2006

Web based monitoring system in wireless sensor networks

Kanishk Panwar

Masters Project Report

Web Based Monitoring System in Wireless Sensor Networks

Kanishk Panwar
<kxp0079@cs.rit.edu>
Computer Science Department
Rochester Institute of Technology

Chair
Prof. Leon Reznik…….

Reader
Dr. Hans-Peter Bischof……..

Observer
Dr. Roxanne Canosa……..

Abstract

In the recent past, Wireless Sensor Technology has come up as an advanced autonomous monitoring technology and promises to be the next research frontier for many enthusiasts. But, due to limited hardware there are many restrictions and the number of applications developed is minimal. Most of the applications developed by software companies are per customer needs that cost a fortune [1]. I have created an application that monitors current readings and that reports anomalies in the sensor network conditions based on previous data collected. This system uses a request-response model over an event driven operating system called TinyOS to provide a user friendly monitoring and anomaly detection interface with flexible testing bed for further experiments. I analyzed the current work in anomaly detection at the application level [2] and develop a better solution to overcome previous shortcomings. Our main goal was to develop an application with more robust algorithm and with enhanced features like monitoring, security and data analysis.

Table of Contents

Introduction

A wireless sensor network (WSN) is a network made of numerous small independent sensor nodes, which consist of a battery, radio, sensors, and a minimal amount of on-board computing power. These nodes do not have a pre-programmed network topology that gives them the flexibility of self-organizing into a network but are low on resources. These nodes are built with power conservation in mind because of lack of resources such as electrical energy etc [13]. Because of this, the production generally cost large amounts. Advances in silicon radio chips, coupled with cleverly crafted routing algorithms and network software are promising to eliminate those wires and their installation and maintenance costs. Figure 1 given below is an example of nodes deployed as a wireless sensor network.

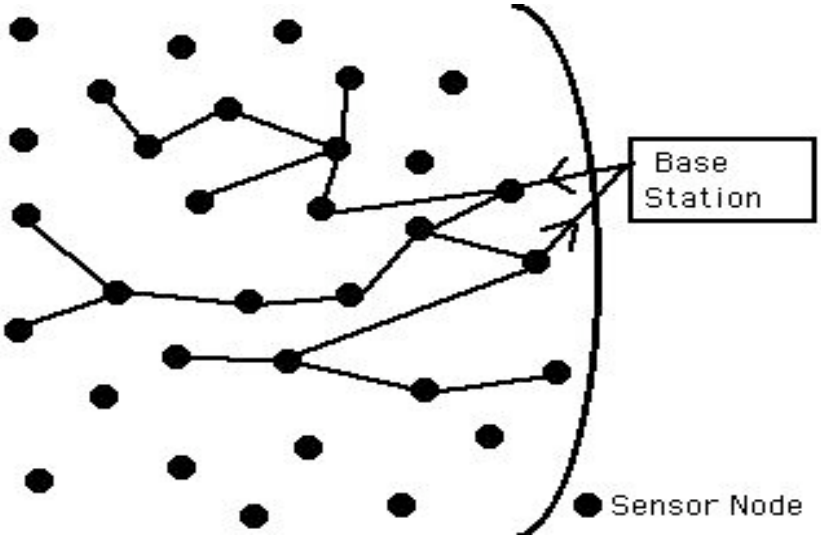


Figure 1

There are numerous nodes/sensors deployed in a field that respond to the message sent by the base station. Typically, the message is broadcasted throughout the network and the node that is supposed to read it and sends a reply message back to the station. If the node is not able to make any direct communication with the station, then the message is routed through the nodes to the base station. These tiny sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks based on collaborative effort of a large number of nodes. These networks can use several different wireless technologies including IEEE 802.11 wireless LANs, Bluetooth and radio frequency identification (RFID). But, right now most of the action is with low-

power radios that have a range of about 30 to 200 feet and data rates of up to around 300K bit/sec [3]. Most of these, with their accompanying network software and APIs, are proprietary products. But, the IEEE last year approved the 802.15.4 low-rate standard for a simple, short-range wireless network whose radio components could run several years on a single battery. The described features ensure a wide range of features in a sensor network that includes military, health and environmental applications. For example, in case of a gas leak, sensors can be deployed from a plane, and as soon as they are released, they form a network and report the level of toxic gas in the air. When they land on the ground, they can read the level of traffic movement on the roads [3, 6].

1 Technologies

1.1 Sensor Nodes

The project has been implemented using Telos Revision B or Telos Sky motes, and there is no functional difference between the 2 motes. Tmote sky has a slight change to the 8-pin JTAG connector on Telos Revision B that allows USB power to be routed to an auxiliary board. The USB power line is useful for systems that use rechargeable batteries and allows these batteries to be charged using the USB power whenever Tmote sky is connected to a PC [3].

Given above is an image of a Telos Rev B mote. Key features of the mote include 250kbps 2.4GHz IEEE 802.15.4 Chipcon Wireless Transceiver, Integrated ADC, DAC, Supply Voltage Supervisor, and DMA Controller. It also has integrated onboard antenna with 50m range indoors / 125m range outdoor. It also has USB based programming and data collection, integrated humidity, temperature, and light sensors, ultra low current

consumption [10]. There is not a complete support for this mote yet from TinyOS community as these do not work properly with TinyDB, a database for TinyOS.

1.2 TinyOS

TinyOS is an open source operating system designed specifically for wireless sensor networks. It features a *component-based architecture* which makes consistent upgrades easy and minimizing the code size [6]. TinyOS was initially developed by the U.C. Berkeley EECS Department, and its component library includes network protocols, distributed services, sensor drivers, and data acquisition tools which can be used to create custom applications. TinyOS in a true sense is an event-driven operating system that enables power management yet allows the scheduling flexibility made necessary by the unpredictable nature of wireless communication.

TinyOS has been ported to various platforms and numerous sensor boards, and programming language is special C that is compiled using a special compiler called NesC. "A wide community uses it to develop and test various algorithms and protocols and over 500 research groups, and companies are using it on various supported platforms such as Linux RedHat 9.0, Windows 2000, and Windows XP" [6]. Various groups are also contributing code to the sourceforge site and working together to establish a standard in the open source development environment. This is the key component for developing this project as the motes itself need TinyOS to run.[6]

1.3 SQL Server 2000

SQL Server 2000 is used in the JDBC layer for storing all the readings from the sensor network and present them on the front end in a user-friendly manner. Microsoft SQL Server is a relational database management system produced by Microsoft. It supports a superset of Structured Query Language SQL, the most common database language. It is commonly used by businesses for small to medium sized databases, and in the past 5 years large enterprise databases have been implemented and it competes with other relational database products for this market segment. [5]

1.4 Java SDK 1.4.2

Java is an open source, cross platform language developed by SUN Microsystems and used to make various kinds of applications. For this project Java 1.4.2 was used as a development platform.

1.5 Apache Tomcat

Apache Tomcat is a servlet container on which the application will be deployed which can be accessed from a web browser. Apache Tomcat is developed in an open and participatory environment and is intended to be a collaboration of the best developers from around the world. Apache Tomcat powers numerous large-scale, mission-critical web applications across a diverse range of industries and organizations [7].

1.6 JFeeChart

JFreeChart is an open source JAVA library that is used to generate complex graphs with various customization options. [14]

1.7 iText

iText is an open source JAVA Library that generates complex PDFs and allows PDF manipulation. [15]

2 Architecture and design

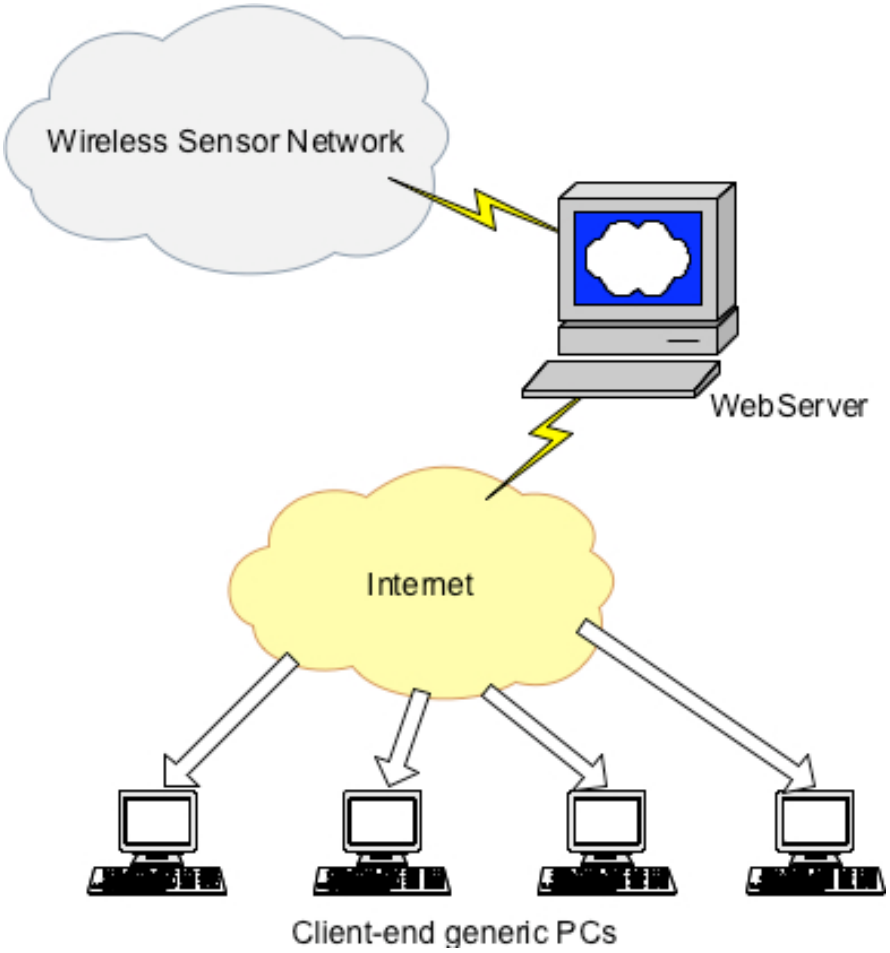
2.1 Requirement

1) This application is developed on top of wireless sensor networks which follows most of the objectives in the software design.
2) The goal of this application is to mobilize the monitoring of an environment by and provide the intrinsic details to the user remotely.
3) This application will not require the end-user to install JVM or TinyOS on their machines.

4) This application is scalable and user friendly.

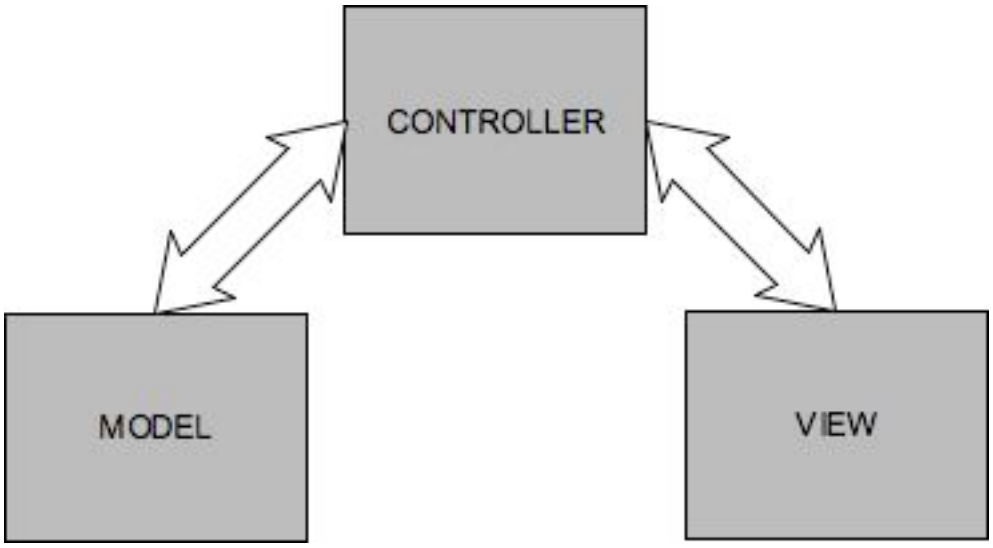
2.2 Architecture

Given below is a top-level design of the architecture.



