Development of Graphical User Interface and Data Acquisition System for Home Healthcare Using Tinyos-2.x Sensor Networks

Baolong Jia

MASTER THESIS WORK 2009
SUBJECT: Embedded System
Development of Graphical User Interface and Data Acquisition System for Home Health Care Using Tinyos-2.x Sensor Networks

Baolong Jia

This thesis work is performed at Jönköping Institute of Technology within the subject area. The work can also be a part of the master’s degree. The authors are responsible for the given opinions, conclusions and results.

Supervisor: Youzhi Xu

Credit points: 30 ECTS Ponits (D-level)

Date: 2009-12-07

Archive number:
Abstract

Wireless sensor network (WSN) is an very important technology that is able to realize a vision of ambient intelligence. One of the application of Wireless Sensor Networks is Wireless E-Care Systems for monitoring the health of senior citizens. The idea is to use simple sensors that distributed in the old man/woman ‘s outfit and home to provide home assistant/ nurse/ healthcare center/ relatives with a degree of ”understanding” of what he is doing, where he is, what is his health and activity state, what help he need, and what is happening around him.

Basically wireless sensor network consist of three parts: WSN, gateway and remote health-care centre. The purpose of this thesis is to develop a Graphical User Interface and Data Acquisition System for Home Health Care Using Tinyos-2.x Sensor Networks. This system is used for remote health-care centre.

This thesis presents design and implementation of a prototype of a system, which includes the following functions, collecting real-time data from tinyos-2.x sensor networks to a database; displaying nodes’ topology and deployment map at home in real time; plotting sampling values versus time in real time and replay time, dissemination parameters or commands to sensor networks.

Results of the experiments show this system is working well and reliable.
Key Words

- Tinyos-2.x
- Wireless sensor network
- Graphical user interface
- Wireless E-health Care system
- Data acquisition system
- NesC language
- IEEE 802.15.4
- ED (Energy detection)
- CCA (Clear Channel Assessment)
- RSSI (Received signal strength indicate)
- LQI (Link Quality Indication)
- Octopus
Acknowledgement

First of all, I would like to thank my supervisor Professor Youzhi Xu, who followed me with his great vision during my entire master study and allowing me to work with him in summer project as well as master thesis.

I would like to thank master program coordinator Alf Johansson for being always helpful and give me a chance to start working on my thesis in this year.

I would like to thank guest researcher Wei Shen. I had a good opportunity of learning so many things about with wireless sensor network from him in the meetings. I would like to thank Dr. DongYang, who gave me many good suggestions to write this report.

I would like to thank all my teachers for teaching me.

I would like to thank to my parents who give me financial support. And all my friends who have help me during this work.
# Table of Contents

Abstract................................................................................................................. 1
Key words.................................................................................................................. 2
Acknowledgement.................................................................................................. 3
List of Figures.......................................................................................................... 6
List of Tables........................................................................................................... 7

1 Introduction ....................................................................................................... 8
   1.1 BACKGROUND.............................................................................................. 8
   1.2 PURPOSE AND AIMS.................................................................................. 8
   1.3 DELIMITS .................................................................................................... 9
   1.4 OUTLINE ..................................................................................................... 9

2 Theoretical background .................................................................................... 10
   2.1 WIRELESS SENSOR NETWORK ................................................................. 10
      2.1.1 Feature of WSN .................................................................................. 10
      2.1.2 WSN Application in E-Health Care system ......................................... 10
      2.1.3 WSN Platform ..................................................................................... 12
   2.2 IEEE 802.15.4 ............................................................................................ 14
      2.2.1 Overview .............................................................................................. 14
      2.2.2 IEEE 802.15.4 Physical Layer ............................................................... 16
      2.2.3 IEEE 802.15.4 MAC Layer .................................................................. 17
   2.3 JAVA COMMUNICATION ARCHITECTURE FOR TINYOS APPLICATIONS .................................................. 21
      2.3.1 Java vs. NesC ......................................................................................... 21
      2.3.2 Java mote communication ..................................................................... 21

3 Design and Implementation ............................................................................. 22
   3.1 DISPLAY NODES’ MAP TOPOLOGY AT HOME AND PLOT SAMPLING VALUES VERSUS TIME IN REAL TIME .................................................................................................................. 23
   3.2 DISPLAY NODES’ DEPLOYMENT MAP AT HOME IN REAL TIME .......... 29
      3.2.1 Design .................................................................................................... 29
      3.2.2 Implementation ...................................................................................... 30
   3.3 SKYPE CALL, SMS AND EMAIL ................................................................ 31
      3.3.1 Design .................................................................................................... 31
      3.3.2 Implementation ...................................................................................... 32
   3.4 DISSEMINATION PARAMETERS OR COMMANDS TO SENSOR NETWORKS .......................................................................................................................... 33
      3.4.1 Mode 1: PC to Gateway by serial communication ................................. 34
      3.4.2 Mode 2: PC to Gateway by network communication ......................... 35
   3.5 DEVELOPING A MIDDLEWARE FOR THE SYSTEM’S INNER AND EXTERNAL COMMUNICATION ................................................................. 37
      3.5.1 Design .................................................................................................... 37
      3.5.2 Implementation ...................................................................................... 38
   3.6 DATA ACQUISITION ..................................................................................... 40
   3.7 DATA REPLAY .............................................................................................. 41

4 Experiment ....................................................................................................... 44
   4.1 OVERVIEW .................................................................................................. 44
   4.2 INSTALLATION ............................................................................................. 44
   4.3 RESULT ......................................................................................................... 45
      4.3.1 Step 1: Receiving data and display data in real time ......................... 45
      4.3.2 Step 2: Data acquisition and replay .................................................. 48
      4.3.3 Step 3: Skype call, SMS, Email working concurrently ...................... 52
      4.3.4 Step 4: Test sending data back to mote ............................................. 53
<table>
<thead>
<tr>
<th>5</th>
<th>Conclusions and Discussions .................................................</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>SUMMARY OF CONTRIBUTION AND RESULT ..................................</td>
<td>54</td>
</tr>
<tr>
<td>5.2</td>
<td>ANSWER TO RESEARCH QUESTIONS ........................................</td>
<td>54</td>
</tr>
<tr>
<td>5.3</td>
<td>LIMITATION AND FUTURE WORK ...........................................</td>
<td>56</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Limitation ........................................................................</td>
<td>56</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Future work .......................................................................</td>
<td>56</td>
</tr>
<tr>
<td>4</td>
<td>References ...........................................................................</td>
<td>57</td>
</tr>
<tr>
<td>6</td>
<td>Appendix ..............................................................................</td>
<td>58</td>
</tr>
<tr>
<td>6.1</td>
<td>SETTINGS AND CONFIGURATION OF VIRTUAL AUDIO CABLE (VAC) FOR AUTOMATIC VOICE CALL</td>
<td>58</td>
</tr>
<tr>
<td>6.2</td>
<td>SETTINGS AND CONFIGURATION OF SKYPE FOR AUTOMATIC VOICE CALL</td>
<td>60</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1: wireless sensor network architecture ................................ 8
Figure 2: E-Care System Architecture ........................................... 11
Figure 3: the picture of one sensor node ...................................... 12
Figure 4: MIBS20 USB interface board ........................................ 12
Figure 5: Architecture of sensor node ......................................... 12
Figure 6: star and peer to peer topology model .............................. 14
Figure 7: tree topology model .................................................. 15
Figure 8: 802.15.4 Architecture .............................................. 15
Figure 9: 2.4GHz modulation diagram ......................................... 16
Figure 10: super frame structure ............................................. 17
Figure 11: General MAC frame format ....................................... 18
Figure 12: beacon frame format ............................................... 18
Figure 13: MAC command frame ............................................. 19
Figure 14: communication to coordinator with beacon enable .............. 19
Figure 15: communication to device with beacon enable .................. 20
Figure 16: communication diagram ......................................... 21
Figure 17: system development methods .................................... 22
Figure 18: data flow structure ............................................... 23
Figure 19: Class Diagram Of Octopus ........................................ 23
Figure 20: node topology on MapPanel ..................................... 24
Figure 21: structure of OctopusSnetMsg .................................... 25
Figure 22: structure of OctopusCollectMsg ................................. 26
Figure 23: Chart Panel with data coming from mote_0 and mote_1 ....... 26
Figure 24: the RequestPanel of GUI ....................................... 28
Figure 25: the LegendPanel of Graphical user interface ................. 28
Figure 26: shows the relationship between NewMapPanel and other classes .... 29
Figure 27: shows the class diagram about relationship among these six classes .... 31
Figure 28: shows the thread implementation of sending message .......... 32
Figure 29: shows the GUI of calling, SMS and Email sending function .. 33
Figure 30: PC sending moteId and Count number to gateway procedure ... 34
Figure 31: the flowchart of server .......................................... 35
Figure 32: the GUI of PC to Mote network communication function ....... 37
Figure 33: class diagram of communication middle-ware .................. 37
Figure 34: the TCP/IP protocol ............................................. 38
Figure 35: class diagram about data acquisition function .................. 40
Figure 36: the flowchart of logger function ................................ 40
Figure 37: a simple class diagram about data replay function .......... 41
Figure 38: GUI of data replay function .................................... 41
Figure 39: the interesting data shows on JTextArea ...................... 42
Figure 40: flow chart of data replay function ............................. 43
Figure 41: Describe nodes in network and relations between nodes ...... 45
Figure 42: shows the sound value in real time .............................. 46
Figure 43: shows four situations of the new network map ................. 46
Figure 44: Receiving packet from stargate ................................. 47
Figure 45: select CSV file .................................................. 49
Figure 46: shows the first 30 data of storefri_aug_21_2009.csv JTextField .... 50
Figure 47: the first 20 data in storefri_aug_21_2009.csv .................. 51
Figure 48: drawing curve by using 30 data of storefri_aug_21_2009.csv .... 51
Figure 49: GUI for Skype call, SMS send and Email Send ............... 52
Figure 50: the graphical user interface of PC to Mote ..................... 53
List of Tables

