Desktop based Software Application for Weather Monitoring System

Pragati Yadav[1] Vikas Lohtia[2] Rugved Durbade[3] Bageshree Khare[4]
[1,2,3,4]BE Student
[1,2,3,4]Department of Computer Science & Engineering
[1,2,3,4]Bharati Vidyapeeth College of Engineering Lavale, India

Abstract— Weather monitoring plays an important role in human life, so the collection of information about the temporal dynamics of weather changes is very important. In any industry during certain hazards it is very important to monitor weather. The fundamental aim of this paper is to develop an embedded system to design a weather monitoring system which enables the monitoring of weather parameters in an industry. Such a system contains pair of sensors like temperature, Gas and humidity will be monitored. This paper presents one such software application bearing in mind some of the hardware that could be used in weather monitoring Implementation.

General Terms: Filtering data, Desktop Computing & Software Implementation
Keywords: File Handling, Desktop based Weather monitoring System, Query based

I. INTRODUCTION

An automated weather station is an instrument that measures and records meteorological parameters using sensors without intervention of humans. The measured parameters can be stored in a built-in data logger or can be transmitted to a remote location via a communication link. If the data is stored in a data logger, recorded data must be physically downloaded to a computer at a later time for further processing. Therefore, the communication system is an essential element in an automated weather station. Today, automated weather stations are available as commercial products with variety of facilities and options [1-3]. Although automated weather stations can be built and implemented in remote parts of Sri Lanka to bring down the cost of maintaining weather stations, until recently, not much emphasis has been given for building and using such instruments locally. Automated weather stations have been developed in universities by interfacing meteorological parameter monitoring sensors to microcomputer/ commercially available data loggers with communication devices or through serial and parallel ports to obtain hard.

Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. Unlike AWT, Java Swing provides platform-independent and lightweight components. The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc. The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI. Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method. Not be a novel one as there may be chances that the free weather forecasting website's updating intervals frequency.

This paper presents one such novel concept on Desktop based weather monitoring system, where there is no need of dependency on any foreign source of medium for result generation as this system is a self-sustaining network. Here, the system is self-dependent as it forms a complete network with several sensor nodes each containing different sensors for sensing different parameters which are deployed in required geographical locations & these are monitored by a gateway station with a Base Station to input Desktop. The proposed system accepts user queries through Base station & presents the end user with result after invoking & executing relevant services which helps in prediction of weather forecast.

Another system with a feature where the end user can make the Desktop through any hand held devices like PDA, Mobile phones etc. with an application running in it, where the required info on weather can be queried is directly connected to a weather information server which gets the data's from a website supplying a free weather forecast.

The application built on J2ME (Java 2 Platform Micro Edition) in the device enables the user to send Desktop through an interface which connects to information server for further processing of the Desktop to generate required result.

Base station, Sensor node & respective sensor's are the major components forming a basic weather system. This system is designed considering three different sensor nodes with each having three different sensors to sense three different parameters i.e. Temperature, Humidity & Wind. The Desktop is requested by user from the base station which is sent to the server for processing and the returned results are displayed on the base station. In this system, the gateway node is responsible for serving the queries/requests from the clients. Upon receiving the Desktop it parses the Desktop and then validates the node number specified in it. Then it forwards the remaining data items in the Desktop to the sensor node specified. The sensor node is responsible for identifying the type of sensor (temperature, humidity or wind) from which the data is requested. Then it performs the function requested (avg, max or min) over the requested range of values and returns the results to the gateway node. Gateway node then returns this result to the client.