The Web server has a gateway mote attached to typically a USB port that communicates to all the motes in the field. All the sensors send data to the gateway that further sends it to the application that deciphers the information to yield meaningful data. This web server can be accessed using Internet, and the user can monitor the readings for analysis through a web interface. It can only be accessed by the authorized users and configured by administrator. For the project, the chosen web server was Tomcat Apache and backend database server was SQL Server 2000. Web Interface is written in JSP/Servlets. Here, a single process launched by a servlet on the server sends the query to the sensor network (using serial forwarder) and stores the value in the database. The application is

designed using MVC architecture that eased up the process of adding new features or testing newly implemented algorithms on motes. MVC is a software architecture which separates an application's data model, user interface and control logic into 3 separate components. [7]

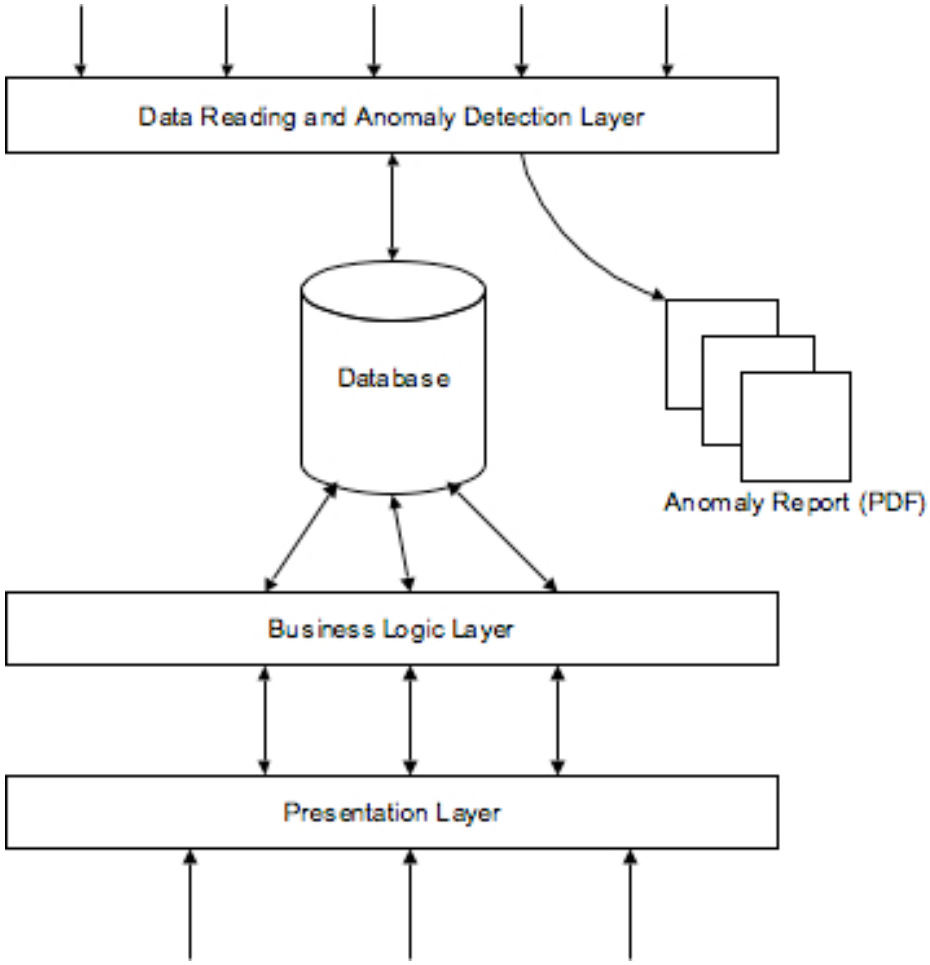


Model: This is the domain-specific representation of the information on which the application operates.[7]

View: This renders the model into a form suitable for interaction, typically a user interface element.[7]

Controller: Typically, this responds to events such as typically user actions, and invokes changes on the model or view as appropriate. [7]

Given below is a low level architecture of the application running on the web server.



Data Reading and Anomaly Layer: This layer is responsible for communicating with the sensors in the field and collecting data from the gateway/base station. This layer also validates if an anomaly has occurred using a robust algorithm and generates a detailed PDF report. If an anomaly has occurred or not, it will be decided by admin controlled thresholds that is configurable.

Database: It stores all the readings from the motes that are used for analysis. It also stores the information about the authorized users. Storing of values allows the admin/user to review the node behavior in the past and help in analyzing the problem in better detail from the web front end.

Business Logic Layer: This layer contains the business logic which generates all the charts and processes all the user requests like queries etc.

Presentation Layer: This layer is responsible for generating the user interface. This includes reviewing the node's statistics and viewing them in graphical aspect such as

frequency graph. Details that are graphically represented are temperature, humidity, light and node power and it also has a settings console where admin can configure the system as per the needs.

3 Implementation Details

This section briefly explains the code implemented for the application.

3.1 Backend Java Classes

3.1.1 DataReadingLayer

This class contains the logic to communicate with TinyOS APIs to read data from the sensors using Oscilloscope Application. This class also acts as validation source for front end content like JSP pages and Servlets. This class implements a subclass called Motes which saves all the information about the motes and the last N number of readings as well. The code takes care of conditions like sensor node failure or rejection of sensor node after X number of continuous anomalies. This class is implemented as a Singleton pattern that allows only one instance of this class to exist at a time. New readings and anomalies are inserted in the database as new Thread instances.

3.1.2 WSNUtil

This class has common methods and variables used by most of the classes. This class also stores the thresholds of different channels and various parameters for JDBC connection. There are various setters and getters implemented for different information like threshold values of the channels and maximum number of readings to be stored for a given channel. A lot of Data in the application needs formatting and they are defined in this class.

3.1.3 DAOConnection

This class is starts a thread that gets an instance of database connection and execute some SQL based methods. Depending on the constructor type called, this class calls appropriate method for data updating. This class calls the method to either store information about the PDF generated for the anomaly or the readings of a channel for a given mote.

3.1.4 JDBCLayer

This class implements Singleton Pattern for database connection. This class has all the methods which return SQL query result sets and insert data into the tables. This class also

3.1.5 Logger

Class implemented for logging information for debugging purpose.

3.1.6 MyDrawingSupplier

This class is implemented as sub class of DefaultDrawingSupplier in JFreeChart API. This class generates eclipse shape as a default for all the readings of different channels.

3.1.7 MyRenderer

This class is implemented as sub class of LineAndShapeRenderer in JFreeChart API. This class provides the feature of painting a reading in the graph as black if it is an anomaly else by default color.

3.1.8 CreatePDF

This class creates a PDF with anomaly details and stores the PDF in a folder where the application resides. This can be used by administrator to analyze the data for further conclusions.

3.1.9 ImageEngine

This class is used to generate a graph of readings for individual motes and average system reading which is displayed on the web pages.

3.2 Servlets

3.2.1 Main

This servlet takes the input from the login page and verifies the credentials of the user. If passed, then the user is directed to secure main page else sent back to login page.

3.2.2 Delegator

This servlet starts the monitoring of the sensor network if not already started and stores the instance of the class in the session of the user. If the monitoring is already running then it is paused. After the execution it redirects the user to main page.

3.2.3 MoteManager

This servlet controls the enabling or disabling of the current mote being monitored. This can be used by administrators to activate the motes that have been invalidated by the application or deactivate the motes which are generating anomalies.

3.2.4 ServletImageGenerator

This Servlets generates the graph of current readings of a given mote or average reading graph of the system and returns the content to the front end as jpeg image. This servlet is called every 5 seconds to update the page with most current readings.

3.2.5 MyReport

This servlet is always launched in the new window and it loads the anomaly report (PDF) saved on the server in the browser window for analysis.

3.2.6 UpdateThreshold

This servlet is called to update the threshold values defined for the different channels in WSNUtil class.

3.3 Front end JSP Pages

3.3.1 Login

This is the login page of the application which takes user input to authenticate him.

3.3.2 Main

This is the main page of the application that is loaded after user authentication. This page displays the username, last time he logged in and current status of the sensor network as one of the contents. The menu on this page has link to anomaly reviews of the whole system or per mote. This page also has a button to start or pause the sensor network monitoring that is only visible to the administrator. If the monitoring is currently active then user can also monitor individual motes. In the center of the page user can see the

average system readings of all the motes as a graph. Most of these features are available in other pages as well.

3.3.3 Anomaly

This page shows the anomaly count status of the system. This page can display the count of total anomalies in the system, system anomalies since last visit, total anomalies for a given mote or anomalies of a mote since last visit. If there are no anomalies then it displays "No records found". If records exist then total number of records is displayed on the page that is linked to AnomalyDetails.jsp .

3.3.4 AnomalyDetails

This page loads up all the anomaly information with a link to their respective reports.

3.3.5 Monitor

This page displays information about the current mote like the readings graph and options of Activating or deactivating a mote if one has admin rights.

3.3.6 AdminConsole

This page displays information about the high and low threshold values defined for all the monitored channels defined in the system.

4 Application Installation and Configuration

4.1 Softwares to be installed

1) TinyOS
2) Cygwin
3) Microsoft SQL Server 2000
4) Microsoft JDBC drivers
5) Tomcat 5.0
6) USB Serial COM Driver

4.2 Libraries to be installed/copied

1) JFreeChart
2) Computer Science Course Library

3) iText

4) JSP library

5) Servlet jars

4.3 Custom configuration

4.3.1 TinyOS

First step is to program the motes with appropriate application using Bash shell of cygwin. Start the Cygwin shell by clicking on the icon as above on the desktop. The installer installs all the tinyOS resource in the /opt/tinyos-1.x/ directory.

Now, plug in the first mote into the USB port and this mote will act as the base mote and will receive packets from other motes in the network. On the bash prompt, execute "cd /opt/tinyos-1.x/apps/TOSBase" and compile the TOSBase program by executing "make telos" on the bash shell

Now change the directory to /opt/tinyos-1.x/contrib/moteiv/apps/Oscilloscope/ and compile the Oscillioscope program by executing "make telos" at the bash prompt. Compiling only needs to be done once and it can be reused for installing on N number of motes.

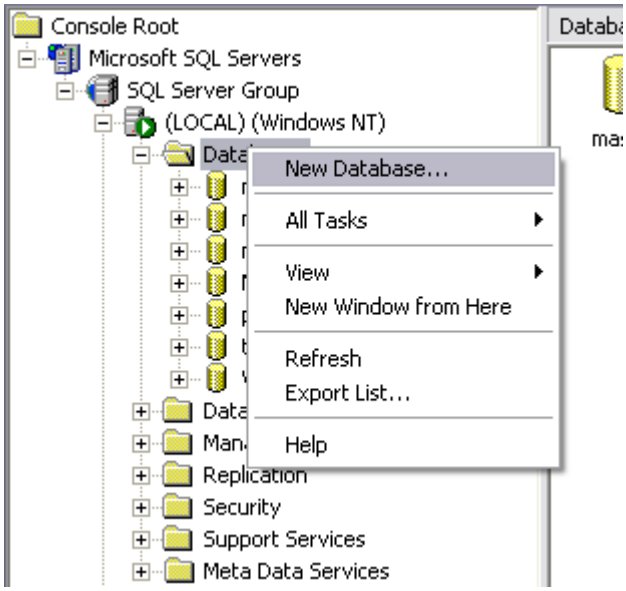
For each of the motes that need to be configured to transmit sensor information like Temperature, Humidity, Light and power readings, plug the mote into the USB port as described above.
Now the binary can be installed on the motes using the command "make telos reinstall,x" on the bash shell where x is the unique network id of the mote that needs to be in the valid range of 1-65535.