Table 1: the properties of event-based programming ........................................13
Table 2: comparing three frequency band.......................................................16
Table 3: format of PPDU ............................................................................17
Table 4: the type of MAC command frame......................................................19
Table 5: 15 different of situations .................................................................29
Table 6: the packet format............................................................................39
Table 7: shows data in store1.csv file............................................................48
Table 8: shows the first 27 data from store2.csv file .....................................49
Table 9: statistic delay ..................................................................................52
I Introduction

Wireless E-health Care System is used to monitor the health of people. The monitoring data is transmitted from home to healthcare centre, then the administer (doctor) in health-care centre could analyse the data, store the data, load previous data and see the location of data. In this report I will present design and implementation of Graphical user interface, data acquisition system and how to use computer to control network node in TinyOS-2.x.

Background

This master thesis is part of Wireless sensor networks for E-healthcare at home project, which is professor Youzhi Xu’s research project.

The purpose of Wireless sensor networks for e-healthcare at home project is to integrate the technologies of wireless sensor networks and public communication networks to construct a healthcare system for senior citizens at home without interfering their daily activities[1].

Purpose and aims

![Diagram](image)

Figure 1: wireless sensor network architecture

The main purpose of the thesis to develop a graphical user interface and data acquisition system for remote healthcare centre. This system runs in a computer, which is illustrated in figure 1 (yellow cycle). The research questions created are mention in below:

- How to Collect real-time data from tinyos-2.x sensor networks to a database?
Introduction

- How to Display nodes’ topology and deployment map at home in real time?
- How to Plot sampling values versus time in real time and replay time?
- How to Dissemination parameters or commands to sensor networks?
- How to Develop a middleware for the system’s inner and external communication?

Delimits

I have made a middle-ware, which can be used to receive data from network. Those data are exactly same as the data get from serial port. Due to I only spend five month on this project, So those data only can show in console, means that GUI does not have function to show them.

This system have not include a data decision function to make a decision if the person is healthy or not.

Outline

In this section, I gave a brief introduction of my thesis, which include background, purpose and Delimits.

The second section provides the reader with detailed background theory about wireless sensor network, IEEE 802.15.4 protocol standard and java communications architecture for TinyOS applications.

In the third section, I illustrate project design diagrams and explain how did I implement it.

The fourth section focuses on performance evolution of functions. All the function running results shows in this section.

The fifth section provide reader the contribution and result of the project and my plan for future work in this field.
Theoretical background

This chapter provides the theoretical background of tinyos-2.x wireless sensor network.

TinyOS is a free and open source component-based operating system. This operating system is used especially for wireless sensor networks, and it is written in the nesC programming language[2]. TinOS programs are built out of software components. Components are connected to each other using interfaces. The feature of TinyOS and NesC language include: Event-based programming model and reusable components software design. Comparing with sequential model the Event-based programming model performance 8 times higher, memory lower and power consumption lower.

Due to TinyOS-2.0 is not backwards compatible with TinyOS-1.x, which means that TinyOS-1.x code is impossible complies in TinyOS-2.x.

My job is based on Octopus, which is a visualization and control tool for WSNs. It includes a little embedded program for the mote and a GUI in java for the computer. The features of Octopus is interduced in the second chapter.

Wireless sensor network

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices using sensor to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations.[3][4].

Feature of WSN

- Human not in the communication loop.
- Self configuration loop, self origination.
- Random deployment.
- Topology change.
- Data centric instead of address centric.
- In network signal processing.
- Application specific.
- Limited power.

WSN Application in E-Health Care system

Nowadays, people are awarded that wireless sensor networks are more and more useful in many area. Wireless sensor networks are very powerful to support to a lot of different real world applications [5][6]. For instance, home automation, environment monitoring, production or E-health Care system. This project is about WSN which are using in E-health Care system.
Wireless sensor network is an emerging technology. In this section I will explain why wireless sensor network could be use for home health care system, and its advantage. The figure 2 shows the architecture of E-care system.

The advantages of WSNs using in home healthcare system are shown as follows:

- Each node have several sensors, which can provide different type of information. For instance, sound, temperature and so on.
- The sensor nodes are very small and cheap. So the sensor node can be put on the human body.
- The transmitter and receiver have build on each sensor node, so the emergent signal can be immediately sent to remote healthcare centre.
- Due to the reason that sensor node are able to move to any place, so using wireless communication is very important.
- The sensor node can share the resources in its network.
WSN Platform

**Hardware:**

- MicaZ motes 2400MHz-2483.5MHz
- ATMega 128L Processor
- 802.15.4 radio transceiver

*Figure 3: the picture of one sensor node.*

*Figure 4: MIB520 USB interface board*

The architecture of this sensor node as shown in figure 5.

*Figure 5: Architecture of sensor node*

The CC2420 is a single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver. This chip is designed for low-power and low-voltage wireless applications.

- Low supply voltage 2.1v to 3.6v with integrated voltage regulator.
- Low supply voltage 1.6v to 2.0v with external voltage regulator.

CC2420 includes a digital direct sequence spread spectrum baseband modem providing a spreading gain of 9 dB and an effective data rate of 250 kbps[12].
Operating system

The operating system of wireless sensor network includes: programming model, to execute a protocol stack and energy management.

TinyOS is the first open-source operating system designed for wireless embedded sensor networks. It uses modern and advanced operating system and language techniques and it is commercial available. We can program sensor node with minimum hardware and microcontroller language.