In this system socket connection has been used to connect between base station and server. So this enables the base station on one computer and server on other computer to get connected wirelessly or both the server and base station can be connected to each other (using cal host) via ports. A block diagram of the presented systems is given below

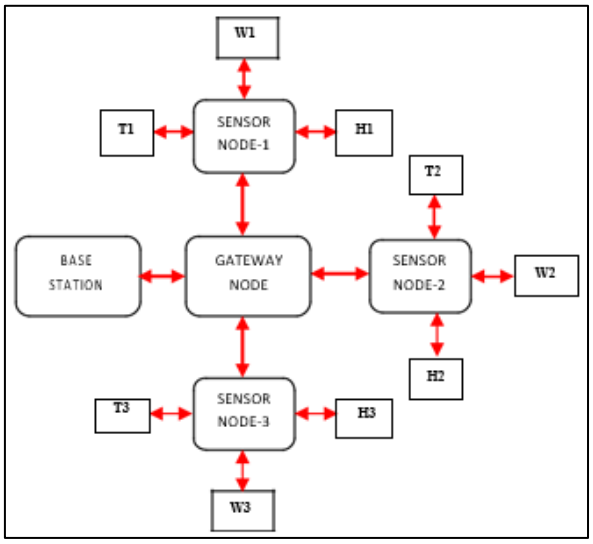

Fig. 1: Block Diagram of WSN Weather monitoring system

II. HARDWARE CONSIDERATIONS

The developed system under this project can be utilized on 'Gumstix' (a type of Mote). Gumstix is named as such cause of its size which resembles that of a Glue stick. Gumstix is actually a small computing system which comprises of Memory (RAM & ROM), Processor, OS & Networking capabilities. There are basically two manufacturing companies for Gumstix, one is 'Overo' series under Texas Instrument & the other is 'Verdex Pro' series under Marvell XScale. Upon the two said models 'Overo Earth' is more advanced with Enhanced memory & several special features like microSD slot, I2C, UART, SPI pins included in it which doesn't come with the latter 'Verdex Pro' model. Although these two said models use different processors the speed of them remains same at 600MHz i.e. 'Overo Earth' model uses OMAP 3503 application with ARM Cortex-A8 processor & 'Verdex Pro' uses Marvell PXA270 with XScale. Temperature sensor (Thermocouples or Thermistor), Humidity sensor & Wind sensor can be connected thro' GPIO/I2C/UART/USB etc to any of the above said mote to run the developed system as it is designed to work as a Weather station detecting all three different parameters mentioned.

III. SOFTWARE IMPLEMENTATION

The Desktop system software is developed using Java language. A Java Virtual Machine (JVM) can be setup on the gum Stix which would run the Java code on the Linux operating system. JamVM(6) is an example of a virtual machine used frequently. It is small in size and complies with Java Virtual Machine (JVM) specification.

The software system is comprised of classes that help the system to be operated in a desired manner. In the developed system, different Sensor's class generates arbitrary values for respected sensor type to represent temperature, humidity & wind speed using a random generator Java component. The Sensor node class collects data from the sensor class and processes the information that is collected. Data is stored at the Sensor node in arrays and same data is replaced with new data every 10 seconds & data's older than 24 hours are over written on FIFO (First in First Out) mode.

The Base station class provides operator three different types of queries to be made on three different sensor nodes & each having three different sensor types. The Desktop on either one of the parameters (temperature/ humidity/wind speed) with specification of time range (start time & stop time with respect to present time) for the parameter's average, maximum and minimum can be extracted respectively. When the operator makes a Desktop the base station class enquires the Gateway node class for response to the Desktop. The computed data is returned to Base station and in turn shown to the operator through a graphical user interface (GUI). GUI source code for Client lies in the Client GUI class (node input) & output GUI source in Server class which are constructed to handle all the input and output of the system. The GUI of base station (Input) & Desktop server (to see Output) is shown below.

Fig. 2: GUI of Base Station & Desktop Server

IV. DESIGN & WORK FLOW OF THE DESKTOP

Base station forms the input for the Desktop through GUI provided; the Desktop then from the client is wirelessly forwarded to the Gateway node for further validation of the Desktop format & processing is done to obtain result.

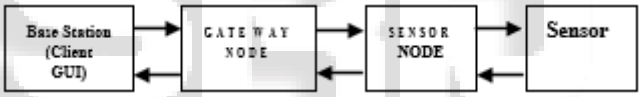

Fig. 3:

Based on the above function analysis the structural flow & classes involved for Desktop to be executed from (Base station point of view) is show below in the next page.