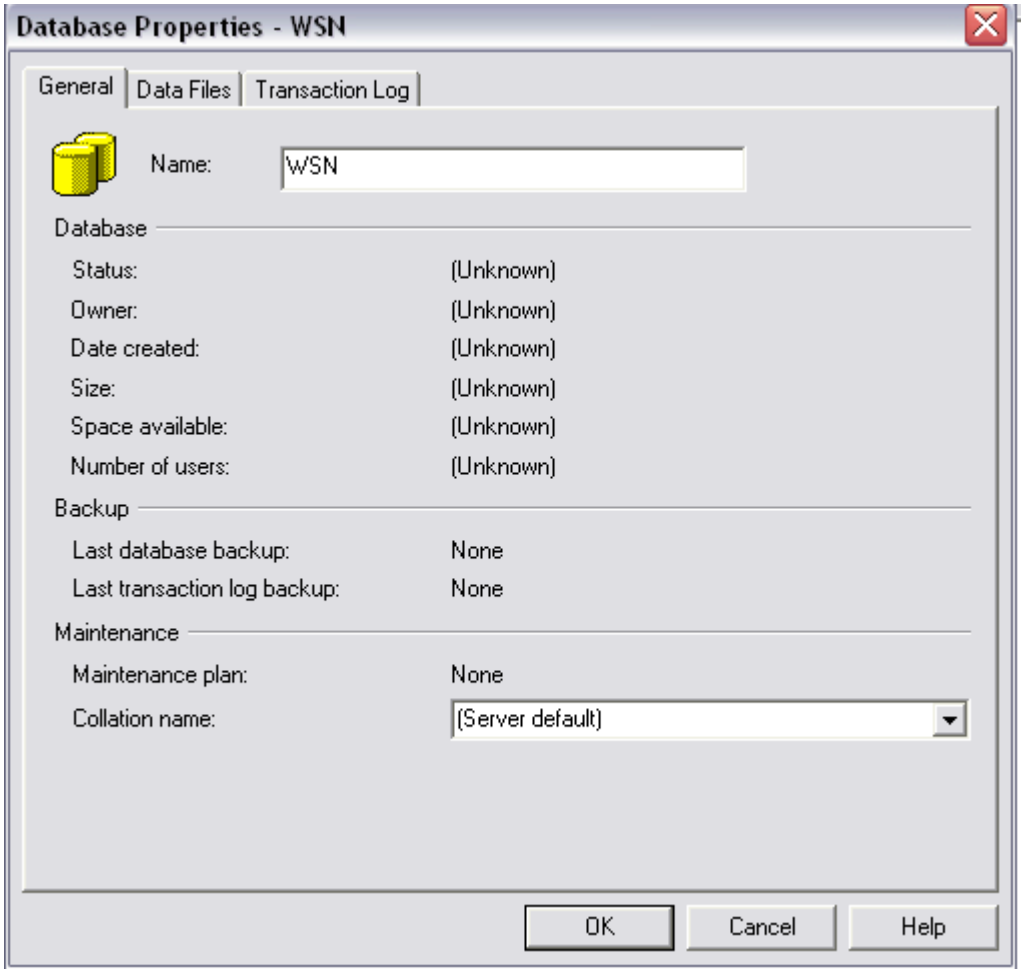
$ make telos reinstall, <mote number>

When the mote has finished programming, remove it from the USB port, install the batteries and it will begin broadcasting readings.

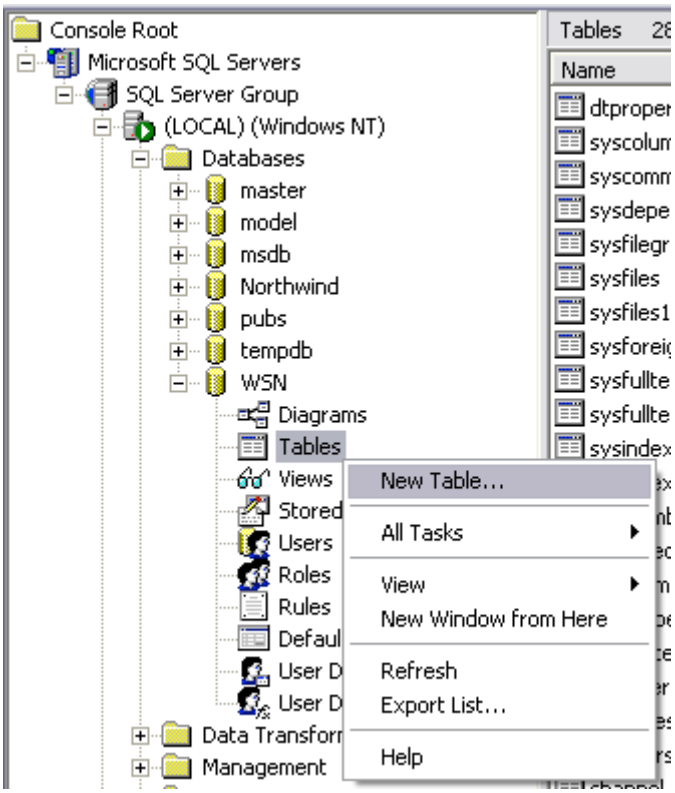
4.3.2 SQL Server

Launch Enterprise manager and create a new Database named WSN as shown below.


Console Root
Microsoft SQL Servers
SQL Server Group
(LOCAL) (Windows NT)
New Database...
All Tasks
View
New Window from Here
Refresh
Export List...
Help
Replication
Security
Support Services
Meta Data Services


Database Properties - WSN
General
Data Files
Transaction Log
Name:
WSN
Database
Status:
(Unknown)
Owner:
(Unknown)
Date created:
(Unknown)
Size:
(Unknown)
Space available:
(Unknown)
Number of users:
(Unknown)
Backup
Last database backup:
None
Last transaction log backup:
None
Maintenance
Maintenance plan:
None
Collation name:
(Server default)
OK
Cancel
Help

Once the database is created, create new table as shown below. Also, create the following tables shown after the image below. Set the user authentication to SQL Authentication.



Tables for the system are as follows

channel_table

Column Name	Data Type	Length	Allow Nulls
id	int	4	
table_name	varchar	50	

This table (above) contains the channel number of readings (Temperature, humidity etc) as the id and the actual names of the channels as table_name. This table is referred when Data Reading Layer updates the database for a given channel number. The query get the table name from this table and then updates the actual table.

motes

Column Name	Data Type	Length	Allow Nulls
mote_id	int	4	

This table (above) stores the id of all the motes that will be monitored by the system. All the other mote readings are rejected by the application.

users

Column Name	Data Type	Length	Allow Nulls
username	varchar	50	
password	varchar	50	
fname	varchar	50	
lname	varchar	50	
is_admin	bit	1	
last_visit	varchar	50	✓

This table (above) stores all the user information which is used in rendering the front end JSP pages and authenticating user access.

reports

Column Name	Data Type	Length	Allow Nulls
mote_id	int	4	
type	varchar	50	
anomaly_date	datetime	8	
file_path	varchar	500	

This table (above) stores the details of the anomaly and stores the path and the filename of the PDF report.

The tables given below store the channel reading information and the reading type is reflected in the table name. Table channel_table is referred to get the name of the table in which data is to be inserted.

humidity

Column Name	Data Type	Length	Allow Nulls
mote_id	int	4	
reading	float	8	
date_time	datetime	8	
anomaly	bit	1	

power

Column Name	Data Type	Length	Allow Nulls
mote_id	int	4	
reading	float	8	
date_time	datetime	8	
anomaly	bit	1	

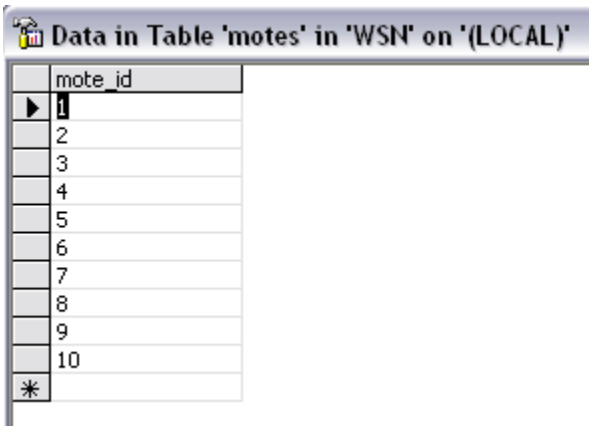
light

	Column Name	Data Type	Length	Allow Nulls
	mote_id	int	4	
	reading	float	8	
	date_time	datetime	8	
	anomaly	bit	1	

temperature

	Column Name	Data Type	Length	Allow Nulls
	mote_id	int	4	
	reading	float	8	
	date_time	datetime	8	
	anomaly	bit	1	

Once the tables are created, launch SQL Server Enterprise Manager and open motes table. Then add the mote ids in each data row one by one. This is needed by the data reading layer to decide which motes are to be monitored and which to be ignored. This adds an extra layer of security in the system. Given below is a screenshot of the Enterprise Manager data insertion window.



4.3.3 Tomcat 5.0

First of all set the environment variable in the system as JAVA_HOME=C:\Program Files\UCB\jdk1.4.1_02\j2sdk1.4.1_02. Then add all the jars being used for the application in the classpath.

Go to the folder where Tomcat is installed (C:\Program Files\Apache Software Foundation\Tomcat 5.0, by default) and change directory to <Tomcat path>\conf\Catalina\localhost.

Create a new file called wsn.xml and copy the content below (in bold), replace

<application path> with actual path where application resides and save.

<Context path="/wsn" docBase="<application path>" crossContext="false" debug="0"/>

Then copy servlet-api.jar, mssqlserver.jar, msutil.jar, msbase.jar and jsp-api.jar to <Tomcat path>\common\lib. Unzip Computer Science Course Library to C:\cscl and switch to <Tomcat path>\bin. Open setclasspath.bat in a text editor and add the following line (in bold) into the file and save.

Set CLASSPATH=%JAVA_HOME%\lib\tools.jar;C:\Program Files\UCB\cygwin\opt\tinyos-1.x\tools\java\;C:\cscl\cscl\lib

4.3.4 Custom Application

Launch the command prompt and go to the folder where TinyOS is installed. Default location of the installation is C:\Program Files\UCB. Now, go to <TinyOS install folder>\cygwin\opt\tinyos-1.x\apps and create a new folder named "WSN". At prompt execute "cd WSN" and create a folder named "WEB-INF".