Berkeley have developed two versions of TinyOS until now. Those are TinyOS-1.x and TinyOX-2.x. I decided choose TinyOS-2.x for my project, the reasons are shown as follows:

- Many new designs in TinyOS-2.x make the system much more reliable than TinyOS-1.x.
- At present, the open source community of TinyOS Alliance gives a lot of supports of TinyOS2.x.
- TinyOS1.x was designed based on mica family platform. TinyOS2.x supports more platforms.

TinyOS and its applications are written in nesC programming language. It’s a component-based C dialect. It is similar to C language and it is designed for wireless sensor node.

There are some advantages of TinyOS and nesC:

<1> it is event-based programming model. The advantage of event-based programming model shows in table 1.

<table>
<thead>
<tr>
<th>Performance</th>
<th>8 times ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>1/(30) ↓</td>
</tr>
<tr>
<td>Power consumption</td>
<td>1/12 ↓</td>
</tr>
</tbody>
</table>

*Table 1: the properties of event-based programming.*

<2> It is reusable components software design.
IEEE 802.15.4

IEEE 802.15.4 is a wireless standard, and the main features of IEEE 802.15.4 standard are network flexibility, low cost, low power consumption and low speed communication between devices.

Overview

1. The standard defines two types of network node:
   <1> Full-function device (FFD). It can serve as the coordinator, and it can communicate to any devices. It implements complete protocol set.
   <2> Reduced-function device (RFD). They are simple and cheap device. They can only communicate with FFD’s and can never act as coordinators. It implements reduce protocol set.

2. Network topology
There are two types of topologies: star topology and peer to peer topology as shown in figure 6:

![Figure 6: star and peer to peer topology model](image)

Star topology:

In the star topology, the communication is established between devices and PAN coordinator. The PAN coordinator is mains powered while the devices will most likely be battery powered. After an FFD is activated for the first time, it may establish its own network and become the PAN coordinator. Each start network choose a PAN identifier. Afterwards other devices can join this network. All the star networks are working independently.

Peer to peer topology:

There is also one PAN coordinator in peer-to-peer topology. Comparing with star topology, any device can communicate with any other device as long as they are in range of one another. Peer to Peer topology is suitable for Wireless Sensor Network application, because peer to peer topology can self-originating and self-healing and it allows multiple hops to route messages.
Cluster-tree Topology:

Cluster-tree Topology is used in this Project. The tree topology model as shown in figure 7. The PAN coordinator forms the first cluster by establishing itself as the cluster head with a cluster identifier of zero, choosing an unused PAN identifier, and broadcasting beacon frames to neighboring devices. A candidate device receiving a beacon frame may request to join the network at the CLH. If the PAN coordinator permits the device to join, it will add this new device as a child device in its neighbor list. The newly joined device will add the CLH as its parent in its neighbor list and begin transmitting periodic beacons such that other candidate devices may then join the network at that device. Once application or network requirements are met, the PAN coordinator may instruct a device to become the CLH of a new cluster adjacent to the first one. The advantage of this clustered structure is the increased coverage area at the cost of increased message latency [7].

3. 802.15.4 Architecture

**PHY**: contains the radio frequency transmitter and receiver and low level control mechanism.

**MAC**: provides access to physical channel.

**Upper Layers**: mainly indicate network layer here. It provides network origination, message routing and so on.

The 802.15.4 architecture shows in figure 8.
IEEE 802.15.4 Physical Layer

The physical layer provides the data transmission service. Physical layer manages the physical RF transceiver and performs channel selection and energy and signal management functions. It operates on one of three possible frequency bands as shown in Table 3:

<table>
<thead>
<tr>
<th>Channel</th>
<th>868MHz (EU)</th>
<th>915MHz (USA)</th>
<th>2.4GHz (World)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central frequency</td>
<td>$f_c=863MHz$</td>
<td>$f_c=906+2(k-1)MHz$</td>
<td>$f_c=2045+5(k-11)MHz$</td>
</tr>
<tr>
<td>Channel bandwidth</td>
<td>1MHz</td>
<td>2MHz</td>
<td>5MHz</td>
</tr>
<tr>
<td>Channel Bit rate</td>
<td>20kb/s</td>
<td>40kb/s</td>
<td>250kb/s</td>
</tr>
<tr>
<td>Symbol rate</td>
<td>20kS/s</td>
<td>40kS/s</td>
<td>62.5kS/s</td>
</tr>
<tr>
<td>Chip rate</td>
<td>300chips/s</td>
<td>600chips/s</td>
<td>2Mchips/s</td>
</tr>
</tbody>
</table>

Table 2: comparing three frequency band

Note: 868MHz and 915MHz modulation using BPSK. 2.4 modulation using O-QPSK.

The 2.4GHz modulation diagram of transmitter part shows in figure 9.

![2.4GHz modulation diagram](image)

Figure 9: 2.4GHz modulation diagram.

Other functions:

- Receiver Energy Detection (ED): energy detection within the current channel. It is used for measure which link is reliable, frequency selection and CSMA/CA.
- Received signal strength indicate (RSSI): is a measurement of the power present in a received radio signal. It is used for rout selection, frequency selection and estimation of locations.
- Link Quality Indication (LQI): The LQI measurement is a characteristic of quality of received packet. LQI may be implemented by using packet loss rate or SNR.
- Clear Channel Assessment (CCA): There are three methods can perform CCA. Those are energy above threshold, Carrier sense only and Carrier sense and energy above threshold.
Physical protocol data format (PPDF): The PPDF packet structure is illustrated in figure 7:

<table>
<thead>
<tr>
<th>Preamble</th>
<th>SFD</th>
<th>Frame length</th>
<th>Reserved</th>
<th>PSDU</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bytes</td>
<td>1 byte</td>
<td>7 bits</td>
<td>1 bit</td>
<td>Variable</td>
</tr>
</tbody>
</table>

Table 3: format of PPDU

Preamble and SFD: allows a receiving device to synchronize.
SFD: start of frame delimit.
Frame length and reserved: contains frame length information.
PSDU: physical service data unit from MAC layer.

IEEE 802.15.4 MAC Layer

The MAC layer provides two kinds of services, MAC data service and management service. The medium access control allows the transmission MAC frames through the use of the physical channel. The general functions are shown below:

1. Superframe Structure:

   The format of superframe structure is defined by the coordinator. The superframe structure is shown in figure 10. The superframe is between two network beacons, and it contains 16 slots. Every slot is the same size.

   **Figure 10: Superframe structure**

   Red part: network beacon. Transmitted by PAN coordinator. Contains network information, frame structure, and notification of pending node messages.

   Black part: beacon extension period.

   Pink part: Access by any node using CSMA/CA

   Green part: Guaranteed Time slot.

   Normally superframe has two active portions and an inactive portion. The active portion is divided into two periods, which are contention access period (CAP) and contention free period (CFP). These two periods are shown in figure 8, pink part and green part.

   In CAP period, the time slot is not guaranteed, so any node wants to communicate by using time slot in this period, they need compete. The mechanism is CSMA/CA.

   CFP contains guaranteed time slots (GTSs). CFP at most allocates seven time slots.
In inactive portion, the coordinator cannot interact with its PAN and may enter a low-power mode.

2. **General MAC frame format (Figure 11):**

<table>
<thead>
<tr>
<th>Octets</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0/2</td>
</tr>
<tr>
<td>0/2/8</td>
</tr>
<tr>
<td>0/2</td>
</tr>
<tr>
<td>variable</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Frame control</th>
<th>Sequence number</th>
<th>Destination PAN Identifier</th>
<th>Destination address</th>
<th>Source PAN Identifier</th>
<th>Source address</th>
<th>Frame payload</th>
<th>Frame check sequence</th>
</tr>
</thead>
</table>

**Note:**
Frame type includes: beacon, data ack, command and reserved.
Security enabled: '1' means yes. '0' means no.
Intra PAN: check if personal ID address is included or not.

