go to the Temp Line graph tab, check the checkboxes next to nodes 1, 2 and 3 in the nodes table, view the temp line graph for 1 minute, go the Light line graph tab and then close the visualiser.

*Note these have been adapted from section 2.3 - Visualiser user scenarios.*

4. Evaluation from supervisors
The feedback from supervisors during the design and development process of our system has been used to guide and help through the semester. First, internal supervisors (i.e. Selvakennedy Selvadurai and Uwe Roehm) have evaluated our work after each stage including: implementing the simulation, designing and implementing our own protocol, implementing the visualiser, and migrating the designed software to Sun SPOT nodes. In addition, external supervisor (i.e. Dr. Cristina Cifuentes) also attended our demo (in week 12) and gave us useful feedbacks for the visualiser as well as the simulation and the routing protocol.

B. TESTING RESULTS

1. Performance
Some small tests were created to test the time it took to retrieve node and nodedata from the database to ensure it did not exceed greater than a second. But the results showed that each method ran successfully under 1/2 a second with getNodeDataByName (DataNodTable.java) being the slowest showing to take 0.26 seconds to complete. Below are the results gathered from all scripted tests run against the core modules of the database code.

![Performance Results Chart](image)

Other tests were also created to measure the performance of the simulation and sensor nodes. The main purpose here was to test the time it took the simulation and sensor nodes to write data into the database. Since the visualiser needs to read data from the database at real time, a long delay time is not expected. We set up a threshold of 0.5 second, and the results show that it took maximum of 0.45 second to update data to the database.
2. Correctness
As mentioned in the previous section, JUnit tests were developed for the database module. Each of these tests were created before the development of the code and were used to ensure the correctness of the code developed. Once a test passed the developer knew that the code was correct. Thankfully, all the tests have passed, which has ensured the codes correctness.

For the simulation and sensor nodes, most of the tests have passed. Since the multi-hop routing protocol was not perfect, in some complex scenarios it was not the shortest-path that was chosen. However almost all the tests for the simulation and sensor nodes had results as expected.

3. Usability

- Preparation notes

Each user was given some preparation notes to guide them through the test which included:
Instructions on how to create, delete, resize and move a selected area, where the node table is located and how to graph a node.

- User summary

Here is a quick summary of each user that took part in the test.

User 1: Computer Science student, 20 years old, male
User 2: Computer Science student, 21 years old, female
User 3: Computer Science student, 20 years old, male
User 4: Computer Science student, 22 years old, female

We felt that the users selected for the tests should have some form of computer science background, since our clients from Sun who will be using the system have such knowledge. To be fair to both genders, we took the same number of males as females. And in terms of age, we decided to choose fellow students that we knew for convenience whose ages are grouped between 20 and 22. It is important to note that no user tested had any prior knowledge of the system, and possessed no background knowledge in wireless sensor networks.

- Observations

Below is a summary of the errors or abnormal actions performed by the users in each scenario.

S1: Open the visualiser, go to the data tab, view the current temperature and light for node 2, press refresh after 15 seconds, inspect the values have changed and then close the visualiser.
No real errors detected by users. But user 2 and 4 each did not wait 15 seconds before clicking refresh. User 2 waited approx. 10 seconds and user 4 waited approx. 5 seconds.

S2: Open the visualiser, go to topology tab, show the map as the background, create a selection rectangle, move it to the desired location (the university tennis courts), resize the rectangle if needed, delete the selected region and then close the visualiser.
All users were able to show the map as the background, create a selected area and delete a selected area. User 2 struggled a little bit when attempting to resize the rectangle, as they attempted to drag the wrong corner. But they soon corrected themselves and used the bottom right corner to resize. User 3 found it difficult to locate the required area to select on the map (which was the tennis courts). And
User 4 decided to resize the rectangle before moving it into position which did not cause any problem, as they still managed to cover the required area.

**S3**: Open the visualiser, go to the Temp Line graph tab, check the checkboxes next to nodes 1, 2 and 3 in the nodes table, view the temp line graph for 1 minute, go the Light line graph tab and then close the visualiser.
All users were able to complete this task successfully, without any major problems. User 1 did select the checkbox next to node 4 instead of 3, but they corrected themselves by deselecting that node and selecting the correct node.

- **Results/Conclusions**

From the above observations it was clear that the usability of the system can be generally considered good, as all users were able to complete each task successfully. A couple of issues arose from scenario 3 where user 2 had forgotten which corner to drag when resizing the rectangle, but that issue was quickly resolved by the user and as a result no further action in terms of development was needed. User 3 found the map hard to read, but this is not a direct fault of the system as the map was provided to us, so having a clear map for the topology will be considered as a requirement in the future.

It is important to note that all of the listed scenarios run by the users cover all the core functional requirements of the visualiser. And as a result of all users completing the given scenarios successfully, we can safely say that the visualiser works as required by the client.

### 3.2 INTERPRETATION OF THESE RESULTS

**A. WORK QUALITY CONCLUSION**

We have been working on this project for more than 10 weeks and produced three main products, they are: (1) The simulation, (2) The visualiser, and (3) The multi-hop routing protocol. It is not easy to evaluate our own work ourselves, however based on the Requirements and Planning Report (Appendix 1) that we set up at the beginning of the semester and compare the project’s expected outputs with its actual outputs we could have some conclusion about the work that we have done.

Firstly, the simulation that was created can simulate a wireless sensor network with a flexible number of nodes and other network factors successfully. The simulation can be extensible both in terms of network size as well as network complexity. Its outputs are stored in the central database so that users are able to see the behaviors of the network simulation on the visualiser. This simulation can be considered at a good quality as it serves the basic goals quite well.

The visualiser can be considered a success as well. Its basic functionalities can help users monitor the network easily. With a friendly and easy-to-use user interface, users may not have any trouble of navigating. In addition, beyond the requirements that were originally planned, the visualiser provides the function to display the average temperature/light at any place in the area. At the moment, the visualiser is quite basic; however it is very easy to be extended in the future.

Lastly, developing the multi-hop routing protocol and migrating the code to Sun SPOT nodes can be considered the hardest part in my project since we need to work with real hardware – Sun SPOT sensor nodes. We constructed simple multi-hop routing protocol and a base station that can gather data from all
the nodes at real time which is an extension of the project and can be very helpful to other purposes as well. The routing protocol does work though quite simple at the moment. Hence, this part should be under further improvement.