Copy all the html, jsp, css, jpg files for web pages to WSM folder. At prompt execute "cd WEB-INF" and then create 3 folders called "classes","lib","anomaly_reports". In WEB-INF folder create a new document and paste the content given below (in bold) and save it as web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
   PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
   "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <servlet>
        <servlet-name>Main</servlet-name>
        <servlet-class>wsn.Main</servlet-class>
  </servlet>
  <servlet>
        <servlet-name>Delegator</servlet-name>
        <servlet-class>wsn.Delegator</servlet-class>
```

```
</servlet>
<servlet>
        <servlet-name>ServletImageGenerator</servlet-name>
        <servlet-class>wsn.ServletImageGenerator</servlet-class>
</servlet>
<servlet>
        <servlet-name>MoteManager</servlet-name>
        <servlet-class>wsn.MoteManager</servlet-class>
</servlet>
<servlet>
        <servlet-name>MyReport</servlet-name>
        <servlet-class>wsn.MyReport</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>Main</servlet-name>
        <url-pattern>/Main</url-pattern>
</servlet-mapping>
<servlet-mapping>
        <servlet-name>Delegator</servlet-name>
        <url-pattern>/Delegator</url-pattern>
</servlet-mapping>
        <servlet-mapping>
        <servlet-name>ServletImageGenerator</servlet-name>
        <url-pattern>/ServletImageGenerator</url-pattern>
</servlet-mapping>
        <servlet-mapping>
            <servlet-name>MoteManager</servlet-name>
            <url-pattern>/MoteManager</url-pattern>
</servlet-mapping>
            <servlet-mapping>
            <servlet-name>MyReport</servlet-name>
```

<url-pattern>/MyReport</url-pattern>

</servlet-mapping>

</web-app>

At command prompt execute “cd lib” and copy jfreechart-1.0.1.jar, jcommon-1.0.0.jar, gnujaxp.jar, junit.jar and itext-1.4.jar in that folder. Now, at the command prompt execute “cd ..\classes” and create a folder named “wsn”. At the command prompt execute “cd wsn” and copy all the java files in this folder.