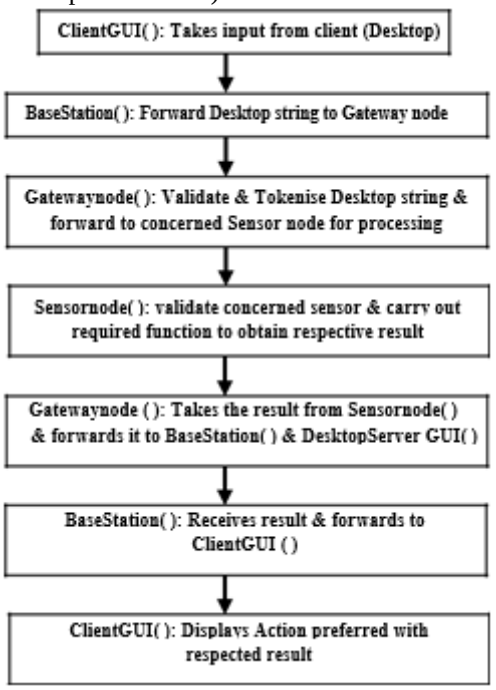

Fig. 4: Structural Work Flow of Desktop

A. Methods & Functions of the Gatewaynode Class

Gateway Node forms the entry point to the server, where the Desktop through Base station is received through socket connection. Gate way node parses the Desktop & then validates the sensor node specified in it, thus forwarding further Desktop to be processed to the concerned sensor node.

SI No	Method	Function description
1	Server Socket ( )	To listen Desktop from Base station.
2	Execute Desktop ( )	To execute which sensor node to be selected based on Desktop
3	Sn1 ( )	To select Sensor Node-1.
4	Sn2 ( )	To select Sensor Node-2.
5	Sn3 ( )	To select Sensor Node-3.
6	Validate & Tokenise Desktop St ring ( )	Forward's to concerned Sensor node for further processing
7	Server Gui ( )	To display final queried result on server GUI

Table 1: Methods & Functions of GatewayNode Class

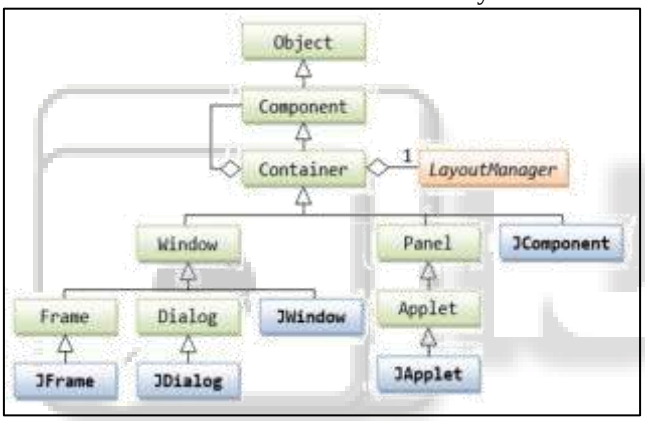

Fig. 5: Swing GUI for Java

B. Methods & Functions of the Sensornode Class

Methods & functions of SensorNode Class which receives Desktop validate Sensor & processes on Desktop are given below in Table. 2.

SI No.	Method	Function description
1	ActivateSensors ( )	To activate concerned sensor required to process Desktop.
2	ReportTemperatureData ( )	Obtain data from Temperature Sensor.
3	ReportHumidityData ( )	Obtain data from Humidity Sensor.
4	ReportWindData ( )	Obtain data from Wind Sensor.
5	ProcessDesktop ( )	Process data in order to achieve Desktop related result
6	StopSensors ( )	To Stop sensors that are running