B. STRENGTHS AND WEAKNESSES

- **Strengths**
  - The visualiser is easy to use and is capable of displaying useful statistics such as the latest information of each node and the comparison in terms of temperature or light among several sensor nodes.
  - Temperature and light at any particular place in the area can be calculated based on collected data from all sensor nodes around.
  - The code written for Sun SPOT sensor nodes is useful. Each node can read data from the environment and transmit that data to the base station for further processing.
  - The database used is MySQL which has light weight nature and is open source. These features could make it easy to install the database as well as less memory is needed to store it.

- **Weaknesses**
  - Our multi-hop routing protocol is relatively simple at the moment
  - The visualiser lacks the function of displaying the temperature/light of an area whose size can be variable (i.e. enlarging/narrowing the rectangle).
  - The code for the simulation is quite hard to reuse by people outside our group since originally J-Sim code is relatively complex, however instead of creating sub classes of main classes in J-Sim, we made changes directly to J-Sim’s source code. Thus, it would be quite hard for other people to follow our simulation’s code.

C. FURTHER EXTENSIONS AND IMPROVEMENTS

There are several areas for further extensions as well as improvements in our project.

- Introduce “clustering” technique in the multi-hop routing protocol.
- Make the visualiser reflect the temperature/light of an area with a variable size instead of a small fixed size.
- Change “shortest-path” criteria to “least-power” criteria in the routing protocol.
CHAPTER 4 – PROJECT MANAGEMENT

4.1 KIM NGUYEN’S PART

A. PROJECT PLAN AND EXECUTION

This project’s start date and end date are 24-July (beginning of week 1) and 27-October (end of week 13) of semester 2, 2006 respectively. The project initial plan was created at the beginning of the semester and can be seen in the Appendix 1. In the initial plan we divided our project into 9 main tasks throughout the semester.

- Task 1: Choose group and project
  Scheduled time: 1 week (week 1).
  Actual time: 1 week.
  Six INFO3600 students were provided with more than 20 projects at the beginning of the semester. Parampal and I were both interested in the project “A Multi-hop Building Monitoring System Using Sun SPOT Sensor Nodes” so we formed a group of two doing this project.

- Task 2: Research and decide simulation tool
  Scheduled time: 1 week (week 2).
  Actual time: 1 week.
  Both of us explored the two simulation tools provided (J-Sim and bSPOT), then compared to choose the most suitable one.

- Task 3: Research Wireless Sensor Network (WSN) and related background
  Scheduled time: 1 week (week 3).
  Actual time: 1 week.
  I had some previous experience with WSN so I spent less time than Parampal did for this task. We also studied and did some sample experiments with J-Sim. After week 3 we both had relative background knowledge about WSN and J-Sim.

- Task 4: Design and implement protocol in simulation
  Scheduled time: 3 weeks (week 4, 5, 6).
  Actual time: 3 weeks.
  We chose a distance-vector protocol available in J-Sim to implement the simulation. We also planned to introduce “clustering” into the protocol; however using J-Sim to build the simulator was more complex than expected, thus we had to spend more time to make the simulation work and left “clustering” for later development. We had to extend this task one more week (i.e. finished in week 7).

- Task 5: Develop a visualiser for application data
  Scheduled time: 4 weeks (week 4, 5, 6, 7).
  Actual time: 4 weeks.
Parampal was responsible for developing the visualiser. Since he was good at GUI Java programming, we completed this task on time.

- Task 6: Develop routing protocol further  
  Scheduled time: 1 week (week 8).  
  Actual time: 1 week.  
  We planned to change the cost between two nodes from constant to actual distance and to change from “shortest hop” criteria to “least energy” criteria then to implement the design in the simulation. However later on we decided not to implement second change since it was probably over-complex for our project.

- Task 7: Test and prove-correctness to supervisors  
  Scheduled time: 1 week (week 9).  
  Actual time: 1 week.  
  We scheduled to make the final test of the whole integrated system (i.e. the simulation and the visualiser); however the integration process was still in progress so we delayed this task until week 10 (i.e. after the break). We also had feedback and further instructions from our supervisors.  
  Since the simulation was not completed perfectly, we extended one week (the semester break) for the simulation. In addition, we also added one more task “Study and prepare for Squawk migration”.

- Task 8: Migrate developed software to Sun SPOT sensor nodes  
  Scheduled time: 3 weeks (week 10, 11, 12).  
  Actual time: 4 weeks.  
  Major Milestone  
  We spent these three weeks playing with sensor nodes provided and migrating code to the nodes, from simple (using one-hop routing protocol) to more complicated code (simple multi-hop routing protocol). A majority of the migration to the Sun SPOT sensor nodes was completed and this task was still needed to be extended into week 13.

- Task 9: Finish up the project  
  Scheduled time: 1 week (week 13).  
  Actual time: 1 week.  
  We used the last week to make final changes to the simulation as well as the visualiser. Also, we used this week to complete migrating code to sensor nodes and prepared for the final demo and the presentation.

B. SCOPE AND SCHEDULE
Throughout doing the project, I have found that our group in general and I in particular have set quite suitable scope for each given time. However some scope was a bit too high such as introducing “clustering” to the protocol so we had to reduce the scope.  
In general, the detailed schedule (and also the revised one) that we created has been proven to be feasible and reasonable. We could follow the schedule and used it to control our progress.

C. TOOLS AND SKILLS
- Tools: Choosing a right tool was an important part of our project. As being mentioned in the Requirement and Planning Report, the tool that we must choose was the simulation tool. We found out that there were only 2 suitable simulation tools: J-Sim and bSpot and we decided that
J-Sim would be more suitable for our project however it was not perfect. Since we did explore the tools quite carefully before choosing one, so we knew that J-Sim has some limitations like: more complex to use, may contain overhead etc. however J-Sim still was the best tool that we could choose.

- **Skills:** In my opinion, I think I have quite enough right skills for doing the project. In terms of technical skills, I had some knowledge of network protocol & programming as well as some previous experience with Wireless Sensor Network. In addition, I am also familiar with Java programming language. Hence, it did not take me much time to study background knowledge. In terms of group working skill, I was quite confident since I had some experiences working in group before.