Start cygwin by launching the icon on the desktop and execute “cd /opt/tinyos-1.x/apps/WSN/WEB-INF/classes/wsn” and hen execute “javac *.java”. This action will compile the source in .class and ready to be executed.

5 Application Launching

5.1 Launching SerialForwarder

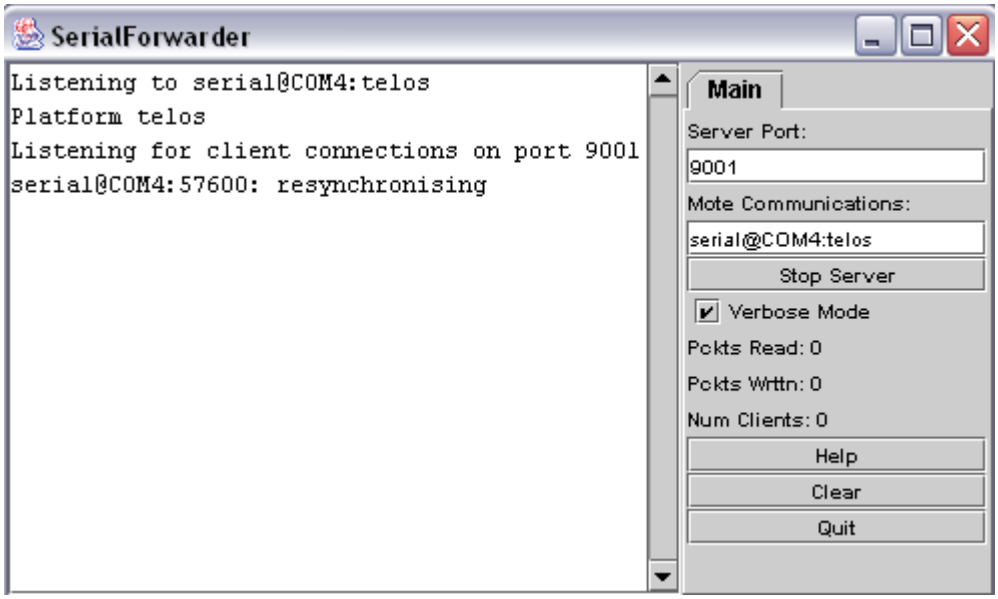
Plug the mote that has the TOSBase program installed into the USB port. This mote will not require batteries as it draws power from the USB port. Open Cygwin and type motelist to display the motes currently connected to the system.

In this way the Tmote acts as a “Tmote module adapter” making a bridge between the PC and Tmote networks. SerialForwarder is a java tool that listens for TinyOS packets on a serial port and forwards them over a local TCP network socket. This allows more than one application to send and receive packets to the attached module. [2]

First plug in a Tmote module to your PC’s USB port. Run the SerialForwarder by entering the following command on the shell

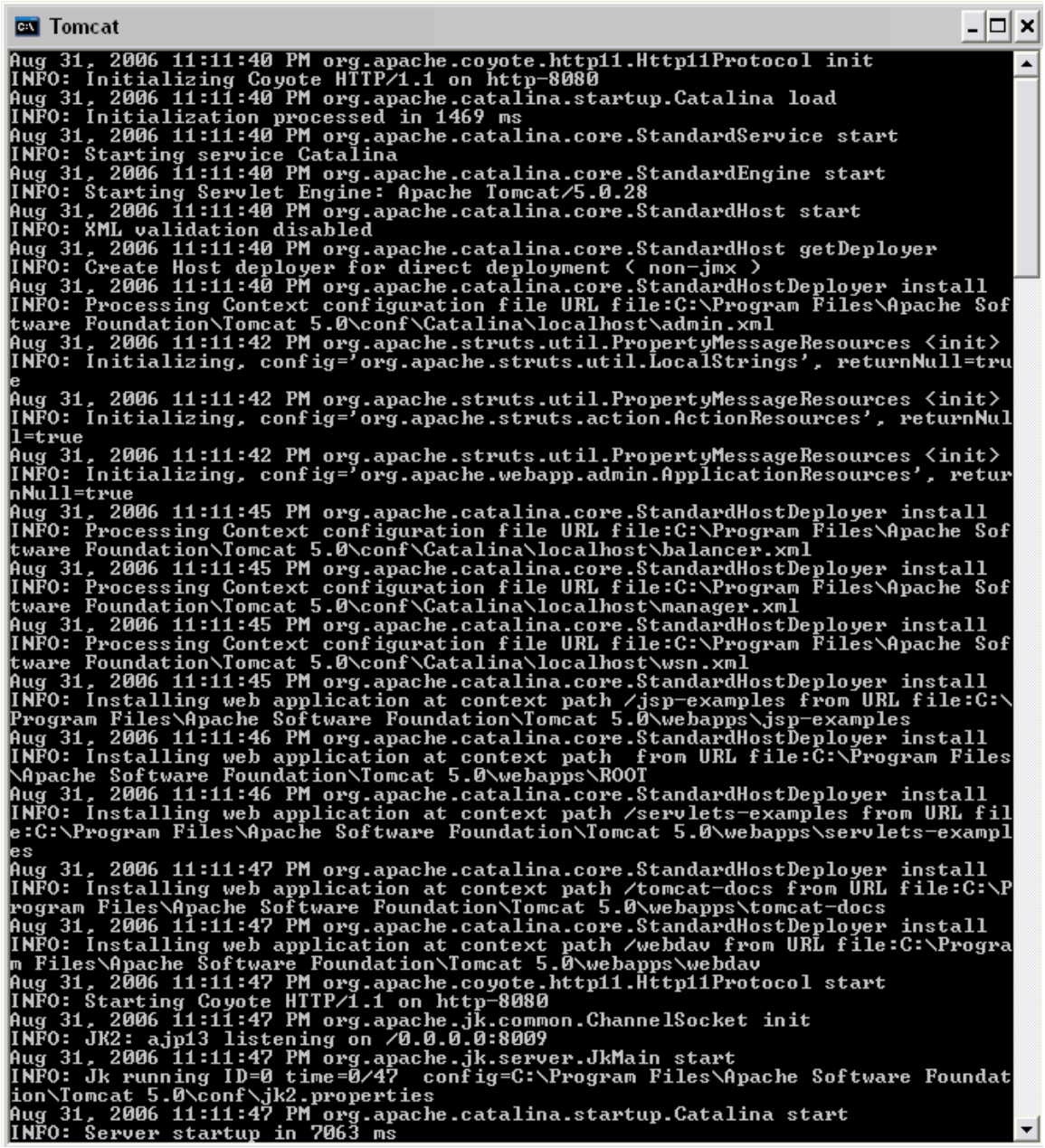
$ java net.tinyos.sf.SerialForwarder –comm serial@COM4:telos

Given below is a screenshot of the SerialForwarder Application



5.2 Launching Tomcat

Launch command line window and go to directory where Tomcat is installed. Execute "cd bin" at the prompt and then execute "startup.bat". This action will launch the tomcat initialization in a new window. Given below is a screen shot of the tomcat initialized and ready to be used.



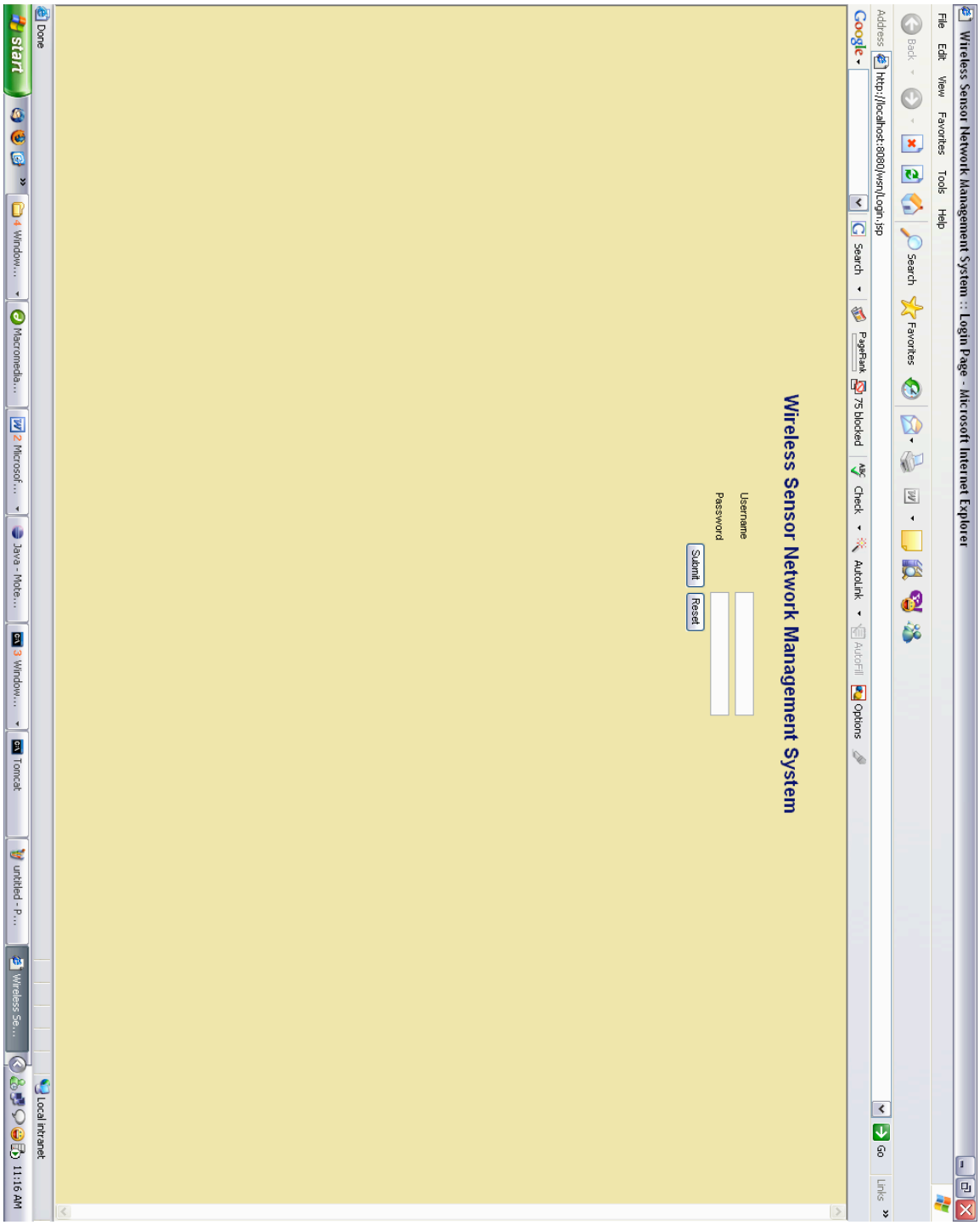
5.3 Launching Web Application

Launch Internet Explorer and type in "http://localhost:8080/wsn/". This will launch the login page of the Application. Given below is the screen shot of the Login Page.

5.3.1 Login Page

Given below is the initial page of the application that is used to login to the secure

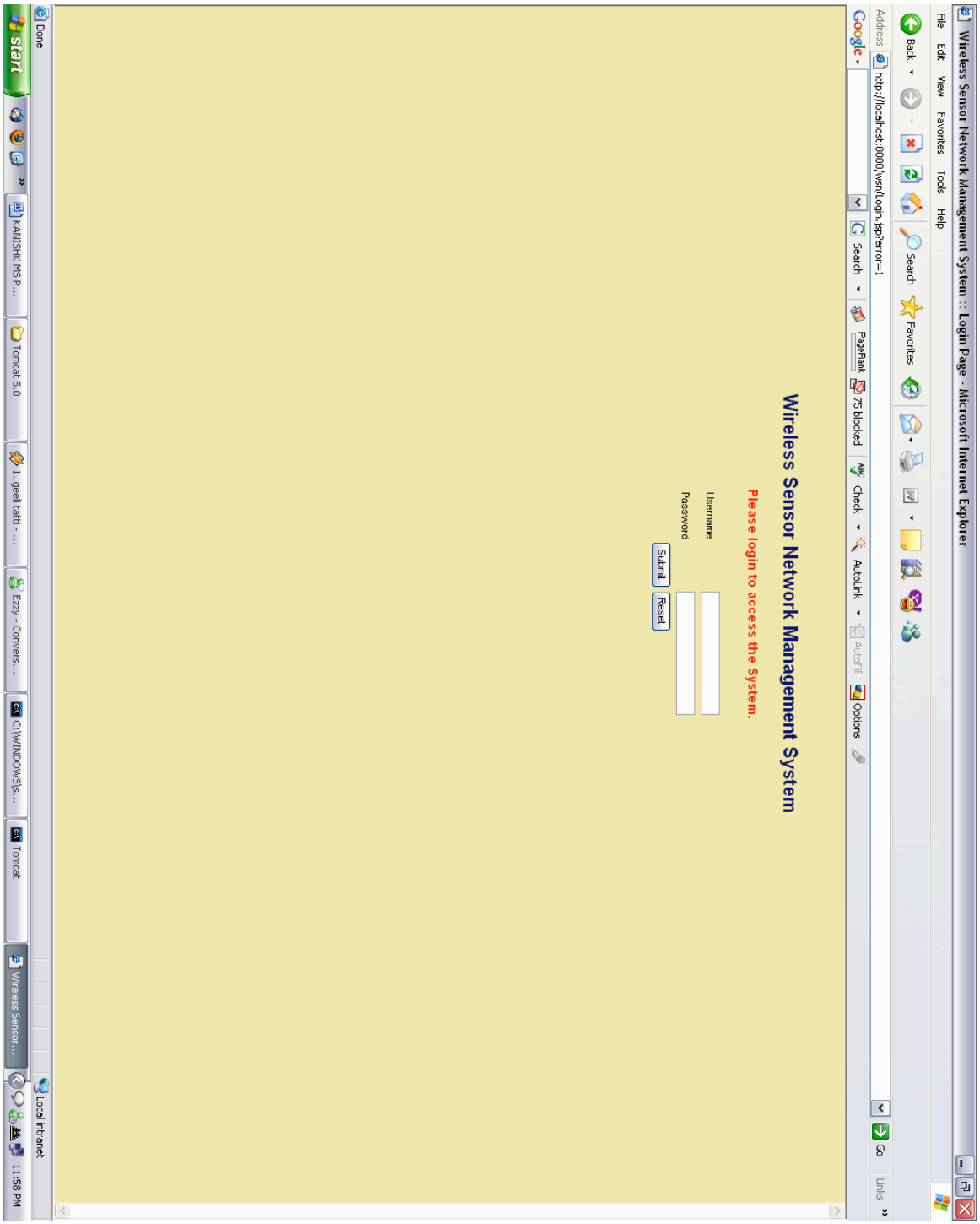
monitoring area.



5.3.2 Login Page with error