3. **Beacon frame format:**

The beacon is allocate at the first slot of each super frame. The coordinator can decide whether using super frame structure, by turn on or turn off the beacon transmission. The beacons are use for synchronous devices. The beacon frame structure shows in figure 12.

```
<table>
<thead>
<tr>
<th>Octets</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>4 or 10</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>variable</td>
</tr>
<tr>
<td>variable</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Frame control</th>
<th>Beacon sequence number</th>
<th>Source address information</th>
<th>Superframe specification</th>
<th>OTS fields</th>
<th>Pending address fields</th>
<th>Beacon payload</th>
<th>Frame check sequence</th>
</tr>
</thead>
</table>
```

**Figure 11: General MAC frame format**

**Figure 12: beacon frame format**
4. **MAC command frame:**
There are many kinds of MAC command frame. The types as shown in table 6:

<table>
<thead>
<tr>
<th>Association request</th>
<th>Beacon request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Association response</td>
<td>GTS request</td>
</tr>
<tr>
<td>Disassociation notification</td>
<td>Data request</td>
</tr>
<tr>
<td>PAN ID conflict notification</td>
<td>Coordinator realignment</td>
</tr>
</tbody>
</table>

**Table 4: the type of MAC command frame.**

The MAC command frame format as shown in figure 13.

5. **Data transfer model:**
There are three types of data transfer transactions exist: from a coordinator to a device, from a device to a coordinator and between two peer devices. Here I will describe first two types.

<i>1</i> Data from device to coordinator (beacon enable)

When a device wishes to transfer data to a coordinator in a beacon-enabled coordinator

When the beacon is found, it synchronous to the super frame structure.

At the right time, it transmits its data frame, using slotted CSMA-CA, to the coordinator. There is an optional acknowledgement at the end as shown in Figure 14.

**Figure 14: communication to coordinator with beacon enable.**
<2> Data from coordinator to device (beacon enable)

When a coordinator wishes to transfer data to a device in a beacon-enabled network, it indicates in the network beacon that the data message is pending. The device periodically listens to the network beacon, and if a message is pending, transmits a MAC command requesting this data, using slotted CSMA-CA. The coordinator optionally acknowledges the successful transmission of this packet.

*Figure 15: communication to device with beacon enable.*

The pending data frame is then sent using slotted CSMA-CA. The device acknowledged the successful reception of the data by transmitting an acknowledgement frame. Upon receiving the acknowledgement, the message is removed from the list of pending messages in the beacon as shown in Figure 15.
Java communications architecture for TinyOS applications

Java vs. NesC

As I mentioned in theory background part, NesC is a component-oriented language that runs on embedded system. The software written with NesC is built using an event-driven model and interfaces.

Java is an object oriented language that runs on a computer. Instead of interface, java applications rely on object oriented techniques to hide the implementation by using public, private, and protected to access to methods.

Java mote communication

A mote interacts with a java application through a Serial Forwarder instance. The Serial Forwarder program is used to read packet data from a serial port and forward it over an Internet connection, so that java GUI can be written to communicate with the sensor network over the Internet. The architecture shows in figure 16.

Figure 16: communication diagram
Design and Implementation

I choose system development research method to development a new system for TinyOS-2.x as figure 17. This is a way of testing wireless sensor network theory. Demonstration of system will prove the theory.

Step 2:
- Design system architecture and define system functionality and components and so on. To find the general ideal to solve those problems. Chapter 3.1 shows the details of design architecture.
- Due to my project is making a software system. I need to draw class diagram for each function before implementation.
- I separate the whole system into several parts. Each part corresponding to one or two functions. At last I combine all the functions together.

Step 3:
Making evaluation of this system, which is I have developed. The system evaluation is based on Experiment. Chapter 4 describes the running result of this system. The purpose of evaluation is to prove the validity and reliability of this system.

Data collection
The system receives data from gateway, which is primary data. The data includes temperature, humidity, light, sound, and time. To collect those quantitative data, I chose measurement and observation.

In the testing chapter I will describe how to analysis those data. Basically I will use excel to make statistics chart.
Display nodes’ map topology at home and Plot sampling values versus time in real time

Graphical user interface and Data Acquisition system design is based on the Octopus. Due to Octopus just a general solution, which is means it is not specific for our research, So I need to modify it and add some other functions. First of all I will explain the global communication structure of this system, which is illustrate in figure 18.

![Figure 18: data flow structure](image)

As shown above PC connect with gateway by serial communication and network communication. There are eight components in PC. Each component at least corresponds to a Java Class. The communication protocol between gateway and GUI is also defined as java class. Here I will explain the details about each component and the relationship among the components. The relationship among components we can be seen from class diagram.

First of all, figure 19 shows the class diagram of graphical user interface. The arrow line describe the relationship between two classes.

![Figure 19: basic Class Diagram of my system](image)
All of the classes in this class diagram are created based on Octopus. There are many other classes also developed for this system for some special function. Those classes have not shown in this class diagram. I will explain it in next section.

**OctopusGui Class:**

This is the main class, and program starts to execute from this Class.

All the panels are created and added into a JFrame. The threads are also created and started. This Class has modified by me. It because I need add some more JPanels on this JFrame. For instance, NewRequestPanel, SelectPanel and so on.

**Scout Class:**

This class implements a interface, Runnable. It means that this class is a thread. There are two important methods in this Class, except constructor.

The one is main method is run(), which is called when the thread start to execute. Another method is messageReceived() that is used to receive message from the serial port.

**Mote Class & MoteDatabase Class**

One object of Mote class represents of a mote, according to the packets received from real mote.

A link list is created in Mote Database class. The link list is used to store the instance of mote object.

**MapPanel Class**

This class is used to describe a map of wireless sensor network. The map as shown in figure 20.

![Figure 20: node topology on MapPanel](image)

The red cycle is described as coordinator (gateway).
Those blue cycles are described as common mote.

Basically there are five methods in this class, which are shows in below:

**Paint ():** when the Panel need to repaint, this method will be invoked.
drawCircleMote(): Checking if it is gateway, then decide to chose which color for cycle.

```java
if(mote.isGateway())
    g2.setPaint(Color.red);  // gateway
else
    g2.setPaint(Color.blue); // mote
```

drawMoteText(): this method is used to draw text from a mote.

drawParentRoute(): this method is used to draw the route between a mote and its parents.

drawParentRouteLabel(): display some information about route.

**ReadingRecordListPanel Class:**

This class extend a LinkList, which is a container. It store objects, and one object corresponding to a mote.

**OctopusSentMsg Class:**

This Class is automatically generated by mig. It defined the structure of a message, which is designed for communication between GUI and gateway and from gateway to regular mote. It is used for request from user. the structure of this message shows in figure 21.

```
| targetId   | 16 bits  |
| parameters | 16 bits  |
| request    | 8 bits   |
```

*Figure 21: structure of OctopusSentMsg*

**OctopusCollectMsg Class:**

This class defined the structure of a message, which is designed for the communication from regular mote to the gateway, and from gateway to GUI. The message include motId, count, reading, quality, parented, reply. The message structure shows in figure 22 next page:
Figure 22: structure of OctopusCollectMsg

- **moteId 16 bits**: the message can be transmitted by many motes before it arrives to gateway.
- **Count 16 bits**: It is a kind of sequence number, incremented at each reading of the sensors.
- **Reading 16 bits**: it is the value read from the sensors.
- **Quality and parented 16 bits**: they are the related to the connection of the mote which has emitted the packet, and its parent.
- **Reply 16 bits**: it is used for checking if a packet is a reply or just a reading.