D. DIFFICULTIES

My part in the project includes developing the simulation and migrating code to sensor nodes (work with Parampal). As being stated in Part 6: SWOT Analysis of the Requirement and Planning Report ([Appendix 1](#)), we did predict some difficulties that we might have to cope with. Firstly, J-Sim was a great tool for building the simulation; however it was not ideal and easy to study, hence we had to spend more time than expected to work with J-Sim. Secondly, time management was a significant problem in my project. I had from 2 other 3rd year subjects to do this semester; hence sometimes I could not finish the task on time since I had to spend time for other assignments and quizzes as well. To overcome this difficulty I tried to schedule my time table reasonably and to adjust my original plan flexibly. Lastly, working with Sun SPOT hardware (i.e. sensor nodes) was very new to me. In order to migrate code to real sensor nodes I had to spend time and effort working with some limited examples first, then figured out the problem myself with little help from outside since Sun SPOT is quite a new project.

E. GROUP MANAGEMENT

Honestly, this is the best group that I have ever had. Group working would be disaster if you lack of effective communication and agreement. However, our group’s communication has been great and efficient. We have met regularly and also made use of Wiki [10] which has already been proven to be a great supporter for our group working. The project integration process was not easy but run quite smoothly. We were both responsible to our own part and understood the other member’s part, thus we could work together effectively to integrate all the parts. The best thing of our group management is that we have shared the work fairly and effectively supported each other.

Our group management has been very good; however there are still areas for improvement. Firstly, in the last a couple of weeks at the end of semester we were both very busy so we did not update our Wiki very frequently. In addition, should we start working on this final report earlier we would make it much better.

F. LESSONS LEARNED

There are many valuable things that I have learned from doing this project. The project has given me a chance to study and work with Wireless Sensor Network which is a very interesting area and attracts me a lot. In addition, the assessment tasks including writing Requirement and planning report, Progress report, Final report, and Presentation have helped me improve my writing as well as my presentation skill which would be very helpful in my later career. Additionally, I have gained some good experiences in group working, if you start early with a well-designed plan then it will be much easier for you to do the project.
4.2 PARAMPAL POONI’S PART

A. PROJECT PLAN AND EXECUTION

At the start of the semester Kim and I decided on an initial plan that would divide up the 9 main tasks that were given to us. We also estimated on the duration of each task, based on its difficulty level which was assessed by our supervisors and ourselves. During week 8 of the semester, this project plan was revised and the duration of some activities were extended and some were decreased. Also based upon the availability, knowledge gained and efficiency of individual group members, some tasks that were assigned to the entire group, were re-assigned and done individually.

The project lasted for a duration of 13 weeks and started on the 24/7/06 and has ended on the 27/10/06. The following is a detailed plan which outlines what we planned to accomplish and what we actually accomplished on a task-by-task basis. Major milestones have also been identified on tasks 4, 7 and 8.

* Task 1: Choose group and project
Scheduled time: 1 week (starting week 1)
Actual time: 1 week (starting week 1)
During this period each of the 6 INFO3600 students identified there favorite projects among those listed on the course website. On Thursday of week 1 we all met online (msn messenger) to decide upon projects and who would be paired with who. As a result Kim Nuygen and I (Parampal Pooni) were paired together to undertake the “A Multi-hop Building Monitoring System Using Sun SPOT Sensor Nodes” project due to our common interest in the topic. We then formally notified our course coordinator, supervisors and client on our choice and then begun work.

* Task 2: Research and decide simulation tool
Scheduled time: 1 week (starting week 2)
Actual time: 1 week (starting week 2)
After we were told of the tasks that we needed to complete during this semester, both our group and supervisors decided that we would need to pick a suitable simulation tool. We had two possible options, bSpots and J-Sim, and by comparing each tools pros and cons individually on our wiki [10], our group decided to pick J-Sim. As seen above, this task was finished within the one week time frame as originally planned.

* Task 3: Research Wireless Sensor Network (WSN) and related background
Scheduled time: 1 week (starting week 3)
Actual time: 1 week (starting week 3)
After further talks with our supervisors, we decided that researching wireless sensor networks (WSN's) initially would be better than learning the field as we go along, so we assigned this week for research on WSN's. Luckily Kim had prior knowledge in this field and was able to assist me in my learning by pointing out certain materials that I should read. During this period we also managed to read up on J-Sim. So by the end of week 3 we both had a sound knowledge in WSN's and J-Sim, and as a result achieved what we had planned to do for this period.

* Task 4: Design and implement protocol in simulation
  * Major Milestone *
Scheduled time: 2 weeks (starting week 4)
Actual time: 3 weeks (starting week 4)
Task 4 consisted of choosing an appropriate routing protocol in J-Sim and implementing it as specified by the supervisors in week 2. We planned to introduce “clustering” into our protocol which was another
concept introduced to us by our supervisors, but this proved to be too complex to achieve in our given timeframe. As a result we were forced to extend this task's deadline, since we had only completed about 65% of the protocol after 2 weeks. By week 7 (3 weeks from starting this task) we were able to complete the task. It is also important to note that initially we planned for this task to be completed by both group members, but since Kim was making greater progress in this task and I was making greater progress in the visualiser task, we decided to assign each task to separate group members to become more efficient.

* Task 5: Develop a visualiser for application data
Scheduled time: 4 weeks (starting week 4)
Actual time: 4 weeks (starting week 4)
During this period I was able to learn and implement the core functionality required of the visualiser. And as mentioned in the previous task, this task was originally assigned to both group members but I became the sole developer to increase productivity in the project. So as scheduled, I completed the set task within the given period.

* Task 6: Develop routing protocol further
Scheduled time: 1 week (starting week 8)
Actual time: 1 week (starting week 8) **not completed**
For task 6 we had originally planned to add additional functionality to the routing protocol, such as changing the cost between two nodes from constant to actual distance and changing from “shortest hop” path to the “least energy” path to the sink. After a week of researching we realized that these features would have to be left out, due to their complexity and our lack of time available.

* Task 7: Test and prove-correctness to supervisors
* Major Milestone *
Scheduled time: 1 week (starting week 9)
Actual time: 2 weeks (starting week 9) **Task changed**
By this period we had planned to complete a working simulation protocol and visualiser. Unfortunately bits of the simulation protocol needed to be completed and furthermore we had been given further instructions from our supervisors to study and prepare for migration to the Squawk environment. So we extended this task by one week to complete the simulation protocol. In addition, we also added another task which was to be done simultaneously in this extra week which was “Study and prepare for Squawk migration”.