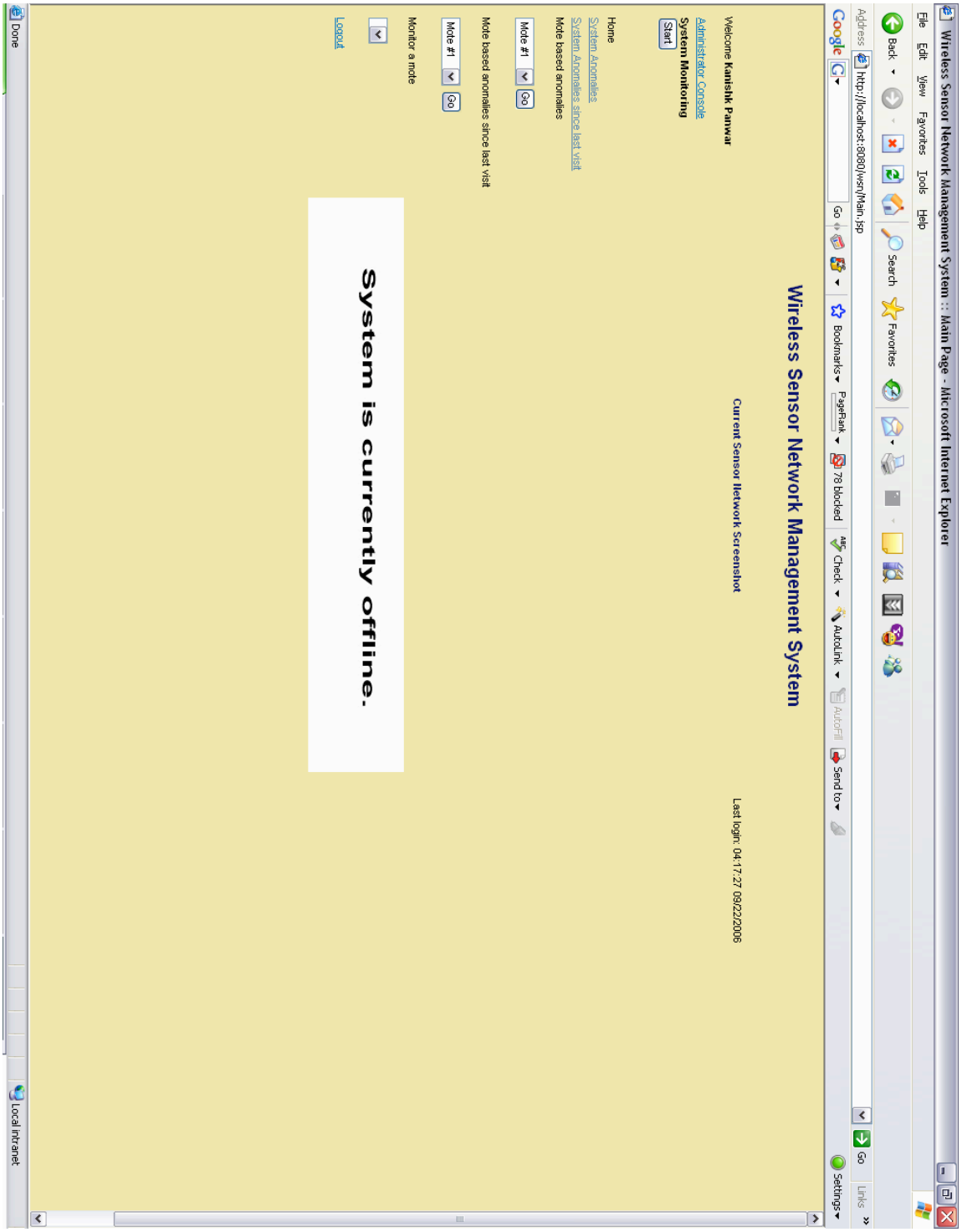
Given below is the error page of the application that is loaded when user credentials are

not correct or when user tries to access the secure area without logging in.

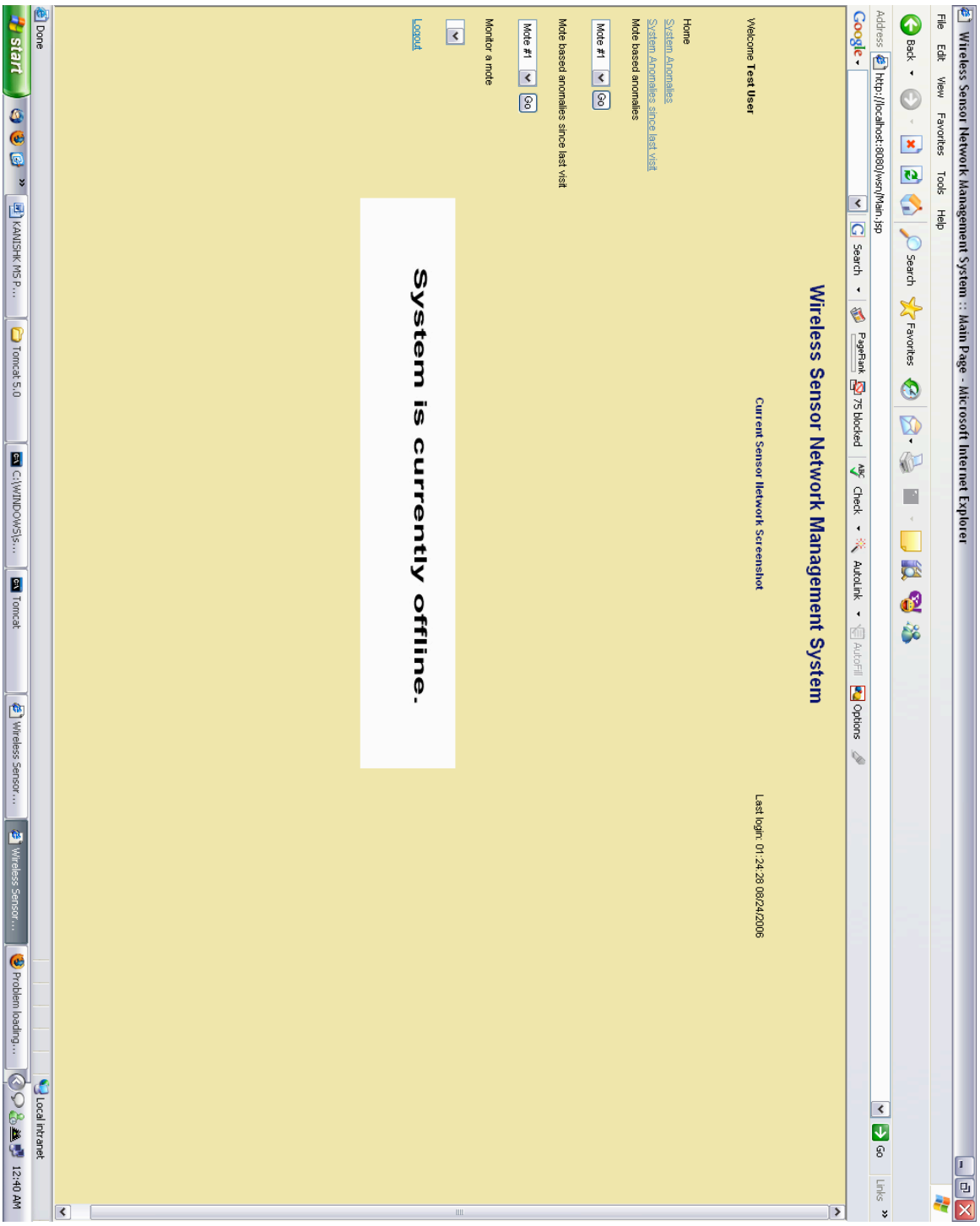


5.3.3 Main Page

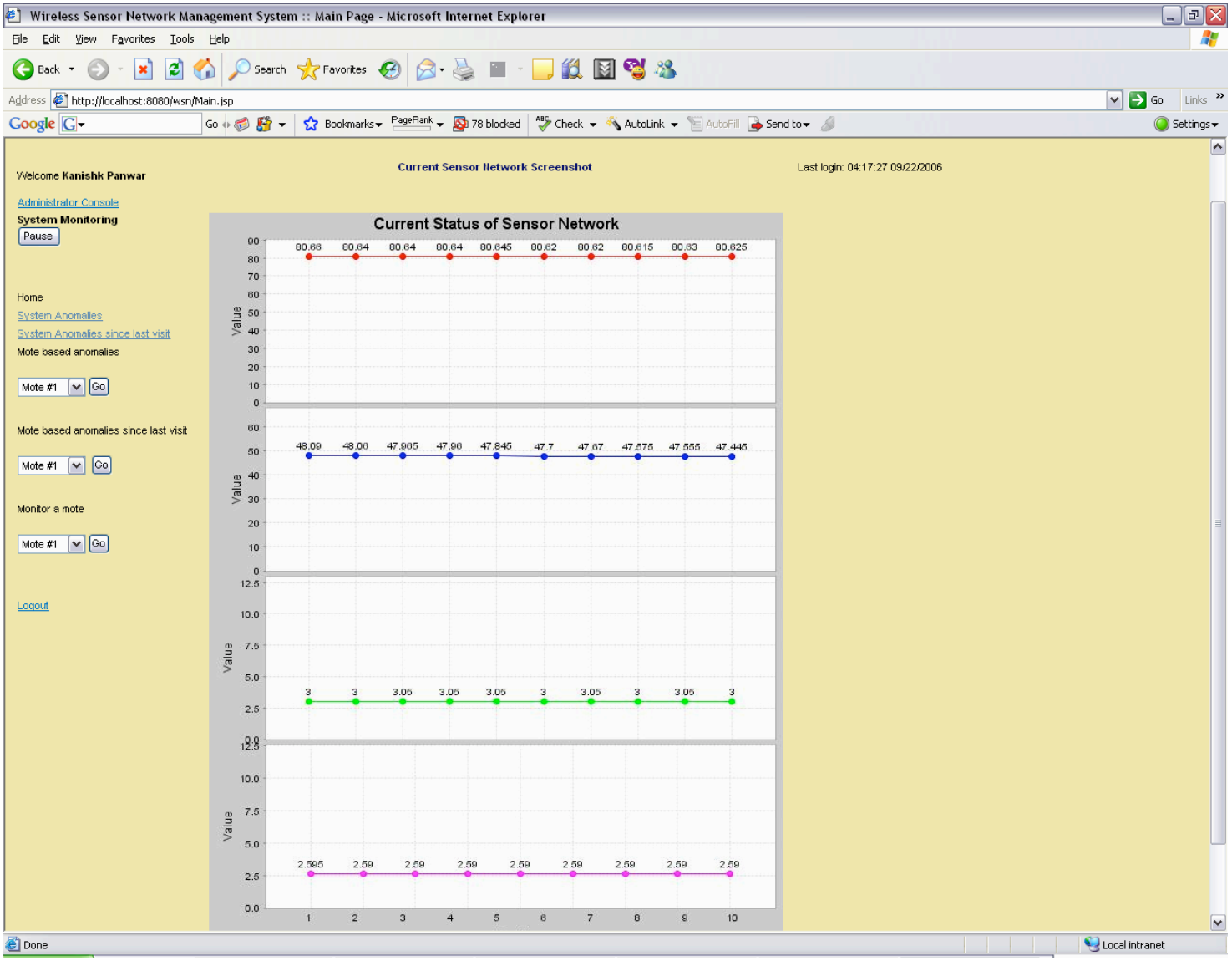
Given below is the screen shot of the main page of the application with admin access. There is a “System Monitoring” button on the left hand side which is only accessible to admins that can be used to start or pause the sensor network monitoring. If the monitoring is not activated then “System is currently offline” is displayed as the message and there is no option of monitoring a mote.



Given below is a screen shot of the main page with User level access.

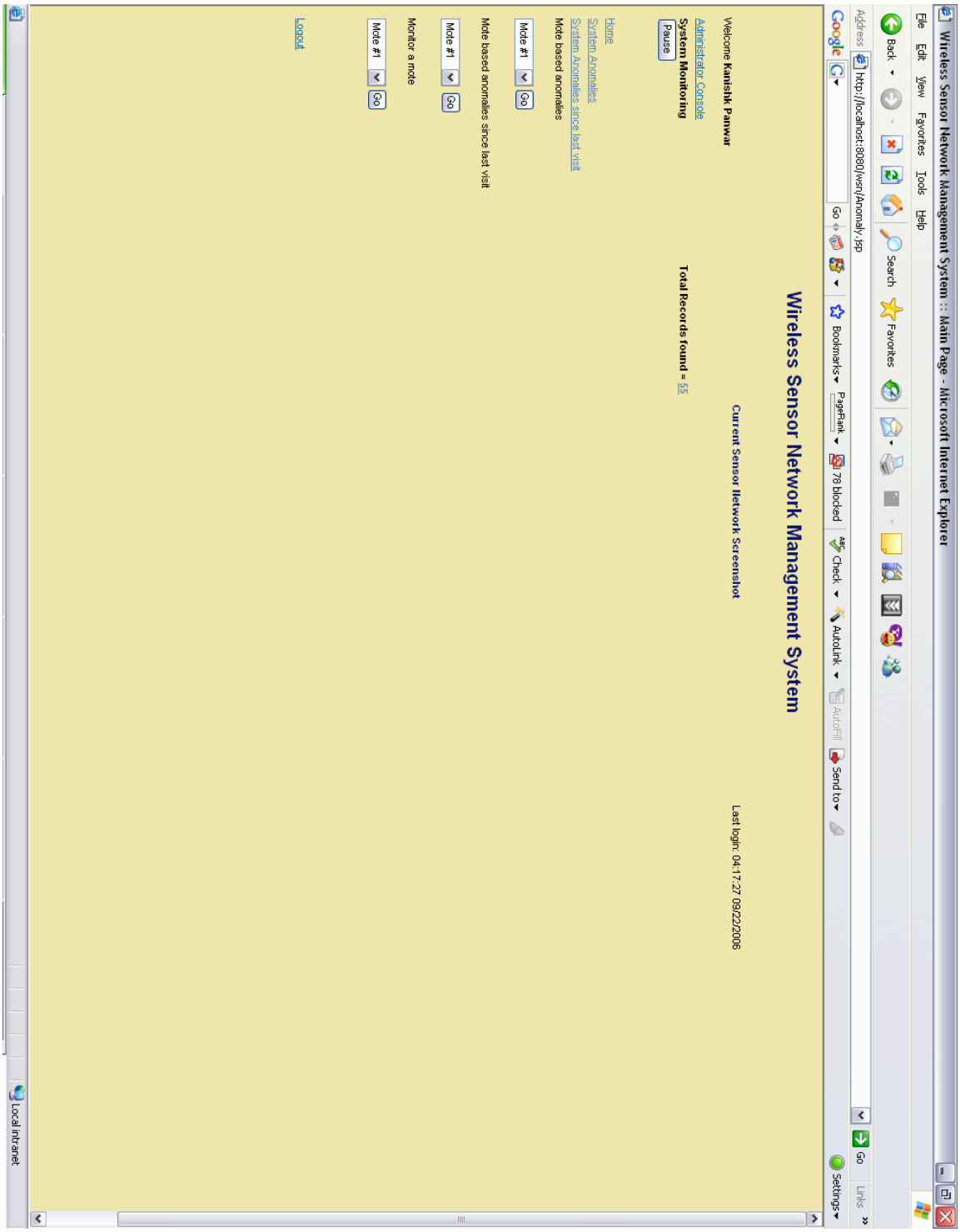


Given below is the screen shot of the Main page that refreshes the readings graph every 5 seconds. The readings are average of readings of all the motes.

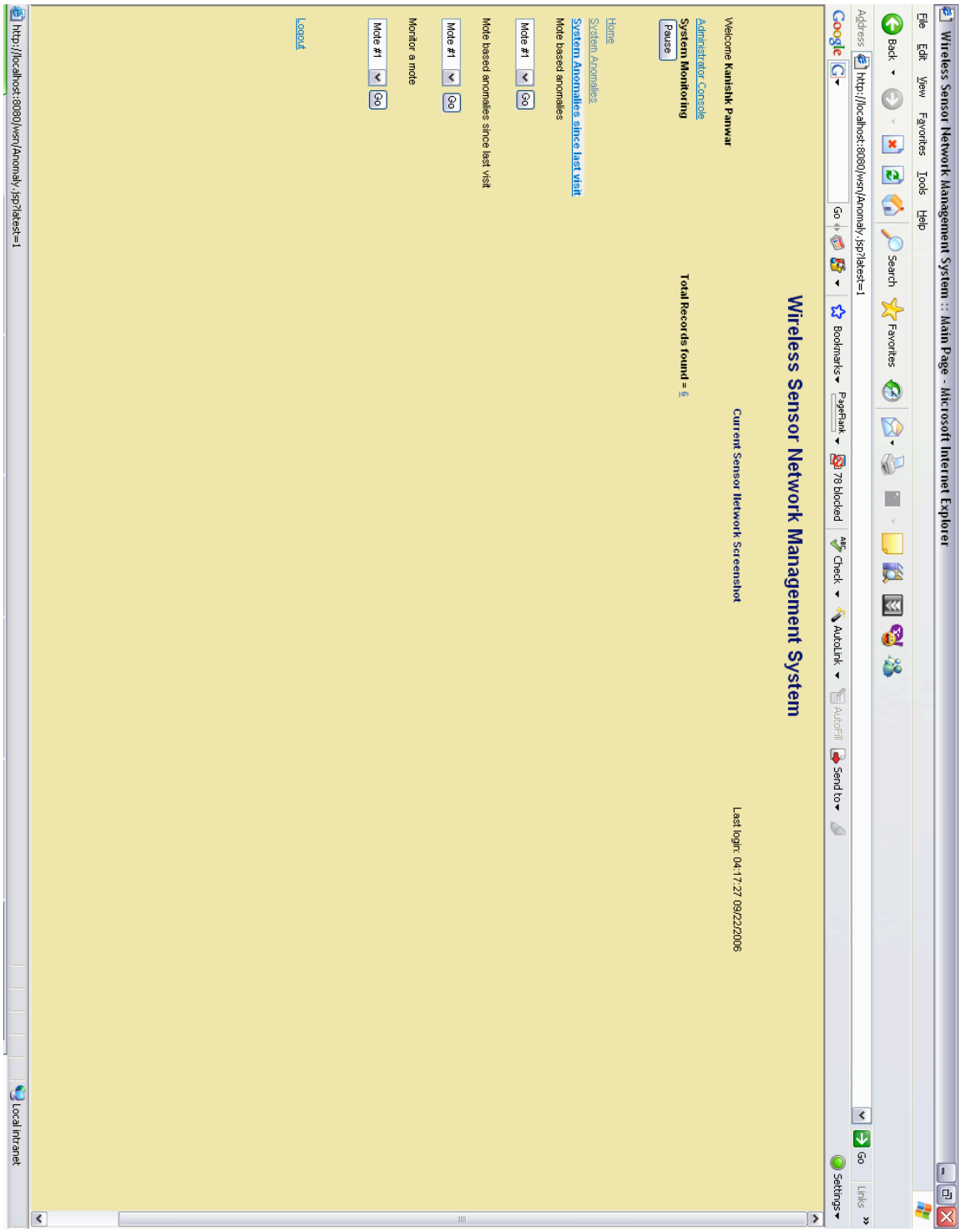

Wireless Sensor Network Management System :: Main Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/wsn/Main.jsp
Go Links
Google Go Bookmarks PageRank 78 blocked Check AutoLink AutoFill Send to Settings
Welcome Kanishk Panwar
Current Sensor Network Screenshot
Last login: 04:17:27 09/22/2006
Administrator Console
System Monitoring
Pause
Home
System Anomalies
System Anomalies since last visit
Mote based anomalies
Mote #1 Go
Mote based anomalies since last visit
Mote #1 Go
Monitor a mote
Mote #1 Go
Logout
Current Status of Sensor Network
Value
80.66 80.64 80.64 80.64 80.645 80.62 80.62 80.615 80.63 80.625
48.09 48.06 47.965 47.96 47.845 47.7 47.67 47.575 47.555 47.445
3 3 3.05 3.05 3.05 3 3.05 3 3.05 3
2.595 2.59 2.59 2.59 2.59 2.59 2.59 2.59 2.59 2.59
1 2 3 4 5 6 7 8 9 10
Done
Local intranet

5.3.4 System Anomalies Page

Given below is the screen shot of the System Anomalies page where total number of anomalies occurred in the system is displayed.

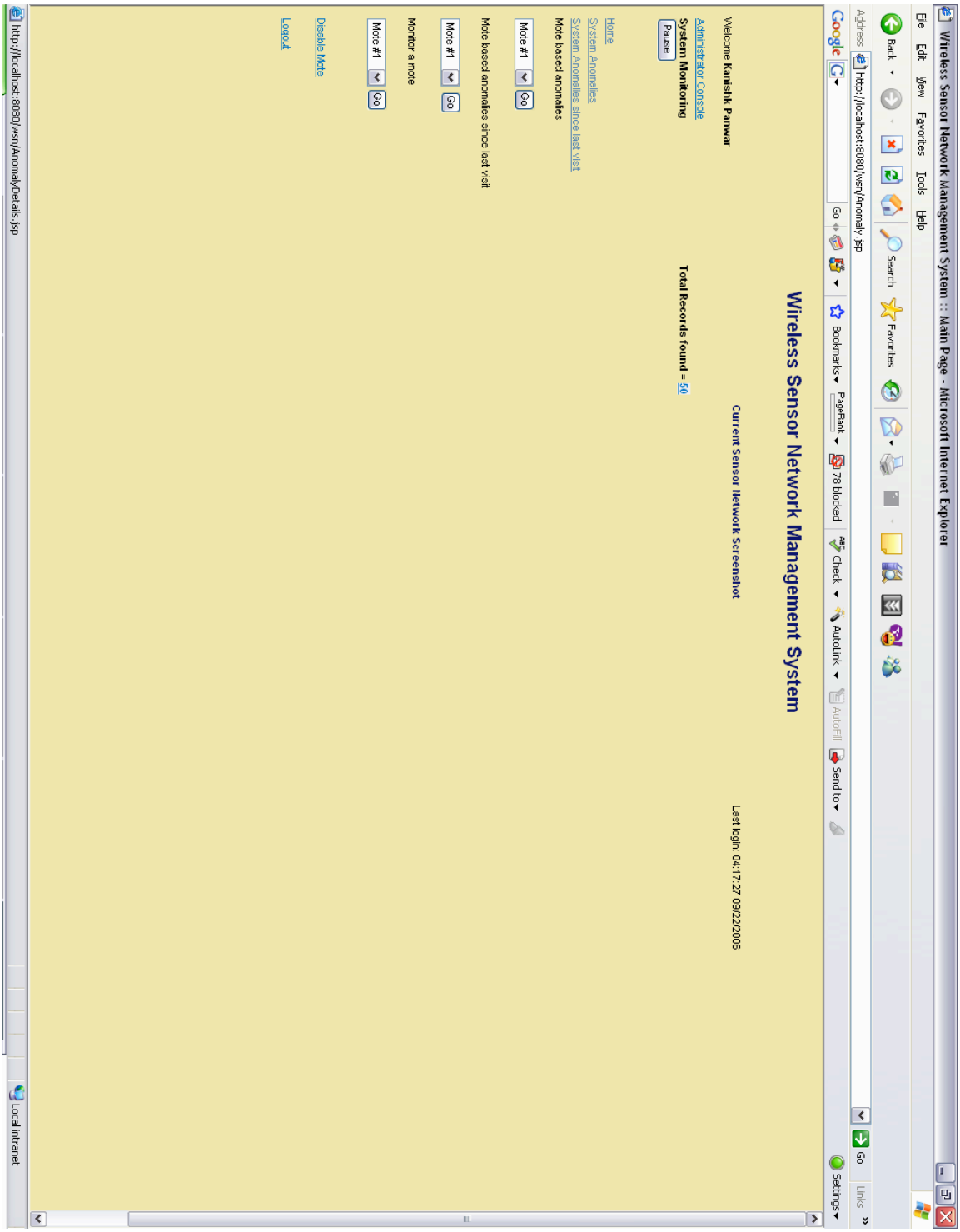