**ChartPanel Class**

This class is used to display the value of a mote in real time. The curve display is shown in figure 23.

Figure 23: Chart Panel with data coming from mote_0 and mote_1.
RequestPanel Class:

This class is used for user to send request to network. We can see that Mote0 have been selected by user from top of figure 24. the check box is used to broadcast request of this mote.

The "Auto" button is corresponding to a time-driven mode. "Query" button is a query-driven mode.

To Sleep button is used to make the mote in sleep state. The "wake up" button is used to wake up the mote.

The user can adjust the Sleep duty cycle, wake up duty cycle, Sampling period and threshold of the mote. It as shown in the middle of figure 24.

The 'Read the sensor' button is used to request a reading from mote.

There are three buttons in below of Read the sensor button, those are Add Selected button, Add All button, and Clear button. The function of them are use to display motes value in chart.

LegendPanel Class

User can use Legend Panel to control the map panel. When a new legend is chosen, the map will automatic update.

In the middle of the Legend Panel in figure 25, there is a slider to choose time out. This time is about how long time the route appear on the map. The code is shown below:

Mote.getTimeSinceLastTimeScreen()<timeout

The CSV database file is created when the checkbox “Log in a file” is checked.
**Figure 24:** The Request Panel of GUI.

**Figure 25:** The Legend Panel of Graphical user interface.
Display nodes’ deployment map at home in real time.

3-D picture with person shows in newMapPanel. This function is based on Hairong Yan’s algorithm (person detection algorithm). There are totally 15 saturation of this apartment, which are shown in table 5.

<table>
<thead>
<tr>
<th>person not in apartment</th>
<th>Person on bed</th>
<th>Person stand in bed room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person is cooking</td>
<td>Person beside Refrigerator</td>
<td>Person beside water tap</td>
</tr>
<tr>
<td>Person beside micro-oven</td>
<td>Person in the centre on kitchen</td>
<td>Person is washing hand</td>
</tr>
<tr>
<td>Person in toilet</td>
<td>Person sit on sofa</td>
<td>Person taking bath</td>
</tr>
<tr>
<td>Person stand in living room</td>
<td>Person in corridor</td>
<td>Person in balcony</td>
</tr>
</tbody>
</table>

*Table 5: 15 different of situations*

Design

In order to shows the person in 3-D picture, I created a new panel, which is called NewMapPanel. This class is similar with MapPanel. The difference between NewMapPanel and MapPanel shows in below:

1. The cycle (mote) have fixed location in NewMapPanel.
2. 3-D picture as background in NewMapPanel, which is able to change.

NewMapPanel is working like MapPanel, it also relate with other classes.

The instance of NewMapPanel in attribute of OctopusGui class, Scout class, and LegendPanel class.

The instance of LegendPanel class, MoteDatabase class, Mote class and RequestPanel class in attribute of NewMapPanel class.

Class diagram of 3-D as shown in figure26.

*Figure 26 shows the relationship between NewMapPanel and other classes.*
Implementation

1. In order to give the fixed location for each mote, I need to defined the location for each mote in Mote class. The program as shown in below:

```java
int[] xpostion = {350,450,400,600,500,700};int[] ypostion = {850,500,200,400,200,400};
this.x = xpostion[moteId];
this.y = ypostion[moteId];
This.x1 = xpostion[moteId];
This.y1 = ypostion[moteId];
```

NewMapPanel class have two methods are used to check moteId, which are drawCircleMote() and drawCircle(). when moteId is gateway then getting the x value and y value. Afterwards draw the red cycle in corresponding location. When moteId is equals 1, then getting the x1 and y1. Afterwards draw the blue cycle in corresponding location. By using this way all the motes will be displayed on this Panel.

2. To make 3-D picture as background of NewMapPanel, I need to do two steps:
Step 1: read picture. The code shows in below:

```java
try{
    BufferedImage image = javax.imageio.ImageIO.read(new File("picture1.jpg"));
    this.image=image;
}
catch(IOException ex){}
```

Step 2: draw image. The code shows in below:

```java
super.paintComponent(g);
g.drawImage(image, 0, 0, this);
```
Skype call, SMS and Email

Calling, message sending, and email sending function have implemented by Saad Mubeen at last year. There are three reasons for me to improve his work.

At first, he implement these three function in TinyOS-1.x, as we know the code cannot direct using in TinyOS-2.x.

Secondly, he did not using thread for each function, which is means that his code working in sequence. Those functions can not working concurrently. We need to turn this program into separate, independently running subtasks.

Thirdly, Users can not type in telephone number. They only can call the fixed telephone number.

Design

- JSkypeSMSTest Class: the function of sending message through skype is implemented in this class.
- JSkypeTest Class: the function of calling through skype is implemented in this class.
- SendEmail Class: the function of sending email is implemented in this class. The java application to send email using google gmail account
- SelectPanel Class: this class extend JPanel, it is added into OctopusGui.
- OctopusGUi Class: this class extend JFrame.

I have draw the basic class diagram, before I implement these functions. SelectPanel class have three instances, and it own can be a instance of OctopusGUI. As we can see from figure 27.

Figure 27 shows the class diagram about relationship among these six classes.
Implementation

1. Make the code suitable for TinOS-2.x:

   - Software requirements:
     - Install Java SE Development Kit (JDK) 6.
     - Download Javamail 1.4 to use SMTPS protocol.
     - Skype.jar include into java package.
     - Package "net" include into java package. This is the original package developed by Lamot and contains java files related to Skype API changes[9].
     - Install skype and Virtual Audio Cable.

   - Setting of Environment variables:
     - There is a simple way to set environment variables. It is shown in handover Document [10].

   - The java code shows in appendix.

2. Using a thread for each function:

   Each independent subtask is corresponding to a thread. Each thread runs by itself and has the CPU to itself, so we can say those threads running concurrently.

   Generally there are two ways to achieve multiple threads. The one is inheriting from thread. Another one is implement runnable interface. Due to the common resource is big, so I chose the second one.

   Implement runnable interface following steps:

   Step 1: Create JSkypeSMSTest class implementing runnable interface.

   Step 2: Making run method in this class, and method type is public

   Step 3: Create a object of JSkypeSMSTest class.

   Step 4: Create a new thread, passing the object as target.

   Step 5: Call start method on the thread.

   Figure 28 gives the example code of these five steps.

```java
public class JSkypeSMSTest implements Runnable,
MessageListenerInterface{
    public void run(){
        // Implementation
    }
}
```

Figure 28 shows the thread implementation of sending message.
3. New GUI for Skype call, SMS and Email

I have set a default telephone number for user. It is as shown in figure 29. User can press CA directly call this number.

```
3. New GUI for Skype call, SMS and Email

I have set a default telephone number for user. It is as shown in figure 29. User can press CA directly call this number.

<table>
<thead>
<tr>
<th>Telephone number</th>
<th>460766838382</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>CA</td>
</tr>
</tbody>
</table>
```

The user can also press the button to call any numbers, which they want. User can press C to remove numbers, which they before typed in.

There are two send button on this panel. The one is used to send message, another one is used to send mail.

**Figure 29 shows the graphical user interface of calling, message sending and Email sending function.**

Telephone number (0-9) buttons:

- After user press one button, for instance, button '1', then this part of code will be executed, which is shows in below:
  ```java
  jtf1.setText(jtf1.getText() + "1"); // jtf1 is a instance of JTextField
  ```

- After user type in the whole telephone number, who need to press 'CA' to call this telephone through Skype. The code shows in below:
  ```java
  String number="Call +"+jtf1.getText(); //get the whole telephone number
  new JSkypeTest(number); //the parameter (number) passing to constructor of JSkypeTest class
  ```

User can clear the jtf1 by press 'C' button. the code shows in below:

```
jtf1.setText("");
```

The layout manager have not used in SelectPanel class. The position of components are seted by setBounds() method.