* Task 8: Migrate developed software to Sun SPOT sensor nodes
* Major Milestone *
Scheduled time: 3 week (starting week 10)
Actual time: 4 weeks (starting week 10) **Major Milestone**
During this period we completed a majority of the migration of our simulation protocol to the Sun SPOT sensor nodes. Initially we began by running sample programs on the nodes and understanding how to send and receive data along nodes. Then we managed to implement the most of the multi-hop protocol required of us onto the sensor nodes within the given timeframe. This task needed to be extended into week 13.

* Task 9: Finish up the project
Scheduled time: 1 week (starting week 13)
Actual time: 1 week (starting week 13)
The final week of semester was spent making presentation adjustments to the code as well as the visualiser GUI. We also used this week to complete any final touches to the multi-hop protocol on the sensor node's and prepared for our final demonstration.
B. SCOPE AND SCHEDULE

I believe the core of the project that was given to us to complete within the timeframe of a semester was very reasonable, especially for a group of two B.I.T students. However a few exceptions in the form of adding clustering within our protocol, made some tasks beyond a reasonable scope to complete within the set duration.

I believe that that the schedule that we produced in week 5 and the revised schedule made in week 8 were very reasonable and realistic. As a result we were able to use it successfully to complete our projects requirements gradually throughout the semester.

C. TOOLS AND SKILLS

- **Tools**
  Some of the tools we used to build our system included J-Sim (a simulation tool), Eclipse (IDE to develop our java code) and Java (the programming language we developed in).

  We felt that we did a good job in deciding which tools to use at the beginning of the semester, by analysing each tool (and alternatives) for there pros and cons on the wiki. For example when deciding on a simulation tool each group member looked into J-Sim's and bSpots (alternative simulator) pros and cons and by analysing the results as a group we chose J-Sim. In practice we did find J-Sim difficult to use but based on the requirements set, J-Sim was the best simulator for the task.

  Upon the completion of the project I can say that each tool we have chosen has worked well for us and me in particular. As we saw when we performed our evaluations, each tool was able to successfully perform all operations required and assist in the completion of the project.

- **Skills**
  With the advantage of hindsight, coming into the project i had sufficient skills in programming and networking to complete this project successfully. My prior knowledge included basic networking concepts and extensive java knowledge and I believe that this has helped both myself and team produce a quality end product.

D. DIFFICULTIES

The main parts of the project that i developed were the visualiser and the database layer. I also assisted with the migration of the routing protocol to the sensor nodes.

A major problem i came across was learning and developing in java2d to create the topology and selection of areas on a map. For most areas in Java there are common examples that can be found on the internet or text books, but concepts such as manipulating selected areas unfortunately are not as readily available. So the development of this feature required extra time, in an already tight schedule towards the end of semester. To help solve any problems that I did have, I used an online message boards to ask questions and I also referred to the Java API documentation whenever needed.

I also found working with new hardware like sun sensor nodes difficult. Fortunately i was able to work with Kim, where as a group we were able to overcome initial difficulties with coding on the nodes and develop a working system. This was done by referring to the squawk API, and further documentation/examples given to use by our supervisors.
Finally managing my time with other subject became difficult towards the end of semester. This semester I have undertaken 5 units instead of the usual 4 which meant at times I had multiple assignments due close together. Luckily Kim was able to undertake more work on our project during these periods, and similarly I did the same during her busy times during semester.

E. GROUP MANAGEMENT

Personally I felt that our group worked well together and as a result we managed to develop an impressive working system. The way that our group integrated the project was simple, through the use of a database. We were able to work effectively on our own modules and then create a working system, through simple interactions with the database layer. Furthermore if Kim needed to get different data from a specific table, she would let me know (via a meeting or email) and I would assist here as soon as possible.

We were also able to maintain strong communication between group members through the use of our wiki and email. Also multiple group meetings per week and code reviews ensured that our group was able to discuss key issues related to the project and resolve any outstanding issues that may have arisen.

The best thing about our group management was the ability to adapt to change. For instance based on a team member’s schedule, weakness/strength or the realization that working separately would be more efficient, we were able to change our assigned tasks or help each other out to increase the efficiency of our group.

F. LESSONS LEARNED

There were many valuable concepts and theories that I learnt throughout this project. Some of these included the ability to work with emerging technologies i.e. Sun Sensor Nodes, learning the workings and theory of wireless sensor networks and being able to develop my java programming knowledge further in database communication and GUI programming.

As a group, I learnt how to work effectively, together and separately. I realized that sometimes working apart can be more productive than working together on a single module. I have also learnt the importance of early planning and scheduling, as it can help you stay on track and deliver the application when needed.
As a result of the development carried out by our group throughout the semester the required system has been developed.

One of our product’s strengths is that it has a fully tested user interface which displays temperature and light recordings of each node in ways that are beyond those required of us. For instance we used line graphs, topologies and area selection to provide better visual ways to analyze the required data. The main weakness came from the simple nature of the simulation protocol that was developed. This was not a big issue, as the main application which was placed on the nodes was completed and fully functional.

To avoid learning difficulties late in the semester, if our group had its time over we would have begun researching how to migrate onto the nodes earlier. But this would always be hard to achieve with many constraints such as time and other course work placed upon us.

In the future, further enhancements could be made to the system, via the multi-hop protocol (i.e. choosing a different path algorithm) or even further enhancements to the visualiser (i.e. different types of graphs such as bar or scatter graphs).
REFERENCES


[8] SunSpotWorld – Home of Project Sun SPOT, URL: http://www.sunspotworld.com/


Visualiser's class diagram
Simulations's class diagram
Sensor nodes’ diagram
APPENDIX 2

Here is the requirements and planning report that was developed in week 4 of the semester.
INFO3600 REQUIREMENTS AND PLANNING REPORT

PROJECT: A MULTIHOP BUILDING MONITORING SYSTEM USING SUNSPOT SENSOR NODES

PART 1: OVERVIEW

This project involves the development of hardware (ie. SunSPOT nodes) specific code for our sponsor Sun Microsystems. SunSpot nodes can be used as part of a Wireless Sensor Network (WSN) which has a variety of uses within society, for example monitoring an object or environment for certain changes or disruptions.