Table 2: Methods & Functions of SensorNode Class

C. Work Flow of File to Database Class

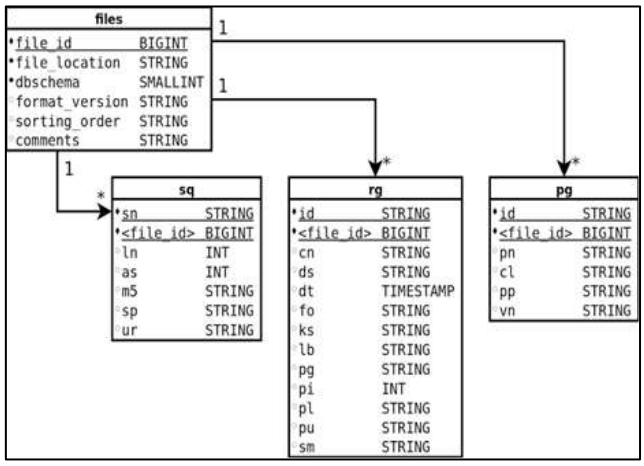

Fig. 6:

V. DESKTOP FORMAT & EVIDENCE OF SYSTEM PROCESSING

As explained in the previous sections, the proposed system contains three different sensor node's which further is connected to three different types of sensors for sensing three different parameters. To extract a particular data from any individual sensor & processing it calls a need for a specific path for the system to reach for it & this should be specified by the user. A specifically designed Desktop Format is used in this system in order to instruct the system & perform required processes as per the Desktop. It is recommended for the users to use the suggested format in order to place any Desktop without which this system is non-responsive & may end up with error messages. The Desktop format consists of five tokens separated by comas as shown in figure below.

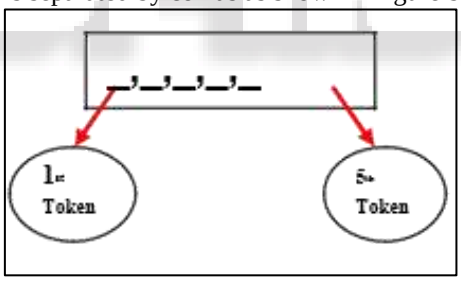

Fig. 7: Desktop Format used

Description of individual tokens & their meaning is given below in Table 3 which is given below.

1st Token	This token specifies the node number (ex: 1 or 2 or 3). As the system has only 3 sensor nodes so only 1 or 2 or 3 can be entered depending on the node required.
2nd Token	This token specifies the sensor type in the specified node (ex: temperature or humidity or wind). For temperature sensor need to enter 'T'or 't' likewise for humidity 'H' or 'h' and for wind 'W' or 'w'. It is case insensitive.
3rd Token	This token specifies the start time in minutes in the past with respect to the current time, i.e. it specifies the starting point of the range of data over which the Desktop should be processed. (E.g. if the user needs to start from 2 Hrs in the past then the value '120' should be entered here.)

4th Token	This token specifies the time period in minutes over which the required function will be performed, i.e. the range of data values over which the Desktop should be processed. (E.g. to request a the data over a 60 minute period of time beginning from 2 hours in the past, the third token value should be '120' and this token's value should be 60).
5th Token	This last token specifies the type of function to be processed (ex:- 'AVG' for average, 'MAX' for maximum and 'MIN' for minimun). This is case- insensitive. For example if a user wants to know the maximum temperature from node 1 and starting from 2 mins in the past and 1 min after that, then the user should enter in the Desktop in a format like '1,T,2,1,MAX'.

Table 3: Detailed information of Desktop Format

A Desktop with the mentioned format as explained above is shown in the figure 8 with its details explained after the figure 8.

Fig. 8: An example of Desktop in previously explained format.

The above figure shows an example know the average humidity of sensor node 1 over a period of 1 minute starting from 2 minutes in the past.

A. Various Methods used to perform File operation:

- writeBytes (String s): Writes the string to the file as a sequence of bytes.
- readLine(): Reads the next line of text from this file.
- getFilePointer(): Returns the current offset in this file.
- length(): Returns the length of this file and return type is long.
- close(): Closes this random access file stream and releases any system resources associated with the stream.
- setLength(long newLength): Sets the length of this file.
- seek(long pos): Sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.