**Dissemination parameters or commands to sensor networks.**

To control the mote by user, PC need send data back to mote. There are many ways to solve this problem. For examples, by serial communication, by network communication (TCP/IP protocol) or by GSM and so on. I have implemented two modes, the one is by serial communication and another one is network communication.
Mode 1: PC to gateway by serial communication

**Design:** I have draw the flowchart of this task before programming. The flowchart as shown in figure 30.

![Flowchart](image)

*Figure 30: PC sending moteId and Count number to gateway procedure*
Implementation:
1. The implementation flow the procedure of flowchart.
2. Create a new JButton, which is called 'send', then add it into NewRequestPanel, afterwards add ActionListen on this button.
3. The rest of code write in actionPerform of this button. getText() method is used to get moteld and count number.
4. My_radio_msg class is automatically generated by mig. This class implements a Java interface to the 'my_radio_msg' message type.
5. Integer.parseInt() method is used to convert String value to integer.
6. MoteIF provides an Java interface for sending messages to a mote through a serial port. Gateway is an object of MoteIF class.

Mode 2: PC to Gateway by network communication
This is a simple function. I have created a simple GUI and a server. The user can input data, and the data can send out by the server.

Design:
A server program typically provides resource to a network of client program. Client program send request to the server program, and server program responds to the client. Figure 31 gives an idea to create a server by using java language.

Start

Step 1
Create a server socket

Step 2
Wait for connection

Step 3
Get input/output stream

Step 4
Process connection

Step 5
Close connection

End

Figure 31: the flowchart of server
Implementation:

Step 1: create a server socket

The server program begins by creating a new ServerSocket object to listen on a specific port. ServerSocket is a java.net class. The example as shown in below:

```java
try {
    serverSocket = new ServerSocket(PORT);//a new ServerSocket
} catch (IOException e) {
    System.out.println("Exception on new ServerSocket constructor: " + e);
} // throws the IOException
```

If the port is already being used, it have to throws the Exception.

Step 2: Wait for connection

When the connection is successful established, then accept connection. execute the code in below:

```java
Socket socket = serverSocket.accept();
```

The accept method waits until a client starts up and requests a connection on the host and port of this server.

Due to the aim of this server can provide data for many client, so we need create a new thread after connection accepted. In this case each thread corresponding to a client.

Step 3: get input/output stream

Step 4: process connection

Step 5: close connection

After this system started running, the user can type in MoteId and count number in GUI (NewRequestPanel) as we can see from figure 32 next page.

```
try {
    PrintWriter a = new PrintWriter(socket.getOutputStream());
    a.write("modeID " + id + "\nchannel number " + channel);
    a.flush();
} catch (IOException ioe) {
    System.out.println("IOException on socket listen: " + ioe);
    ioe.printStackTrace();
}
```

Figure 32: the GUI of PC to Mote network communication function.
Developing a middleware for the system’s inner and external communication

This task is to use Java programs to implement a middleware system to link Stargate (Netbridge) based WSN system and laptop via TCP/IP protocol.

The Stargate NetBridge is an embedded Sensor Network gateway device. Its purpose is to connect Crossbow Sensor Nodes to an existing Ethernet network[11]. It features one wired Ethernet and two USB 2.0 ports. The Stargate NetBridge runs the Linux operating system. This is a fully fledged standard Linux distribution for the ARM architecture. It comes preloaded with Crossbow’s Sensor Network management and data visualization software packages, XServe and MoteExplorer.

First of all we should learn the communication flow char of TCP/IP protocols. To achieve this, a Protocol Sniffer and Analyzer tool is used. The server is started and we log in to it using XServe software. All communication is logged in the tool, yielding the following results.

Design

![class diagram of communication middleware](image)

*Figure 33: class diagram of communication middleware.*

The middleware include a client and a server. The client is used to get data from the server of stargate, then send data to the server of middleware. The server in middleware will send data to client of GUI. In this way addition of multiple clients is easily possible.

MicaZConnectionManager Class: this is the main class of these five class.

To implement this class in have following four steps:

Step 1: Create a client socket.

Step 2: Get input/output stream.
Step 3: process connection.

Step 4: close connection.

MicaZProtocol Class: In order to receiving packet from stargate, we have to have a protocol. This protocol is the rules for communication between stargate and laptop. I have define the communication protocol in this class.

MicaZPacket Class: this class is used to define the packet format.

Implementation

To find the TCP/IP protocol, a protocol sniffer and analyzer tool is used. It as shown in below figure 34.

![Figure 34: the TCP/IP protocol](image)

As we can see, in the first packet with information sent from the server, it sends as data two bytes, 0x54 followed by a 0x21. Afterwards, the Xserver sends to the computer periodically the sensor information, with the following TinyOS 2.x message structure shows in table 6 next page:

**Header:** Destination address, Link source address, Message length, GroupID, Active Message handler type. **Payload:** data.

![Table 6: the packet format](image)
The ability to retrieve information from a sensor node through internet seems very normal nowadays, when getting access to a computer with internet is fairly simple. Additionally, it is important to notice that a friendly display of the information is necessary for users. Finally, the ability to reuse software components to decrease the time to market is a must today to be able to keep up the pace of the technology development.

In order to get correct data from data Stargate, at first gives Position of data in MicaZ packet

```
public static int BYTE_DATA_LENGTH = 4;
public static int BYTE_DATA_LENGTH_LENGTH = 1;
public static int BYTE_length = 5;
public static int BYTE_modeID = 8;
public static int BYTE_modeID_length = 2;
public static int BYTE_sample_count = 10;
public static int BYTE_sample_count_length = 2;
public static int BYTE_reading = 12;
public static int BYTE_reading_length = 2;
public static int BYTE_q = 14;
public static int BYTE_q_length = 2;
```

Secondly, store the data into a container. for instance,

```
Integer[] data = new Integer[5];
data[0] = getLength();
```

In this case the packet length store into the first position of integer array.

The received data from this packet also print on command window, so the data need convert to string first. The example code shows in below:

```
String stringRepresentation, temp;
stringRepresentation =  "\nPacket size: " + data.length + " \nPacket data: ";

for (byte b : data)
{
  if( b >= 0 )
    temp = Integer.toHexString(b);
  else
    temp = Integer.toHexString(256+b);
  stringRepresentation += "[0x" + temp.toUpperCase()+"]";
}

//Add payload data

stringRepresentation +=  "\nmessage_length:" + getLength()
+"\nmodeID:" + getProperty(BYTE_modeID,BYTE_modeID_length)
+"\nsample:" + getProperty(BYTE_sample_count,BYTE_sample_count_length)
+"\nreading:" + getProperty(BYTE_reading,BYTE_reading_length)
+"\nQuality:" + getProperty(BYTE_q,BYTE_q_length);
```
3.6 Data acquisition

Design:

As I have mentioned, each mote class is a representation of a mote, based on the packets received from this mote. Logger class gets data from Mote object. Then the data will store into a Linklist, afterwards write linklist to csv file. The class diagram illustrate in figure 35.

Implementation:

The flowchart of logger function illustrate in figure 36.

At first import com.csvreader.*; // import lib. String filename = "Store " + date.toString() + ".csv"; // create a new file name, the file name consist of Store, current time and csv type. startLogging(filename); // start store data.

record[0] = "Mote Id"; record[1] = "Time in ms"; //store MoteId into first position of linklist. //Store time into second position of linklist.

csvWriter.writeRecord(record); // write into csv.
Design and Implementation

3.7 Data replay

Design:

In order to replay the previous session, at first the system need to read the data from csv file, then the data shows on the GUI, at last the curve replay on a new panel. The class diagram of this function shows in below figure 37.