This project will require us to complete three major tasks: 1) A simulation for a many to 1 data collection system, 2) A visualiser (GUI) for monitoring the temperature of an enclosure, 3) Once these core tasks have been proven to work, the third task will be to migrate all code to the Squawk environment and then run the code on the required hardware (ie. Sunspot nodes). The minimum requirements for each of these tasks are listed below:

<table>
<thead>
<tr>
<th>Data Collection System</th>
<th>Requirement ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC1</td>
<td></td>
<td>Uses a Simulation tool (ie. JSim) to develop the data collection simulation</td>
</tr>
<tr>
<td>DC2</td>
<td></td>
<td>Uses a multi-hop protocol</td>
</tr>
<tr>
<td>DC3</td>
<td></td>
<td>Each node sends its data to one central node called the sink, ie. many to 1 relationship</td>
</tr>
<tr>
<td>DC4</td>
<td></td>
<td>Correctly uses a shortest path algorithm (ie. Dijkstra or Bellman-Ford) to find the shortest path from any node to the sink</td>
</tr>
<tr>
<td>DC5</td>
<td></td>
<td>The routing protocol makes use of clustering</td>
</tr>
<tr>
<td>DC6</td>
<td></td>
<td>Duty-cycles are used for sending data periodically</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Temperature monitor</th>
<th>Requirement ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM1</td>
<td></td>
<td>Each sensor node in the area being monitored is shown at the relative location</td>
</tr>
<tr>
<td>TM2</td>
<td></td>
<td>Visualises the captured values in an intuitive way and in real-time ie. As new values are captured, the visuals are updated</td>
</tr>
<tr>
<td>TM3</td>
<td></td>
<td>Has options to display more than one physical characteristics such as temperature and light intensity simultaneously</td>
</tr>
<tr>
<td>TM4</td>
<td></td>
<td>If the network changes (may be a new node is added or existing one dies), they are reflected on the node visuals</td>
</tr>
<tr>
<td>TM5</td>
<td></td>
<td>All inputs are pre-configured (hard-code all nodes' location)</td>
</tr>
<tr>
<td>TM6</td>
<td></td>
<td>Nodes' send data periodically, ie. every 10 seconds</td>
</tr>
<tr>
<td>TM7</td>
<td></td>
<td>Each node has different colours representing different statuses, e.g. 10-15C: dark blue, 16-25C:</td>
</tr>
</tbody>
</table>
Migrate to Squawk Environment

<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS1</td>
<td>Characterise's the SunSpot radio properties to determine its range, energy usage and packet reception probability</td>
</tr>
<tr>
<td>MS2</td>
<td>Upload's the data collection application and test collection within a single-hop (segment) network</td>
</tr>
<tr>
<td>MS3</td>
<td>Implements a simple multihop network as line of nodes to minimise interference and study's the data collection behavior. Noting only one node generates data at one end and the base-station at the other</td>
</tr>
<tr>
<td>MS4</td>
<td>Tests the full network deployed across a room in a uniform fashion with full data collection application, and routing and clustering protocols.</td>
</tr>
</tbody>
</table>

Other Requirements

<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OT1</td>
<td>No looping occurs in routing protocol</td>
</tr>
<tr>
<td>OT2</td>
<td>Tests which ensure all requirements have been implemented and work correctly before using SunSpot nodes</td>
</tr>
</tbody>
</table>

Target User
The target user for this application is Sun Microsystems. It also important to note that if they choose, they may adopt the software we develop and present it for use by their own clients. Taking this into account, we will still consider Sun as our primary user, and develop the software so it meets their needs and usability requirements.

PART 2: TESTING

TESTING PLAN

1. UNIT TESTING

During development we will be using unit tests to validate each module in our source code and to ensure the system is working correctly. We will do so by making use of test driven development, which essentially involves writing tests before the source code and only writing code to make the tests pass. This technique is apart of extreme programming and has numerous benefits, such as building the system faster, increasing trust in code and catching defects earlier.

In terms of our project, unit tests will be particularly useful as they will help prove to our supervisors that our code works correctly and is ready for migration to the Sun Sensor Nodes. The tests cases will cover all methods of each class, so each class can guarantee correctness according to the tests provided.

To assist us with writing and running these tests effectively, we will be making use of JUnit, a popular testing framework for Java. The following are key reasons and benefits that JUnit provides for testing our code, reduce redundant test code, unit tests can be composed into a hierarchy of test suites and it allows you to write code faster.
while increasing quality.

To ensure old bugs are not re-introduced each time new functionality is implemented, we will be using regression
testing throughout development. To carry this out we need to run every test that has been written whenever new
code has been implemented. This will allow us to detect if correct code that had been written previously is broken
from a later change.

Please note that since we have not begun designing the system, we are unable to provide specific unit tests for
methods of each class.

2. ACCEPTANCE TESTING

To satisfy our client and supervisors our code must perform certain minimal operations i.e. implement certain
requirements. These requirements are clearly listed with identifies in sections one, under each of the three main tasks.
Listed below are specific black box tests or scenarios and when run they will prove that each requirement listed has
been implemented.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Purpose</th>
<th>Assumptions</th>
<th>Method</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send data from a node (X) to the sink from 50 meters away, via the shortest path</td>
<td>Ensure data from node X to sink takes the shortest route based on distance and no looping occurs</td>
<td>The sink node is located at (0,0), not using clustering. Note the same test using clustering would give a different path for the data to travel</td>
<td>Setup 4 nodes along a straight path leading to X, as well as various other nodes that are not used in the shortest path to the sink</td>
<td>Hard code nodes into the following positions, N1 (0,10), N2 (0,20), N3 (0,30), N4 (0,40), X (0,50), n5 (0,100), n6 (50,100), n7 (100,100)</td>
<td>Data takes the following route: X, N4, N3, N2, N1, sink</td>
</tr>
<tr>
<td>View temperature results periodically every 10 seconds</td>
<td>Ensure temperature results can be read and are periodically captured correctly</td>
<td>The nodes are spaced relative to their actual location</td>
<td>Setup 4 nodes around the room and simply run the program</td>
<td>Hard code all nodes into the following positions, N1 (0,10), N2 (0,20), N3 (0,30), N4 (30,30), N5 (60,30), n6 (60,20), n7 (60,10), n8 (30,10). The by going to the temperature results tab, view the data changing periodically</td>
<td>The temperature data changes every 10 seconds based on the temperature in the room</td>
</tr>
<tr>
<td>Remove a node while the simulation is running</td>
<td>Ensure that nodes are removed from display if they die</td>
<td>The node that will be removed is functioning correctly and is connected to the wsn</td>
<td>Setup 5 nodes in the wsn, disconnect one and check if it disappears from the visualisers display</td>
<td>Remove a selected node from the wsn</td>
<td>The selected node disappears from display</td>
</tr>
<tr>
<td>Add a node while the</td>
<td>Ensure that new nodes to the</td>
<td>The node that will be added is</td>
<td>Setup 5 nodes in the wsn, add a</td>
<td>Add a selected node to the wsn</td>
<td>The selected node appears in</td>
</tr>
</tbody>
</table>
Please note that more tests will be added after further research is complete and development has begun. For instances some calculations can not be carried out yet so some tests currently have unknown expected Outputs and have therefore been omitted from the above table. For example calculating the shortest path from a node to the sink using clustering.