Given below is the screen shot of the System Anomalies since last visit page where total number of anomalies occurred in the system since user's last login is displayed.



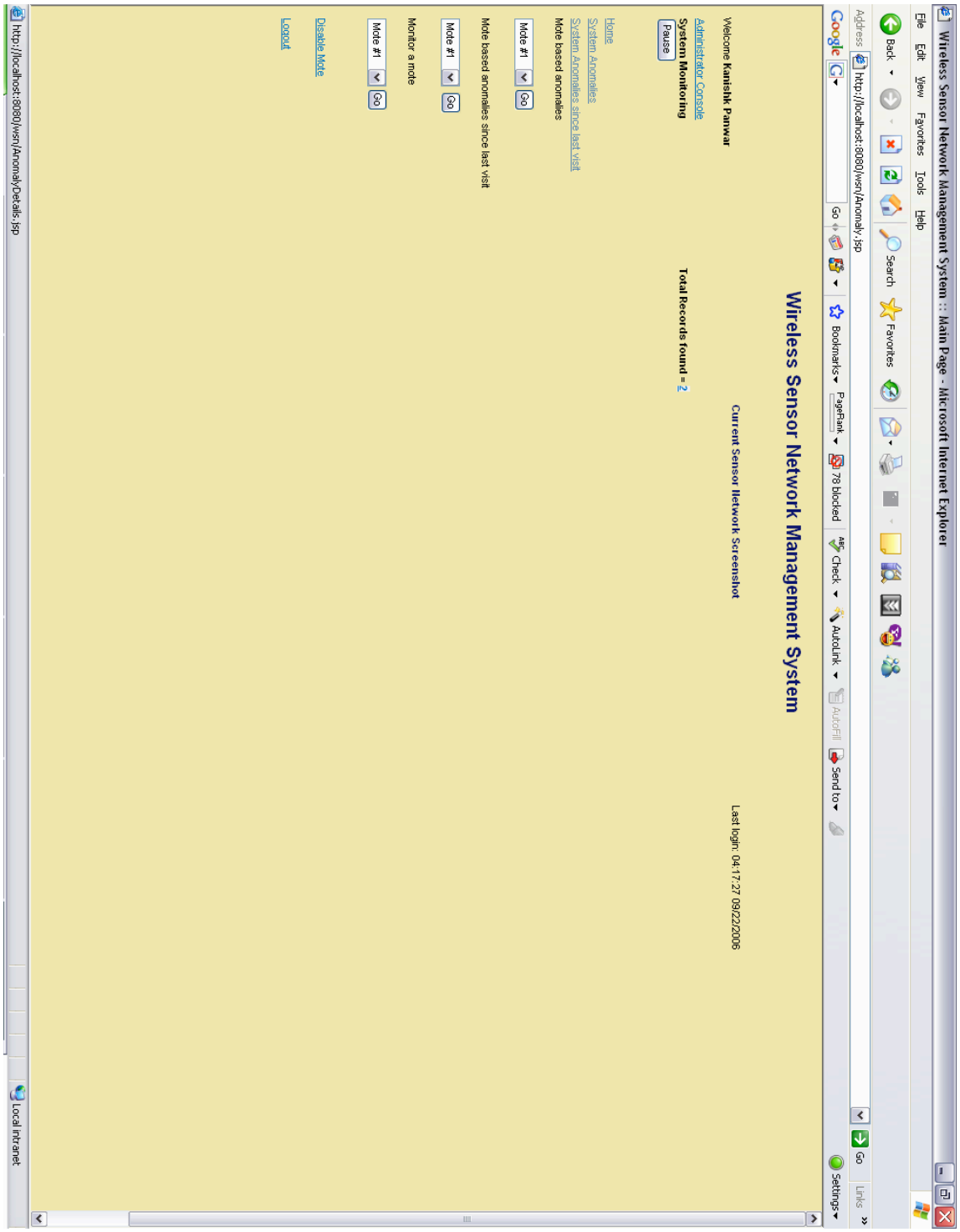
5.3.5 Mote Anomalies Page

Given below is the screen shot of the Mote Anomalies page where total number of anomalies occurred in the system is displayed. This page is similar to System anomalies page except for that fact that this page displays only individual mote anomalies.



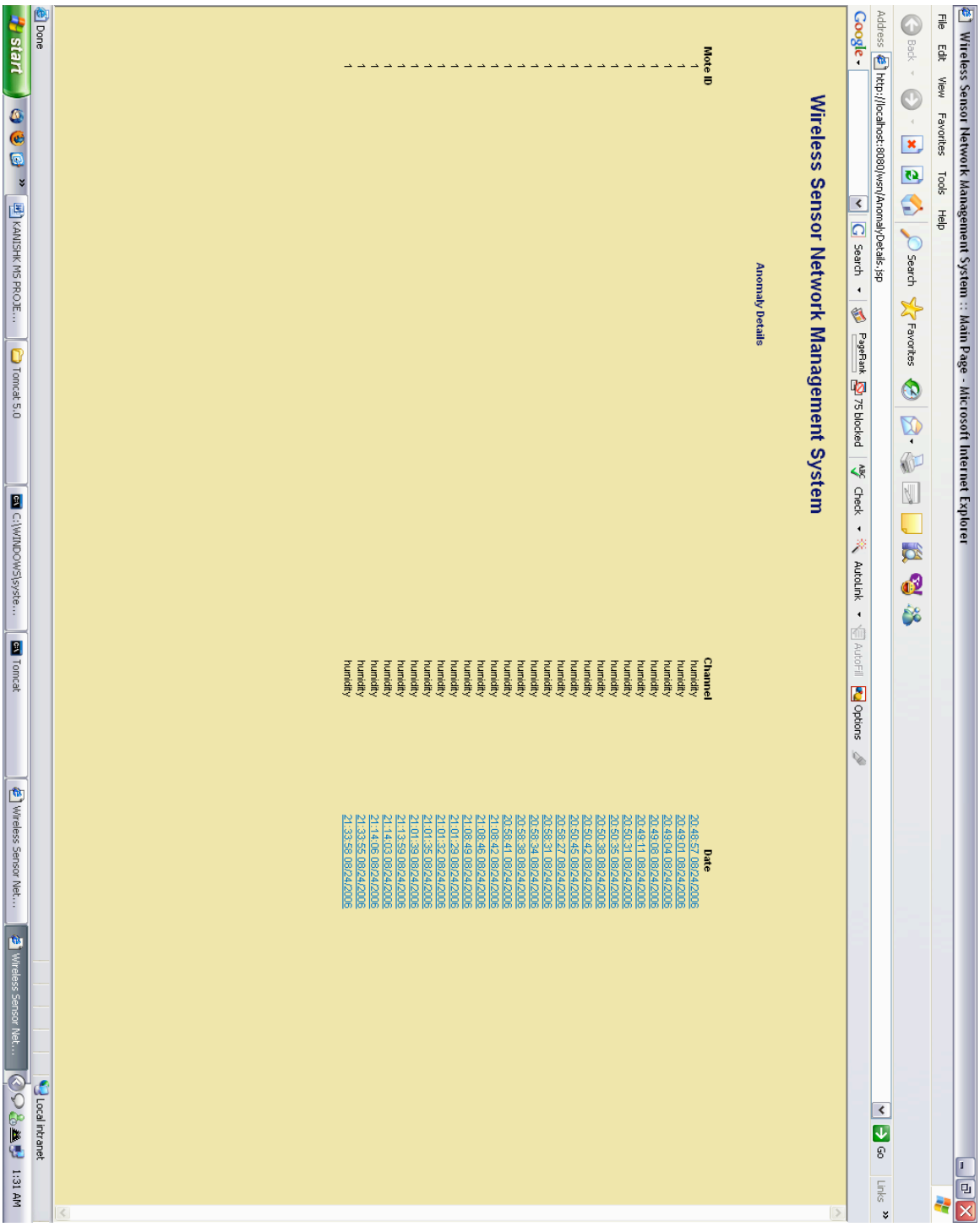
Given below is the screen shot of the Mote Anomalies since last visit page where total number of anomalies occurred for a given mote since user's last login is displayed.

This page is similar to System anomalies since last visit page except for that fact that this page displays only individual mote anomalies since user's last visit.



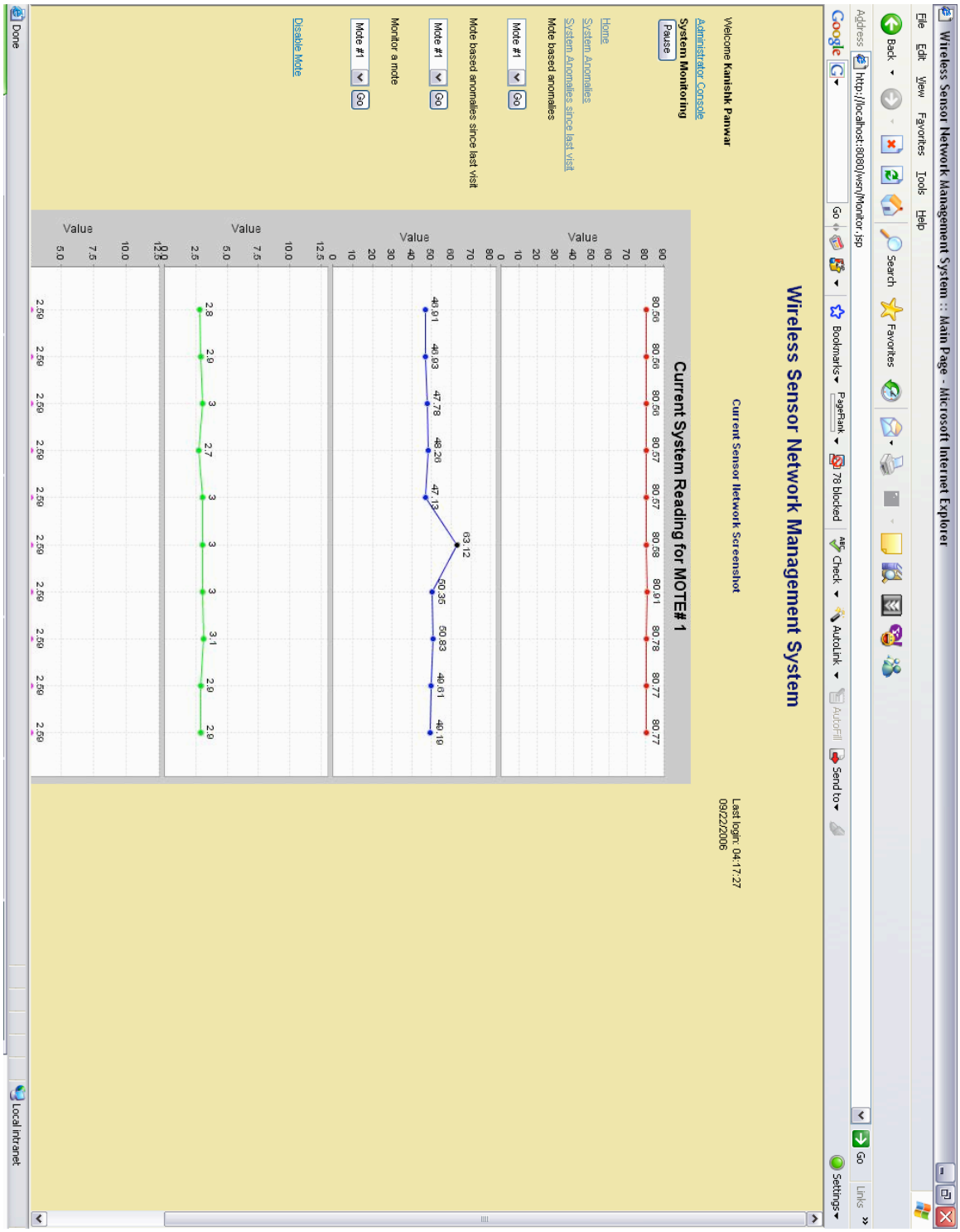
5.3.6 Anomaly Report Page

Given below is the screen shot of the Anomaly Report page where all the Returned results from anomaly search are displayed. This page displays the mote ID, the channel type and Date/Time when the anomaly occurred. Clicking on the Date/Time link the PDF report is launched in a new window.



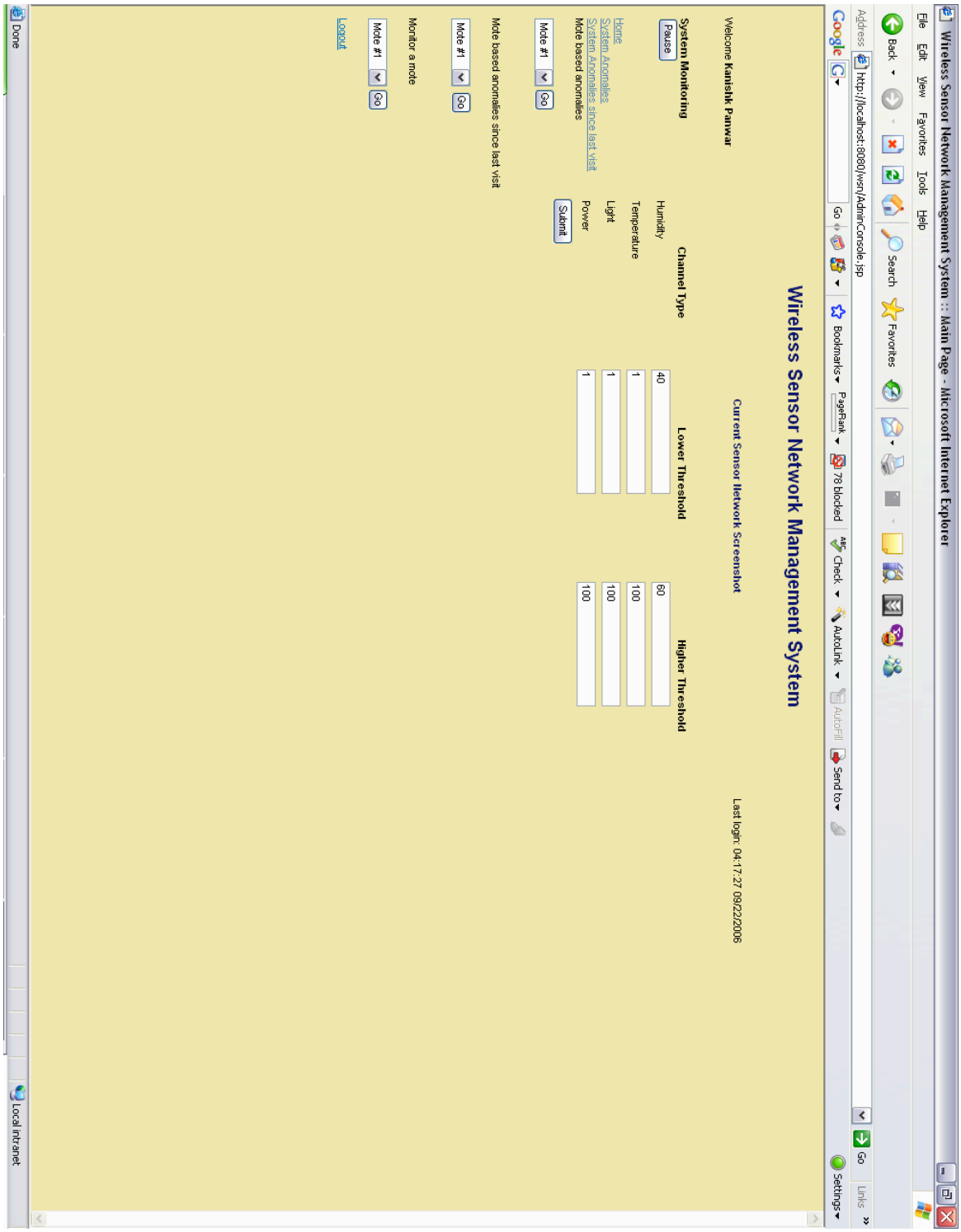
5.3.7 Mote Monitoring Page

Given below is the screen shot of the Mote Monitoring page that refreshes the readings graph every 5 seconds. All the anomaly data points are displayed in black.



5.3.8 Administrator Console Page

Given below is the screen shot of the Administrator Console where an admin can change the threshold values of the channels and those are immediately reflected on the reading graphs displayed on the main page and mote monitoring page.



In the application, the administrator can enable or disable a mote explicitly as per the requirements. Though the motes keep sending the readings to the base station but they are ignored by the application.

6 Executive Summary

6.1 Goals Achieved