When the user wants to replay a part of previous session, normally they just care about a certain period of a sensor node data. So the user should be able to choose the csv file, Mode_ID, range of data and so on. The graphical user interface for this function shows in figure 38.

![Class Diagram of Data Replay Function](image)

*Figure 37: A simple class diagram about data replay function.*

![GUI of Data Replay Function](image)

*Figure 38: GUI of data replay function*
Design and Implementation

SelectPanel Class:

The graphical user interface of data replay function implement in this class. Basically there are four methods are used to get the information, which is from user typed in.

<1> Getting file name.

```java
FileChooser fileChooser = new FileChooser();
fileChooser.showOpenDialog(this);
File fileName = fileChooser.getSelectedFile();
name = fileName.getName();
```

The function of this part of code let users find a csv file, which is they want to replay, and register the file name.

<2> Getting Mote_ID

The user type the Mote_ID into text file. The method `s = m.getText();` is used to register it.

<3> Getting data number

The user can choose start load value and end load value. So there are two text fields, the one is used for get start load value, another one is used for get end load value.

<4> Due to system need to replay different type of data, so the user should be able to choose different kinds of scale and unit. I have created three RecordChartPanel classess for replay different type of data.

All these information will send to CsvRead class by passing parameter. The load data from CSV file list in figure 39.

![Data from database](image)

*Figure 39: the interesting data shows on JTextArea*
CsvRead Class:

Figure 40 gives the idea for implement CsvRead class.

1. At first import csv read library.

2. Create a new object of CsvReader class.

```
CsvReader reader = new CsvReader(name);
```

3. Read the header of this file.

```
reader.readHeaders();
```

4. Store information in arraylist, for instance MoteId.

```
mid = reader.get("Mote Id");
list1.add(mid);
```

5. Get the interesting data.

```
for (int i=0;i<list1.size();i++){
    a = (String)list1.get(i); // get moteID
    int i2 = Integer.parseInt(a);//convert to int type
    if (i2==c){ // find concentrate moteID
        count++;
        b = (String)list2.get(i);
        //get the reading data related with this moteId
        b2 = (String)list4.get(i);
    }
}
```

6. Store interesting data into arraylist, for instance, time.

```
list3.add(b);
```

5. Transfer the interesting data to Tarea class. The Tarea will shows the data on TextArea. for instance, `new Tarea(list3);`

6. Transfer the interesting data to Osci class to use these data draw a curve. For instance, `new Osci(list3,list5,1);// list3 store time value, list5 store sound value, 1 indicate RecordChartPanel1.`
Experiment

Overview

This part of report illustrates system running results.

To show the results of whole system, I will use four steps. Each step corresponding to one or two functions.

Step 1: shows the result of receiving data and display data in real time function.

Step 2: shows the result of store data in database and replay previous session function.

Step 3: shows the result Skype call, message send & Email send in concurrently.

Step 4: shows the result of Sending data back to mote in serial communication and network.

Installation

In order to run this system, we have to install the cygwin or Xubuntu and so on. I will describe how to install relative software in this section.

At first install Cygwin or Xubuntu. Cygwin is like Linux environment for windows. It consist of two parts: Part 1: cygwin1.dll dynamic link library it works like Linux API Part 2: a collection of software tools, which provide Linux feel.

After cygwin or Xubuntu successfully installed. The user can use the follow instruction start with this system application.

1. Compile the root and node code:
   
   # (cd mote; Make micaz)

2. Install the code on a root mote with ID = 0:
   
   # for instance, make micaz install.0 mib510,/dev/ttyS0

3. Install the code on some number of micaz motes, giving each mote a distinct id.
   
   # for instance, make micaz reinstall.22 mib510,/dev/ttyS0

4. Connect the root micaz mote to your PC and switch on all motes.

5. Compile and run the java application.

   $ cd java

   $ make

   $ export MOTECOM=serial@/dev/ttyS0:micaz

   $ java Gui

Some options are available only before the compilation process. These options can be chosen in the file "motes/OctopusConfig.h". For instance, DEFAULT_SAMPLING_PERIOD and MINIMU_SAMPLING_PERIOD.
Result

Step 1: Receiving data and display data in real time

<1> topology

Result:

As we can see from figure 41, the red point represent gateway, and the blue point represent regular mote. The routes between motes are described as green line, if the link quality is good.

![Figure 41: Describe nodes in network and relations among nodes](image)

This graph is generated before the packet send. When the mote has not send any packet, the route will disappear. Nodes are organized into a tree topology rooted at the Bs. Every node in the tree has a depth. Each node advertises its depth. When a node receives advertising packets from serveral nodes and each link quality is good enough. It selects the node with smallest depth to be its parent. It sends its samples and forwards data it received to its parent. If the link quality with it parent is bad, it will selects a new parent.

<2> shows the data in real time by using curve.

Result:

This function show the value of read from a sensor node. Figure 42 shows a curve, which is drawn by sound value and time.
Figure 42: shows the sound value in real time

<2> 3-D Distribution of location-Fixed sensor node

Result:
The result of implementation as shown in figure 43.

1. Shows no person stay in this apartment.
2. Shows the person stay in the kitchen and washing vegetables.
3. Shows the person stay in the kitchen and beside micro oven.
4. Shows the person stay in corridor.
The functions of location-fixed sensors are:

- Form a wireless sensor network at home.
- The mobile node work into the sensor network, and it can provide location address.
- Due to each room have at least one sensor, so this system can easily monitor the environment of the whole apartment.
- Easily to find the old people’s activities and health in this apartment.
- The alarm signal can be generated by sensor node, when emergency happens.

In order to let the user easily to distinguish location in this apartment, I add a 3-D picture as the background. The background changing are base on the signal from wireless sensor node.

<3> Receiving data from Stargate by wireless

Result:

```
Packet size: 19
Packet data: [0x0][0xFF][0x0][0x8][0x0][0x0][0x93][0x0][0x0][0x0][0x1][0x67][0x6c][0x0][0x0][0x0][0x8][0x0][0x0][0x0]
message_length:11
time:0
sample:23
reading:106
Quality:0
```

```
Packet size: 19
Packet data: [0x0][0xFF][0x0][0x8][0x0][0x0][0x93][0x0][0x0][0x0][0x1][0x67][0x6c][0x0][0x0][0x0][0x8][0x0][0x0][0x0]
message_length:11
time:0
sample:24
reading:195
Quality:0
```

*Figure 44: Receiving packet form Stargate.*

The first seven bytes in each packet are about destination address, link source address, message length, group ID and active message handler type. The rest of data are payload.

Payload:

```c
nx_am_addr_t moteId; /* Mote id of sending mote. Byte 9 and 10 */
nx_uint16_t count; /* The readings are samples count. Byte 11 and 12 */
nx_uint8_t type;    // byte 13
nx_uint16_t reading; // byte 13 and 14
nx_uint16_t quality;// byte 15 and 16
nx_am_addr_t parentId;// byte 17 and 18
nx_uint8_t reply;// byte 19
```

All these data are used for draw topology of node network.
Step 2: Data acquisition and replay

Result:

<1> Data acquisition

When the system starts running, the user can choose to log in a file to store data into CSV file. It shows in figure 23. The transmission data rate is about 8ms, and this system has run for a week without any problem. It stores millions of data. Table 7 and table 8 show the data have been stored in CSV file.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mote Id</td>
<td>Time in nHops</td>
<td>Reading</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>563</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>563</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>563</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>563</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>563</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>563</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>563</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>594</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>594</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>610</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td>625</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>16</td>
<td>641</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>657</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>672</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
<td>704</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>6</td>
<td>735</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>14</td>
<td>760</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>16</td>
<td>766</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>782</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>5</td>
<td>787</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>5</td>
<td>813</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>11</td>
<td>829</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>5</td>
<td>844</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>6</td>
<td>860</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>6</td>
<td>875</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>5</td>
<td>891</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>6</td>
<td>907</td>
<td>0</td>
</tr>
</tbody>
</table>

*Table 7: shows data in store1.csv file. This table only list the first 30 data from the file.*

Mote ID: 5, 6 micro phone adc output.