EVALUATION PLAN

1. USABILITY TESTING

We plan to use usability testing to evaluate our work. The key point of usability testing is to get the feedback from users during the design/development process to ensure that our application is effective to use and satisfies all proposed requirements. From the users’ point of view, we have decided the following main features of our system should be tested:

- Visualize the temperature profile in an effective way so that the user can keep track of and monitor the system easily.
- Each sensor node is placed correctly at a relative location.
- All captured values are visualized sensibly, and as new values are captured the visuals are updated with a constrained time.
- The changes of each node’s status as well as the changes of the whole network (e.g. a new node is added or an existing node dies) must be reflected visually.
- There are suitable options to display the characteristics and statistics of the network such as in form of a chart or a table.

Nevertheless, only after developing a working visualiser can we create complete and thorough usability tests since more functions might be added during the development process. In addition, asking for frequent feedback from our supervisors would greatly help us to make good choices of features for usability testing.

Also please note that features such as a think aloud experiment will be added in and completed in later reports. This is important because such an experiment will allow us to identify if our Temperature Monitor is useful and usable by those who didn’t actually develop the system.

2. EVALUATION FROM SUPERVISORS

The feedback from supervisors during the design and development process of our system would guide and help us a lot. First, internal supervisors (i.e. Dr. Selvakennedy Selvadurai and Dr. Uwe Roehm) would evaluate our work after each stage including: implementing the simulation, designing and implementing our own protocol, implementing the visualiser, and migrating the designed software to Sun SPOT nodes. Then, after having a complete simulation we will show it to the supervisor from SUN. Her feedback again, will help us to find any problem (if existent) in our software before it is migrated to real sensor nodes.
PART 3: RELATED WORK

1. COMMERCIAL SENSOR NODES

Wireless Sensor Network has emerged as a very hot topic in the networking area recently due to its widespread applications in industry, military, environment as well as real life. There are many commercial Wireless Sensor nodes available nowadays, and some of the most popular ones such as: TMote Sky, MICA Mote, Intel Mote, or Sun SPOT. The first three are described below, and Sun SPOT will be described in the next part.

- **TMote Sky**: TMote Sky [4] is one of the most popular Wireless Sensor Node today. It is the next-generation mote platform for extremely low power, high data rate sensor network applications. It has integrated sensors, radio, antenna, microcontroller and programming capabilities. With 1MB Memory, 10 KB RAM, 48 KB Memory Flash and 250 kbps Radio bandwidth, TMote Sky can offer a robust solution for many wireless sensor applications in industry, environment and so on. TMote Sky supports TinyOS as the operating system for each sensor node and applications can be wirelessly programmed to the TMote Sky module.

- **MICA Mote**: MICA [3] is a second generation mote module used for research and development of low power, wireless sensor networks. It was developed by UC Berkeley’s research group on wireless sensors. MICA Mote has 4KB RAM, 4 Mb Memory Flash and the Radio bandwidth of 40kbps. In comparison with TMote Sky, MICA Mote is less powerful and its memory has less capacity. MICA Mote also uses TinyOS as the operating system.

- **Intel Mote**: Intel Mote [5] was developed by Intel Mote project team and it is an on-going project. Intel Mote hardware is a modular, stackable design that includes 64MB RAM, 512KB Flash and a very high Radio bandwidth of 600kbps. Like TMote Sky and MICA Mote, Intel Mote software is based on TinyOS and in the future it will be supposed to incorporate security features such as authentication and encryption.

2. OPERATING SYSTEM AND PROGRAMMING LANGUAGE

Almost all current commercial WSN nodes operate based on an Operating System (OS) that is specialized for wireless sensor networks. All three commercial sensor nodes MICA Mote, Intel Mote and TMote Sky support the use of TinyOS which has become the premier operating system for wireless sensor networks. TinyOS has been adopted by thousands of developers worldwide, on many platforms. It provides an execution environment which meets unique requirements of WSNs such as resource-constrained, low-power etc. TinyOS is very helpful since it deals with the radio and power management systems for you and makes it easier to write software for the mote. However, using such an operating system may contain overhead and consume quite a significant amount of memory which is precious in each wireless sensor node.

3. SIMULATOR

There are a number of simulators that have been used to simulate wireless sensor networks. However, not many of them are suitable for doing this project that is under a Java environment to monitor the temperature of surrounding environment. There are two simulators: bSPOT and JSim which were suggested by our supervisors. The first one, bSPOT is described below and the chosen one (JSim) will be described in the next Part of this report.

- **bSPOT**: bSPOT [1] is a simulator written entirely in Java to particularly simulate a Wireless Sensor Network of Sun SPOTs. Its strength is the simplicity to understand and use. In addition, bSPOT is strictly Sun SPOT-oriented, thus its protocols and network topologies are very close to the need of our project. However, bSPOT is a newly-created simulator and limited in functionalities. It does not have many physical parameters for a sensor network like signal attenuation, energy consumption or power mode. What is more,
it does not simulate the actual hardware (e.g. batteries and antennas). In addition, packet loss and transmission error are just random, not like a real scenario. Thus, bSPOT does not seem to fulfill the requirements of our project well.