B. Calculations Needed after Filtering

The acceptability of the data provided by the sensors by the meteorologists is only possible if the sensors are calibrated against conventional weather stations. In order to take care of this, calibration of the sensors was carried out using conventional weather stations at KMD; Chiromo Observatory (run by the Department of Meteorology, University of Nairobi) was for the initial experiments. After the calibration exercise, the sensor boards were used to design SenseWeather. Calibration experiments were conducted to evaluate the field readiness of the Libelium's Waspmote [7] platform to support weather forecasting.

The calibration exercise was carried out for a period of one year between July 2011 and July 2012. It entailed taking readings from both the sensors and the weather stations in parallel on 24-7 basis. The calibration model followed a 3-steps experimental process with three types of experiments: (1) pilot experiments; (2) explanatory experiments; and (3) confirmatory experiments. A systematic error analysis based on three error types: Mean Absolute percentage Error (MAPE), Mean Error (ME) and Root Square Error (RSE) then followed. We also checked for inherent errors that are inbuilt in the sensor boards.

From Fig 10, it can be seen that every 10 seconds, the sensors Report data to their respective sensor nodes. When Desktop is entered the validation of the sensor node takes place (i.e. sensor node 1 or 2 or 3) in the gateway node and the remaining part of the Desktop is forwarded to specified sensor node.

C. Evidence of Desktop Processing

The systems dependability can be judged upon the accuracy with which the system presents the result for the Desktop fetched. So, an example to show the accuracy of the result is shown below which compares the theoretical values with the system presented values taking an example of finding an average temperature value for a time period of 1 minute from a Temperature sensor in Sensor node 1 is considered.

We finally used correlation coefficients and plots such as side-by-side boxplots to run similarity tests between various datasets. With impressive accuracies ranging from 92 to 99 %, this gave us the confidence to move to the next step; system design and implementation.

The sensors were programmed to take readings every 30 minutes while the readings from the weather station were taken on hourly basis. That is, at hour t (say 1 GMT), the sensor boards recorded two readings for each of the sensors. For example, with 6 sensor boards, each fitted with 3 sensors (temperature, humidity and pressure), this would result in 12 readings for temperature, 12 readings for humidity and 12 readings for pressure. In order to aggregate these readings for the purpose of comparing them with the respective readings from the weather station, two options were pursued:

1) Option 1: Average All Sensor Readings Taken Within the Hour: Where S

S is the aggregated reading for Sensor S; for example, S could be temperature sensor or humidity sensor. Si1 and S i2 are the sensor reading for Sensor S on Sensor Board i. For instance, in the case of five sensor boards, the aggregated reading for temperature sensors would be computed.

2) Option 2: Average Readings Taken Closest to the Hour

In this case, only one out of the two sensor reading from each sensor board is considered; the one closest to the weather station readings' observation time. That is: In the case, given five sensor boards, the aggregated reading for the temperature sensor.

3) Theoretical calculations:

Sensor reports value in every 10 seconds

Specified time in minutes, start time = 2min and range in minutes = 1 min.

2min = 120 seconds = 12 values

1min = 60 seconds = 6 values

So according to the Desktop entered, the maximum temperature value can be found by considering first 6 of last 12 values from the current reported value as shown in snapshot. From snapshot

Current reported value = 17

First 6 of last 12 values = 21, 16, 11, 20, 17, 13. Average of these values = 21+16+11+20+17+13 = 98, so, 98/6 = 16.33.