Mote ID: 14, 15, 16, received signal strength indicator dBm
Table 8: shows the first 27 data from store2.csv file.

Data replay

At first user select which file they want to replay. The name of each file is used by data and time describe. The graphical user interface shows in figure 45.

![Select CSV file](image1)

![Select CSV file](image2)

Figure 45: select CSV file
After the file have been selected, the user can type in the MoteId to get data from indicate mote(location). Then the user need choose how many data will be replayed, and choose the time scale for x-axis.

At last the user choose the different type of chart for y-axis. The GUI shows in the middle of figure 46.

The left side of figure 46 shows load the time value from file. The right side of figure 46 shows load the reading value from file. Reader can compare those values with figure 47.
Figure 47: the first 20 data in storefri_aug_21_2009.csv

At last use the time value and reading value draw the curve in a new panel, which is shows in figure 48.

Figure 48: drawing the curve by using the first 30 data of storefri_aug_21_2009.csv
Experiment

Step 3: Skype call, SMS, Email working concurrently

Result:

After press 'CA' button, the system will call the telephone by using Skype, meanwhile system can also sending SMS and sending Email. The system have not stop read data from sensor node.

<table>
<thead>
<tr>
<th></th>
<th>Max. Delay (second)</th>
<th>Min. Delay (second)</th>
<th>Average Delay (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice Call</td>
<td>20</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>SMS</td>
<td>29</td>
<td>28</td>
<td>28.4</td>
</tr>
<tr>
<td>Email</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

*Table 9: statistic delay*

The GUI shows in figure 49.

*Figure 49: the graphical user interface for Skype call, message send and Email Send.*

Analysis:

Due to each of these three function corresponding to a thread, so all these three function can run in concurrently. Normally these three function runs when emergency happens. So the data could very important at this time. The system should be a able to report alarm and receive data at the same time.

For instance, the old person have not have breakfast, the system will report a alarm. But system will also read and store the temperature data and sound data of this period.
Step 4: Test sending data back to mote.

<1> PC send data to mote by serial communication

Result:

I have used this system to successfully transmit count values to the network node. The count numbers are described using LEDs on the node. The graphical user interface is shown in figure 50.

![Figure 50: the graphical user interface of PC to Mote.](image)

This function gives a good start for using a computer to control a wireless sensor network.

<2> PC send data to gateway by wireless communication

Result:

I did this test together with Yaoming, who made a smart gateway. The gateway can receive data wirelessly. After the gateway received the data from my computer, the data will appear on the display of the gateway.

Normally, we put the computer in healthcare-care, but the gateway could be placed in the old person's apartment. In this case, the computer cannot send data to the gateway by serial communication, so we need to find another solution. The gateway could be moiled, so it is not good to use wire. Using GSM/GPRS the costs are very high. So using wireless send data back to the gateway is a suitable solution.
Conclusions and Discussions

Summary of contribution and result

In this thesis, a Graphical user interface and Data Acquisition system is developed.

- Collecting real-time data from tinyos-2.x sensor networks to a database.
- Displaying nodes’ topology and deployment map at home in real time.
- Plotting sampling values versus time in real time and replay time?
- Dissemination parameters or commands to sensor networks.
- Developing a middleware for the system’s inner and external communication.
- Voice call, SMS and Email can send out concurrently.
- designed specifically for tinyOS-2.x.
- successfully test with MicaZ.
- IEEE 802.15.4 friendly.

Answer to research questions.

a. How to Collect real-time data from tinyos-2.x sensor networks to a database?

I have create a Logger class, which is used for collect real-time data to a database. The Logger is a class that automatically records data in CSV format in a file. This class uses a library CSV. To log a file the startLogging() method should be called at first. Next the addRecord() method should be called with in parameter the mote used. Some flags are available to specify the fields that will be recorded. This flags are set or unset by the LegendPanel.

b. How to Display nodes’ topology and deployment map at home in real time?

At first I define a class, which is called OctopusCollectedMsg. This class automatically generated by msg, and it define the format of message. This message includes moteld, count, reading, quality, parentId and reply.
Secondly I crate a class, which is call scout. I have made a messageReceived() method in this class. messageReceived method is called when a message is received on the serial port. Then the message is processed. If the mote which has emitted the message is ever in the database, the database is updated, else a new entry is added to the database. If the message is a reply to a previous request, the database is updated with this reply.

At last I created a class, which is called mapPanel. This class is used to draw the circlemote and route. Drawing circlemote depends on Moteld. moteld get from the database. If the Moteld is gateway, will draw by Red colour. If the moteld is regular mote, will draw by blue colour.

In order to draw route first of all get the parentId of this mote from MoteDatabase. Then draw 2D-line between mote and its parent. The colour of the route depends on the quality value. The quality value also get from MoteDatabase.

Deployment:

c. How to Plot sampling values versus time in real time and replay time?

I have create a ChartPanel class. Mainly I did two things in this class. At first, getting current time from computer system for x-axis. Secondly, Getting reading value from MoteDatabase. The reading value is used for y-axis. At last using paint 2-D line method to draw the curve with time value and reading value.

d. How to develop a middleware for the system’s inner and external communication?

This middleware include a TCP/IP server and a TCP/IP client. The client is used to get data from stargate (external communication), and the server is used to send data to the client of graphical user interface (inner communication).

This server can have multiple thread, each thread corresponding to a client GUI.

e. How Dissemination parameters or commands to sensor networks?

There are two ways to achieve this task. The one is sending parameters to gateway then to sensor network by wireless network communication.

Another way is sending data to sensor network by serial communication.
Limitation and future work

Limitation

<1> the first limitation of graphical user interface and data acquisition system is about the session replay function. It only can replay at most 1000 data at a time. Because all the data store into ArrayList, which has limited size.

<2> This system cannot directly get data from gateway, so SerialForwarder program must run to get packet from gateway, then send the packet to this system.

Future work

Time flies, If I have time I will not only try to solve these two limitations, but also try to add the below functions

<1> Compatible with the location algorithm.
<2> A data decision system to make a decision if the person is healthy or not.
<3> Map with gradient.
<4> Battery support.
<5> Label of the Id of a Mote on the graph.
4. References


1,2006

[7] Sinem Coleri Ergen September 10, 2004 ZigBee/IEEE 802.15.4 Summary page 7


[12] Stargate_NetBridge datasheet
Appendix

Settings and Configuration of Virtual Audio Cable (VAC) for Automatic Voice Call

<1> Open “Sound recorder”. This can be done as follows:

<2> Click on “Edit” drop down menu and click “Audio Properties” shows below:
<3> In the “Sound Playback” select the Default device as “Virtual Cable 1” as shown below:

![Image](image)

<4> Finally press “OK”. Virtual Audio Cable has been configured.
Settings and Configuration of Skype for Automatic Voice Call

<1> Click on “Tools” dropdown menu and click “Options” as shown below:

<2> A new window will open with a title “Skype Options”. Select “Audio Settings” and in the “Microphone” dropdown menu, select “Virtual Audio Cable 1”. This can be seen from the following screenshot.

<3> Press “Save”. The Skype is configured for automatic voice call.