PART 4: SUMMARY OF TOOLS EXPLORED

Compare sensor nodes' hardware

<table>
<thead>
<tr>
<th></th>
<th>MICA Mote</th>
<th>Intel Mote</th>
<th>TMote Sky</th>
<th>Sun SPOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>128 KB</td>
<td>N/A</td>
<td>1 MB</td>
<td>N/A</td>
</tr>
<tr>
<td>Ram</td>
<td>4 KB</td>
<td>64 KB</td>
<td>10 KB</td>
<td>512 KB</td>
</tr>
<tr>
<td>Flash</td>
<td>4 Mbit</td>
<td>512 KB</td>
<td>48 KB</td>
<td>4 MB</td>
</tr>
<tr>
<td>OS supported</td>
<td>TinyOS</td>
<td>TinyOS</td>
<td>TinyOS</td>
<td>Java VM (no OS)</td>
</tr>
</tbody>
</table>

1. SENSOR NODES

As we can see from the table, Sun SPOT has much larger Ram and Flash memory capacity than MICA Mote, Intel Mote or TMote Sky has. With the capacity of 512MB, Sun SPOT’s RAM is 8 times more powerful than TMote Sky’s RAM (64KB); meanwhile, Sun SPOT’s Flash Memory (4MB) is 8 times larger than Intel Mote’s Flash Memory (512K). In addition, Sun SPOT is based on a 32-bit ARM CPU and 11 channels 2.4GHz, IEEE 8.15.4 radio chip which are among the latest Wireless Sensor Network technologies.

It is obvious that Sun SPOT nodes are the most powerful commercial WSN nodes at this moment. With large memory on-device, high speed CPU and many other advantages, Sun SPOT has a great ability to perform complex process and control operations; it also radically simplifies the process of developing wireless sensor and transducer applications. Furthermore, Sun SPOT is very promising in solving several technical challenges in the Wireless Sensor Network area such as:

- Current development tools for creating and investigating wireless sensor are difficult to use and unproductive.
- Within tight resource constraints of sensor nodes and affordable cost, effective security mechanisms are really difficult to be implemented.
- Current sensor nodes have limited processing capability, which restricts signal analysis and control processes.
- Unique characteristics of these new small devices present challenges for networking. New protocols and standards must be created for sensor nodes to communicate with each other.

2. SQUAWK VIRTUAL MACHINE AND PROGRAMMING LANGUAGE

Unlike many other commercial WSN nodes, Sun SPOT system features the “Squawk VM”, a small J2ME Virtual Machine (VM) written almost entirely in Java. It provides the ability to run wireless transducer applications “on the metal”, that means directly on the CPU without any underlying OS. There are many advantages when using a VM instead of an operating system in wireless sensor nodes. Firstly, the overhead due to Operating System can be reduced, hence we can make better use of small memory space and the performance can be greatly enhanced. Secondly, with “Squawk VM”, users can gain the flexibility to experiment with implementations that involve low-level services such as network protocols which are often buried inside an OS. Lastly, a VM can protect hardware from bad or malicious code.

Another feature of Sun SPOT system is that it uses Java technology up-level programming. Java is a high-level and
portable programming language (“write once, run everywhere”), hence it is simpler for developers when building new wireless sensor devices using off-the-shelf hardware component. Java also eliminates many low-level tasks of traditional programming languages such as C.

3. SIMULATOR

JSim \[2\] was first developed in 1999 and is a component-based, compositional simulation environment. JSim has been developed entirely in Java that makes it a truly platform-neutral, extensible, and reusable environment. On top of this simulation framework, a generalized packet switched network model was built which includes many network protocols. JSim provides support for sensors and physical phenomena. It also models many aspects of sensor networks such as Sensor Channel, Sensor Node, Target Node, and Sink Node. Battery and Radio are also modeled in JSim that makes the simulation much more realistic.

In general, JSim’s functionalities are relatively enough for fulfilling the requirements of our project. It has been used widely for quite a long time and received many positive feedbacks from users worldwide. In comparison with some other simulators, JSim was considered the most suitable one for our project. However, we also discovered that JSim still has some limitations, they are:
1) JSim’s protocols are complex and contain overhead, while the protocol used in our project is pretty simple.
2) JSim only provides general routing protocols (e.g. many-to-many transmissions) but not data collection mechanism (i.e. many-to-one transmissions), thus we have to replace with our own protocol in the second phase of the project.

However, after evaluating and comparing the pros and cons of the two suggested tools JSim and bSPOT, we decided to choose JSim since it is able to fulfill our project’s requirements much better. For more detail of our evaluation please see our Wiki here: http://wsn.schtuff.com/choosing_a_node_simulator

PART 5: RESOURCES NEEDED

Listed below are the resources that will be needed to successfully complete the project. Thankfully we have acquired all these items listed, with the exception of the SunSpot nodes which we will be given permission to use as soon as we show our software works correctly.

- SunSpots SDK with all needed software and documents (given to us by Dr. Selvakennedy Selvadurai)
- Use of SunSpot Nodes (we will be granted permission when we prove our software works correctly)
- Access to a computer which has the necessary software installed, ie. Java (JDK 5.0), SunSpot SDK, JSim (Both group members have computers at home which have all these software packages installed.)

PART 6: SWOT ANALYSIS

1. STRENGTHS

Having a strong Java programming background would be considered a significant strength, as this is the language that will be used for development. Fortunately all members in the group have completed at least 3 courses which involved Java and as a result have gained a detailed understanding of the language.

Both group members also have basic networking knowledge about network protocols and network programming. We have both completed two courses in network protocols and programming which is a significant strength, since
designing the protocol for a Wireless Sensor Network is far simpler than standard network protocols that have been learnt.

Kim also has some prior knowledge about Wireless Sensor Networks, where she did a small research project using WSN's last year. As a result, the project gave her a strong understanding of basic concepts involved with WSN's and a considerable advantage/strength for our group.

2. WEAKNESSES

Parampal is relatively new to WSN's, which means that he needs to read up alot more than Kim on the topic. But this weakness has been softened, by Kim providing Parampal with web links and e-books of concise material on WSN's.

A common weakness the team shares is lack of previous experiences with JSim. This is the simulation tool which our team chose to work on and will continue to for the next month while completing the data collection system.

Kim also has a lack of experience with some of Java's advanced capabilities, such as Networking or Thread programming skills, even though she is very familiar with Java. This isn't to concerning as she should be able to learn these quickly when needed, due to her vast experience with java.

3. OPPORTUNITIES

For us as students we will gain a great experience, simply by working on the “hot research topic”, Wireless Sensor Network. There has been an increasing need for Wireless Sensor Networks recently, and there are many opportunities of both working and researching in this area, which makes any skills learnt very useful for our careers in IT.

In terms of the project, when higher layers, namely routing, clustering and application layers for Sun Microsystem’s sensor nodes (called SunSPOT) are successfully developed we can make a network of SunSPOTs gain information about the surrounding temperature. After the information from all nodes is gathered, it is analyzed by traditional computers so that a comprehensive picture of a surrounding environment can be created. This picture would be very useful in many applications for example tracking and monitoring the temperature in a farm, in a swimming pool, in a hospital and so on. In addition, our designed software can be developed further so that a sensor network of Sun SPOT nodes can monitor not only the temperature but also some other factors of the environment such as light, humidity or vibration.