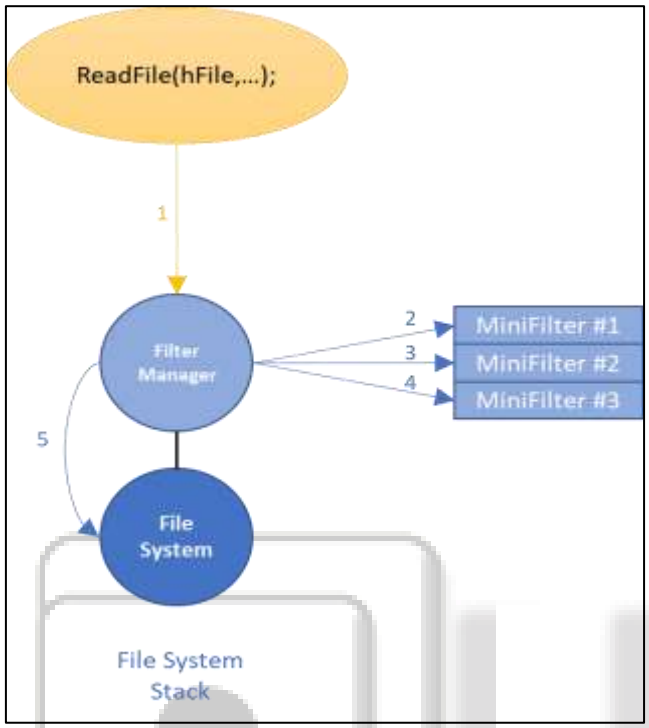

Fig. 9: File Handling Operations

Hence, System presented Values were verified with theoretical values. This shows that this system id dependable.

VI. CONCLUSION

This paper demonstrates Design and Implementation of Weather Monitoring & Controlling System used for controlling the devices as well as monitoring the environmental parameters. Embedded controlled sensor networks have proven themselves to be a reliable solution in providing remote control and sensing for environmental monitoring systems. The sensors have been integrated with the system to monitor and compute the level of existence of Accelerometer, gas, temperature and humidity in atmosphere using information and communication technologies. The sensors can upload the data in Lab view using serial Communication. Implementation was done considering some of hardware's available in the market and this system could easily be implemented while using the suggested hardware. The systems versatility was tested considering different aspects which have been noted in previous sections. The performance on the implemented software was evaluated & its real time performance can be only evaluated through hardware implementations. As the first step, the software implementation was considered & successfully completed. The process is ongoing for its hardware counter par implementation

VII. FUTURE SCOPE

Adding of more sensors to monitor other environmental parameters such as Soil PH Sensor, CO2 and oxygen Sensor while allowing the replacing of current sensors if a wider range of measurements is desired. And also Integration of additional monitoring devices such as a Wi-Fi camera to monitor growth of agricultural product. And also the data can be uploaded to web server continuously.

REFERENCES

[1] Tinghuai Ma, Hao Cao, Donghai Guan and Sung Young Lee "An Efficient Distributed Weather Data Sharing System Based on Agent" the International Arab J. Of Inform Tech., vol 2, pp. 170-178, March 2012.
[2] Jinbiao Huo, "Research on Design of a Desktop System of Weather Information in a Mobile Telephone based on MIDP", in Proc. WISA, 2009, pp. 309-311.
[3] David Malan, Thaddeus R.F. Fulford-Jones, Matt Welsh and Steve Moulton "CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care " Internet:http://lcawww.epfl.ch/luo/WAMES%202004.htm> , 2004 [Jun. 23,2013]
[4] T. Sun, B. Peng, "Developing an End to End Wireless Application Using MIDP and J2EE," Microcomputer Information, 2006, pp. 159-161.
[5] C. Li, J.R. Luo, H.Z. Wang et al, "Program Composition Method Summary on CLDC/MIDP Client," Computer and Modernization, 2005, pp. 5-10.
[6] Sparks L. & Sumner G., "Microcomputer Based Weather Station Monitoring System", Journal of Microcomputer Applications, 7, pp. 233-24, 1984.
[7] Matt Welsh and Geoff Mainland. Programming sensor networks using abstract regions. In the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04), March 2004.
[8] Alec Woo, Terence Tong, and David Culler. "Taming the underlying challenges of reliable multihop routing in sensor networks", In the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), November 2003.
[9] V. Kumar, A. Pentland, "DiaBetNet: Learning and Predicting for Better Health and Compliance", Diabeties Technology Society meeting, Foster City, CA Oct 31-Nov 2
[10] L. Li, I. Harrocks, "A Software Framework For Matchmaking Based on Semantic Web Technology" in Proc. WWW Conference, 2003
[11] V. Agarwal et al., "Synthy. A System for End to End Composition of Web Services" in Journal of Web Semantics, Vol. 3, Issue 4, 2005