In this application, I have successfully managed to achieve the following goals.

1) User Scalability
2) Improved Graphical User Interface
3) Storage of reading for data analysis.
4) Mote rejection feature that allows the application to disable a mote after a certain amount of time if the mote does not send any reading.
5) Software installation independent.
6) Application security.
7) Real time monitoring.
8) PDF report generation.

6.2 Future Improvements

Given below are some possible improvements or project alterations that can be useful.

1) Implementing Monitoring termination from the web console because currently administrator can only pause the monitoring. That still allows the motes to send reading but they are ignored by application.
2) Implement more data analysis and statistical algorithms.
3) Make the application multi administrator compatible. Currently it is assumed that there is only one administrator.
4) Implement the data plotting graphs in AJAX and improve the image updating time.

7 References

[1] http://www.octavetech.com

[2] Wireless Sensor Networks: SMART ALARM APPLICATION; MS Project – Chintan Patel, Computer Science Dept, RIT.

[3] *Wireless Sensor Networks: Architectures and Protocols*, by Edgar H., Jr. Callaway, Edgar H. Callaway ISBN: 0849318238

[4] *Wireless Sensor Networks : An Information Processing Approach*, by Feng Zhao, Leonidas Guibas ISBN: 1558609148

[5] http://en.wikipedia.org/wiki/SQL_Server_2000

[6] http://www.tinyos.net

[7] http://apache.org

[8] http://en.wikipedia.org/wiki/Event-driven_programming

[9] http://en.wikipedia.org/wiki/Mvc

[10] http://moteiv.com

[11] *Wireless Sensor Networks - New Challenges in Software Engineering*, Jan Blumenthal, Matthias Handy, Frank Golatowski, Marc Haase, Dirk Timmermann

[12] *The Design Space of Wireless Sensor Networks*, KAY ROMER, FRIEDEMANN MATTERN, ETH ZURICH

[13] http://en.wikipedia.org/wiki/Wireless_sensor_networks

[14] http://www.jfree.org/jfreechart/

[15] http://www.lowagie.com/iText/