4. THREATS

One of the threats comes from the lack of materials related to Sun SPOT. Indeed, Sun SPOT is a new project which was developed by Sun since 2004, thus there have not been many researches working on it. Therefore the main source of Sun SPOT materials is from the CD which was provided by our supervisors.

There are many states in the project, including: Designing the protocol – Implementing the simulation – Developing a visualiser – Migrating all codes to J2ME/Squawk environment – Programming and testing on real SunSPOT equipments, however the constrained time for us doing this project is only 11 weeks (excluding the first 2 weeks dedicated for choosing group and project). Thus time management is really a problem for us. If we want to finish the project on time we must make a great effort, and we also need a good plan and job schedule.

The simulation platform that we are using (JSim) is actually not an ideal simulator for Wireless Sensor Networks. It was originally designed to simulate a general network thus its protocols are complex and contain overhead. To combat this we have to make changes to the simulator's code and replace certain sections with our own protocol.
PART 7: DETAILED PLAN

In addition to the following plan, we have used Microsoft Project to create a detailed grant chart for the project. You will find it attached at the end of the report for reference.

**Task 1:** Choose group and project (1 week) (Kim + Parampal).
**Task 2:** Research and decide simulation tool (1 week)

- 2.1. Research and evaluate suggested simulation tools. (Kim + Parampal)
- 2.2. Make comparison among them (pros/cons) then choose the most suitable one (J-Sim). (Kim + Parampal)

**Task 3:** Research Wireless Sensor Network (WSN) and related background knowledge (1 week)

- 3.1. Study to understand key concepts of WSN including protocols, special features, etc. (Kim + Parampal)
- 3.2. Study and experiment with the chosen simulation tool (J-Sim). (Kim + Parampal)

**Task 4:** Design and implement protocol in simulation (3 weeks)

- 4.1. Choose a protocol available in J-Sim that is as close to our need as possible (Kim + Parampal); then implement the simulation with that protocol with unit testing. (Parampal)
- 4.2. (After having a working simulation) Design our own protocol based on the requirements suggested. (Parampal)
- 4.3. Implement a simulation with that protocol with unit testing. (Kim)
- 4.4. Introduce “clustering” into the protocol and implement it with unit testing. (Kim)

**Task 5:** Develop a visualiser for application data (4 weeks)

- 5.1. Design the story-board of the GUI, i.e. rough design of each possible screen/view. (Parampal)
- 5.2. Design in more detail the functionalities of each GUI component such as buttons, figures, etc. (Parampal)
- 5.3. Implement the GUI with unit testing. (Kim)

**Task 6:** Develop routing protocol further (1 week)

- 6.1. Change the cost between two nodes from constant to actual distance then implement the design in the simulation. (Kim)
- 6.2. Consider changing from “shortest hop” criteria to “least energy” criteria then implement the design in the simulation. (Parampal)

**Task 7:** Test and prove-correctness to supervisors (1 week)

- 7.1. Make a final test of the whole integrated system. (Kim + Parampal)
- 7.2. Prove correctness of the simulation to supervisors. (Kim + Parampal)
**Task 8:** Migrate developed software to Sun SPOT sensor nodes (3 weeks)

- 8.1. Study the hardware/structure of Sun SPOT nodes. (Kim + Parampal)
- 8.2. Test the designed software on Sun SPOT nodes with full data collection application, and routing and clustering protocols. (Kim + Parampal)

**Task 9:** Finish up the project (1 week)

- Check, test and make final changes before the demo.

---

**PART 8: ROLE ALLOCATION**

1. **GROUP ORGANIZATION:**

   Our group has two members: Parampal Pooni and Kim Nguyen and we decided that there is no need for a leader in a group of two. Throughout the project we will be sharing the work very fairly, and as a result, both group members will lead and be responsible for each specific task they have chosen. But for the whole project there will be no fixed leader.

2. **DOCUMENT MANAGEMENT:**

   To manage our documents throughout the project we will be using a Wiki [8] which has already been proven to be a great supporter for our group working.

   The wiki allows us to share duties in tasks that have been assigned to the both of us. A simple example would be this report, the wiki allowed us to work on the same report at the same time without worrying about losing data, and provided us with the ability to review each others work quickly.

   For our project the Wiki contains important information such as:

   a. Links that are crucial to the project.
   b. Reference documents that are useful for the project.
   c. Meeting minutes for each discussion/meeting between two of us as well as between us and our supervisors.
   d. Jobs reminder for each week.
   e. Tasks, reports and ideas raised during the project (e.g. Draft of the planning report, Pending meeting questions etc.).
   f. Any other concerns that we may have.

3. **TESTING:**

   The allocation of testing roles will be done simply as follows:

   - Unit tests during coding will be done by each group member in his/her own part.
   - Integration/Acceptance tests will be done by both group members. (using pair programming)
   - Usability tests will be done by Parampal

4. **CODE QUALITY:**
Each group member will be responsible for the quality of code and tests they write. On a weekly basis or during a major update each member’s code and tests will be reviewed to ensure it meets a high standard and as few bugs get through as possible. In some instances, we will make use of pair programming, which is where one member writes code while the other observes for errors or a better solution, in which case they swap roles. In most instances this will improve code quality by reducing the amount of bugs and design errors that can occur.

To ensure that the source code written by each group member follows the same format in terms of spacing, brackets etc, a refactor function will be used in the eclipse IDE. This will automatically format the code to a standard which we provide. This standard was agreed upon by both group members and we will attempt to follow it throughout development, and hence only use the IDE’s refactors functionalities if needed.

5. CLIENT LIAISON:

All members in the group keep in constant contact with the supervisors and when needed our client from Sun. A weekly meeting is arranged with our supervisors, in which both group members attend and take notes. If further information is required, for example certain requirements, another meeting may be scheduled or an email containing the questions is sent to all parties involved. This ensures that all group members are able to contact the supervisor when needed, for whatever reason. It is also important to note that contact with our client from Sun has been limited, this is due to their limited availability as they are in a different country. But after all major updates we intend on meeting with our client via video link to update them on our progress and this will also give us a chance to ask important questions.

REFERENCES


